Lecture Notes in Computer Science 2119
Edited by G. Goos, J. Hartmanis and J. van Leeuwen

Springer
Berlin
Heidelberg
New York
Barcelona
Hong Kong
London
Milan
Paris
Singapore
Tokyo

Vijay Varadharajan   Yi Mu (Eds.)

Information Security and Privacy

6th Australasian Conference, ACISP 2001
Sydney, Australia, July 11-13, 2001
Proceedings

Springer

Series Editors

Gerhard Goos, Karlsruhe University, Germany
Juris Hartmanis, Cornell University, NY, USA
Jan van Leeuwen, Utrecht University, The Netherlands

Volume Editors

Vijay Varadharajan
Yi Mu
Macquarie University, Department of Computing
North Ryde, NSW 2109, Australia
E-mail: {vijay,ymu}@ics.mq.edu.au

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Information security and privacy : 6th Australasian conference ; proceedings / ACISP 2001, Sydney, Australia, July 11 - 13, 2001. Vijay Varadharajan ; Yi Mu (ed.). - Berlin ; Heidelberg ; New York ; Barcelona ; Hong Kong ; London ; Milan ; Paris ; Singapore ; Tokyo : Springer, 2001
(Lecture notes in computer science ; Vol. 2119)
ISBN 3-540-42300-1

CR Subject Classification (1998): E.3, K.6.5, D.4.6, C.2, E.4, F.2.1, K.4.1

ISSN 0302-9743
ISBN 3-540-42300-1 Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Boller Mediendesign
Printed on acid-free paper SPIN: 10839883 06/3142 5 4 3 2 1 0

Preface

ACISP 2001, the Sixth Australasian Conference on Information Security and Privacy, was held in Sydney, Australia. The conference was sponsored by Information and Networked System Security Research (INSSR), Macquarie University, the Australian Computer Society, and the University of Western Sydney. I am grateful to all these organizations for their support of the conference.

The aim of this conference was to draw together researchers, designers, and users of information security systems and technologies. The conference program addressed a range of aspects from system and network security to secure Internet applications to cryptography and cryptanalysis. This year the program committee invited two international keynote speakers Dr. Yacov Yacobi from Microsoft Research (USA) and Dr. Clifford Neumann from the University of Southern California (USA). Dr. Yacobi's talk addressed the issues of trust, privacy, and anti-piracy in electronic commerce. Dr. Neumann's address was concerned with authorization policy issues and their enforcement in applications.

The conference received 91 papers from America, Asia, Australia, and Europe. The program committee accepted 38 papers and these were presented in some 9 sessions covering system security, network security, trust and access control, Authentication, cryptography, cryptanalysis, Digital Signatures, Elliptic Curve Based Techniques, and Secret Sharing and Threshold Schemes. This year the accepted papers came from a range of countries, including 7 from Australia, 8 from Korea, 7 from Japan, 3 from UK, 3 from Germany, 3 from USA, 2 from Singapore, 2 from Canada and 1 from Belgium, Estonia, and Taiwan.

Organizing a conference such as this one is a time-consuming task and I would like to thank all the people who worked hard to make this conference a success. In particular, I would like to thank Program Co-chair Yi Mu for his tireless work and the members of the program committee for putting together an excellent program, and all the session chairs and speakers for their time and effort. Special thanks to Yi Mu, Laura Olsen, Rajan Shankaran, and Michael Hitchens for their help with local organization details. Finally, I would like to thank all the authors who submitted papers and all the participants of ACISP 2001. I hope that the professional contacts made at this conference, the presentations, and the proceedings have offered you insights and ideas that you can apply to your own efforts in security and privacy.

July 2001 Vijay Varadharajan

AUSTRALASIAN CONFERENCE ON INFORMATION SECURITY AND PRIVACY ACISP 2001

Sponsored by
Macquarie University
Australian Computer Society

General Chair:

Vijay Varadharajan	*Macquarie University, Australia*

Program Chairs:

Vijay Varadharajan	*Macquarie University, Australia*
Yi Mu	*Macquarie University, Australia*

Program Committee:

Ross Anderson	*Cambridge University, UK*
Colin Boyd	*Queensland University of Technology, Australia*
Ed Dawson	*Queensland University of Technology, Australia*
Yvo Desmedt	*Florida State University, USA*
Paul England	*Microsoft*
Yair Frankel	*Columbia University, USA*
Ajoy Ghosh	*UNISYS, Australia*
Dieter Gollman	*Microsoft*
John Gordon	*ConceptLabs, UK*
Kwangjo Kim	*ICU, Korea*
Chuchang Liu	*DSTO, Australia*
Masahiro Mambo	*Tohoku University, Japan*
Wenbo Mao	*Hewlett-Packard Lab., UK*
Chris Mitchell	*London University, UK*
Eiji Okamoto	*University of Wisconsin, USA*
Joe Pato	*Hewlett-Packard Lab., USA*
Josef Pieprzyk	*Macquarie University, Australia*
Bart Preneel	*Katholieke University, Belgium*
Steve Roberts	*Witham Pty Ltd, Australia*
Qing Sihan	*Academy of Science, China*
Rei Safavi-Naini	*University of Wollongong, Australia*
Jennifer Seberry	*University of Wollongong, Australia*
Yuliang Zheng	*Monash University, Australia*

Table of Contents

A Few Thoughts on E-Commerce
Keynote Lecture

Yacov Yacobi

Microsoft Research, USA

Abstract. I discuss a few notions related to e-commerce, such as: trust, privacy, and the economies of piracy and anti-piracy.

Trust

We have been using the term trust without any quantification for a long time. We need a technical term that will capture some of its meaning and enable quantification. The parallel may be Shannon's quantification of Information. It does not capture all of the meaning of information, but is useful enough. I suggest equating the amount of trust that a system needs with the value that this system is supposed to protect. It seems to me that we cannot get around this. We may push trust in different directions, we may distribute it, but we cannot do without it. For example, one important difference between symmetric and asymmetric key cryptography, is that the latter assigns trust to potentially more trustworthy entities.

Privacy

ID theft is the major issue; much more so than exposure of shopping patterns. ID-theft occurs when somebody issues a credit card on my name, max it out, and disappears, leaving me with the tedious task of salvaging my credit profile (most of the $$ damage is eaten by the credit card company). It happens because today when we want to prove that we know some secret, we expose it. The annual dollar amount in damages is already in many Billions, and rapidly increasing.

Public Key cryptosystems make it possible to prove knowledge of secrets without exposing them. Widespread deployment of PKI will solve most of this problem.

But the privacy issue that gets the headlines is exposure of shopping patterns. Long ago we traded this kind of privacy for credit. Credit card companies know what, where and when we buy, in real time. They can trace us better than the KGB in their heydays could trace citizens of the Soviet Union. We could use cash and avoid it, but we overwhelmingly chose the convenience of credit. Later we chose to trade even more of our location privacy, for mobility. The cell phone companies can now trace our physical location to within a few hundred feet on a continuous basis.

Now we have to choose a tradeoff between privacy and bandwidth. The bandwidth bottleneck is in our heads; there is only so much that we can absorb in a day. Some knowledge of our shopping patterns can help in targeted ads that will alleviate this bottleneck. My bet is that if done well, and if users are free to choose, most of them will choose to trade some privacy for this service.

On the Economies of Piracy and Anti-piracy

We consider the following players in the piracy game: Defense and offense which is further subdivided into transmitters and receivers of piracy. We assume that all the players are economically rational, and try to maximize their profits. With each player we associate an inequality of the general type costs ¡ profits. We scale the inequality per a client machine. Let v denote the average aggregate value of protected objects on a client machine. Each offense player has a different cost of attack per machine, which is compared to v. A system for which the inequality holds for every player is sound. We consider active and passive protected objects (SW and content, respectively). We consider two types of protecting systems: open and closed systems. The former can run protected and unprotected objects. The latter runs only protected objects. A non-protecting system is promiscuous.

Napster-like systems are covered in the sense that if the Napster offense were economically motivated (either receivers or transmitters) then sound systems would deter them. Offenders who are not economically motivated (vandals) would not be deterred by a sound system no matter what the delivery mechanism is. We outline a few open problems on the way to sound anti-piracy systems.

New CBC-MAC Forgery Attacks

Karl Brincat*[1] and Chris J. Mitchell[2]

[1] Visa International EU, PO Box 253, London W8 5TE, UK,
brincatk@visa.com

[2] Information Security Group, Royal Holloway, University of London, Egham, Surrey TW20 0EX, UK,
c.mitchell@rhul.ac.uk

Abstract. This paper is concerned with a particular type of attack against CBC-MACs, namely *forgery attacks*, i.e. attacks which enable an unauthorised party to obtain a MAC on a data string. Existing forgery attacks against CBC-MACs are briefly reviewed, together with the effectiveness of various countermeasures. This motivates the main part of the paper, where a family of new forgery attacks are described, which raise serious questions about the effectiveness of certain countermeasures.

1 Introduction

1.1 Use of MACs

MACs, i.e. *Message Authentication Codes*, are a widely used method for protecting the integrity and guaranteeing the origin of transmitted messages and stored files. To use a MAC it is necessary for the sender and recipient of a message (or the creator and verifier of a stored file) to share a secret key K, chosen from some (large) keyspace. The data string to be protected, D say, is input to a MAC function f, along with the secret key K, and the output is the MAC. We write MAC $= f_K(D)$. The MAC is then sent or stored with the message.

1.2 A Model for CBC-MACs

MACs are most commonly computed using a block cipher in a scheme known as a CBC-MAC (for *Cipher Block Chaining* MAC). This name derives from the CBC 'mode of operation' for block ciphers, and a CBC-MAC is computed using the same basic process. There are several variants of the CBC-MAC, although the following general model (see [1,9]) covers most of these.

The computation of a CBC-MAC on a bit string D using a block cipher with block length n, uses the following six steps.

1. *Padding.* The data string D is subjected to a padding process, involving the addition of bits to D, the output of which (the *padded string*) is a bit string of length an integer multiple of n (say qn).

* The views expressed in this paper are personal to the author and not necessarily those of Visa International

2. *Splitting.* The padded string is divided (or 'split') into a series of n-bit blocks, $D_1, D_2, \ldots, D_q$.
3. *Initial transformation.* Initial transformation I, which may be key-controlled, is applied to D_1 to give the first *chaining variable* H_1, i.e.
$$H_1 = I(D_1).$$
4. *Iteration.* Successive chaining variables are computed as
$$H_i = e_K(D_i \oplus H_{i-1})$$
for $i := 2, 3, \ldots, q$, where, as throughout, K is a block cipher key, $e_K(X)$ and $d_K(X)$ denote block cipher encryption and decryption of block X with key K, and $\oplus$ denotes bit-wise exclusive-or of blocks.
5. *Output transformation.* The n-bit *Output block* G is computed as
$$G = g(H_q)$$
where g is the output transformation (which may be key-controlled).
6. *Truncation.* The MAC is set equal to the leftmost m bits of G.

Most CBC-MACs adhere to this model, and such MACs will be the main focus of this paper.

1.3 Types of CBC-MAC Scheme

The latest version of the relevant international standard, namely ISO/IEC 9797-1, [1], contains six different CBC-MAC variants. These are based on combinations of two Initial transformations and three Output transformations.

- Initial transformation 1 is defined as:
$$I(D_1) = e_K(D_1)$$
where K is the same key as used in the Iteration step. I.e. Initial transformation 1 is the same as the Iteration step, and is the one used in both the original CBC-MAC, as defined in ANSI X9.9, [4], and CBC-MAC-Y (also known as the *ANSI Retail MAC*), standardised in ANSI X9.19, [3].
- Initial transformation 2 is defined as:
$$I(D_1) = e_{K''}(e_K(D_1))$$
where K is the same key as used in the Iteration step, and K'' is a block cipher key distinct from K.
- Output transformation 1 is defined as:
$$g(H_q) = H_q,$$
i.e. Output transformation 1 is the identity transformation, and is the one used in the original CBC-MAC, [4].

– Output transformation 2 is defined as:

$$g(H_q) = e_{K'}(H_q),$$

where K' is a block cipher key distinct from K.
– Output transformation 3 is defined as:

$$g(H_q) = e_K(d_{K'}(H_q)),$$

where K' is a block cipher key distinct from K. Output transformation 3 is the one used in CBC-MAC-Y, [3].

These options are combined in the ways described in Table 1 to yield four of the six different CBC-MAC schemes defined in ISO/IEC 9797-1, [1]. Note that algorithms 5 and 6 do not fit the general MAC model given above; as a result we do not consider these last two algorithms further in this paper.

Table 1. CBC-MAC schemes defined in ISO/IEC 9797-1

Algorithm number	Input transformation	Output transformation	Notes
1	1	1	The 'original' CBC-MAC scheme.
2	1	2	K' may be derived from K.
3	1	3	CBC-MAC-Y. The values of K and K' shall be chosen independently.
4	2	2	K'' shall be derived from K' in such a way that $K' \neq K''$.

Finally note that three Padding Methods are also defined in [1]. Padding Method 1 simply involves adding between 0 and $n-1$ zeros, as necessary, to the end of the data string. Padding Method 2 involves the addition of a single 1 bit at the end of the data string followed by between 0 and $n-1$ zeros. Padding Method 3 involves prefixing the data string with an n-bit block encoding the bit length of the data string, with the end of the data string padded as in Padding Method 1.

When using one of the six MAC algorithms it is necessary to choose one of the three padding methods, and the degree of truncation to be employed. All three Padding Methods can be deployed with all six MAC algorithms.

In the remainder of this paper the discussions primarily apply to MAC algorithms 1–4 from ISO/IEC 9797-1, used with Padding Methods 1–3. We also use the terminology of ISO/IEC 9797-1. In fact, these algorithms cover almost all CBC-MAC variants in common use today.

2 Attacks on CBC-MACs

There are two main types of attack on MAC schemes.

- In a *MAC forgery* attack [6], an unauthorised party is able to obtain a valid MAC on a message which has not been produced by the holders of the secret key. Typically the attacker will need a number of valid MACs and corresponding messages to use to obtain the forgery.
- A *key recovery* attack enables the attacker to obtain the secret key used to generate one or more MACs. Note that a successful key recovery attack enables the construction of arbitrary numbers of forgeries.

We introduce a simple way of quantifying the effectiveness of an attack. Following the approach used in [1], we do this by means of a four-tuple which specifies the size of the resources needed by the attacker. For each attack we specify the tuple $[a, b, c, d]$ where a denotes the number of off-line block cipher encipherments (or decipherments), b denotes the number of known data string/MAC pairs, c denotes the number of chosen data string/MAC pairs, and d denotes the number of on-line MAC verifications. The reason for distinguishing between the numbers c and d is that, in some environments, it may be easier for the attacker to obtain MAC verifications (i.e. to submit a data string/MAC pair and receive an answer indicating whether or not the MAC is valid) than to obtain the genuine MAC value for a chosen message.

3 Simple MAC Forgeries

We start by considering three 'simple' types of MAC forgery. All these forgery attacks apply regardless of the MAC algorithm in use.

- *MAC guessing.* The attacker selects a message and simply guesses the correct MAC value. The probability that the guess will be correct is 2^{-m}. Such attacks can be avoided by making m sufficiently large.
- *Verification forgery.* This is a simple development of the 'MAC guessing' technique. The attacker chooses a message, and then works through all possible MACs, submitting the chosen message combined with each MAC value for verification. This attack has complexity $[0, 0, 0, 2^m]$. Thus, even if an attacker only has access to a MAC verification function, selective verifiable forgeries are possible unless m is sufficiently large.
- *Trailing zeros forgery.* The third attack only applies when Padding Method 1 from [1] is in use. The attack works because of the observation that, if a padded message has final block D_q and the last '1' bit appears at position i (out of n) in D_q, then there are $n+1-i$ (unpadded) messages which, when padded, give the padded message. This means that, unless a message contains a multiple of n bits and ends in a '1' bit, given any message and MAC it is possible to discover other messages with the same MAC by deleting zeros from, or adding zeros to, the end of the message.

The same general type of attack would apply to any scheme using a padding method where the mapping from messages to padded messages is not injective. Fortunately, Padding Methods 2 and 3 do not suffer from this problem — indeed, the main motivation for the design of Padding Method 2 was to avoid this problem.

4 More Sophisticated Forgeries

We now consider further attacks which apply only to particular variants of the CBC-MAC.

4.1 Simple Cut and Paste Attack

Suppose the MAC function in use is the 'original' MAC scheme, i.e. ISO/IEC 9797-1 MAC algorithm 1. Then, given two messages with valid MACs (computed using the same secret key K), we can compute a third 'composite' message with a valid MAC without knowing the key. For further details see, for example, [1]

4.2 Birthday Attack

For this attack we suppose that Padding Method 3 is not being used. We also suppose that no truncation is employed, i.e. so that $m = n$. Suppose, by some means, an attacker discovers two messages with the same MAC. That is, suppose the attacker has found that the two messages with padded data strings $D_1, D_2, \ldots, D_q$ and $E_1, E_2, \ldots, E_r$ have the same MAC. Then it follows immediately that any pair of padded messages that have the form $D_1, D_2, \ldots, D_q, X_1, X_2, \ldots, X_t$ and $E_1, E_2, \ldots, E_r, X_1, X_2, \ldots, X_t$ will also have the same MAC, regardless of the choice of $X_1, X_2, \ldots, X_t$.

By elementary probability theory relating to the so called 'Birthday Paradox' (see, for example, [9]), given a set of $2^{n/2}$ messages there is a good chance that two of them will have the same MAC. Thus, to find such a collision requires only approximately $2^{n/2}$ known message/MAC pairs. Armed with such a pair, the attacker now needs only persuade the user to generate a MAC on one more message to obtain a MAC forgery. Thus the total complexity of this attack is $[0,2^{n/2},1,0]$.

Finally note that this attack applies to all of ISO/IEC MAC algorithms 1–4, as long as Padding Method 3 is not used. Unfortunately, Padding Method 3 does not prevent another, slightly more sophisticated, forgery attack, described immediately below.

4.3 Van Oorschot-Preneel Attack

This attack is based on an observation of Preneel and van Oorschot, also independently made by Kaliski and Robshaw, which is summarised as Lemma 1 in

[9]. The attack relies on finding an 'internal collision' for a pair of padded messages. That is, suppose $D_1, D_2, \ldots, D_q$ and $E_1, E_2, \ldots, E_r$ are two sequences of n-bit blocks obtained as a result of applying the padding and splitting processes to a pair of messages D and E. Suppose also that the chaining variables for the MAC computations for D and E are H_i, $(1 \leq i \leq q)$, and J_i, $(1 \leq i \leq r)$, respectively. Then an internal collision is where:

$$H_s = J_t$$

for some pair (s, t), where $s \leq q$ and $t \leq r$.

Given knowledge of an internal collision (by some means), the attacker immediately knows that the padded messages

$$D_1, D_2, \ldots, D_s, F_1, F_2, \ldots, F_u$$

and

$$E_1, E_2, \ldots, E_t, F_1, F_2, \ldots, F_u$$

will have the same MAC, regardless of $F_1, F_2, \ldots, F_u$. That is, given a known internal collision, a forgery requires only one chosen MAC. Note that, if Padding method 3 is used, then the above attack will work if and only if the values s, t and u satisfy $u = q - s = r - t$.

The problem remains of finding the internal collision. If Padding Method 3 is not in use then the attack works if we set $s = q$ and $t = r$ and look for MAC collisions amongst a large set of messages, i.e. the attack is the same as the Birthday Attack (see Section 4.2). However, when Padding Method 3 is in use, finding a 'useful' internal collision, i.e. one for which $s < q$, is a little more difficult albeit not impossible, as we now describe.

Suppose, that the attacker obtains the MACs for a set of $2^{n/2}$ messages, all of which agree in their final u n-bit blocks for some $u > 0$. As before, suppose also that $m = n$. Then, there is a good chance that two of these messages will have the same MAC. Since these two messages have their final u blocks the same, then we know that there will be an internal collision.

Specifically, suppose we know that the MACs for the sequences of blocks $D_1, D_2, \ldots, D_q$ and $E_1, E_2, \ldots, E_r$ are the same, and suppose we also have $D_i = E_{i+r-q}$ for $q - u + 1 \leq i \leq q$. If H_i and J_i denote chaining variables (as above), then we must have $H_s = J_t$ where $s = q - u$ and $t = r - u$.

If we regard the set of $2^{n/2}$ messages as chosen texts, then the attack has complexity [0,0,$2^{n/2}$,0], which, although large, is still more effective than the 'simple' MAC forgeries. However, it may be easier than this to obtain the desired MACs, bearing in mind that many messages are highly formatted. Thus it may be true 'by accident' that large numbers of messages for which a MAC is computed all end in the same way. If this is the case then the attack complexity might more reasonably be described as [0,$2^{n/2}$,1,0], i.e. the same as the Birthday forgery attack.

In any event it should be clear that Padding Method 3 does not protect against forgery attacks using internal or external collisions. This point is also made in Section III.B of [9]. Thus, to prevent such attacks, further countermeasures are needed. This is the subject of the remainder of this paper.

5 Countermeasures

Over the past few years a number of countermeasures to various forgery attacks have been proposed. Of course, there are certain forgery attacks which cannot be avoided, and serve as a baseline against which other attacks can be measured. We now review some of the proposed countermeasures.

– *Truncation.* Perhaps the most obvious countermeasure to the attacks described in Sections 4.2 and 4.3, is to choose the MAC length m such that $m < n$, i.e. to truncate the MAC. However, Knudsen, [5], has shown that, even when truncation is employed, the same attacks can still be made at the cost of a modest amount of additional effort. Moreover, if m is made smaller, then the MAC guessing and Verification forgery attacks (described in Section 3) become easier to mount.
– *Padding Methods 2 and 3.* Padding Method 2 was introduced specifically to deal with the Trailing zeros forgery (see Section 3). Padding Method 3 was introduced to counter certain key recovery attacks, and was also originally believed to counter Birthday forgeries (for further details see [9]). However, as we have seen, neither Padding Method is able, on its own, to prevent the attack described in Section 4.3.
– *Serial numbers.* A further countermeasure is briefly described in [1]. The idea is to prepend a unique serial number to data prior to computing a MAC. That is, every time a MAC is generated, the data to be MACed is prepended with a number which has never previously been used for this purpose (within the lifetime of the key).
  Although it is not stated explicitly in [1], it would seem that it is intended that the serial number should be prepended to the message prior to padding. Note also that it will be necessary to send the serial number with the message, so that the intended recipient can use it to help recompute the MAC (as is necessary to verify it).
  It is fairly simple to see why this approach foils the attacks of Sections 4.2 and 4.3. Both attacks require the forger to obtain the MAC for a chosen data string. However, because of the insertion of a serial number, the attacker is now no longer in a position to choose the data string. Thus it was believed that this countermeasure was effective against the non-trivial forgery attacks. However, as we will show below, serial numbers do not protect against 'short-cut' forgery attacks, even when combined with Padding Method 3. It is believed that this is the first time a forgery attack more efficient than the verification attack has been demonstrated against the serial number enhancement to CBC-MACs.

6 A New Forgery Attack

We now describe a new type of forgery attack. To simplify the presentation we start by describing the attack as applied to MAC algorithms 1, 2 or 3 with Padding Method 1 or 2 and no Serial Number prefix. Later we consider the scenario where Serial Numbers are used and lastly we consider the implications of this attack in the case where both Padding Method 3 and Serial Numbers are used.

6.1 The Basic Attack

We first consider the case where one of MAC algorithms 1, 2 or 3 is used together with Padding Method 1 or 2 and where the data is not prefixed with a Serial Number. As previously, we consider the case where there is no truncation and the size of the chaining variable is equal to the size of the final output, this common value being denoted by n. Assume that the attacker somehow obtains the corresponding MACs for approximately $2^{n/2}$ (padded) $(r+q+1)$-block messages $E'_1, E'_2, \ldots, E'_q, X, F_1, F_2, \ldots, F_r$ where $E'_1, E'_2, \ldots, E'_q$ are arbitrary n-bit blocks, $F_1, F_2, \ldots, F_r$ are arbitrary but fixed n-bit blocks, and X is an n-bit block that is different for each message. The attacker also obtains the corresponding MACs for approximately $2^{n/2}$ padded $(r+1)$-block messages of the form $Y, F_1, F_2, \ldots, F_r$, with the same fixed blocks F_i, $1 \leq i \leq r$, and a different n-bit block Y for each message.

Using an extension to the Birthday Paradox, [7,8], given the number of MACs obtained there is a high probability that a MAC from the set of $(r+q+1)$-block messages is equal to a MAC from the set of $(r+1)$-block messages. In other words, $MAC(E_1, E_2, \ldots, E_q, X_0, F_1, F_2, \ldots, F_r) = MAC(Y_0, F_1, F_2, \ldots, F_r)$ for some particular known values of $E_1, \ldots, E_q, X_0$ and Y_0. Since the n-bit blocks $F_1, \ldots, F_r$ are the same for the two messages, it is an immediate consequence that $MAC^*(E_1, E_2, \ldots, E_q, X_0) = MAC^*(Y_0)$, where $MAC^*(Z)$ denotes the computation of the MAC on the message Z without the Output Transformation. This final relation is equivalent to

$$MAC^*(E_1, E_2, \ldots, E_q) = X_0 \oplus Y_0.$$

As a result of this, if the attacker knows that the MAC for some (padded) message $Z, P_1, P_2, \ldots, P_t$ $(t \geq 1)$ is equal to M, then the attacker knows that the MAC for the message $E_1, E_2, \ldots, E_q, X_0 \oplus Y_0 \oplus Z, P_1, P_2, \ldots, P_t$ is also equal to M. This means that the complexity of this MAC forgery attack on a MAC algorithm with an n-bit output with no truncation is approximately $[0, 1, 2^{n/2+1}, 0]$. In the case of *DES* with no truncation, this is a forgery attack of complexity $[0, 1, 2^{33}, 0]$.

6.2 A Forgery Attack for the Serial Number Case

Now consider the case where Serial Numbers are used with MAC algorithm 1, 2 or 3 and Padding Method 1 or 2. Note that serial numbers are meant to be prefixed to the messages to be MACed, that is, if $P_1, P_2, \ldots, P_t$ is a padded t-block message, then the MAC is calculated on the $(t+1)$-block message $S, P_1, P_2, \ldots, P_t$ where S is the serial number associated with the message. With this observation we point out that the above attack described for MAC algorithms not using

Serial Numbers works *unchanged* for MAC algorithms using Serial Numbers. Only the interpretation of the first block of the various chosen texts used in the attack is different.

Note that for the $(r+q+1)$-block messages $E'_1, E'_2, \dots, E'_q, X, F_1, F_2, \dots, F_r$ described above, the E'_i's, $1 \leq i \leq q$ were arbitrary and not necessarily fixed. In the case of use of serial numbers, an attacker could submit $(r+q)$-block messages $E'_1, E'_2, \dots, E'_{q-1}, X, F_1, F_2, \dots, F_r$ to be MACed. The MAC algorithm returns the MAC for the string $S'_1, E'_1, E'_2, \dots, E'_{q-1}, X, F_1, F_2, \dots, F_r$ where S'_1 is the (unique) serial number selected by the MAC algorithm for the particular message. For verification of the MAC, the value of S'_1 has to be transmitted with the MAC of the message — if S'_1 is encrypted then this attack will not work. Hence the attacker is assumed to know the value of S'_1 for each of the $2^{n/2}$ $(r+q)$-block messages $E'_1, E'_2, \dots, E'_{q-1}, X, F_1, F_2, \dots, F_r$. Similarly, the attacker can submit the r-block message $F_1, F_2, \dots, F_r$ $2^{n/2}$ times, each time obtaining a (different) MAC for the string $S'_2, F_1, F_2, \dots, F_r$, for a known but different serial number S'_2.

As above, there is a non-trivial probability that an $(r+q)$-block message of the first type and one of the r-block submissions of the second type yield the same MAC, that is,

$$MAC(S_1, E_1, E_2, \dots, E_{q-1}, X_0, F_1, F_2, \dots, F_r) = MAC(S_2, F_1, F_2, \dots, F_r),$$

for some known particular values of $S_1, S_2, E_1, \dots, E_{q-1}$ and X_0. This means that $MAC^*(S_1, E_1, E_2, \dots, E_{q-1}, X_0) = MAC^*(S_2)$ and therefore

$$MAC^*(S_1, E_1, E_2, \dots, E_{q-1}) = X_0 \oplus S_2.$$

If the attacker knows that the MAC for a padded message $P_1, \dots, P_t$ $(t \geq 1)$ using serial number S_3 is equal to M, he also knows that the MAC for the padded message $E_1, E_2, \dots, E_{q-1}, X_0 \oplus S_2 \oplus S_3, P_1, P_2, \dots, P_t$ using serial number S_1 is also equal to M. The complexity of this MAC forgery attack is the same as before, i.e. $[0, 1, 2^{n/2+1}, 0]$. The constructed block $X_0 \oplus S_2 \oplus S_3$ is the reason why the attack does not work if the serial numbers are not in the clear, since in this case the attacker does not know S_2 and S_3.

6.3 Combining Serial Numbers with Padding Method 3

The attack can be generalised to cover the case where Padding Method 3 and Serial Numbers are used in combination; there are two ways to combine these two features, and we describe attacks for both combinations.

Firstly, suppose they are combined as implied in [1], i.e. the serial number is prefixed before the message is padded, i.e. the length of the unpadded message is prefixed to the padded and serial numbered message. The attacker submits the r-block message $F_1, F_2, \dots, F_r$ $2^{n/2}$ times, each time obtaining a (different) MAC for the string $L_2, S'_2, F_1, F_2, \dots, F_r$, for a varying serial number S'_2. Note that L_2 is the 'length-encoding block' for the message (it will be the same every time), as inserted by Padding Method 3.

The attacker also sends $2^{n/2}$ messages $E'_1, E'_2, \ldots, E'_{q-2}, L_2, X, F_1, F_2, \ldots, F_r$ to be MACed, where L_2 is as above and X is different for each message. MACs are computed for strings $L_1, S'_1, E'_1, E'_2, \ldots, E'_{q-2}, L_2, X, F_1, F_2, \ldots, F_r$ where S'_1 is the varying serial number, and L_1 is the length encoding block. As before we suppose that the attacker knows the values of S'_1 and S'_2 for each of the messages.

There is a good chance that an r-block message of the first type and an $r+q$-block message of the second type yield the same MAC, that is,

$$MAC(L_1, S_1, E_1, E_2, \ldots, E_{q-2}, L_2, X_0, F_1, F_2, \ldots, F_r) = MAC(L_2, S_2, F_1, F_2, \ldots, F_r),$$

for some particular values of $S_1, S_2, E_1, E_2, \ldots, E_{q-2}$ and X_0. This means that $MAC^*(L_1, S_1, E_1, E_2, \ldots, E_{q-2}, L_2, X_0) = MAC^*(L_2, S_2)$ and therefore

$$MAC^*(L_1, S_1, E_1, E_2, \ldots, E_{q-2}, L_2) = e_K(L_2) \oplus X_0 \oplus S_2.$$

So if an attacker knows the MAC for padded message $(L_2, S_3, P_1, P_2, \ldots, P_r)$ is equal to M, (where S_3 is any serial number), he knows that the MAC for the padded message $L_1, S_1, E_1, E_2, \ldots, E_{q-2}, L_2, X_0 \oplus S_2 \oplus S_3, P_1, P_2, \ldots, P_r$ is also equal to M. The complexity of this MAC forgery attack is the same as before, i.e. $[0, 1, 2^{n/2+1}, 0]$.

Secondly we consider the alternative way of combining serial numbers with Padding Method 3, i.e. where we first pad the message, then prefix the length of the unpadded message, and finally prefix the resulting string with the selected serial number. That is, for a (padded) message $P_1, P_2, \ldots, P_t$, the MAC algorithm is applied to the string $S, L, P_1, P_2, \ldots, P_t$, where S is the serial number block and L is the length block of the unpadded message. We describe yet another attack variant for this case.

Briefly, the attacker submits $2^{n/2}$ $(r+q)$-block padded messages of the form $E_1, E_2, \ldots, E_{q-2}, X, L_2, F_1, F_2, \ldots, F_r$ where $E_1, E_2, \ldots, E_{q-2}$ are arbitrary n-bit blocks, $F_1, F_2, \ldots, F_r$ are arbitrary but fixed n-bit blocks, L_2 is an n-bit block representing the length of the unpadded string $F_1, F_2, \ldots, F_r$ (as required by Padding Method 3), and X is an n-bit block that is different for each message. The attacker obtains the corresponding MACs and the particular serial number S'_1 used with each message. The attacker also submits the r-block padded string $F_1, F_2, \ldots, F_r$ $2^{n/2}$ times for MACing, obtaining the corresponding MACs and the different serial number S'_2 used for each MAC obtained. There is a non-trivial probability that a MAC for one of the $(r+q)$-block messages is equal to a MAC for the r-block message for some serial numbers S_1 and S_2, that is,

$$MAC(S_1, L_1, E_1, \ldots, E_{q-2}, X_0, L_2, F_1, \ldots, F_r) = MAC(S_2, L_2, F_1, \ldots, F_r).$$

This means that $MAC^*(S_1, L_1, E_1, E_2, \ldots, E_{q-2}) = X_0 \oplus S_2$.

Suppose also that the attacker knows that the MAC for an r-block message $P_1, P_2, \ldots, P_r$, with unpadded length equal to L_2 and serial number S_3, is equal to M. Then he knows that the MAC for the $(r+q)$-block message $E_1, E_2, \ldots, E_{q-2}, X_0 \oplus S_2 \oplus S_3, L_2, P_1, P_2, \ldots, P_r$ of unpadded length L_1 and with serial number S_1 is also equal to M, or

$$MAC(S_1, L_1, E_1, E_2, \ldots, E_{q-2}, X_0 \oplus S_2 \oplus S_3, L_2, P_1, P_2, \ldots, P_r) = MAC(S_3, L_2, P_1, P_2, \ldots, P_r).$$

Note that the complexity of the attack is as before, i.e. it is $[0, 1, 2^{n/2+1}, 0]$.

6.4 Implications

The published version of ISO/IEC 9797-1, [1], indicates that forgery attacks can be avoided by using a combination of Padding Method 3 and Serial Numbers. However, the attacks described in Section 6.3 cast serious doubt on the value of serial numbers as a remedy to forgery attacks even when combined with Padding Method 3.

7 Summary and Conclusions

In this paper we have surveyed some forgery attacks to which MAC algorithms may be subjected, including new attacks which can defeat some proposed countermeasures not successfully attacked before. In particular we have shown that combining Padding Method 3 and Serial Numbers is not as effective as was previously believed in defeating 'shortcut' forgery attacks.

Of course, in practice, other security features may prevent some or all of the described attacks from being a real threat. For example, in certain banking environments the security deployed in the access to a MAC algorithm is such that it is extremely difficult for an unauthorised user to obtain the MAC corresponding to even one chosen text, let alone several. Also, if the MAC scheme is used in such a way that no key is used to compute more than a small number of MACs then certain attacks become impossible.

The use of Padding Method 1 is not automatically excluded because of the attack described in section 3. It is possible that in certain environments messages are highly formatted to the extent that the length of a message to be MACed is fixed or known from the context, and therefore a trailing zeroes forgery is not applicable.

In general it is important for users to carefully assess the significance of the various MAC attacks in the context of the environment in which the resulting MAC algorithm is to be used. There may be no benefit from using certain sophisticated MAC systems in an environment which has other security features in operation which make attacks against simpler MAC schemes impossible to carry out. On the other hand, care should be taken not to assume that a MAC which is secure in a certain environment is automatically secure in others. For example, a 32-bit MAC which may be safely used in a banking environment without any serious threat from a verification forgery, is possibly not safe if used on the Internet or any other environment where large numbers of verifications may be obtained in a short time.

The introduction and use of the AES algorithm [2], with a minimum 128-bit cipher block length, as the encryption function to be used in block-cipher based MACs means that all the attacks described here would become practically infeasible. However, this may not be the case if heavy truncation is used since, in this case, some of the attacks described in section 3 may still be possible.

In this paper we concentrated on block-cipher based MAC algorithms as described in [1]. It is possible that generalisations of the attacks described here may be also applicable to other (dedicated, hash function-based or proprietary) MAC algorithms which are based on iterated functions. Note that all practical MAC algorithms are iterated in construction.

Acknowledgements

The authors would like to thank anonymous referees who suggested the first attack variant described in Section 6.3.

References

1. ISO/IEC 9797-1. *Information technology — Security techniques — Message Authentication Codes (MACs) — Part 1: Mechanisms using a block cipher.* International Organization for Standardization, Genève, Switzerland, 1999.
2. AES, a crypto algorithm for the twenty-first century, Advanced Encryption Standard (AES) development effort, 2000. `http://csrc.nist.gov/encryption/aes`.
3. American Bankers Association, Washington, DC. *ANSI X9.19, Financial institution retail message authentication*, August 1986.
4. American Bankers Association, Washington, DC. *ANSI X9.9–1986 (revised), Financial institution message authentication (wholesale)*, April 1986.
5. L.R. Knudsen. Chosen-text attack on CBC-MAC. *Electronics Letters*, **33**:48–49, 1997.
6. A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, Boca Raton, 1997.
7. K. Nishimura and M. Sibuya. Occupancy with two types of balls. *Ann. Inst. Statist. Math.*, **40**:77–91, 1988.
8. K. Nishimura and M. Sibuya. Probability to meet in the middle. *J. Cryptology*, **2**:13–22, 1990.
9. B. Preneel and P.C. van Oorschot. On the security of iterated Message Authentication Codes. *IEEE Transactions on Information Theory*, **45**:188–199, 1999.

Cryptanalysis of a Public Key Cryptosystem Proposed at ACISP 2000

Amr Youssef[1] and Guang Gong[2]

[1] Center for Applied Cryptographic Research
Department of Combinatorics & Optimization
[2] Department of Electrical and Computer Engineering
University of Waterloo
Waterloo, Ontario N2L 3G1, CANADA
{a2youssef,ggong}@cacr.math.uwaterloo.ca

Abstract. At ACISP 2000, Yoo *et al* proposed a fast public key cryptosystem using matrices over a ring. The authors claim that the security of their system is based on the RSA problem. In this paper we present a heuristic attack that enables us to recover the private key from the public key. In particular, we show that breaking the system can be reduced to finding a short vector in a lattice which can be achieved using the L^3-lattice reduction algorithm.

Key words: public key cryptography, cryptanalysis, L^3-algorithm

1 Introduction

Most practical public key schemes are very slow compared to symmetric key schemes. This motivates extensive research for faster public key schemes. Several lattice-based systems such as [1], [2] are among these schemes. Both of these schemes, which are based on the closest vector problem and the shortest vector problem [6] [7] are broken using the L^3 lattice reduction algorithm. In fact, the L^3 algorithm was successfully used to attack many similar public key systems [5]. Yoo *et al* [11] proposed a fast public key cryptosystem similar to the system proposed in [2]. However, they claim that since the security of their scheme is based on the RSA problem and not the lattice problems, their scheme is secure against these lattice basis reduction attacks. In this paper we show that breaking this system is equivalent to the problem of finding a short vector in a lattice which can be solved using the L^3-lattice reduction algorithm [4]. In particular, our heuristic attack enables us to recover the private key from the public key and hence represent a total break for the proposed system.
The paper is organized as follows. In section 2 we give a description for the system proposed in [11]. In section 3, we describe our attack. Finally we give a numerical example using the same parameters of the encryption-decryption example in [11].

2 Description of the Proposed Scheme

In this section we review the proposed public key scheme. Further details and justification for the bounds on the parameters can be found in [11].
Let n be the dimension of a lattice. The basic steps to choose the parameters are as follows:
1 Choose positive integers $\hat{m}, \hat{e}, d_{ii}$, $1 \leq i \leq n$, primes p, q and a matrix $D \in Mat_n(\mathbb{Z})$ with the following conditions:
1.1 $N = pq$.
1.2 $\hat{m}, \hat{e}$: random integers such that $\hat{m} \approx q^{0.4}, \hat{e} \approx q^{0.3}$, where $\hat{m}$ and $\hat{e}$ are upper bounds of messages and error vectors respectively.
1.3 D : diagonal matrix such that $\hat{m} < |d_{ii}| < q^{0.5}$, where $d_{ii}, 1 \leq i \leq n$ are diagonal entries of D.
2 Choose an invertible matrix $T = (t_{ij})_{1 \leq i,j \leq n} \in Mat_n(\mathbb{Z})$ such that $\sum_{j=1}^{n} t_{ij} < q^{0.2}$.
3 Form the matrices $R = DT$ and $B = B_q UL \bmod N$ where $B_q = R^{-1} \bmod q$, L (respectively U) are uni-modular lower (respectively upper) triangular matrix whose all entries except the diagonal entries are multiples of q.

$B, \hat{e}, \hat{m}$ and N are public information. R, q and T are kept secret.

Encryption: Let $M = (m_1, \cdots, m_n)^t, 0 \leq m_i < \hat{m}$ be a message vector and $E = (e_1, \cdots e_n)^t, 0 \leq e_i < \hat{e}$ be an arbitrary error vector. Then the ciphertext is

$$C = (BM + E) \bmod N.$$

Decryption: At first compute $X = (x_1, \cdots, x_n)^t$:

$$C_q = C \bmod q,$$

$$X = RC_q \bmod q.$$

Then $m_i = x_i (\text{mod } d_{ii})_{1 \leq i \leq n}$.

3 Attacking the Scheme

In this section we will present a heuristic attack that enables us to recover the private key from the public key. In particular, this attack enables us to factor N using the matrix B only. As mentioned in [11], once q is revealed, one can find B_q and D and the system is totally broken. Recall that

$$B^{-1} \text{mod } q = R.$$

The following lemma follows by noting that for $N = pq$ and for any integer a we have

$$a \bmod q = (a \bmod N) \bmod q.$$

Let $V = B^{-1} \text{mod } N$. Then we have

Lemma 1.

$$V \ mod \ q = (B^{-1} \ mod \ N) \ mod \ q = (B^{-1} \ mod \ q) = R.$$

Let r_{max} denote $\max_{\{i,j\}} |r_{ij}|$. Then from Section 3 in [11] we have

$$r_{max} < \frac{q - \hat{m}}{n\hat{e}} \approx \frac{q - q^{0.4}}{nq^{0.3}} < q^{0.7}.$$

Thus every element of the matrix V can be represented as

$$v_{ij} = a_{ij}q + r_{ij},$$

where $0 \leq a_{ij} < p, r_{ij} < r_{max}, 1 \leq i, j \leq n$.

The basic steps in the attack are as follows:

1. Calculate the matrix $V = B^{-1}$ mod N.

2. Pick an $m, m \leq n^2$, elements from the set $\{v_{ij}\}_{\{1 \leq i,j \leq n\}}$. Let $S = \{s_i\}_{\{1 \leq i \leq m\}}$ denote the set formed from the elements above.

3. Use the L^3 algorithm to find a reduced basis B for the $(m+1)$-dimensional lattice L which is generated by the rows of the matrix

$$\begin{bmatrix} N & 0 & 0 & \cdots & 0 & 0 \\ 0 & N & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & N & 0 \\ -s_1 & -s_2 & -s_3 & \cdots & -s_m & 1 \end{bmatrix}.$$

4. For each row $l = (l_1, l_2, \cdots, l_m, l_{m+1})$ in B such that $l_{m+1} \neq N$ do the following:

- Evaluate $gcd(N, l_{m+1})$.
- If $gcd(N, l_{m+1}) \neq 1$, return $p = gcd(N, l_{m+1})$.

5. Return (Failure).

The following lemma is used to justify the success of the attack.

Lemma 2. *The vector*

$$x = ((a_1N - ps_1), (a_2N - ps_2), \cdots, (a_mN - ps_m), p) = (-p\delta_1, -p\delta_2, \cdots, -p\delta_m, p)$$

is in L and has length less than approximately $(\sqrt{m+1}\ pq^{0.7})$.

Proof. The first part follows by noting that x is a linear combination of the rows of L. The second part follows by noting that each of the elements s_i can be represented as $s_i = a_iq + \delta_i$ where $\delta_i < q^{0.7}$.

Note that our lattice has dimension $(m+1)$ and volume N^m. From the lemma above, x is short compared to the $(m+1)^{th}$ root of the volume of the lattice. Hence, there is a good possibility that the L^3 algorithm will produce a reduced basis which include the vector x. If no solution exists then we can try another subset of elements $\{v_{ij}\}$. Our experimental results show that the L^3 algorithm finds p with high probability.

Let $\{b_1, b_2, \cdots, b_{m+1}\}$ denote the basis of the lattice L above. Let $C \in \mathbb{R}$ be such that $|b_i|^2 \leq C$ for $i = 1, 2, \cdots m+1$ and $|b_i|$ denote the length of the basis b_i. From [4], the number of arithmetic operations needed by the L^3 algorithm is $O((m+1)^4 logC)$, on integers of size $O((m+1)logC)$.

Remark 1. The lattice used in step 3 is the standard lattice used in the Simultaneous Diophantine Approximation (SDA) [4]. I.e., our problem can also be formulated in terms of SDA. It was noted by Nguyen and Shparlinski [9] that this formulation leads to unconditional provable attack provided that p and q are much unbalanced ($q > p^{10/3}$) because we would have an unusually good SDA (See Fact 3.107 in [4]). In fact, in this case, we can easily solve the problem using the continuous fraction approximation [3]. It was also noted in [9] that while the attack in [8] can be applied to this cryptosystem, it is not an improvement of our attack and our attack is much simpler in this case.

4 Numerical Example

In order to illustrate the steps in our cryptanalysis, we will use the same numerical example given in [11]. Let $q = 10570841$ and $p = 10570837$. Then $N = 111742637163917$. Let

$$D = \begin{bmatrix} 612 & 0 & 0 & 0 \\ 0 & 681 & 0 & 0 \\ 0 & 0 & 697 & 0 \\ 0 & 0 & 0 & 601 \end{bmatrix},$$

and

$$T = \begin{bmatrix} 5 & 2 & 3 & 7 \\ 4 & 3 & 1 & 2 \\ 4 & 7 & 1 & 3 \\ 2 & 3 & 4 & 9 \end{bmatrix}.$$

Then

$$R = \begin{bmatrix} 3060 & 1224 & 1836 & 4284 \\ 2724 & 2043 & 681 & 1362 \\ 2788 & 4879 & 697 & 2091 \\ 1202 & 1803 & 2404 & 5409 \end{bmatrix}.$$

Choose

$$U = \begin{bmatrix} 1 & -10570841 & 10570841 & -10570841 \\ 0 & 1 & 10570841 & -10570841 \\ 0 & 0 & 1 & 10570841 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 10570841 & 1 & 0 & 0 \\ 10570841 & -10570841 & 1 & 0 \\ -10570841 & 10570841 & 10570841 & 1 \end{bmatrix}.$$

Then we have

$$B = \begin{bmatrix} 85902782524529 & 7783949494261 & 108645955098741 & 62082137341722 \\ 37207086894442 & 97811933363455 & 31492859166426 & 47829503460547 \\ 43940929239657 & 99629428908384 & 64015171957907 & 95852228892018 \\ 100737337377789 & 6871742549039 & 58298211039553 & 15913440226477 \end{bmatrix}.$$

Since B and N are public information, we can calculate

$$V = B^{-1} \text{mod } N =$$

$$\begin{bmatrix} 72960716256453 & 4761772750607 & 47819503708674 & 64505037116731 \\ 18354764339802 & 34264334590284 & 25746128923461 & 46666277809305 \\ 28770435964827 & 105706232411633 & 39730135919762 & 9119580812042 \\ 89276407646137 & 79398453561765 & 94718657144415 & 99534468035995 \end{bmatrix}$$

Then we arbitrarily select the set

$$S = \{v_{11}, v_{12}, v_{13}, v_{14}\} =$$

$$\{72960716256453, 4761772750607, 47819503708674, 64505037116731\}.$$

Using the L^3 algorithm (See algorithm 3.101 in [4], [10]), the basis to be reduced is:

$$\begin{bmatrix} 111742637163917 & 0 & 0 & 0 & 0 \\ 0 & 111742637163917 & 0 & 0 & 0 \\ 0 & 0 & 111742637163917 & 0 & 0 \\ 0 & 0 & 0 & 111742637163917 & 0 \\ -72960716256453 & -4761772750607 & -47819503708674 & -64505037116731 & 1 \end{bmatrix}.$$

The L^3-reduced basis is:

$$\begin{bmatrix} -32346761220 & -12938704488 & -19408056732 & -45285465708 & \mathbf{10570837} \\ -87078711029 & 39709857984 & 7883945327 & 11690435622 & 4385339758 \\ -12420733475 & -4968293390 & -7452440085 & -17389026865 & 182590067501 \\ -102740951106 & -253999460687 & 146924464771 & 79909317394 & 26579009212 \\ 1917450399 & -58848334744 & -420915726704 & 231779925047 & 78377734153 \end{bmatrix}.$$

Hence we get $\mathbf{p} = \mathbf{10570837}$. Once p is revealed we calculate $q = N/p$. Then we get $R = V^{-1}$ mod q. After this we calculate $d_{ii} = gcd(r_{i1}, r_{i2}, \cdots r_{in}), 1 \leq i \leq n$.

It is worth noting that it only took us 91, 520 and 4802 seconds to break the algorithm for the size of N = 256, 512 and 1024 bits respectively. We set $m = 10$ through step 2 of the attack. We performed our experiments with Maple V Release 5.1 running on a SUN ULTRA-80 workstation.

References

1. *M. Ajtai and C. Dwork, A public key cryptosystem with worst-case/average case equivalence*, Proc. of the twenty ninth annual ACM symposium on theory of computing, pp. 284-293, 1997.
2. O. Goldreich, S. Goldwasser and S. Halevi, *Public key cryptosystems from lattice reduction problems*, Advances in Cryptology, Pro. of CRYPTO' 97, Springer-Verlag, LNCS 1294, pp. 112-131.
3. G. H. Hardy and E. M. Wright, *An introduction to the theory of numbers,* 5^{th} *edition*, Oxford University Press, 1979.
4. A J. Menezes, P. C. van Oorschot and S A. Vanstone, *Handbook of Applied Cryptographic Research*, CRC Press, 1996.
5. P. Nguyen and J. Stern, *Lattice reduction in cryptology : An update*, Algorithmic Number Theory, Proc. of ANTS-IV, Springer-Verlag, LNCS 1838.
6. P. Nguyen, *Cryptanalysis of the Goldreich-Goldwasser-Halevi Cryptosystem from Crypto '97*, Advances in Cryptology, Pro. of CRYPTO'99, Springer-Verlag, LNCS 1666, pp. 288-304.
7. P. Nguyen and J. Stern, *Cryptanalysis of the Ajtai-Dwork Cryptosystem*, Advances in Cryptology, Pro. of CRYPTO' 98, Springer-Verlag, LNCS 1462, pp. 223-242.
8. P. Nguyen, J. Stern: Cryptanalysis of a Fast Public Key Cryptosystem Presented at SAC '97. Selected Areas in Cryptography 1998, Springer-Verlag, LNCS 1556, pp. 213-218.
9. P. Nguyen and I. Shparlinkski, Private communications, Jan 24, 2001.
10. Pate Williams, *Algorithms from Handbook of Applied Cryptography* , C code available at `http://www.mindspring.com/~pate/crypto/chap03.html.`
11. H. Yoo, S. Hong, S. Lee, O. Yi and M. Sung, *A Proposal of a New Public Key Cryptosystem using Matrices Over a Ring*, Fifth Australian Conference on Information Security and Privacy (ACISP 2000), Springer-Verlag, LNCS 1841, 2000, pp. 41-48.

Improved Cryptanalysis of the Self-Shrinking Generator

Erik Zenner[⋆], Matthias Krause, and Stefan Lucks[⋆⋆]

Theoretische Informatik
University of Mannheim (Germany)
{zenner,krause,lucks}@th.informatik.uni-mannheim.de

Abstract. We propose a new attack on the self-shrinking generator [8]. The attack is based on a backtracking algorithm and will reconstruct the key from a short sequence of known keystream bits. We give both mathematical and empirical evidence for the effectiveness of this attack. The algorithm takes at most $O(2^{0.694L})$ steps, where L is the key length. Thus, our attack is more efficient than previously known key reconstruction algorithms against the self-shrinking generator that operate on short keystream sequences.

1 Introduction

The self-shrinking generator [8] is a keystream generator for the use as a stream cipher. It is based on the shrinking principle [2] and has remarkably low hardware requirements. So far, it has shown considerable resistance against cryptanalysis.

In cryptanalysis of a keystream generator, the attacker is assumed to know a segment of the keystream. The system is considered broken if the attacker can predict the subsequent bits of the keystream with success probability higher than pure guessing. One way to achieve this goal is to reconstruct the initial state of the generator, which allows prediction of the remaining keystream sequence with probability 1.

In this paper, we propose a new attack against the self-shrinking generator. It reconstructs the initial state of the generator from a short keystream sequence, requiring $O(2^{0.694L})$ computational steps. The fastest attack previously known that operates on a short keystream sequence [8] requires $O(2^{0.75L})$ steps. The only attack that has the potential to achieve a better running time [9] needs a much longer keystream sequence.

The paper is organised as follows: In section 2, we give an introduction to both the shrinking and the self-shrinking generator, the former providing the working principle for the latter. Section 3 surveys some of the previous work on cryptanalysis of the self-shrinking generator.

Sections 4-6 describe our attack and its properties. After giving a description of the algorithm in section 4, we prove the running time to be upper bounded

[⋆] Supported by the LGF Baden-Württemberg
[⋆⋆] Supported by DFG grant Kr 1521/3-1

by $O(2^{0.694L})$ steps in section 5. Section 6 provides some supplementary experimental results.

We conclude in section 7 by giving some design recommendations that help in strengthening a self-shrinking generator against our attack.

2 Description of the Cipher

2.1 The Shrinking Generator

In [2, 6], Coppersmith, Krawczyk and Mansour introduced a new pseudorandom keystream generator called the **shrinking generator**. It consists of two linear feedback shift registers (LFSR) A and S, [1] generating the m-sequences $(a_i)_{i\geq 0}$ (denoted as A-sequence) and $(s_i)_{i\geq 0}$ (denoted as S-sequence), respectively. The keystream sequence $(z_j)_{j\geq 0}$ is constructed from these two sequences according to the following selection rule: For every clock i, consider the selection bit s_i. If $s_i = 1$, output a_i. Otherwise, discard both s_i and a_i.

This way, a nonlinear keystream is generated. Even a cryptanalyst who knows part of the keystream sequence can not tell easily which z_j corresponds to which a_i, since the length of the gaps (i.e., the number of a_i that have been discarded) is unknown.

In [2], the shrinking generator is shown to have good algebraic and statistical properties. For a generalisation of some of these results, refer to [10]. Also in [2], a number of algebraic attacks that reconstruct the initial state of A and S are given. Note that all of them require exponential running time in the length $|S|$ of LFSR S.

A probabilistic correlation attack against the shrinking generator is discussed in [4, 11]. The authors give both mathematical and empirical treatment of the necessary computation. The resulting attack reconstructs the initial state of A, requiring an exponential running time in the length $|A|$ of LFSR A. Note that in order to reconstruct the initial state of S, another search is required.

As a consequence, a shrinking generator with $|A| \approx |S|$ still remains to be broken by an algorithm that is significantly more effective than the one presented in [2] (for a description, see section 4.1).

2.2 The Self-Shrinking Generator

The **self-shrinking generator** is a modified version of the shrinking generator and was first presented by Meier and Staffelbach in [8].

The self-shrinking generator requires only one LFSR A, whose length will be denoted by L. The LFSR generates an m-sequence $(a_i)_{i\geq 0}$ in the usual way. The selection rule is the same as for the shrinking generator, using the even bits $a_0, a_2, \ldots$ as S-Bits and the odd bits $a_1, a_3, \ldots$ as A-Bits in the above sense.

[1] The shrinking principle can be applied to any two binary symmetric sources; it is not restricted to LFSR. All of the algebraic results on the shrinking generator, however, are based on the assumption that LFSR are used as building blocks.

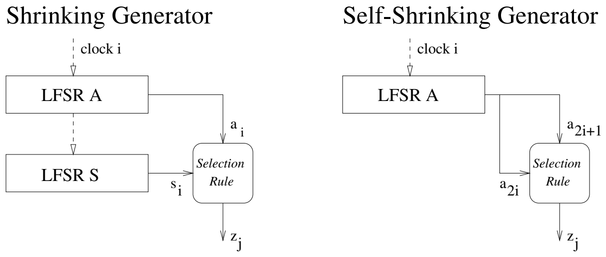


Fig. 1. The Shrinking Generators

Thus, the self-shrinking rule requires a tuple (a_{2i}, a_{2i+1}) as input and outputs a_{2i+1} iff $a_{2i} = 1$.

The close relationship between shrinking and self-shrinking generator is shown in figure 1. In [8], an algorithm is given that transforms an L-bit self-shrinking generator into a $2L$-bit shrinking generator. It is also shown that a shrinking generator with register lengths $|A|$ and $|S|$ has an equivalent self-shrinking generator of length $L = 2 \cdot (|A| + |S|)$. Notwithstanding this similarity, the self-shrinking generator has shown even more resistance to cryptanalysis than the shrinking generator. The next section gives a short description of the most efficient key reconstruction attacks that have been proposed in recent years.

3 Previous Work on Cryptanalysis

First, note that the attacks that have been proposed against the shrinking generator can not be transferred to its self-shrinking counterpart. The shrinking generator is best broken by attacking either LFSR A or S, thus effectively halving the key length. The self-shrinking generator, however, has molded both registers into an inseparable unit, namely a single LFSR. For this reason, "separation attacks" can not be employed without major modifications.

3.1 Period and Linear Complexity

The period Π of a keystream sequence generated by a self-shrinking generator was proven to be $2^{\lfloor L/2 \rfloor} \leq \Pi \leq 2^{L-1}$ in [8]. Experimental data seems to indicate that the period always takes the maximum possible value for $L > 3$.

It was also shown that the linear complexity C is always greater than $\Pi/2$. On the other hand, C was proven in [1] to be at most $2^{L-1} - (L-2)$. If $\Pi = 2^{L-1}$, we have $C \in \Theta(2^{L-1})$.

As a consequence, a LFSR with length equal to C can be constructed from about 2^L keystream bits in $O(2^{2L-2})$ computational steps, using the Berlekamp-Massey algorithm [7]. For realistic generator sizes of $L > 100$, this attack is thus computationally unfeasible.

3.2 Attacks Using Short Keystream Sequences

Even if the feedback logic of the LFSR is not known, there is a simple way of reducing the key space [8]. Consider the first two bits (a_0, a_1) of the LFSR (unknown) and the first bit z_0 of the keystream (known). Then there are only three out of four possible combinations (a_0, a_1) that are consistent with the keystream, since $(a_0, a_1) = (1, \bar{z}_0)$ is an immediate contradiction. The same rule can be applied for the next bit pair (a_2, a_3), and so on. Consequently, only

$$3^{L/2} = 2^{(log_2(3)/2)\cdot L} = 2^{0.79L}$$

possible initial values for the LFSR A consistent with the keystream.

The running time that is needed to search through the reduced key space can be further reduced on average by considering the likelihood of the keys. Note that the following holds:

$$\begin{aligned} \Pr[(a_0, a_1) = (0,0)|z_0] &= 1/4 \\ \Pr[(a_0, a_1) = (0,1)|z_0] &= 1/4 \\ \Pr[(a_0, a_1) = (1,z_0)|z_0] &= 1/2. \end{aligned}$$

Thus, the entropy of the bit pair is

$$H = -(1/4)\log(1/4) - (1/4)\log(1/4) - (1/2)\log(1/2) = 3/2.$$

The total entropy of an initial state consisting of L/2 such pairs is thus $2^{0.75L}$. At the same time, this is the effort for searching the key space if the cryptanalyst starts with the most probable keys. Surprisingly, this is still the most efficient reconstruction algorithm using short keystream sequences that has been published.

3.3 Attack Using Long Keystream Sequences

In [9], Mihaljević presented a faster attack that needs, however, a longer part of keystream sequence. Let the length of this known part be denoted by N. Then the attacker assumes that an l-bit section of the keystream has been generated by the current inner state of the LFSR. Consequently, l out of the $L/2$ even bits of A must be equal to 1. The attacker guesses these bits and checks whether or not this guess can be correct, iterating over all l-bit sections of the keystream. It is shown that cryptanalysis is successful with high probability after 2^{L-l} steps.

Since this procedure only makes sense for $L/4 \leq l \leq L/2$, the running time can vary from $2^{0.5L}$ in the very best case to $2^{0.75L}$ under more unfavourable circumstances. The efficiency of the attack depends mainly on the number of keystream bits that are available, since the value l must be chosen such that the following inequality holds:

$$N > l \cdot 2^{L/2} \cdot \binom{L/2}{l}^{-1}$$

value l:	$0.25L$	0.306	$0.50L$
Time:	$2^{0.75L}$	$2^{0.694L}$	$2^{0.5L}$
Bits :			
$L = 120$	$2^{8.19}$	$2^{10.17}$	$2^{65.91}$
$L = 160$	$2^{8.81}$	$2^{11.37}$	$2^{86.32}$
$L = 200$	$2^{9.30}$	$2^{13.07}$	$2^{106.64}$
$L = 240$	$2^{9.69}$	$2^{14.03}$	$2^{126.91}$
$L = 280$	$2^{10.02}$	$2^{14.94}$	$2^{147.13}$
$L = 320$	$2^{10.31}$	$2^{15.81}$	$2^{167.32}$

Table 1. Number N of keystream bits required for Mihaljević attack

In order to get a feeling for the number of bits required for this attack, table 1 gives some examples of required bitstream lengths for different register sizes L. The number of bits is given in logarithmic form in order to enhance readability. We concentrate on three cases:

– In order to beat the best key reconstruction algorithm described above, we need $l = 0.25L$, yielding a running time of $2^{0.75L}$ steps.
– Improving the running time to $2^{0.694L}$ (which is the performance of the algorithm to be presented in section 4) requires $l = 0.306L$.
– In order to achieve the best possible running time of $2^{0.5L}$ steps, we need $l = 0.5L$. Note that for realistic register lengths, the sheer amount of required data (namely, $N > \frac{L}{2} \cdot 2^{L/2}$) should make such an attack a mere theoretical possibility.

4 The Backtracking Algorithm

The goal of our cryptanalysis is the reconstruction of an inner state of the generator that is consistent with the keystream. We assume thus that a short keystream sequence of length $\approx L$ bits is known to the attacker.

We also assume that the feedback polynomial of the generator is known. Note that none of the attacks given in section 3.2 makes use of the feedback logic. It can be expected that the use of additional information should lead to a more efficient attack.

4.1 Basic Idea: Attacking the Shrinking Generator

First, consider cryptanalysis of the shrinking generator. If the feedback polynomials are known, an obvious way of reconstructing the inner states is as follows.

1. Guess the inner state of the control register S. From this, we can determine as many bits of the S-sequence as required.
2. Knowing the S-Sequence and part of the keystream sequence, we can reconstruct single bits of the A-Sequence.

3. Each known bit of the A-sequence gives a linear equation. If we can find $|A|$ linear independent equations, we can solve the system and thus reconstruct the inner state of register A.
4. We run the shrinking generator, using the reconstructed inner states for A and S. If the keystream sequence thus generated matches the known keystream sequence in the first $|A| + |S| + \epsilon$ positions (where ϵ is a security margin), we have found with high probability the correct inner state.

The running time of this attack (that was also presented in [2]) is obviously upper bounded by $O(|A|^3 \cdot 2^{|S|})$, since there are at most $2^{|S|} - 1$ inner states of register S and the solving of a system of $|A|$ linear equations takes at most $|A|^3$ steps.

4.2 Applying the Idea to the Self-Shrinking Generator

The principle of guessing only the S-Bits and deriving the A-Bits by solving a system of linear equations can be applied to the self-shrinking generator as well. It is, however, not as straightforward as with the shrinking generator, since guessing all S-Bits in the initial state (i.e., all even bits) will not enable the cryptanalyst to compute the rest of the S-sequence (unless the generator has a non-primitive characteristic polynomial). Thus, we will guess the even bits one at a time, using a backtracking approach similar to the procedure proposed by Golić in [3] for cryptanalysis of the A5/1 stream cipher.

Before we describe the details of the attack, we give the following property of the key (i.e. the initial state of the LFSR)[2]:

Proposition 1. *For each key $K = (a_0, \ldots, a_{L-1})$ with $a_0 = 0$, there exists an equivalent key $K' = (a'_0, \ldots, a'_{L-1})$ with $a'_0 = 1$.*

Proof. Consider the sequence $(a_i)_{i \geq 0}$ generated by the inner state K. Suppose the first '1' on an even position appears in position $2k$. Then clock the register by $2k$ steps, deriving the new inner state $K' = (a_{2k}, \ldots, a_{2k+L-1})$. Obviously, both inner states yield the same keystream sequence, since in transforming K to K', no output is generated. □

It is thus safe to assume that $a_0 = 1$ and $a_1 = z_0$. This way, we will reconstruct a key that is not necessarily equal to the original key, but it is equivalent in a sense that it will create the same keystream sequence.

From now on, we will have to guess the even bits of the sequence $(a_i)_{i \geq 0}$. This way, we obtain two different types of equations as follows:

– Every guess can be represented by a linear equation $a_{2i} = b_i$. These equations will be referred to as being of *type 1.*
– If $a_{2i} = 1$, we obtain a second equation of the type $a_{2i+1} = z_j$, where $j = \sum_{c=0}^{i} a_{2c}$. These equations will be denoted as being of *type 2.*

This approach will be implemented using a tree of guesses as shown in figure 2.

[2] The same property also holds for the shrinking generator. In this context, it was discussed in [11].

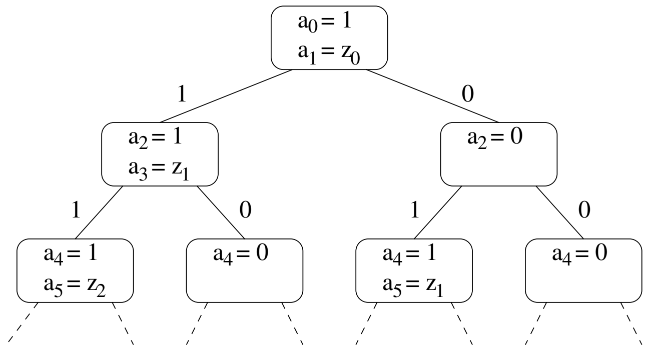


Fig. 2. The Tree of Guesses

As long as $i \leq \lfloor L/2 \rfloor - 1$, the development of the tree is straightforward. We get exactly two new equations whenever we follow a '1' branch and exactly one new equation when following a '0' branch. All of these equations are linearly independent, since no variable a_k appears more than once. Thus, we get a complete binary tree with height $\lfloor L/2 \rfloor - 1$.

After that point, however, the tree becomes irregular, since the indices of the new equations become larger than $L - 1$. Thus, the feedback recurrence must be used to convert the simple equations into a representation using only $a_0, \ldots, a_{L-1}$. Depending on the equations that are already known, there is an increasing probability that the new equations are linearly dependent of the earlier ones. That means they are either useless (in case they are consistent with the existing equation system) or lead to a contradiction. In the latter case, we have chosen a path in the tree that is not consistent with the known keystream sequence. We can thus ignore the current branch and start backtracking.

If we find a branch that ultimately gives us L linearly independent equations, we can solve the equation system and derive a key candidate. This candidate is evaluated by running the self-shrinking generator with this initial value, generating a candidate keystream of length $L+\epsilon$ (where ϵ is a small number of additional bits). We compare the candidate keystream with the known keystream segment. If they match, the key candidate is equivalent to the original key with high probability.

5 Upper Bounding the Running Time

In this section, we establish an asymptotical upper bound on the running time of our algorithm. For this purpose, we first give an upper bound C_L for the number of *leaves* in the tree of guesses (sections 5.1-5.3). Then, in section 5.4 we derive an upper bound for the number N_L of *nodes* in the tree and conclude that the total running time of the algorithm can be upper bounded by $O(L^4 \cdot 2^{0.694L})$.

5.1 Well-Formed vs. Malformed Trees

Let T_ℓ denote a tree of guesses such that ℓ linearly independent equations are still missing in the root to allow the solving of the equation system. Note that for the search tree given in section 4, we have $\ell = L - 2$.

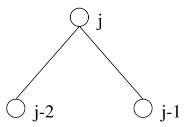
In order to formally prove the maximum number C_ℓ of leaves in T_ℓ, we label the nodes as follows: Each node is labelled by the number of linearly independent equations still needed in order to solve the equation system. The root is thus labelled by ℓ. For technical reasons, we allow a leaf of the tree to take both the labels 0 and -1, both meaning that the system is completely specified.

Assumption 1 *For the following average case analysis, we assume that an equation that is linearly dependent of its predecessors will lead to a contradiction with probability 1/2.*

This assumption is reasonable, since the bits a_{2i} and a_{2i+1} are generated by an m-LFSR, meaning that a variable takes values 0 and 1 with (almost) equal probability.

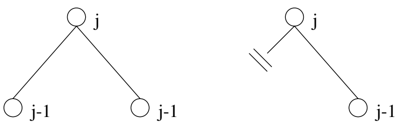
Now consider an arbitrary node V of depth $i - 1$, $i \geq 1$, and its two children, V_0 and V_1 (reached by guessing $a_{2i} = 0$ or $a_{2i} = 1$, resp.) Let V be labelled by j. The labelling of the child nodes depends on whether a_{2i} and/or a_{2i+1} are linearly dependent of the previous equations or not:

A) *Both are independent.* In this case, no contradiction occurs. The left child is labelled $j - 2$ and the right child is labelled $j - 1$.



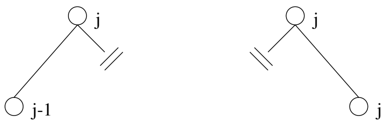
Prob = 1

B) a_{2i} *is independent,* a_{2i+1} *is not.* Both children are labelled $j - 1$. However, a contradiction occurs in V_1 with probability of 1/2.



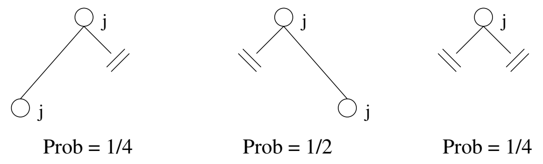
Prob = 1/2 Prob = 1/2

C) a_{2i} *is dependent,* a_{2i+1} *is not.* The left child is labelled $j - 1$, while the right child is labelled j. However, a contradiction occurs either in V_1 or in V_0, with equal probability.



Prob = 1/2 Prob = 1/2

D) *Both are dependent.* In this case, both child nodes have the same label as the parent node. Due to the linear dependency of a_{2i} there occurs a contradiction in either V_1 or V_0, with equal probability. In addition, there is an additional probability of $1/2$ that a_{2i+1} leads to a contradiction in V_1.



Definition 1. *A* ***well-formed tree*** T^*_ℓ *is a binary tree where only branchings of type A occur, i.e., for every node that is not a leaf, the following rule holds: If the label of the node is* j*, then the label of its left child is* $j-2$ *and the label of its right child is* $j-1$*.*
A ***malformed tree*** *is an arbitrary tree of guesses that contains at least one branching of a type B, C or D.*

Essentially, the notion of a well-formed tree describes the tree of guesses under the assumption that all linear equations (of both type 1 and 2) are linearly independent. Note that such a tree is highly unlikely for large ℓ. Nonetheless, the well-formed tree plays an important role in establishing the overall number of leaves for the tree of guesses. We proceed now to prove that on average, a malformed tree has at most the same number of leaves as a well-formed tree.

Theorem 1. *Let* C^*_ℓ *denote the number of leaves of a well-formed tree* T^*_ℓ*. Let* C_ℓ *denote the maximum number of leaves in a tree* T_ℓ *that may or may not be malformed. Then in the average case,* $C_\ell \leq C^*_\ell$ *holds.*

Proof. The proof is by induction. Obviously, the inequality holds for C_{-1} and C_0, since trees T_{-1} and T_0 consist only of a root without a child. Thus, $C_{-1} = C^*_{-1} = 1$ and $C_0 = C^*_0 = 1$.
Now consider C_ℓ, $\ell \geq 1$. First note that since the theorem holds for $C_{\ell-1}$ and $C_{\ell-2}$, it follows that

$$C_{\ell-1} + C_{\ell-2} \leq C^*_{\ell-1} + C^*_{\ell-2} = C^*_\ell. \tag{1}$$

Also note that even in the worst possible branching case, we have

$$C_\ell \leq 2 \cdot C_{\ell-1}. \tag{2}$$

for all ℓ. Using these two facts, we can prove an upper bound for C_ℓ by distinguishing the following cases (identical to the ones given above):

A) Let the tree T^A_ℓ be composed of a subtree with at most $C_{\ell-2}$ leaves and a subtree with at most $C_{\ell-1}$ leaves. It follows for the maximum number C^A_ℓ of such a tree that
$$C^A_\ell \leq C_{\ell-2} + C_{\ell-1} \leq C^*_\ell.$$

B) The tree T_ℓ^B is composed of either one or two subtrees, having at most $C_{\ell-1}$ leaves each. Consequently, $C_\ell^B \leq 1/2 \cdot C_{\ell-1} + C_{\ell-1}$. Using (2), we have

$$C_\ell^B \leq C_{\ell-2} + C_{\ell-1} \leq C_\ell^*.$$

C) The tree T_ℓ^C is composed of only one subtree with at most $C_{\ell-1}$ or C_ℓ leaves, resp. (with equal probability). We have $C_\ell^C \leq 1/2 \cdot (C_{\ell-1} + C_\ell)$, and using (2), derive

$$C_\ell^C \leq \frac{1}{2}(2C_{\ell-2} + 2C_{\ell-1}) = C_{\ell-2} + C_{\ell-1} \leq C_\ell^*.$$

D) The tree T_ℓ^D has one of the forms given in case D. Then, for the average number C_ℓ^D of leaves in this tree, we have $C_\ell^D \leq \frac{3}{4} \cdot C_\ell$. Using (2) repeatedly, we get

$$C_\ell^D \leq \frac{3}{2} \cdot C_{\ell-1} = C_{\ell-1} + \frac{1}{2}C_{\ell-1} \leq C_{\ell-1} + C_{\ell-2} \leq C_\ell^*.$$

Since $C_\ell = \max(C_\ell^A, C_\ell^B, C_\ell^C, C_\ell^D)$, we have $C_\ell \leq C_\ell^*$. □

5.2 Size of a Well-Formed Tree

We have shown that the number C_ℓ of leaves in an arbitrary tree of guesses is on average not bigger than the number C_ℓ^* of leaves in a well-formed tree. In the next section, we will prove an estimate for C_ℓ^* and thus an upper bound for C_ℓ.

Theorem 2. *Let C_ℓ^* denote the size of a well-formed tree T_ℓ^*. Then we have $a^\ell \leq C_\ell^* \leq \frac{2}{a}a^\ell$ for all $L \geq 1$, where $a = \frac{1+\sqrt{5}}{2} \approx a^{0.6942419}$.*

Proof. Note that for all $\ell \geq -1$, C_ℓ^* satisfies the recursion $C_{\ell+2}^* = C_{\ell+1}^* + C_\ell^*$ with $C_{-1}^* = C_0^* = 1$.
Let a be the unique positive solution of $x^2 = x + 1$, i.e. $a = \frac{1+\sqrt{5}}{2}$. In this case, the function $F(\ell) = a^\ell$ also satisfies the recursion $F(\ell+2) = F(\ell+1) + F(\ell)$ for all $\ell \geq 0$. Since $C_0 = F(0)$ and $C_1 = \frac{2}{a}F(1)$, we have $a^\ell \leq C_\ell \leq \frac{2}{a}a^\ell$ for all $\ell \geq 0$. □

Note that $\frac{2}{a} \approx 1.236068$. Thus, we have found the upper bound of the average search tree to be $C_\ell \leq \frac{2}{a} \cdot 2^{0.694\ell} \approx 2^{0.694\ell+0.306}$.

5.3 Worst Case Considerations

The above result can be applied directly to the tree of guesses in section 4. Remembering that such a search tree actually has a root labelled $\ell = L - 2$, we can upper bound the average number of leaves by $C_L \leq 2^{0.694L-0.918}$.

This upper bound seems to holds even for the worst case, provided that L is large enough. Remember that assumption 1 stated that in case of a linearly dependent equation, contradiction occurs with probability 1/2. Now remember from section 4.2 that linearly dependent equations do not occur before depth $\lfloor \frac{L}{2} \rfloor$ is reached. This, in turn, means that for large L, there exists a large number of nodes labelled j for each $j < L - \lfloor \frac{L}{2} \rfloor$. We can thus apply the law of large numbers, stating that the actual number of contradictions is very close to the expected number of contradictions. Thus, the number of leaves should be close to the above bound not only for the average case, but for almost any tree of guesses.

In order to give some more weight to this rather informal argument, we will provide some empirical evidence for this conjecture in section 6.

5.4 Running Time of the Algorithm

The asymptotically most expensive single step of the backtracking algorithm presented in section 4 is the testing of the linear dependency of new equations. This operation in itself takes $O(L^3)$ elementary steps and has to be repeated once or twice in each node of the tree of guesses.

Thus, we have to establish an upper bound for the maximum number of *nodes* in the tree. Since the tree will be malformed, it contains nodes that have only one child. It is thus impossible to upper bound the number of nodes by $2 \cdot C_L - 1$, as could be done for a proper binary tree. We can, however, prove that the maximum depth of the search tree is $L - 1$.

Proposition 2. *If the linear recurrent sequence $(a_i)_{i \geq 0}$ is of maximum length, then the tree has maximum height of $L - 1$.* [3]

Proof. In any node of depth i, we have exactly $i + 1$ equations of type 1 at our disposal (and a varying number of equations of type 2). Thus, at depth $L - 1$, we have exactly L such equations, namely $a_0, a_2, \ldots, a_{2L-2}$.
By a theorem on maximum length linear recurrent sequences (see e.g. [5], p. 76), there exists a k such that the following holds:

$$(a_k, a_{k+1}, \ldots, a_{k+L-1}) = (a_0, a_2, \ldots, a_{2L-2})$$

Since $a_k, \ldots, a_{k+L-1}$ must be linearly independent, the same holds for $a_0, a_2, \ldots,$ a_{2L-2}. Consequently, we have L linearly independent equations of type 1 in any node of depth $L-1$, allowing us to solve the system and derive a key candidate. Thus, no node of the tree will have depth $\geq L$. $\square$

We can use this fact to upper bound the number of nodes. Consider the largest binary tree (w.r.t. the number of nodes) with height $L-1$ and C_L leaves. This tree is a complete binary tree from depth 0 to $p = \lfloor \log C_L \rfloor$. From depth $p + 1$ to depth $L - 1$, the tree has constant width of C_L.

[3] Note that this proposition only holds for maximum length sequences. The use of shorter sequences, however, would be a breach of elementary design principles, since it would facilitate a number of other attacks. It does not seem to increase resistance against our attack either, it just makes the proof harder.

Let N_L denote the number of nodes in a search tree. It follows that N_L is at most the size of this worst possible tree.

$$N_L \leq (2^{p+1} - 1) + (L - p - 1) \cdot C_L$$

Note that both 2^{p+1} and C_L are in $O(C_L)$. Ignoring all constant summands and factors to C_L, we obtain:

$$\begin{aligned} N_L &\in O((L-p) \cdot C_L) \\ &= O(0.306L \cdot 2^{0.694L-0.918}) \\ &= O(0.162L \cdot 2^{0.694L}) \end{aligned}$$

Remembering that in each node, a linear equation has to be inserted into an equation system, and ignoring constant factors again, we derive a total asymptotic running time in $O(L^4 \cdot 2^{0.694L})$.

6 Experimental Results

6.1 Results on the Number of Leaves

In the section 5, we have proven the number of leaves in the search tree to be upper bounded by $2^{0.694L-0.918}$ in the average case. This result leaves a number of interesting questions open: Since we have only derived an upper bound: How close is this value to the average number of leaves that do occur in an actual search?[4] And what about the conjecture in section 5.3? Is C_L also an upper bound for the worst case, for large L?

In order to answer those questions, the key reconstruction algorithm from section 4 has been implemented and tested against all keys and all primitive polynomials for $L = 3, \ldots, 16$. The main results of this simulation are given in the left half of table 2. Here, C_{avg} and C_{max} denote the average and maximum number of leaves encountered in the experiments. $C_{bound} = 2^{0.694L-0.918}$ denotes the upper bound as calculated in section 5. For ease of comparison, all values are given in logarithmical notation.

First observe that values C_{avg} and C_{max} are very close; they differ by a factor ϕ with $1 < \phi < 1.33$. Of course, this may or may not hold for larger values of L, but for small L, the maximum number of leaves does not stray very far from the average.

Also observe that for $L > 8$, C_{bound} seems to be a proper upper bound not only for the average case, but also for the maximum number of leaves in the search tree. Note especially that for $L > 8$, the gap between C_{max} and C_{bound} seems to be widening with increasing L. Nonetheless, additional empirical or mathematical evidence for larger L might be necessary before our conjecture from section 5.3 can be considered confirmed.

[4] We must take care not to confuse the average case of the analysis with the average number of leaves in the search tree; they are quite different mathematical objects.

	Number of leaves			Number of nodes		
L	C_{avg}	C_{max}	C_{bound}	N_{avg}	N_{max}	N_{bound}
3	$2^{1.00}$	$2^{1.00}$	$2^{1.16}$	$2^{1.58}$	$2^{1.58}$	$2^{1.04}$
4	$2^{1.55}$	$2^{1.58}$	$2^{1.86}$	$2^{2.57}$	$2^{2.81}$	$2^{2.15}$
5	$2^{2.29}$	$2^{2.58}$	$2^{2.55}$	$2^{3.28}$	$2^{3.46}$	$2^{3.17}$
6	$2^{2.85}$	$2^{3.17}$	$2^{3.25}$	$2^{4.21}$	$2^{4.64}$	$2^{4.12}$
7	$2^{3.53}$	$2^{3.81}$	$2^{3.94}$	$2^{4.92}$	$2^{5.43}$	$2^{5.04}$
8	$2^{4.23}$	$2^{4.64}$	$2^{4.63}$	$2^{5.61}$	$2^{5.93}$	$2^{5.93}$
9	$2^{4.88}$	$2^{5.29}$	$2^{5.33}$	$2^{6.35}$	$2^{6.79}$	$2^{6.79}$
10	$2^{5.53}$	$2^{5.88}$	$2^{6.02}$	$2^{7.05}$	$2^{7.55}$	$2^{7.64}$
11	$2^{6.22}$	$2^{6.57}$	$2^{6.72}$	$2^{7.75}$	$2^{8.24}$	$2^{8.47}$
12	$2^{6.87}$	$2^{7.26}$	$2^{7.41}$	$2^{8.46}$	$2^{8.89}$	$2^{9.29}$
13	$2^{7.56}$	$2^{7.92}$	$2^{8.10}$	$2^{9.16}$	$2^{9.73}$	$2^{10.10}$
14	$2^{8.25}$	$2^{8.56}$	$2^{8.80}$	$2^{9.85}$	$2^{10.20}$	$2^{10.90}$
15	$2^{8.92}$	$2^{9.23}$	$2^{9.49}$	$2^{10.56}$	$2^{11.26}$	$2^{11.69}$
16	$2^{9.61}$	$2^{9.90}$	$2^{10.19}$	$2^{11.25}$	$2^{11.64}$	$2^{12.48}$

Table 2. Empirical Results

6.2 Results on the Number of Nodes

In the right half of the table, we give the results on the number of nodes. Again, N_{avg} and N_{max} denote the average and maximum values encountered in the experiments, while $N_{bound} = 0.162L \cdot 2^{0.694L}$ denotes the mathematical bound as given in section 5.4.

It seems that for $L > 7$, N_{bound} is an upper bound for the number of nodes in the worst possible case. As with the results on the number of leaves, the gap between N_{max} and N_{bound} seems to be widening with increasing L, but again, more data for larger L would be helpful. We also note that N_{avg} and N_{bound} are very close to each other.

An interesting side observation is that $N_{avg} \approx 2 \cdot C_{bound}$, i.e. that the average number of nodes appears to be almost exactly twice the mathematical upper bound for the number of leaves as derived in section 5.3. This is not apparent from the mathematical analysis in section 5 and may thus be an interesting starting point for future research.

7 Design Recommendations

The effective key size against our attack is less than 70% of the key length. For a register length of 120 bit, the backtracking attack runs in $O(2^{83})$ steps and is probably not feasible in today's practice. Our attack, however, is easily parallelised, allowing an adversary to use as many parallel processors at once as he can afford. Since each processor can operate on its own segment of the tree (without any need of communication with the other ones), k processors can reduce the running time by a factor of k. Thus, a generator using a shorter register is in real danger of being compromised. We conclude that the **minimum length** of a self-shrinking generator should exceed 120 bit.

Note that our attack relies on the feedback logic of the register to be known. If it is not, the attack has to be repeated for all primitive feedback polynomials of length L, yielding an additional working factor of $\phi(2^L - 1)/L$. Security of the self-shrinking generator can thus be increased significantly by following the proposal given in [2, 8]: Use a **programmable feedback logic** and make the actual feedback polynomial a part of the key.

Finally, observe that the use of sparse feedback polynomials makes our attack slightly more effective. If the more significant bits depend on only a few of the less significant bits, the probability of linear dependent equations increases, yielding a tree of guesses that is more slender than the average case tree considered above. However, as stated in section 6.2, the sizes of worst case and best case trees seem to differ by less than the factor 2. Nonetheless, **sparse feedback polynomials** should be avoided in designing most stream ciphers, the self-shrinking generator being no exception.

Acknowledgement We would like to thank one of the anonymous referees for a number of very helpful comments.

References

[1] S.R. Blackburn. The linear complexity of the self-shrinking generator. *IEEE Transactions on Information Theory*, 45(6):2073–2077, September 1999.

[2] D. Coppersmith, H. Krawczyk, and Y. Mansour. The shrinking generator. In D.R. Stinson, editor, *Advances in Cryptology - EUROCRYPT '93*, volume 773 of *LNCS*, pages 22–39, Berlin, 1993. Springer-Verlag.

[3] J.D. Golić. Cryptanalysis of alleged A5 stream cipher. In W. Fumy, editor, *Advances in Cryptology - EUROCRYPT '97*, volume 1233 of *LNCS*, pages 239–255, Berlin, 1997. Springer-Verlag.

[4] J.D. Golić and L. O'Connor. Embedding and probabilistic attacks on clock-controlled shift registers. In A. De Santis, editor, *Advances in Cryptology - EUROCRYPT '94*, volume 950 of *LNCS*, pages 230–243, Berlin, 1995. Springer-Verlag.

[5] S.W. Golomb. *Shift Register Sequences.* Aegean Park Press, Laguna Hills (CA), revised edition, 1982.

[6] H. Krawczyk. The shrinking generator: Some practical considerations. In R. Andersen, editor, *Fast Software Encryption '93*, volume 809 of *LNCS*, pages 45–46, Berlin, 1994. Springer-Verlag.

[7] J.L. Massey. Shift register synthesis and BCH decoding. *IEEE Transactions on Information Theory*, 15:122–127, 1969.

[8] W. Meier and O. Staffelbach. The self-shrinking generator. In A. De Santis, editor, *Advances in Cryptology - EUROCRYPT '94*, volume 950 of *LNCS*, pages 205–214, Berlin, 1995. Springer-Verlag.

[9] M.J. Mihaljević. A faster cryptanalysis of the self-shrinking generator. In J. Pieprzyk and J. Seberry, editors, *Advances in Cryptology - ACISP '96*, volume 1172 of *LNCS*, pages 182–189, Berlin, 1996. Springer-Verlag.

[10] I. Shparlinski. On some properties of the shrinking generator. `http://www.comp.mq.edu.au/~igor/Shrink.ps`.

[11] L. Simpson, J.D. Golić, and E. Dawson. A probabilistic correlation attack on the shrinking generator. In C. Boyd and E. Dawson, editors, *Advances in Cryptology - ACISP '98*, volume 1438 of *LNCS*, pages 147–158, Berlin, 1998. Springer-Verlag.

Attacks Based on Small Factors in Various Group Structures

Chris Pavlovski[1,2] and Colin Boyd[1]

[1] Information Security Research Centre, School of Data Communications
Queensland University of Technology, Brisbane, Australia
boyd@fit.qut.edu.au
[2] IBM Global Services, Brisbane, Australia
chripavl@au1.ibm.com

Abstract. We describe new attacks that can be launched on some well known signature schemes. The attacks are related to Lim and Lee's key recovery attacks in prime order subgroups. Several new attacking scenarios are described where the group order can be either prime, composite, or unknown. These attacks are able to compromise certain properties of complex protocols such as identity revelation by the revocation manager in a group signature setting, or owner tracing in fair electronic cash. It is suggested that safe primes must be considered for use in all such protocols, together with a proof of safe parameter selection.

1 Introduction

Many cryptographic protocols operate in a subgroup of some larger group, generally placing restrictions on the parameter selection of the larger group. In many cases group operations are performed within a prime order subgroup of a much larger group, whilst in others, operations occur in a group of composite modulus within a larger group. The most obvious primitive protocols that satisfy these characteristics include the Diffie-Hellman [14], Schnorr [22] and other ElGamal-type protocols.

These well known primitives serve as the basis for signature and encryption schemes; more complex schemes build upon these to provide additional properties such as undeniablity [12], blindness [11], and group membership [9]. Further work has also given rise to complex protocols furnishing a variety of security related applications, including electronic cash [5] and election protocols [13].

Many researchers have noted the usefulness of setting a protocol in a prime order subgroup of a larger group. In this paper it is shown that, unless the *larger group* takes on a specific form, then potential exists for protocol exposure. The special form we refer to is that the order of the large group should have no factors smaller than the order of the subgroup (with the possible exception of the factor 2 which often cannot be avoided). For example, in a subgroup of $\mathbb{Z}_p^*$ of prime order q, $(p-1)/2$ should have no factors smaller than q.

Lim and Lee [19] demonstrated how a key recovery attack can be launched against many published scheme given such parameters. They also describe how to launch a general attack on schemes working in prime order subgroups. Their basic idea is to employ a substitution $a \rightarrow \gamma a \bmod p$, where γ is a generator of a small subgroup of $\mathbb{Z}_p^*$; we call this the *direct low order attack*. They demonstrate a direct low order attack on the undeniable signature algorithm used in Brands' cash scheme and also show how to compromise discrete log based systems in a prime order subgroup by obtaining a signature on γa, where the order of γ is a factor of the order of the larger group $\mathbb{Z}_p^*$.

With the fundamental notion of the direct low order attack in mind, we show in this paper several new attacks and weaknesses based on the existence of small order elements in various multiplicative groups. Some of these attacks apply to protocols published subsequent to the Lim and Lee attack [19], while others apply to earlier protocols that use alternative algebraic structures, specifically subgroups with non-prime order. Some such protocols continue to select parameters based solely upon primality. As these security protocols are extended to take on additional properties the possibility exists for new attacks.

1.1 Contribution

This paper demonstrates several new attacks by extending the fundamental notion of Lim and Lee [19], applied to prime order subgroups, to various different group structures and protocols settings. Recall that a prime p is called *safe* if $(p-1)/2$ is also prime. We show that fixing the schemes may be dependent upon selecting parameters as safe primes (or the product of safe primes) and proofs that they have this safe form.

We demonstrate our attacks on a group signature scheme of Camenisch and Stadler [9]. Schemes which employ this as a primitive, such as cash, voting, and auction protocols, may all inherit a similar exposure. Specifically, an attack is shown where a member of a group may sign on behalf of the group in a manner that prevents the group manager from revoking the identity of the group member. Further attacks on specific electronic cash schemes, which make use of various signature schemes as primitives, are also presented.

We analyse possible precautions for avoiding the attack, concluding that selection of a safe prime is required together with a proof of such a safe selection (required when the group order is unknown). In addition, there still exists the need to check protocol elements due to the presence of the element of order two. Finally, it is suggested that selection of safe primes is desirable for security protocols in general. This is based on the evolutionary pattern that follows such original protocols, where new properties are later devised which may possibly be thwarted by the presence of factors in large multiplicative groups.

We regard the following as the main contributions of this paper.

- We show that attacks based on factors may be applied to various multiplicative group structures[1], outlining how attacks may be mounted where the group order is prime, non-prime, or unknown.

[1] Lim and Lee's attack applies to prime order subgroups, where the prover applies the signature key to elements supplied by another party [19].

- We demonstrate ways to attack group signature schemes where the group manager cannot identify the group member who generated a signature.
- We detail exposures and protocol weaknesses of some electronic cash schemes.
- We show how protocols in their original form may exhibit no exposure, but when used as a primitive in a more complex protocol an attack can be launched.
- We propose several techniques to avoid such attacks.

1.2 Organisation of Paper

The next section provides a background to the central observations of the attack and the related work of attacks based on subgroups. We then outline specific attacks that can be made on some published schemes to illustrate the exposure. This includes a specific attack on group signature schemes, exposures to protocols based upon these signature techniques, and some weaknesses in electronic cash schemes. We then sketch possible ways to avoid this type of attack.

2 Background and Related Work

Attacks exploiting subgroup structures are not new, and several researchers have used their properties to expose protocol weaknesses. Lim and Lee [19] used elements of smooth order in key recovery attacks on discrete log based schemes that use a prime order subgroup. Burmester [6] observed that a false Schnorr public key may be issued that is able to fool verification. Each of these works will now be reviewed.

Lim and Lee [19] outline several key recovery attacks on discrete log based schemes operating in the large group $\mathbb{Z}_p^*$, or some prime order subgroup. These attacks are able to disclose the secret key in key agreement protocols and ElGamal signature schemes. The specific attacks depend on finding partial discrete logarithms y_i to the base α, where α is an element of low order. This is possible by breaking down the discrete logarithm problem over $\mathbb{Z}_p^*$, into several smaller problems as subgroups using the Pohlig-Hellman decomposition [20], defined by low order elements of $\mathbb{Z}_p^*$.

Burmester [6] previously pointed out that it is possible to cheat using the Schnorr identification scheme. Recall that the Schnorr scheme uses primes p and q where $q|p-1$, and a generator g of G of order q. The private key is x and the public key $h = g^{-x} \bmod p$. The prover first chooses $w \in_R \mathbb{Z}_q$ and forwards $a = g^w \bmod p$ to the verifier. The verifier generates a challenge $c \in_R \mathbb{Z}_q$ and the prover sends the response $r = cx + w \bmod q$. The verification equation is $g^r = ah^c \bmod p$. Burmester points out that if the signer publishes some public key $h' = \beta^{-x} \bmod p$, where β is of order $2q$ and x is odd, then the probability that the verification will hold is 1/2. To achieve this the prover sends $a = g^w \bmod p$ or $a = g^w h' \bmod p$ (in an attempt to guess the parity of the verifier's challenge c), and then returns $r = cx/2 + w \bmod q$ if c is even and otherwise $r = x(c-1)/2 + w \bmod q$. The verifier will find that $g^r = ah'^c \bmod p$ holds if the prover has correctly guessed the parity of the challenge; this occurs with probability 1/2. To prevent the attack the verifier should check that h' lies within the prime order subgroup G, by checking $h'^q \bmod p = 1$. Burmester also points out an exposure in the Fiat-Shamir [15] scheme using a composite modulus.

Recently Boudot [2] observed a weakness in the proof of security for Girault's scheme [17] by exploiting the element of order two in the setting of discrete logarithm with a composite modulus.

We here detail the central observation and then show how one may construct alternative attacks. Let G be a prime order subgroup of $\mathbb{Z}_p^*$ of order q, where $q|p-1$ and p, q prime, and g be a generator of G. Let $a \in G$ and $\gamma \in \mathbb{Z}_p^*$ be of order $t < q$. Then given an integer $x \in \mathbb{Z}_q$ such that $g^x = a$, it follows that $(\gamma g)^x \equiv a \bmod p$ when $t|x$. When there are no conditions on the choice of the prime p, $p-1$ will likely have many factors less than q. A similar observation can be made regarding composite modulus systems. Note that in the case of composite modulus systems a test for group membership may not be possible if the order of the group is unknown. The *direct low order attack* of Lim and Lee makes the transformation $a \rightarrow \gamma a$, where $ord(\gamma) < ord(a)$, and the signing entity directly applies its signature key (exponent). In this paper we demonstrate how alternative attacks may be launched on several well known protocols that use various multiplicative group structures.

3 Attacks on Group Signature Schemes

In this section we show how an attack may work against group signature schemes that allows a group member to sign on behalf of the group in a manner that prevents the group manager from revealing the member's identity. We then outline a collusion attack between a group member and the group manager, showing that the group manager must prove to each group member that the order of the group contains no small factors.

Group signatures were introduced by Chaum and van Heyst [8]. Camenisch and Stadler [9] designed a group signature scheme where the public key of the group remains constant in size and complexity with an increasing group size. Group signature schemes enable a member of a group to generate a signature on behalf of the group. Each group member has a unique signing key, however signed messages may be verified using the single public key for the group. There is an additional entity, the *group manager*, which is able to reveal the identity of the group member who is responsible for creating any signature on behalf of the group. There are five operation phases of a group signature scheme.

Setup - generates the group public key and secret administration key of the group manager.
Join - enables a member to join the group, creating a membership certificate on her private key.
Sign - group member signs on behalf of the group using her private key.
Verify - checks the signature using the public key of the group.
Open - reveals the identity of the group member who generated a signature.

The responsibilities of the group manager may be split into those of the *membership manager* who administers the Join operation, and the *revocation manager* who is responsible for the Open operation.

Camenisch and Stadler [9] stated that group signature schemes must satisfy the property that 'group members can neither circumvent the opening of a signature nor sign on behalf of other group members'. In the attack that follows it is shown that the opening protocol may be thwarted: a valid group member may generate a signature on behalf of the group in way that prevents the revocation manager from revealing the identity of the member.

The scheme of Camenisch and Stadler relies on two 'signatures of knowledge', known as *SKLOGLOG* and *SKROOTLOG*. The first of these provides a signature of a message m that proves that the signer knows the double discrete logarithm of a given value with respect to two base values. The second provides a signature of m that proves that the signer knows the e-th root of a discrete logarithm of a known value for a known e and base value. Our attack is based on the observation that these signatures may be generated in an incorrect form if the known values are not in the subgroup G where they are intended to lie.

3.1 Preventing Opening of Signatures

The setup of the Camenisch-Stadler scheme [9] involves the selection of security parameter l by the group manager and publication of the public key $(n, e, G, g, a, \lambda, \nu)$:

- RSA public and private keys [21] are (n, e) and d respectively.
- a cyclic group $G = \langle g \rangle$ of order n in which computing discrete logarithms is infeasible. We assume that G is a subgroup of $\mathbb{Z}_P^*$ where P is prime and $n|P-1$, as suggested by Camenisch and Stadler.
- a collision resistant hash function $\mathcal{H}$ which takes strings of any length to strings of a fixed length (suggested as 160 bits by Camenisch and Stadler).
- $a \in \mathbb{Z}_n^*$ of large multiplicative (unknown) order modulo both prime factors of n.
- an upper bound λ on the length of the secret keys and a constant $\nu > 1$.

Definition 1. *A signature of knowledge, for the message m, of the double discrete logarithm of z to the bases g and a is a tuple $(c, s_1, \ldots s_l)$ satisfying*

$$c = \mathcal{H}(m \parallel z \parallel g \parallel a \parallel t_1 \parallel \ldots \parallel t_l) \text{ with } t_i = \begin{cases} g^{(a^{s_i})} & \textit{if } c[i] = 0 \\ z^{(a^{s_i})} & \textit{if } c[i] = 1 \end{cases}$$

and is denoted $SKLOGLOG[\alpha : z = g^{(a^\alpha)}](m)$.

Camenisch and Stadler state that $SKLOGLOG[\alpha : z = g^{(a^\alpha)}](m)$ can only be computed with knowledge of $\alpha = x$, the double discrete logarithm of z. We now show that this statement is not quite accurate unless additional checks are made beyond what is in Definition 1. The following attack is a small variation on the generation of a correct signature as specified by Camenisch and Stadler.

1. Choose $\epsilon \in \mathbb{Z}_P^*$ of low order t, and compute $\tilde{z} = \epsilon z \bmod p$. Finding such an ϵ is not hard: first find a generator g_0 of $\mathbb{Z}_P^*$ by trial and error and then set $\epsilon = g_0^{P/t}$. In general there may be many small values of t with $t|(P-1)/n$, and $t = 2$ will always be a possible choice.
2. Choose $r_i \in_R \{0, \dots, 2^{\lambda\nu} - 1\}$, for $i = 1$ to l, such that $a^{r_i - x} \bmod n$ is a multiple of the order t. Note that, although this property is ensured for all i, it will only be required for around half the values, depending on the value of c found in the next step. Of course, the attacker is not able to guess in advance for which i values the property will be required. If $t = 2$, then on average this will require two trials for each r_i, so that $a^{r_i - x} \bmod n$ is even.
3. Next compute $c = \mathcal{H}(m \parallel \tilde{z} \parallel g \parallel a \parallel g^{a^{r_1}} \parallel \dots \parallel g^{a^{r_l}})$ and set

$$s_i = \begin{cases} r_i & \text{if } c[i] = 0 \\ r_i - x & \text{if } c[i] = 1 \end{cases}$$

The signer now claims that $(c, s_1, \dots, s_l)$ is a valid signature of knowledge $SKLOGLOG[\alpha : \tilde{z} = g^{(a^\alpha)}](m)$. It can be seen that the verification equation still holds since the choice of r_i in step 1 has ensured that a^{s_i} in Definition 1 will always be a multiple of t when $c[i] = 1$. Yet the signer certainly does not know the double discrete logarithm of $\tilde{z}$. In fact if n is a good RSA modulus then $\epsilon \notin G$ and so no such double discrete logarithm even exists!

A very similar forgery is possible for the second signature of knowledge, *SKROOTLOG* used by Camenisch and Stadler. Now we describe the group signature setup in which these two signatures of knowledge are used.

A group member obtains her membership certificate $(y+1)^d \bmod n$, by choosing $x \in_R \{0, \dots, 2^\lambda - 1\}$, computing $y = a^x \bmod n$ and membership key g^y. The pair (y, g^y) is forwarded to the group manager whilst proving knowledge of $\log_a(y)$. When convinced the group manager returns the membership certificate $\beta = (y+1)^d \bmod n$.

Signing Procedure To sign a message m the group member chooses $r \in_R \mathbb{Z}_n^*$, computes $\tilde{g} = g^r \bmod p$, $z = \tilde{g}^y \bmod p$, and forms $V_1 = SKLOGLOG[\alpha : z = \tilde{g}^{a^\alpha}](m)$ and $V_2 = SKROOTLOG[\beta : z\tilde{g} = \tilde{g}^{\beta^e}](m)$.

To mount the attack, the following steps are performed by the group member.

1. Choose $r \in_R \mathbb{Z}_n^*$ and computes $\tilde{g} = g^r$, where $r \in_R \mathbb{Z}_n^*$.
2. Choose ϵ of order $t < n$, compute $\tilde{z} = \epsilon z$.
3. Form $V_1 = SKLOGLOG[\alpha : \tilde{z} = \tilde{g}^{a^\alpha}](m)$ and $V_2 = SKROOTLOG[\beta : \tilde{z}\tilde{g} = \tilde{g}^{\beta^e}](m)$ as shown above.

The signature consists of the tuple $(\tilde{g}, \tilde{z}, V_1, V_2)$ and the verification consists of checking that V_1 and V_2 are correct signatures of knowledge. The purpose of V_1 is to show that the signer is a valid group member because V_1 confirms that $\tilde{z}$ is of the form $\tilde{g}^{a^\alpha}$; this proves the signer knows some value $\alpha = x$. Then, since $\tilde{z}\tilde{g} = \tilde{g}^{a^{\alpha}+1}$, then V_2 proves that the signer has a membership certificate $(a^\alpha + 1)^d$ for the secret value $\alpha = x$.

At some later point, if foul play is suspected, the signature $(\tilde{g}, \tilde{z}, V_1, V_2)$ may be forwarded to the revocation manager to attempt to reveal the identity of the group member. This is usually done by checking if $\tilde{g}^{y_i} = \tilde{z}$ for the key y_i of each group member i. However in this case $\tilde{g}^{y_i} \neq \tilde{z}$ and hence the revocation manager is unable to identify the group member.

As noted above, the simplest attack is where the order of ϵ is 2. This can be varied by choosing an element of higher order, whilst ensuring that the order $t < n$. A larger order, however, requires a longer running time in the signing procedure to determine r_i values that will enable the attack to succeed.

It is worth noting that Ateniese and Tsudik [1] have shown that there is a weakness in the Camenisch and Stadler scheme, unrelated to the attack presented here. A suggested fix, which alters the definition of the membership certificate, does not affect the validity of our attack. The attack as described here can be prevented if the verifier checks that $\tilde{z}$ is actually in the group G (as implicitly claimed). This can be achieved at the cost of one exponentiation by verifying that $\tilde{z}^n \bmod P = 1$. However, we will see below that this check by itself is not enough to prevent the attack in general.

An essentially identical attack to that described here may be made against the scheme of Stadler [23] for publicly verifiable encryption of discrete logarithms. (Indeed the $SKLOGLOG$ is based on that scheme.) Such a scheme is designed to ensure that any party can verify that shares of a discrete logarithm of a known value have been correctly distributed to a set of trustees. The attack can be used by the dealer of the secret to prevent trustees from recovering the correct shares even though the verification equation has been checked.

In Stadler's scheme the prover forms an ElGamal ciphertext, which is a pair (A, B). The prover then proves to the verifier that when decrypted by the private key of a designated party the corresponding plaintext for (A, B) will be the discrete logarithm of a known value V. The proof is identical to the $SKLOGLOG$ discussed above and the same attack will work. This allows the prover to use a value $\tilde{V} = \epsilon V$, for any low order element ϵ, and 'prove' that (A, B) will decrypt to the discrete logarithm of $\tilde{V}$. Whether this results in a meaningful attack will depend on the exact application for which verifiable encryption is used.

A related group signature scheme has been published more recently by Camenisch and Michels [7]. In this more recent scheme it is required that the RSA modulus used is a product of safe primes, and that it is required to be proven by the membership manager that this is the case. Therefore the attack described here does not apply.

3.2 Collusion Attacks

We now show a collusion attack where the membership manager and one group member may generate a signature that once again prevents the revocation manager from revealing the identity with the correct use of the revocation protocol. Suppose that the membership manager chooses $n = pqw$. The manager may then collaborate with one group member and reveal the factor w of n. The membership manager can reveal w while it is still computationally infeasible for the group member to factorise n, and thus obtain p and q. Then, using the attack outlined in the previous section, the group member is able to generate a group signature on some message which prevents the identity from being revealed with the correct use of the revocation protocol. The attack proceeds as follows:

– The membership manager selects $n = pqw$ and generates the prime P such that $n|P - 1$.
– The membership manager reveals w to one group member but not to any other party (including the revocation manager).
– The membership manager generates g to be an element of order pq and publishes the usual parameters including g.
– The group member finds an element ϵ or order w. This can be achieved with high probability by choosing a random element $\delta \in \mathbb{Z}_P^*$ and setting $\epsilon = \delta^{n/w} \bmod P$.
– At signature time the group member replaces z with $\tilde{z} = \epsilon z$ and the attack proceeds as above.

In distinction to the previous attack, checking that $\tilde{z} \in G$ in this instance will not reveal any problem. This is because the small order element is now actually in the group that it is supposed to be in. A means to detect this attack is to insist that the group manager publishes a proof that n is the product of only two primes. Efficient methods for such a proof are known [18].

4 Electronic Cash Schemes

Many electronic cash protocols employ as primitives one or more of the signature schemes discussed above, and hence may be exposed. Techniques which may be used to repair or avoid the attacks shown here are discussed in Section 5.

4.1 Traoré's Scheme

Traoré [24] proposes an innovative electronic cash scheme based upon group signatures where the customers form a group. We show that the above attack may be extended to work against the scheme[2].

In Traoré's scheme the RSA primes are chosen to be safe but the large group $\mathbb{Z}_P^*$ may contain many small factors. P is chosen by first generating $n = pq$, where $p = 2p' + 1$, $q = 2q' + 1$ and p, q, p', q' are all prime) and then choosing P by searching for $P = kn + 1$, where k is an integer. In this case it is very likely that $P - 1$ will contain many small factors and hence there are small order elements easily found in the large group $\mathbb{Z}_P^*$. Now, it is possible to devise a similar attack to the group signature scheme of Camenisch and Stadler [9], although there are a few extra details to consider.

Traoré's scheme requires users to obtain a *license* which is a pair of integers which must be used during each coin withdrawal and 'embedded' in each coin

[2] A revised protocol with repairs against this attack is detailed by Traoré [25].

withdrawn. A license is obtained through a protocol executed with the group manager during which the member proves that he knows the discrete log of a value ID_{U_i} which will be later used to revoke anonymity if necessary. This uses a proof of knowledge called $Proof_{LOG+RANGE}$. In the following l_1, l_2 and $\epsilon > 1$ are constants; α is a quadratic residue in a different multiplicative group Z_N^* where N is a product of different safe primes and l_N is the bit-length of N. Finally, k is a security parameter.

Definition 2. *A proof of knowledge of the discrete logarithm of* h *with respect to* g *and of* δ *with respect to* α*, which also proves that* $\log_g h = \log_\alpha \delta$ *and that* $\log_g h$ *is in* $[2^{l_1} - 2^{\epsilon(l_2+k)+1}, 2^{l_1} + 2^{\epsilon(l_2+k)+1}]$ *is a pair* (c, r) *satisfying*

$$c = \mathcal{H}(g \parallel h \parallel \alpha \parallel \delta \parallel g^{r-c2^{l_1}} h^c \parallel \alpha^{r-c2^{l_1}} \delta^c)$$
$$r \in [-(2^k - 1)(2^{l_2} - 1), 2^{\epsilon(l_2+k)}]$$

and is denoted $Proof_{LOG+RANGE}(g, h, \alpha, \delta, l_1, l_2, l_N, \epsilon, k)$*.*

The attack works when a group member is able to obtain a license from the group manager in a way that prevents the group manager from revoking the identity of the user. The idea is similar to the attack in the last section in that the proof is accepted even though the attacker does not possess the knowledge claimed. This is achieved in the following steps:

1. During the license generating protocol, the group member obtains a license (X, δ) by sending various parameters and two proofs to the group manager. One of these proofs contains identifying information for the user using two strings $ID_{U_i} = g^{x_i}$ and $id_{U_i} = \alpha^{x_i}$. The required proof is as follows.

$$Proof_{LOG+RANGE}(g, ID_{U_i}, \alpha, id_{U_i}, l_1, l_2, l_N, \epsilon, k)$$

The malicious user can forge this proof by introducing the low order element ϵ in the following:

$$ID_{U_i} = \epsilon g^{x_i}$$
$$id_{U_i} = \alpha^{x_i}$$

To construct $Proof_{LOG+RANGE}$ the prover chooses $a \in_R \{0,1\}^{\omega(l_2+k)}$, computes

$$c = H(g||ID_{U_i}||\alpha||id_{U_i}||g^a||\alpha^a)$$

and then computes $r = a - c(x_i - 2^{l_1})$. In general, a proof constructed in this way will fail verification by the group manager with a probability of $1/ord(\epsilon)$. Therefore (for 100% success) it is necessary to determine a correct value for c by choosing a and evaluating c until $ord(\epsilon)|c$. This may be performed off-line with minimal computation. When the desired value of c is found the verification $g^a = g^{r-c2^{l_1}}(ID_{U_i})^c$ will be satisfied as required for the proof.

2. During the withdrawal and payment protocols the group member would generate coins using g^{x_i}, rather than ϵg^{x_i}, and these would still pass verification by the merchant. During withdrawal the customer must supply an ElGamal encryption of g^{x_i} using the revocation manager's public key h_R. This is the pair $(g^{x_i} h^z, h_R^z)$ and is included in the coin. This value is checked by the merchant at payment time.
3. When necessary, the revocation manager attempts to reveal the identity using the owner tracing string $ot = h_R^z$ which has been included in the coin. To do this the manager first decrypts the ElGamal ciphertext to obtain g^{x_i}. Then the group manager attempts to determine the identity of the group member's identity $ID_{U_i} = g^{x_i}$ by performing a search on the registered group members. However, the group manager will find no ID'_{U_i} equal to ID_{U_i}, since $ID_{U_i} = \epsilon g^{x_i} \bmod P$.

4.2 An Anomaly in Brands' Cash Scheme

An anomaly in Brands' original electronic cash scheme [5] is now outlined. It is related to the observation of Lim and Lee [19] that a customer is able to double spend coins while avoiding identity revocation, as long as the bank never checks that the customer's identity element is in the correct group during registration[3]. The attack of Lim and Lee was based upon the prover applying her secret key to some supplied element, so the customer must interact with the bank to mount the attack. In contrast we show how the customer can mount an alternative attack by himself, through the selection of blinding invariants. In the initial remark that follows the customer is easily detected; however different assumptions must be made regarding the security of the scheme. Consequently the overall protocol must be altered to accommodate the detection of false coins.

Recall that in Brands' scheme computations take place in a subgroup G of $\mathbb{Z}_p^*$ of prime order q. Three independent generators of G, denoted g, g_1 and g_2, are made public. A coin consists of a pair (A, B) and a signature $\sigma(A, B)$ issued by the bank. The signature is a 4-tuple (a, b, z, r) which is valid if the two verification equations $g^r = h^{\mathcal{H}(A,B,z,a,b,))}a$ and $A^r = z^{\mathcal{H}(A,B,z,a,b,))}b$ where $h(= g^x)$ is the bank's public key. During the withdrawal phase the customer obtains (A, B) blindly in such a way that he knows the representation of (A, B) with respect to (g_1, g_2). At payment time the customer supplies the coin and engages in an interactive protocol designed to prove that he knows this representation.

In the 'attack', the customer substitutes A with $\epsilon A \bmod p$ during the withdrawal where, as before, ϵ is an element of low order t, such that $t|p-1$. The attack may be applied to Brands' scheme using the following steps.

1. During coin withdrawal the customer chooses blinding invariant $s \in_R \mathbb{Z}_q$ as normal and chooses ϵ such that $ord(\epsilon) = t$.
2. The customer computes $A = (g_1^{\mu} g_2)^s \bmod p$ as normal.

[3] It seems that this check was proposed by Brands [4] before the publication of Lim and Lee [19].

3. The customer sets $\alpha = \epsilon A \bmod p$ and replaces A by α in the calculation of $c' = \mathcal{H}(\alpha, B, z', a', b')$.
4. At payment time the customer presents (α, B) and the bank's signature $\sigma(\alpha, B)$ to the merchant as a valid coin.

The customer will obtain a valid bank signature $\sigma(\alpha, B)$ in the case that $\alpha^{r'} = A^{r'}$, where r' is the exponent calculated during withdrawal. Since this element is chosen randomly this will happen with probability $1/t$. In practice the customer could examine the response to confirm whether merchant verification of the false coin will pass with the derived response r', thus guaranteeing that verification will succeed during payment. In addition, the representation check will pass with probability $1/t$ since d will also be of the required value $1/t$ of the time.

When the merchant deposits the coin, the bank is equally likely to accept the coin as valid. If the coin fails the representation check, then the coin is rejected. It is interesting to note that in Brands original paper he suggests that the customer may in fact determine d, reducing the payment to a single move protocol. In this case the customer is able to spend an invalid coin with a 100% success rate; as suggested above during the withdrawal protocol the customer is able to check that an appropriate value r is provided that will enable a false coin to pass verification using $r' = ru + v \bmod q$.

Regardless of this, the question remains of any real exposure. The answer is negative, since if the customer spends a second instance of α, or even A itself, he must supply r values during payment that pass the representation criterion. Since these values encode the customer's true identity the customer will be caught. So this observation is rather an anomaly of the scheme, where an invalid coin (α, B), such that $\alpha \notin G$ and $B \in G$, may be in circulation. But to be able to spend these invalid coins the customer must have taken part in the withdrawal protocol. Since the customer's account is debited during withdrawal and double spending will reveal the identity there is no incentive to fool the system.

It should be pointed out that this observation invalidates theorems presented by Brands [5]. For it can be seen that a customer can present a coin $(\alpha, B), \sigma(\alpha, B)$, and not know the representation of α with respect to (g_1, g_2). The following formal statements are made [5] with reference to a customer U.

Lemma. *If U in the payment protocol can give correct response with respect to two different challenges, then he knows a representation of both A and B with respect to (g_1, g_2).*

Corollary. *U can spend a coin if and only if he knows a representation of it.*

From the results above it is clear that the lemma and corollary have been invalidated. The customer cannot possibly know a representation of α with respect to (g_1, g_2), since $\alpha \notin G$ and (g_1, g_2) are both generators of G.

In order to prevent the customer from spending invalid coins, the merchant should check that the value A lies in the group G. This is an added expense in performing the check $A^q \bmod p = 1$. Due to the existence of the element of order

2, even when p is a safe prime, it will always be necessary to allow that either (A, B) or $(-A, B)$ is a valid coin.

We now show how other schemes, such as fair electronic cash [16], that extend the Brands protocols are susceptible to some attacks. This reinforces the observation that, whilst the original protocol may exhibit no real exposure, extended versions may be compromised. This viewpoint is in contrast to the suggestion by Brands [3] that if the secret key is not applied by the prover to base numbers supplied by other parties then the issue may be avoided [3][4]. Whilst this position is reasonable for the original form of the protocol [5], we suggest that a more prudent approach is to ensure safe parameter selection in the first instance to ensure that extensions remain secure.

4.3 Fair Off-Line Cash

Frankel, Tsiounis and Yung [16] show how Brands' scheme may be extended into a fair electronic cash protocol by providing owner tracing and coin tracing. The subgroup exposure shown above can be modified so that the additional property of owner tracing can be thwarted in their scheme. Once more, this exposure is due to the selection of primes p and q such that $p = \gamma q + 1$, where γ is a small integer [16].

We first note that the customer may again spend invalid coins as in Brands' scheme. The only difference between the withdrawal in Brands' scheme and that of Frankel *et al.* is that the B value is split into two parts, B_1 and B_2, with $B = B_1 B_2$. The customer performs the following modified steps in order to withdraw the false coin.

1. Choose blinding invariant $s \in_R \mathbb{Z}_q$, and ϵ where $ord(\epsilon) = t$.
2. Compute $A = (g_1^{\mu} g_2)^s \bmod p$.
3. Set $\alpha = \epsilon A \bmod p$, $B_1 = g_1^{x_1}$ and $B_2 = g_2^{x_2}$.
4. Calculate $c' = h(\alpha, B, z', a', b')$, where $B = [B_1, B_2]$.

When the customer spends a coin the following modified payment ensues.

1. $A_1 = g_1^{\mu s}, A_2 = g_2^s$.
2. Set $\alpha_1 = \epsilon A_1$.
3. Present $\alpha_1, A_2, A, B_1, B_2$ and $\sigma(\alpha, B)$ to the merchant as a valid coin.

When the merchant validates the coin he will find that $\alpha = \alpha_1 A_2$, and will also find the signature will be valid (since this may be checked at withdrawal time by the customer to ensure that the derived value r is a multiple of order of ϵ). As in the attack on Brands' scheme, the representation check will work with probability $1/t$.

To provide owner tracing, Frankel *et al.* added messages in the payment protocol. These messages assume the existence of a trustee T whose public key

[4] Moreover, the attacks we outline are based upon the use of blinding invariants by the customer.

$f_2 = g_2^{X_T}$ is known to all concerned parties. The additional messages include an ElGamal encryption of the identity of the customer using f_2 and a proof by the customer that this is the same identity as that hidden in the coin in the component A. We now show how the above attack may be extended to thwart the additional property of owner tracing. The customer proceeds as follows.

1. Forms the ElGamal encryption of I as $D_1 = If_2^m$, $D_2 = g_2^m$ where $m \in_R \mathbb{Z}_q$.
2. Sets $\delta_1 = \epsilon D_1$ where ϵ is of small order t.
3. Presents δ_1, D_2 to the merchant as an encrypted identity during payment.

To verify the encryption, the merchant selects $s_0, s_1, s_2 \in_R \mathbb{Z}_q$, and computes $D' = \delta_1^{s_0} g_2^{s_1} D_2^{s_2}$ and $f_2' = f_2^{s_0} g_2^{s_2}$ and sends these two values to the customer. The value $V = \mathcal{H}_1((D')^s/(f_2')^{ms})$ is returned by the customer and the merchant checks that $V = \mathcal{H}_1(\alpha_1^{s_0} A_2^{s_1})$. This verification will succeed as long as $t|s_0$ which occurs with probability $1/t$.

It should be noted that Frankel *et al.* [16] assume that $D' \in G_q$ and $f_2' \in G_q$, hence their proofs are correct from this standpoint. However, as shown above, in practice one cannot necessarily assume that this is the case without the appropriate checks in place. If follows that when the trustee attempts to decrypt the ciphertext, the trustee is unable to obtain the true identity of the customer.

$$\frac{\delta_1}{D_2^{X_T}} = \epsilon I \neq I \bmod p$$

5 Methods to Avoid Exposures

In this section we briefly consider some techniques that may be employed to avoid the attacks described.

Testing each element. Explicitly test that each presented element is within the group. In some cases this will be effective, but when working in the large multiplicative group $\mathbb{Z}_P^*$, this will not protect schemes. Moreover, if $P - 1$ contains many factors then a test for elements within the group will not overcome any problems. For example, this approach is ineffective against the collusion attack in Section 3.2 because a membership check on a supplied element a, using the group order n will yield a positive result.

Restricting Exponents. Another approach is to avoid exponents which are multiples of the factors of the group order. This ensures that any element outside the correct group G will remain outside after exponentiation. The exponent must be prime with respect to all factors of the large group. One may use prime exponents, but this greatly restricts the range available for randomised exponents.

Safe Primes for Multiplicative Groups. The most appropriate solution is to ensure that factors do not exist. If p is a safe prime then the multiplicative group $\mathbb{Z}_p^*$ has no subgroups, except the subgroup of order 2. Thus use of safe primes can avoid these attacks except that it may be necessary to modify the protocols to check for the factors ± 1.

In the schemes of Camenisch and Stadler [9] and the scheme of Traoré [24] the prime P must be chosen in such a way that $(P-1)/n$ has as few small factors as possible. In the best case $P-1=2n$; primes of this form can be generated by first generating n and checking if $2n+1$ is prime.

Proving Composite is the Product of Two Safe Primes. When using a composite modulus, the number of subgroups is minimised by choosing the modulus to be a product of safe primes. Camenisch and Michels [10] have provided a protocol to prove that an RSA modulus is the product of two safe primes. This protocol is not as efficient as would be desirable if it has to be checked at each use of the modulus but can be checked by a certification authority in a one time registration procedure. This can then prevent the collusion attack presented in Section 3.2.

6 Conclusions and Summary

There exist a number of protocols that operate in some subgroup of $\mathbb{Z}_p^*$, where the order is prime or non-prime, with no extra restrictions on p. We have shown that it is possible to attack some of these systems, compromising one or more properties. We show that more elaborate attacks may be mounted that operate in various known and unknown group structures, exploiting weakneses either within the subgroup or within the larger group of the scheme. It is also shown that whilst these schemes may possess no true weakness in their original form, as these protocols are extended (perhaps with additional properties) the potential for exposure is also extended. We suggest that using safe primes together with a proof that this is the case is a sensible precaution even if the specific requirement is not apparent. Finally, it may be necessary that the protocol be modified to cater for elements of order 2.

Acknowledgments

We would like to thank Jacques Traoré for many discussions and helpful remarks regarding the attacks presented.

References

1. Giuseppe Ateniese and Gene Tsudik, Some Open Issues and New Directions in Group Signatures, Financial Cryptography'99, Springer-Verlag, 1999.
2. F. Boudot, On the Soundness of Girault's Scheme. Poster paper at Eurocrypt 2000 Rump Session, 2000.
3. S. Brands, Rethinking public key infrastructures and digital certificates - building in privacy. PhD Dissertation, Ponsen & Looijen BV Publishing, September 1999.
4. S. Brands CS-R9323, Technical Report, 1993.
5. S. Brands. Untraceable Off-line Cash in Wallet with Observers. Advances in Cryptology - Crypto '93, LNCS, Springer-Verlag, Vol. 773, pp302-318, 1993.

6. M. Burmester, A Remark on the Efficiency of Identification Schemes, Advances in Cryptology – Eurocrypt'90, pp.493-495, Springer-Verlag, 1991.
7. Jan Camenisch and Markus Michels, A Group Signature Scheme with Improved Efficiency, Advances in Cryptology – Asiacrypt'98, pp.160-174, Springer-Verlag, 1998. (Revised version available at `www.brics.dk/RS/98/27`).
8. D. Chaum and E. van Heyst, Group Signatures, Advances in Cryptology – Eurocrypt'91, Springer-Verlag, 1991, pp.257-265.
9. J. Camenisch and M. Stadler, Efficient Group Signature Schemes for Large Groups, Advances in Cryptology – Crypto'97, Springer-Verlag, 1997, pp.410-424.
10. Jan Camenisch and Markus Michels, Proving in Zero-Knowledge that a Number is the Product of Two Safe Primes, Advances in Cryptology - EUROCRYPT '99, pages 106-121, Springer Verlag, 1999.
11. D. Chaum, Blind signature system, Advances in Cryptology: Proceedings of Crypto '83, pp. 153–156, Plenum Publishing, 1984.
12. D. Chaum and T. Pedersen. Wallet databases with observers. Crypto '92, pp89-105, 1992.
13. R. Cramer, R. Gennaro and B. Schoenmakers, A Secure and Optimally Efficient Multi-Authority Election Scheme, Advances in Cryptology - Eurocrypt '97, pp. 103–118, Springer-Verlag, 1997.
14. W. Diffie and M. Hellman, New Directions in Cryptography, IEEE Transactions on Information Theory, 1976.
15. A. Fiat and A.Shamir, How to prove yourself: practical solutions to identification and signature problems, Advances in Cryptology - Crypto '86, pp. 186–194, Springer-Verlag, 1986.
16. Y. Frankel, Y. Tsiounis, and M. Yung. Indirect Discourse Proofs: Achieving Efficient Fair Off-Line E-Cash, Asiacrypt '96, Springer-Verlag, pp.286-300, 1996.
17. Marc Girault, An identity-based identification scheme based on discrete logarithms modulo a composite number", Advances in Cryptology - Eurocrypt '90, pp. 481–486, Springer-Verlag, 1990.
18. J. van de Graaf and R. Peralta, A Simple and Secure Way to Show the Validity of Your Public Key, Advances in Cryptology – Crypto'87, pp.128-134, Springer-Verlag, 1987.
19. C.H. Lim and P.J. Lee, A key recovery attack on discrete log-based schemes using a prime order subgroup, Burton S. Kaliski Jr. (Ed.), Advances in Cryptology - CRYPTO '97, LNCS, Springer-Verlag, Vol. 1294, pp.249-263, 1997.
20. S.C. Pohlig and M.E. Hellman. An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance, IEEE Transactions on Information Theory, IT-24(1), pp.106-110, 1978.
21. R. Rivest, A. Shamir and L. Adleman, A Method for Obtaining Digital Signatures and Public Key Cryptosystems, Communications of the ACM, 21, pp.120-126, 1978.
22. C.P. Schnorr, Efficient Signature Generation for Smart Cards, Advances in Cryptology - Crypto '89, Springer-Verlag, pp.239-252, 1990.
23. Markus Stadler, Publicly Verifiable Secret Sharing, Advances in Cryptology – Eurocrypt'96, pp.190-199, Springer-Verlag, 1996.
24. Jacques Traoré, Group Signatures and Their Relevance to Privacy-Protecting Off-Line Electronic Cash Systems, Information Security and Privacy, ACISP'99, Springer-Verlag, 1999, pp.228-243.
25. Jacques Traoré, personal communication.

On Classifying Conference Key Distribution Protocols

Shahrokh Saeednia[1], Rei Safavi-Naini[2], and Willy Susilo[2]

[1] Universite Libré de Bruxelles, Département d'Informatique
Campus Plaine - CP 212, Boulevard du Triomphe
1050 Bruxelles, Belgium
saeednia@ulb.ac.be

[2] Centre for Computer Security Research
School of Information Technology and Computer Science
University of Wollongong
Wollongong 2522, Australia
{rei, wsusilo}@uow.edu.au

Abstract. In this paper, we examine a classification of conference key distribution protocols proposed in [4] and show that no known protocol satisfies the security requirements in class 4, the highest security class of this classification. We show two new attacks on protocols that were believed to belong to the highest security class and show that both protocols in fact belong to class 3. This motivates us to propose a refinement of this classification to allow separating protocols with different security properties while maintaining the classification framework.

1 Introduction

A conference key distribution protocol (CKDP) establishes a common key among a number of users forming a conference. Security analysis of CKDP raises security issues that did not exist in two-party key distribution protocols (KDP)[4] . In particular, in addition to attacks from outsiders that must be considered in both types of protocols, in CKDP there are attacks from malicious insiders with the aim of changing the structure of the conference. For example, a subgroup of participants may attempt to share different keys with different participants, eliminate a participant U_i from the conference or even make other participants believe that U_i is participating in the conference, while the latter is unaware of the conference.

In [4], Saeednia and Safavi-Naini proposed a general framework for defining and classifying security properties of CKDPs. The highest security class in this classification is class 4 security, and requires assurance that no collusion of malicious insiders can break authenticity of the conference key without other participants (outside the coalition) detecting the fraud.

In this paper we show that this requirement is too strong and is not satisfied by any of the examined protocol. We also point out that it is unlikely that the requirement be satisfiable by any other protocol. A result of this is that a broad range of protocols with varying levels of security are put in the same class. In particular, we examine two protocols proposed in [4] and show that none of them achieve the claimed level of security and must be put in the same lower class. We propose a modification of class 4 security that provides a more refined classification of CKDP.

2 Overview of the Proposed Classification

In this section we briefly recall the security classification proposed in [4]. The basic approach is to define properties that must be satisfied by a CKDP, and classify protocols depending on the properties that they satisfy when different types of adversaries are considered. We omit the details of the lower level classes since they are not relevant to this paper.

A conference $\mathcal{C}$ is defined by a subset $\{U_1, \ldots, U_m\}$ of participants from a set $\mathcal{U}$. Participants in $\mathcal{C}$ are called *insiders* while those in $\mathcal{U} \backslash \mathcal{C}$ are called *outsiders*. A CKDP may satisfy one or more of the following properties.

A. All insiders must be able to compute the conference key $K_{\mathcal{C}}$.
B. $K_{\mathcal{C}}$ must be fresh.
C. No outsider, having access to messages of previous runs of the protocol and the corresponding keys, can calculate $K_{\mathcal{C}}$ or share a key with each insider.
D. Every insider can be sure that either he is sharing the same key with all the conference participants, or no two participants share a common key.

Properties A to C are all required to ensure that a protocol functions correctly and produces "good keys" in presence of both passive and active outsiders. They constitute the properties that are shared between KDPs and CKDPs. Property D, however, is only relevant to CKDP and not to two-party protocols. This is true because in these protocols there are exactly two participants and property D reduces to assurance about the sameness of the key computed by the other participant that is always achievable by an extra handshake protocol using the distributed key. This means that KDPs that satisfy properties A to C guarantee confidentiality of the information which in the extreme case could mean messages encrypted by one participant not being readable by anyone else, including other valid recipients.

In a CKDP the situation is different. A protocol that satisfies A to C cannot be considered secure because the protocol might result in subgroups of participants to share a common key different from the conference key. In this case, an encrypted message is readable by one subset without them knowing who is able/unable to read the message and so the protocol cannot be considered secure. This means that D is an essential property of a secure CKDP.

In general we can consider two kinds of adversaries: outsiders and insiders. It is important to distinguish between the two types as security risks in each case is different and a protocol that is secure against attackers of the first kind may not be secure against attackers of the second kind.

Because of this, property D is split into two, resulting in two different classes of security:

D1. It is infeasible for an active outsider to break authenticity of the conference key by tampering with the messages without insiders detecting the fraud.

D2. It is infeasible for any coalition of malicious insiders to break authenticity of the conference key by tampering with the messages with no insider outside the coalition detecting the fraud.

Now security classes 3 and 4 are defined as the classes containing protocols satisfying A, B, C and D1, and protocols satisfying A, B, C, D1 and D2, respectively.

In the following, we show that because of property D2, we cannot distinguish between a protocol in which a group of malicious insiders can force one, or a subgroup of, participants to calculate a different conference key while no other participant, who is neither a colluder nor a victim, detect it, and a protocol in which a group of insiders can establish a subliminal channel [5,2] between themselves. Both these attacks are captured by D2, while they actually have very different implications.

3 Identity-Based Protocols and Attacks

In [4], two identity-based CKDPs have been proposed that were based on the broadcast protocol of [1] and were claimed to be of class 4. In this section we recall these protocols and show attacks that allow a subliminal key other the conference key to be computed by a subgroup of participants.

3.1 Protocol 1

In this protocol, users' keys are chosen by a Trusted Third Party (TTP) that in the initialization phase chooses

- an integer n that is a product of two large distinct random primes p and q such that $p-1 = 2p'$ and $q-1 = 2q'$, where p' and q' are also prime integers,
- two bases α and $\beta \neq 1$ of order $r = p'q'$;
- a large integer $u < \min(p', q')$, and
- a one-way hash function f.

TTP makes α, β, u, f and n public, keeps r secret and discards p and q afterward.

Each user, after his identity is verified by the TTP, receives a pair of public and private keys. The TTP does the following:

- prepares the user's public key, ID, by hashing the string I corresponding to his identity: that is, $ID = f(I)$;
- computes the user's secret key as the pair (x, y) and $x = \alpha^{ID^{-1}} \pmod{n}$, $y = \beta^{-ID^{-1}} \pmod{n}$ and ID^{-1} is computed modulo r.

The protocol is executed in three steps and has two broadcasts by each user.

1. Each U_i, $i = 1, \dots, m$, randomly selects $t_i \in_R Z_u$, computes $z_i = \alpha^{t_i} \pmod{n}$ and broadcasts it.
2. Each U_i, $i = 1, \dots, m$, computes $c = f(z_1 || z_2 || \dots || z_m)$ ("$||$" denotes the concatenation), and then broadcasts $v_i = (z_{i+1}/z_{i-1})^{t_i} \pmod{n}$ and $w_i = y_i^c \cdot x_i^{f(v_i)t_i} \pmod{n}$.
3. Each U_i, $i = 1, \dots, m$, checks whether $w_j^{ID_j} \cdot \beta^c \overset{?}{=} z_j^{f(v_j)} \pmod{n}$; $j = 1, \dots, i-1, i+1, \dots, m$. If so, computes the conference key as

$$K_i = z_{i-1}^{mt_i} \cdot v_i^{(m-1)} \dots v_{i-3}^2 \cdot v_{i-2} \pmod{n}.$$

The common key computed by U_i is

$$K_{\mathcal{C}} = \alpha^{t_1 t_2 + t_2 t_3 + \dots + t_m t_1} \pmod{n}.$$

The pair (z_i, w_i) constitutes a signature of v_i and a witness of the knowledge of t_i, as well as U_i's secret key. Since with a very high probability c is different in each runs of the protocol, the freshness of the signatures can be guaranteed. This means that no signature (for the same v_i and z_i) will be valid twice. Such signature provides assurance about the origin of v_i and z_i and makes it infeasible to impersonate a user by replaying message of a previous session, or eliminate a user from the conference while other participants believe that the user has computed the same key as them. However, it is possible to establish a subliminal key, as we will see in the following subsection.

3.2 Attack on Protocol 1

We show how a colluding group of participants may share a key other than the conference key computed by everyone, without others detecting it. For simplicity, we describe the attack by $m-1$ colluders against U_j. It is easy to modify the attack to work for $m-k$ colluders against the remaining k participants. We assume that attackers can completely control the view of the cheated participant. This means that, it is possible to broadcast a message in such a way that U_j cannot receive it. This is denoted by *broadcast*$(\backslash U_j)$. Also, no participant can distinguish between a received broadcasted message and a message sent only to him.

1. Each U_i, $i = 1, \dots, m$, selects $t_i \in_R Z_u$, computes $z_i = \alpha^{t_i} \pmod{n}$ and broadcasts it.
2. Each U_i, $i \neq j-1$ and $j+1$, computes $v_i = (z_{i+1}/z_{i-1})^{t_i} \pmod{n}$ and $w_i = y_i^c \cdot x_i^{f(v_i)t_i} \pmod{n}$ and broadcasts them.

2'. U_{j-1} and U_{j+1} send to U_j, (v_{j-1}, w_{j-1}) and (v_{j+1}, w_{j+1}) that they have computed following step 2 of the protrocol, compute

$$v'_{j-1} = (z_{j+1}/z_{j-2})^{t_{j-1}} \pmod{n}$$

$$v'_{j+1} = (z_{j+2}/z_{j-1})^{t_{j+1}} \pmod{n}$$

and broadcast$(\backslash U_j)$ them (as v_{j-1} and v_{j+1}), respectively. In addition, they broadcast$(\backslash U_j)$ v_{j-1} and v_{j+1} (those they have sent to U_j) as w_{j-1} and w_{j+1}, respectively.

3. U_j checks validity of all w_k's he has received and computes the key $K_{\mathcal{C}} = \alpha^{t_1t_2+t_2t_3+\ldots+t_mt_1} \pmod{n}$. Other U_i, knowing that they are colluding to exclude U_j, do not check the validity of w_k's, but first compute

$$K^* = \alpha^{t_1t_2+\ldots+t_{j-1}t_{j+1}+\ldots+t_mt_1} \pmod{n}$$

using v_k's they have received (excluding v_j), and second

$$K_{\mathcal{C}} = \alpha^{t_1t_2+t_2t_3+\ldots+t_mt_1} \pmod{n}$$

using values broadcasted($\backslash U_j$) by U_{j-1} and U_{j+1} as w_{j-1} and w_{j+1}, respectively.

Now, the colluders can use the key K^* to communicate (excluding U_j) and also use $K_{\mathcal{C}}$ when they need to send a message that should also be readable by U_j.

3.3 Protocol 2

This protocol is the Burmester-Desmedt protocol with key confirmation. It uses the same parameters and keys as protocol 1, but instead of signatures on the pair v_i and t_i, each participant computes a signature on the conference key. The validity of this signature will be verified by all other participants.

1. Each U_i, $i = 1, \ldots, m$, selects $t_i \in_R Z_u$, computes $z_i = \alpha^{t_i} \pmod{n}$ and broadcasts it.
2. Each U_i, $i = 1, \ldots, m$, computes $c = f(z_1||z_2||\ldots||z_m)$ and then computes and broadcasts $v_i = (z_{i+1}/z_{i-1})^{t_i} \pmod{n}$.
3. Each U_i, $i = 1, \ldots, m$, computes the conference key as

$$K_i = z_{i-1}^{mt_i} \cdot v_i^{(m-1)} \ldots v_{i-3}^2 \cdot v_{i-2} \pmod{n}$$

and then computes $k_i = f(K_i)$ and $w_i = y_i \cdot x_i^{t_ik_i} \pmod{n}$, and broadcasts them.
4. Each U_i, $i = 1, \ldots, m$, verifies whether $w_j^{ID_j} \cdot \beta \overset{?}{=} z_j^{k_i} \pmod{n}$, for $j = 1, \ldots, i-1, i+1, \ldots, m$. If they hold, then U_i accepts, otherwise rejects and halts.

3.4 Attack on Protocol 2

Here again, we describe our attack against a given participant U_j, but it can straightforwardly be generalized to exclude any number of participants.

1. U_i, $i = 1, \ldots, m$, selects $t_i \in_R Z_u$, computes $z_i = \alpha^{t_i} \pmod{n}$ and broadcasts it.
2. U_i, $i \neq j-1$ and $j+1$, computes $v_i = (z_{i+1}/z_{i-1})^{t_i} \pmod{n}$ and broadcasts it.

2'. U_{j-1} and U_{j+1} send to U_j, the values v_{j-1} and v_{j+1} that they have computed following step 2 of the protocol, and compute

$$v'_{j-1} = (z_{j+1}/z_{j-2})^{t_{j-1}} \pmod{n}$$

and

$$v'_{j+1} = (z_{j+2}/z_{j-1})^{t_{j+1}} \pmod{n}$$

and broadcast($\backslash U_j$) them (as v_{j-1} and v_{j+1}), respectively.

3. U_j computes the key $K_{\mathcal{C}} = \alpha^{t_1 t_2 + t_2 t_3 + \ldots + t_m t_1} \pmod{n}$. Other U_i's ($i = 1, \ldots, m$, $i \neq j$, $j-1$ and $j+1$) compute

$$K^* = \alpha^{t_1 t_2 + \ldots + t_{j-1} t_{j+1} + \ldots + t_m t_1} \pmod{n}$$

using v_k's they have received (excluding v_j), and U_{j-1} and U_{j+1} compute both keys.

3'. U_{j-1} and U_{j+1} broadcast($\backslash U_j$) $K_{\mathcal{C}}$ (instead of their signature on the conference key) and send to U_j

$$w_{i-1} = y_{i-1} \cdot x_{i-1}^{t_{i-1} K_{\mathcal{C}}} \pmod{n}$$

and

$$w_{i+1} = y_{i+1} \cdot x_{i+1}^{t_{i+1} K_{\mathcal{C}}} \pmod{n}$$

respectively.

3". U_i, $i \neq j-1$ and $j+1$, computes $w_i = y_i \cdot x_i^{t_i K_{\mathcal{C}}} \pmod{n}$ and broadcasts it.

4. U_j verifies the correctness of w_k and is convinced that other participants have computed the same key as him. Other participants do nothing.

Once again, the colluders can use the key K^* to communicate (excluding U_j) and also use K when they need to send a message that should be readable by U_j.

4 Refining the Classification

The above attacks show that none of the protocols in [4] are in class 4 as collusion of malicious insiders can break authenticity of the key. Here, "breaking authenticity of the key" means that property D is not satisfied and so with this requirement none of the analyzed protocols is in class 4.

Establishing subliminal channels might not need an active attack. Rather, it could use correctly constructed messages of the protocol in a different computation planned by the colluders. An example is the case when participants in the protocol send messages of the form α^{t_i} at some stage of the protocol. This allows two participants, U_i and U_j, to establish a Diffie-Hellman key, $\alpha^{t_i t_j}$ [3] and use it to communicate through a subliminal channel[1]. A similar type

[1] In star based and cyclic systems, any participant may learn the value sent by each participant to another by eavesdropping the communication.

of computation can be also performed by a group of participants resulting in a subliminal channel among the group. This type of channel that could always be established among passive colluding insiders, is called *inherent subliminal channel.* To establish such channels the attackers correctly follow the protocol and it is not necessary for them to modify or suppress any of the messages. In this case, since calculating the conference key uses messages of a correct and untampered run of the protocol, the result is a correct conference key. However some of the participants also follow another pre-agreed computation and calculate a subliminal key and there is no way for other participants to be sure that such a computation has not been performed. Subliminal channels are studied by Simmons [5] and Desmedt et al [2]. It is an open question if it is possible to construct conference key protocols that are subliminal free, that is, protocols that do not allow establishment of these kinds of subliminal channels.

This has motivated us to refine property D so that while it can be used to distinguish between the more secure protocols and the less secure ones.

To clarify this, we compare the Burmester-Desmedt protocol with the protocols in sections 3.1 and 3.3. As shown in [4], Burmester-Desmedt protocol is only secure against outsiders' attacks and no security is provided against malicious insiders: malicious insiders can impersonate other users, or eliminate a number of them from the conference without any other participant that is not part of the colluding group or the victim group, being able to detect the subversion of the protocol.

The protocols analyzed in sections 3.1 and 3.3 only allow a collusion of insiders to share a subliminal channel (inherent or not), but there is no known attack that leaves out at least one of the participants outside both the colluder group and the victim group. Nevertheless, with the current classification, all these protocols belong to class 3 despite some resistance that they provide against insiders' attacks.

We propose an intermediate class to differentiate between the two types of protection offered. We separate attacks that leaves out at least one participant outside the group of colluders and the group of victims. This is reasonable because when a group of colluders impersonate or eliminate a participant U_j, their goal is to make other participants believe that user U_j is involved in the conference while U_j is either impersonated or has computed a wrong key without being aware of it. The attack divides the conference into three subgroups: colluders, victims and others, where the latter two subgroups are both cheated by the colluders, though not in the same way.

In contrast attacks that aim at creating a subliminal channel break the conference into two subgroups: colluders and the victims with colluders' aim being to have private communication among themselves. So property D2 can be rewritten as follows.

D'2. No colluding subgroup $G \in \mathcal{C}$ can impersonate a subgroup $G' \in \mathcal{C}$ such that $G \cap G' = \emptyset$ and $G \cup G' \neq \mathcal{C}$, or force them to compute a wrong key. Here we require that in this fraud there is at least one participant $U \in \mathcal{C} \setminus \{G \cup G'\}$ who cannot detect this fraud.

D"2. No colluding subgroup $G \in \mathcal{C}$ can share a subliminal key other than the conference key.

We note that an inherent subliminal channel between two participant can be extended to a subliminal channel among a group of participants as follows: if user U_i shares a subliminal key K_1 with U_j and K_2 with U_k, then it can encrypt K_2 with K_1 and send it to U_j. In this way all three users share K_2. This can be extended to any number of users.

We define the new classes as follows.

– Class 3 contains protocols that satisfy A, B, C and D1.
– Class 4 contains protocols that satisfy A, B, C, D1 and D'2.
– Class 5 contains protocols that satisfy A, B, C, D1 and D'2 and D"2.

Now, the above protocols belong to the new class 4, while less secure protocols such as the Burmester-Desmedt remain in class 3.

5 Conclusions and Further Works

We argued that the classification of CKDPs given in [4] is very restrictive. We showed attacks on two CKDPs that were previously believed to belong to the highest security class in this classification. We noted that many conference key distribution protocols have inherent subliminal channels and in fact we are not aware of any CKDP that is free from such channels. We argued that there is no reason to separate protocols based on the type of the attack, active or passive, used for establishment of the channel.

However we separate attacks that aim at establishing a subliminal channel and those that remove the assurance that each user can compute the same conference key with all other participants (although he might be excluded from some subliminal discussions). By replacing class 4 in the classification given in [4] with two new classes, 4 and 5, we can retain the original framework and at the same time differentiate between protocols that have different resistance against insiders' attacks. The construction of a CKDP that satisfy the requirements of class 5 remains an open and challenging problem.

References

1. M. Burmester and Y. Desmedt. A secure and efficient conference key distribution system. *Advances in Cryptology - Eurocrypt '94, Lecture Notes in Computer Science 950*, pages 275–286, 1994.
2. M. Burmester, Y. Desmedt, T. Itoh, K. Sakurai, H. Shizuya, and M. Yung. A progress report on subliminal-free channels. *Workshop on Information Hiding*, 1996.
3. W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. Inform. Theory*, vol. 22, pages 644-654, 1976.

4. S. Saeednia and R. Safavi-Naini. Efficient Identity-Based Conference Key Distribution Protocols. *Information Security and Privacy, ACISP '98, Lecture Notes in Computer Science 1438*, pages 320–331, 1998.
5. G. J. Simmons. The subliminal channel and digital signatures. *Advances in Cryptology - Eurocrypt '84, Lecture Notes in Computer Science 209*, pages 364 – 178, 1984.

Pseudorandomness of MISTY-Type Transformations and the Block Cipher KASUMI

Ju-Sung Kang, Okyeon Yi, Dowon Hong, and Hyunsook Cho

Section 0741, Information Security Technology Division, ETRI
161 Kajong-Dong, Yusong-Gu, Taejon, 305-350, KOREA
{jskang,oyyi,dwhong,hscho}@etri.re.kr

Abstract. We examine the security of block ciphers on the view point of pseudorandomness. Firstly we show that the four round (unbalanced) MISTY-type and the three round dual MISTY-type transformations are pseudorandom permutation ensembles. Secondly we prove that the three round KASUMI is not a pseudorandom permutation ensemble but the four round KASUMI is a pseudorandom permutation ensemble. We provide simplified probability-theoretic proofs for non-adaptive distinguishers.

Key words: Distinguisher, (Super-)Pseudorandom permutation ensemble, MISTY-type transformation, KASUMI.

1 Introduction

The notion of pseudorandomness has applied to a method of analyzing provably the security of block ciphers together with the provable security against differential and linear cryptanalysis. Luby and Rackoff[5] introduced a theoretical model for the security of block ciphers by using the notion of pseudorandom and super-pseudorandom permutations.

A block cipher can be regarded as a family of permutations on a message space indexed by a secret key. That is, one secret key determines a permutation on the given message space. A pseudorandom permutation can be interpreted as a block cipher that no attacker with polynomially many encryption queries can distinguish between the block cipher and the truly random permutation. We call a block cipher is a super-pseudorandom permutation if no attacker with polynomially many encryption and decryption queries can distinguish between the block cipher and the truly random permutation. Maurer[8] presented a simplified proof of Luby-Rackoff's results for non-adaptive distinguishers and provided new insight into the relation between complexity-theoretic and probability-theoretic results. Naor and Reingold[9] proposed the revised construction by showing that the two round Feistel-type transformation was sufficient together with initial and final independent permutations to be a super-pseudorandom permutation. Iwata and Kurosawa[3] proved that the five round RC6 and the three round Serpent were super-pseudorandom permutations for non-adaptive distinguishers.

In [5], Luby and Rackoff used the Feistel-type transformation of DES in order to construct a pseudorandom and super-pseudorandom permutation from a pseudorandom function. They showed that the Feistel-type transformation with three rounds yielded pseudorandom permutation and with four rounds yielded super-pseudorandom permutation under the assumption that each round function was a pseudorandom function.

In this paper we examine the pseudorandomness of the MISTY-type transformation which is not a Feistel-type and apply this results to analyze the pseudorandomness of the 3GPP block cipher KASUMI. Sakurai and Zheng[10] showed that the three round MISTY-type transformation is not a pseudorandom permutation ensemble. We prove that the four round (unbalanced) MISTY-type and the three round dual MISTY-type transformations are pseudorandom permutation ensembles. The overall structure of KASUMI is a Feistel-type structure, but its round function doesn't seem to be a pseudorandom function. Thus we cannot straightforwardly apply the Luby-Rackoff's result to KASUMI. We prove that the three round KASUMI is not a pseudorandom permutation but the four round KASUMI is a pseudorandom permutation. Through out this paper we use the simplified probability-theoretic proofs for non-adaptive distinguishers.

Recently, Iwata et al.[4] proved that the five round MISTY-type transformation was super-pseudorandom and Gilbert and Minier[2] showed that the four round MISTY-type and three round dual MISTY-type transformations were pseudorandom and the five round MISTY and dual MISTY-type transformations were super-pseudorandom. These two results included some parts of ours, however ours is obtained found independently and our results about the pseudorandomness of the 3GPP block cipher KASUMI is new.

2 Preliminaries

2.1 Definitions

Let I_m denote the set of all m-bit strings and Ω_m be the set of all permutations from I_m to itself where m is a positive integer. That is,

$$\Omega_m = \{\pi : I_m \to I_m \mid \pi \text{ is a bijection}\} .$$

Definition 1 *Ω_m is called a TPE(truly random permutation ensemble) if all permutations in Ω_m are uniformly distributed. That is, for any permutation $\pi \in \Omega_m$, $Pr(\pi) = \frac{1}{2^m!}$.*

We consider the following security model. Let $\mathcal{D}$ be computationally unbounded distinguisher with an oracle $\mathcal{O}$. The oracle $\mathcal{O}$ chooses randomly a permutation π from Ω_m or from a permutation ensemble $\Psi_m \subset \Omega_m$. For an m-bit block cipher, Ψ_m is the set of permutations obtained from all the secret keys. The purpose of the distinguisher $\mathcal{D}$ is to distinguish whether the oracle $\mathcal{O}$ implements Ω_m or Ψ_m.

Definition 2 *Let $\mathcal{D}$ be a distinguisher, Ω_m be a TPE, and $\Psi_m(\subset \Omega_m)$ be a permutation ensemble. The advantage $Adv_{\mathcal{D}}$ of the distinguisher $\mathcal{D}$ is defined by*

$$Adv_{\mathcal{D}} = \left| p^{\Omega_m} - p^{\Psi_m} \right| ,$$

where

$$p^{\Omega_m} = Pr(\mathcal{D} \text{ outputs } 1 \mid \mathcal{O} \leftarrow \Omega_m)$$

and

$$p^{\Psi_m} = Pr(\mathcal{D} \text{ outputs } 1 \mid \mathcal{O} \leftarrow \Psi_m) .$$

Assume that the distinguisher $\mathcal{D}$ is restricted to make at most $poly(m)$ queries to the oracle $\mathcal{O}$, where $poly(m)$ is some polynomial in m. We call $\mathcal{D}$ is a pseudorandom distinguisher if it queries x and the oracle answers $y = \pi(x)$, where π is a randomly chosen permutation by $\mathcal{O}$. We say that $\mathcal{D}$ is a super-pseudorandom distinguisher if it is a pseudorandom distinguisher and also allowed to query y and receives $x = \pi^{-1}(y)$ from the oracle $\mathcal{O}$.

Definition 3 *A function $h : N \rightarrow R$ is negligible if for any constant $c > 0$ and all sufficiently large $m \in N$,*

$$h(m) < \frac{1}{m^c} .$$

Definition 4 *Let Ψ_m be an efficiently computable permutation ensemble. We call Ψ_m is a PPE(pseudorandom permutation ensemble) if $Adv_{\mathcal{D}}$ is negligible for any pseudorandom distinguisher $\mathcal{D}$.*

Definition 5 *Let Ψ_m be an efficiently computable permutation ensemble. We call Ψ_m is a SPPE(super-pseudorandom permutation ensemble) if $Adv_{\mathcal{D}}$ is negligible for any super-pseudorandom distinguisher $\mathcal{D}$.*

In Definition 4 and 5, a permutation ensemble is efficiently computable if all permutations in the ensemble can be computed efficiently. See [9] for the rigorous definition of this. It is reasonable assumption that Ψ_m is an efficiently computable permutation ensemble if it is obtained from an m-bit block cipher. Hence we assume that any permutation ensemble obtained from a block cipher is efficiently computable. Throughout this paper, we consider a non-adaptive distinguisher which sends all the queries to the oracle at the same time.

2.2 Some Basic Lemmas

Before we proceed to the main results, we state simple but useful lemmas.

Lemma 1. *Let π be a permutation chosen in a TPE Ω_m. Then for any $x, y \in I_m$, $Pr(\pi(x) = y) = \frac{1}{2^m}$.*

Proof. The assertion is straightforward since

$$Pr(\pi(x) = y) = \frac{\#\{\pi \in \Omega_m \mid \pi(x) = y\}}{\#(\Omega_m)} = \frac{(2^m - 1)!}{2^m!} = \frac{1}{2^m} \,.$$

□

Lemma 2. *Let π_1 and π_2 be two permutations independently chosen from a TPE Ω_m. Then for any $x_1, x_2, y \in I_m$,*

$$Pr\left(\pi_1(x_1) \oplus \pi_2(x_2) = y\right) = \frac{1}{2^m} \,,$$

where $\oplus$ denote the bitwise exclusive-or.

Proof. Let Γ be the event of $\pi_1(x_1) \oplus \pi_2(x_2) = y$ and A_i be the event of $\pi_1(x_1) = w_i$ for $1 \le i \le 2^m$, where $I_m = \{w_1, \cdots, w_{2^m}\}$. Then $\Omega_m = \cup_{i=1}^{2^m} A_i$ is a disjoint union and

$$\begin{aligned} Pr(\Gamma \cap A_i) &= Pr(\pi_1(x_1) \oplus \pi_2(x_2) = y,\ \pi_1(x_1) = w_i) \\ &= Pr(\pi_1(x_1) = w_i) \cdot Pr(\pi_2(x_2) = y \oplus w_i) \,, \end{aligned}$$

since π_1 and π_2 are independently chosen. Hence by Lemma 1, we obtain that

$$\begin{aligned} Pr(\Gamma) &= \sum_{i=1}^{2^m} Pr(\Gamma \cap A_i) \\ &= 2^m \cdot \left(\frac{1}{2^m}\right)^2 = \frac{1}{2^m} \,. \end{aligned}$$

□

Lemma 3. *Let π be a permutation chosen from a TPE Ω_m. Then for any $x_1 \neq x_2, y \in I_m$,*

$$Pr(\pi(x_1) \oplus \pi(x_2) = y) = \begin{cases} \frac{1}{2^m - 1} & \text{if } y \neq 0 \,, \\ 0 & \text{otherwise.} \end{cases}$$

Proof. Let Γ be the event of $\pi(x_1) \oplus \pi(x_2) = y$ and A_i be the event of $\pi(x_1) = w_i$ for $1 \le i \le 2^m$, where $I_m = \{w_1, \cdots, w_{2^m}\}$. If $y = 0$, $Pr(\Gamma) = 0$ since $x_1 \neq x_2$ and π is a bijection. So we consider the case of $y \neq 0$. Observe that

$$\begin{aligned} Pr(\Gamma \cap A_i) &= Pr(\pi(x_1) \oplus \pi(x_2) = y,\ \pi(x_1) = w_i) \\ &= Pr(\pi(x_1) = w_i,\ \pi(x_2) = y \oplus w_i) \\ &= \frac{(2^m - 2)!}{2^m!} = \frac{1}{2^m(2^m - 1)} \,. \end{aligned}$$

Thus if $y \neq 0$, we obtain that

$$Pr(\Gamma) = \sum_{i=1}^{2^m} Pr(\Gamma \cap A_i) = 2^m \cdot \frac{1}{2^m(2^m - 1)} = \frac{1}{2^m - 1} \,.$$

□

Lemma 4. *Let π_1 and π_2 be two permutations independently chosen from a TPE Ω_m. Then for any $a, b, c, d, y \in I_m$, such that $a \neq b$ and $c \neq d$,*

$$Pr\left(\pi_1(a) \oplus \pi_1(b) \oplus \pi_2(c) \oplus \pi_2(d) = y\right) < \frac{1}{2^{m-1}} \ , \ \textit{for } m \geq 2.$$

Proof. Let Γ be the event of $\pi_1(a) \oplus \pi_1(b) \oplus \pi_2(c) \oplus \pi_2(d) = y$ and A_j be the event of $\pi_1(a) \oplus \pi_1(b) = w_j$ for $1 \leq j \leq 2^m$, where $I_m = \{w_1, \cdots, w_{2^m}\}$. Then by Lemma 3, we obtain that

$$\begin{aligned} Pr(\Gamma \cap A_j) &= Pr(\pi_1(a) \oplus \pi_1(b) = w_j) \cdot Pr(\pi_2(c) \oplus \pi_2(d) = y \oplus w_j) \\ &\leq \left(\frac{1}{2^m - 1}\right)^2 . \end{aligned}$$

Therefore

$$\begin{aligned} Pr(\Gamma) &= \sum_{j=1}^{2^m} Pr(\Gamma \cap A_j) \\ &\leq 2^m \cdot \left(\frac{1}{2^m - 1}\right)^2 < \frac{1}{2^{m-1}} \ , \ \text{for } m \geq 2. \end{aligned}$$

□

3 Pseudorandomness of the MISTY-Type Transformation

Matsui[6] introduced another structure of block ciphers with provable security against differential and linear cryptanalysis which was different from Feistel-type. This structure was applied to the block cipher MISTY[7] and KASUMI[1] later, so we call this as MISTY-type transformation. In this section we examine the pseudorandomness of the MISTY-type transformation.

Definition 6 *For any n-bit permutation $f \in \Omega_n$, $2n$-bit MISTY-type permutation $\mathbf{M}_f \in \Omega_{2n}$ is defined by*

$$\mathbf{M}_f(L, R) = (R, f(L) \oplus R) \ , \quad \textit{where } L, R \in I_n \ .$$

Definition 7 *For any n-bit permutation $g \in \Omega_n$, $2n$-bit dual MISTY-type transformation $\mathbf{DM}_g \in \Omega_{2n}$ is defined by*

$$\mathbf{DM}_g(L, R) = (g(L \oplus R), L) \ , \quad \textit{where } L, R \in I_n \ .$$

Note that $\mathbf{DM}_{f^{-1}}$ is the inverse permutation of $\mathbf{M}_f$. Sakurai and Zheng[10] showed that $\mathbf{M}_{f_3} \circ \mathbf{M}_{f_2} \circ \mathbf{M}_{f_1}$ was not a $2n$-bit PPE though each f_i is n-bit PPE. We show that $\mathbf{M}_{f_4} \circ \mathbf{M}_{f_3} \circ \mathbf{M}_{f_2} \circ \mathbf{M}_{f_1}$ is a $2n$-bit PPE under the assumption that each $f_i (i = 1, 2, 3, 4)$ is an n-bit PPE.

Theorem 1 *If f_1, f_2, f_3, and f_4 are independently chosen from an n-bit PPE, then the four round MISTY-type transformation $\mathbf{M}_{f_4} \circ \mathbf{M}_{f_3} \circ \mathbf{M}_{f_2} \circ \mathbf{M}_{f_1}$ is a $2n$-bit PPE.*

Proof. Without loss of generality, we assume that f_i's are independently chosen from the TPE Ω_n. Let Ψ_{2n} be the set of all permutations over I_{2n} obtained from $\mathbf{M}_{f_4} \circ \mathbf{M}_{f_3} \circ \mathbf{M}_{f_2} \circ \mathbf{M}_{f_1}$ and the i-th round output of this permutation is defined by (L_i, R_i) for $i = 1, 2, 3, 4$ where (L, R) is the $2n$-bit input. That is, $(L_i, R_i) = (\mathbf{M}_{f_i} \circ \cdots \circ \mathbf{M}_{f_1})(L, R)$.

We assume that the distinguisher $\mathcal{D}$ makes t calls to the oracle $\mathcal{O}$. In the i-th oracle call, $\mathcal{D}$ sends a query $(L^{(i)}, R^{(i)})$ to $\mathcal{O}$ and receives the corresponding output $(Y_L^{(i)}, Y_R^{(i)}) = \pi(L^{(i)}, R^{(i)})$, where π is the randomly chosen permutation by $\mathcal{O}$ from Ω_{2n} or Ψ_{2n}.

Let A_L denote the event that $L_2^{(1)}, \cdots, L_2^{(t)}$ are all distinct and A_R denote the event that $R_2^{(1)}, \cdots, R_2^{(t)}$ are all distinct. If A_L occurs, then we can see that $L_4^{(1)}, \cdots, L_4^{(t)}$ are completely random since $L_4^{(i)} = f_3(L_2^{(i)}) \oplus R_2^{(i)}$ and the outputs of f_3 are completely random. Similarly, if A_R occurs, then $R_4^{(1)}, \cdots, R_4^{(t)}$ are completely random. Therefore, if A_L and A_R occur, then $Adv_{\mathcal{D}}$ is bounded above as follows:

$$Adv_{\mathcal{D}} \le 1 - Pr(A_L \cap A_R) \le \sum_{1\le i<j\le t} Pr(L_2^{(i)} = L_2^{(j)}) + \sum_{1\le i<j\le t} Pr(R_2^{(i)} = R_2^{(j)}) \,.$$

Now we estimate $Pr(L_2^{(i)} = L_2^{(j)})$ and $Pr(R_2^{(i)} = R_2^{(j)})$ for any $1 \le i < j \le t$. Fix $(L^{(i)}, R^{(i)}) \ne (L^{(j)}, R^{(j)})$ arbitrarily. We have the following three cases.

Case 1: $L^{(i)} \ne L^{(j)}$ and $R^{(i)} = R^{(j)} = R_0$. Observe that $L_2^{(i)} = f_1(L^{(i)}) \oplus R_0$ and $L_2^{(j)} = f_1(L^{(j)}) \oplus R_0$. Then we obtain by Lemma 3 that

$$Pr(L_2^{(i)} = L_2^{(j)}) = Pr(f_1(L^{(i)}) = f_1(L^{(j)})) \le \frac{1}{2^n - 1} \,,$$

since f_1 is a truly random permutation. By similar process we also obtain that

$$Pr(R_2^{(i)} = R_2^{(j)}) \le \frac{1}{2^n - 1} \,.$$

Case 2: $L^{(i)} = L^{(j)} = L_0$ and $R^{(i)} \ne R^{(j)}$. In this case it is easy to see that

$$Pr(L_2^{(i)} = L_2^{(j)}) = Pr(R^{(i)} = R^{(j)}) = 0$$

and by Lemma 3,

$$\begin{aligned} Pr(R_2^{(i)} = R_2^{(j)}) &= Pr(f_2(R^{(i)}) \oplus f_2(R^{(j)}) = R^{(i)} \oplus R^{(j)}) \\ &\le \frac{1}{2^n - 1} \,, \end{aligned}$$

since f_2 is a truly random permutation.

Case 3: $L^{(i)} \neq L^{(j)}$ and $R^{(i)} \neq R^{(j)}$. Observe that by Lemma 3,

$$\begin{aligned} Pr(L_2^{(i)} = L_2^{(j)}) &= Pr(f_1(L^{(i)}) \oplus f_1(L^{(j)}) = R^{(i)} \oplus R^{(j)}) \\ &\leq \frac{1}{2^n - 1} \end{aligned}$$

and by Lemma 4,

$$\begin{aligned} Pr(R_2^{(i)} = R_2^{(j)}) &= Pr(f_1(L^{(i)}) \oplus f_1(L^{(j)}) \oplus f_2(R^{(i)}) \oplus f_2(R^{(j)}) = R^{(i)} \oplus R^{(j)}) \\ &< \frac{1}{2^{n-1}} \text{ for } n \geq 2 \ . \end{aligned}$$

Hence, for any case,

$$Pr(L_2^{(i)} = L_2^{(j)}) \leq \frac{1}{2^n - 1} \text{ and } Pr(R_2^{(i)} = R_2^{(j)}) < \frac{1}{2^{n-1}}$$

hold for $n \geq 2$. Therefore we obtain that for all $n \geq 2$,

$$Adv_{\mathcal{D}} < \frac{t(t-1)}{2} \cdot \frac{1}{2^n - 1} + \frac{t(t-1)}{2} \cdot \frac{1}{2^{n-1}} < \frac{t^2 - t}{2^n} \ .$$

Consequently, $Adv_{\mathcal{D}}$ is negligible, since $t = poly(n)$. □

Theorem 2 *The two round dual MISTY-type transformation* $\mathbf{DM}_{f_2} \circ \mathbf{DM}_{f_1}$ *is not a* $2n$*-bit PPE though* f_1 *and* f_2 *are chosen from the* n*-bit TPE* Ω_n.

Proof. Let Ψ_{2n} be the set of all permutations over I_{2n} obtained from $\mathbf{DM}_{f_2} \circ \mathbf{DM}_{f_1}$. Consider a distinguisher $\mathcal{D}$ such as follows;

1. $\mathcal{D}$ chooses two queries $(L^{(1)}, R^{(1)})$ and $(L^{(2)}, R^{(2)})$ such that $L^{(1)} = R^{(1)} = 0$ and $L^{(2)} = R^{(2)} = S \neq 0$.
2. $\mathcal{D}$ sends these two queries to the oracle $\mathcal{O}$ and receives the corresponding answers $(Y_L^{(1)}, Y_R^{(1)})$ and $(Y_L^{(2)}, Y_R^{(2)})$ from the oracle.
3. $\mathcal{D}$ outputs 1 if and only if $Y_R^{(1)} = Y_R^{(2)}$.

If the oracle implements the TPE Ω_{2n}, then for any fixed $x^{(1)} = (L^{(1)}, R^{(1)})$ and $x^{(2)} = (L^{(2)}, R^{(2)})$, we obtain that

$$\begin{aligned} Pr(\mathcal{D} \text{ outputs } 1 \mid \mathcal{O} \leftarrow \Omega_{2n}) &= \frac{\#\{\pi \in \Omega_{2n} \mid \pi(x^{(1)})|_R = \pi(x^{(2)})|_R\}}{\#(\Omega_{2n})} \\ &= \frac{2^{2n} \cdot (2^n - 1) \cdot (2^{2n} - 2)!}{2^{2n}!} \leq \frac{1}{2^n} \ , \end{aligned}$$

where $x|_R$ denotes the right half n-bit of $2n$-bit vector x.

On the other hand, if $\mathcal{O}$ implements Ψ_{2n}, then for $x^{(1)} = (0, 0)$ and $x^{(2)} = (S, S)$,

$$Pr(\mathcal{D} \text{ outputs } 1 \mid \mathcal{O} \leftarrow \Psi_{2n}) = 1 \ ,$$

since $Y_R^{(1)} = g_1(0) = Y_R^{(2)}$.

Consequently we obtain that

$$Adv_{\mathcal{D}} = \left|p^{\Omega_{2n}} - p^{\Psi_{2n}}\right| \geq 1 - \frac{1}{2^n} \,,$$

which is non-negligible. □

Theorem 3 *Let g_1, g_2, g_3 be independently chosen from an n-bit PPE. Then the three round dual MISTY-type transformation $\mathbf{DM}_{g_3} \circ \mathbf{DM}_{g_2} \circ \mathbf{DM}_{g_1}$ is a $2n$-bit PPE.*

Proof. It suffices to show the assertion under the assumption that g_1, g_2, g_3 are independently chosen from the n-bit TPE Ω_n. Let Ψ_{2n} be the set of all permutations over I_{2n} obtained from $\mathbf{DM}_{g_3} \circ \mathbf{DM}_{g_2} \circ \mathbf{DM}_{g_1}$ and the i-th round output of this permutation is defined by (L_i, R_i) for $i = 1, 2, 3$ where (L, R) is the $2n$-bit input.

We assume that the distinguisher $\mathcal{D}$ makes t calls to the oracle $\mathcal{O}$. In the i-th oracle call, $\mathcal{D}$ sends a query $(L^{(i)}, R^{(i)})$ to $\mathcal{O}$ and receives the corresponding output $(Y_L^{(i)}, Y_R^{(i)}) = \pi(L^{(i)}, R^{(i)})$, where π is the randomly chosen permutation by $\mathcal{O}$ from Ω_{2n} or Ψ_{2n}.

Let A be the event that $L_1^{(1)} \oplus R_1^{(1)}, \cdots, L_1^{(t)} \oplus R_1^{(t)}$ are all distinct. If A occurs, then we can see that $L_3^{(1)}, \cdots, L_3^{(t)}$ are completely random since

$$L_3^{(i)} = g_3(L_1^{(i)} \oplus g_2(L_1^{(i)} \oplus R_1^{(i)}))$$

and the outputs of g_2 and g_3 are completely random. Similarly, we can see that if A occurs, then $R_3^{(1)}, \cdots, R_3^{(t)}$ are also completely random. Therefore, if A occur, then $Adv_{\mathcal{D}}$ is bounded above as follows:

$$Adv_{\mathcal{D}} \leq 1 - Pr(A) \leq \sum_{1 \leq i < j \leq t} Pr(B_{ij}) \,,$$

where B_{ij} is the event of $L_1^{(i)} \oplus R_1^{(i)} = L_1^{(j)} \oplus R_1^{(j)}$.

We estimate the value of $Pr(B_{ij})$ for any fixed $1 \leq i < j \leq t$. We have the following three cases as in the proof of Theorem 1.

<u>Case 1</u>: $L^{(i)} \neq L^{(j)}$ and $R^{(i)} = R^{(j)} = R_0$. Observe that $L_1^{(i)} = g_1(L^{(i)} \oplus R_0)$ and $L_1^{(j)} = g_1(L^{(j)} \oplus R_0)$. Then we obtain by Lemma 3 that

$$\begin{aligned} Pr(L_1^{(i)} \oplus R_1^{(i)} = L_1^{(j)} \oplus R_1^{(j)}) &= Pr(g_1(L^{(i)} \oplus R_0) \oplus g_1(L^{(j)} \oplus R_0) = L^{(i)} \oplus L^{(j)}) \\ &\leq \frac{1}{2^n - 1} \,, \end{aligned}$$

since g_1 is a truly random permutation.

<u>Case 2</u>: $L^{(i)} = L^{(j)} = L_0$ and $R^{(i)} \neq R^{(j)}$. We can see easily that

$$Pr(L_1^{(i)} \oplus R_1^{(i)} = L_1^{(j)} \oplus R_1^{(j)}) = Pr(g_1(L_0 \oplus R^{(i)}) \oplus g_1(L_0 \oplus R^{(j)}) = 0) = 0 \,.$$

Case 3: $L^{(i)} \neq L^{(j)}$ and $R^{(i)} \neq R^{(j)}$. If $L^{(i)} \oplus R^{(i)} = L^{(j)} \oplus R^{(j)}$, then

$$Pr(L_1^{(i)} \oplus R_1^{(i)} = L_1^{(j)} \oplus R_1^{(j)}) = Pr(L^{(i)} = L^{(j)}) = 0 .$$

Otherwise, by Lemma 3, we obtain that

$$\begin{aligned} & Pr(L_1^{(i)} \oplus R_1^{(i)} = L_1^{(j)} \oplus R_1^{(j)}) \\ &= Pr(g_1(L^{(i)} \oplus R^{(i)}) \oplus g_1(L^{(j)} \oplus R^{(j)}) = L^{(i)} \oplus L^{(j)}) \\ &\leq \frac{1}{2^n - 1} . \end{aligned}$$

Therefore for any case, we obtain that $Pr(B_{ij}) \leq \frac{1}{2^n-1}$. Thus

$$Adv_{\mathcal{D}} \leq \frac{t(t-1)}{2} \cdot \frac{1}{2^n - 1} = \frac{t^2 - t}{2^{n+1} - 2} ,$$

which is negligible, since $t = poly(n)$. □

From Theorem 1 and 3, we obtain the following result.

Theorem 4 *If f_1, f_2, f_3, and f_4 are independently chosen from an n-bit PPE, then the four round MISTY-type transformation $\mathbf{M}_{f_4} \circ \mathbf{M}_{f_3} \circ \mathbf{M}_{f_2} \circ \mathbf{M}_{f_1}$ is a $2n$-bit SPPE for any non-adaptive distinguisher.*

4 Pseudorandomness of KASUMI

In this section we investigate the pseudorandomness of the 3GPP algorithm KASUMI[1]. KASUMI is a block cipher that forms the heart of the 3GPP confidentiality and integrity algorithms. KASUMI is based on the block cipher MISTY1 which is provable secure against differential and linear cryptanalysis. We can classify the permutation of KASUMI into the following three stages;

– The overall permutation of KASUMI is a 64-bit permutation composed of the eight round Feistel permutation with the two round permutation FO and FL.
– FO function is a 32-bit permutation composed of the three round MISTY-type transformation with the round permutation FI.
– FI function is a 16-bit permutation which is composed of the four round unbalanced MISTY-type transformation obtained from 7-bit S-box $S7$ and 9-bit S-box $S9$.

By the similar argument as the proof of Theorem 1, we can easily obtain the fact that FI function is a 16-bit PPE since $S7$ and $S9$ are bijective. On the other hand we know that FO function is not a PPE, so it doesn't seem that the three round Feistel permutation of KASUMI is a PPE as the Luby-Rackoff cipher. Since the pseudorandomness of KASUMI is guaranteed by FO function mainly, we omit FL function in this paper. On the reasonable assumption that FI function is a PPE, we show that the four round KASUMI is a PPE.

We define two unbalanced MISTY-type transformations to examine accurately the pseudorandomness of FI function.

Definition 8 *Let n and m be two positive integer such that $m \leq n$. Then for any n-bit permutation f and m-bit permutation g, two $(n+m)$-bit unbalanced MISTY-type transformation $\tilde{\mathbf{M}}_f \in \Omega_{n+m}$ and $\mathbf{M}'_g \in \Omega_{n+m}$ are defined by*

$$\tilde{\mathbf{M}}_f(L,R) = (R, f(L) \oplus \tilde{R}) \in I_m \times I_n \ , \ \ \forall (L,R) \in I_n \times I_m$$

and

$$\mathbf{M}'_g(L,R) = (R, g(L) \oplus R') \in I_n \times I_m \ , \ \ \forall (L,R) \in I_m \times I_n \ ,$$

where for any n-bit vector x, x' denotes the m-bit value obtained by discarding the $n-m$ most-significant end and for any m-bit vector y, $\tilde{y}$ denotes the n-bit value obtained by adding $n-m$ zero bits to the most-significant end.

Theorem 5 *Let for any positive integer n and m such that $m \leq n$, $f_1, f_3 \in \Omega_n$ and $f_2, f_4 \in \Omega_m$ be independently chosen from two n-bit and m PPEs, respectively. Then the four round unbalanced MISTY-type transformation $\mathbf{M}'_{f_4} \circ \tilde{\mathbf{M}}_{f_3} \circ \mathbf{M}'_{f_2} \circ \tilde{\mathbf{M}}_{f_1}$ is an $(n+m)$-bit PPE.*

Proof. By the similar process as the proof of Theorem 1, we can obtain that

$$ADv_{\mathcal{D}} \leq (t^2 - t)\left(\frac{1}{2^{n+1}-2} + \frac{1}{2^m}\right) \ .$$

Then the assertion follows easily, since t is a polynomial in n and m. □

From Theorem 5, it becomes a reasonable assumption that FI function of KASUMI is a PPE. In order to investigate the pseudorandomness of KASUMI, we use a simplified figure of KASUMI. The four round simplified KASUMI is illustrated in Figure 1, where $x = (x_1, x_2, x_3, x_4)$ denotes a $4n$-bit input value, $w = (w_1, w_2, w_3, w_4)$, $y = (y_1, y_2, y_3, y_4)$, and $z = (z_1, z_2, z_3, z_4)$ denote corresponding outputs of the two, three, and four round KASUMI, respectively. Each of x_i, w_i, y_i, and z_i is an n-bit value. We first prove the following theorem.

Theorem 6 *The three round simplified KASUMI is not a $4n$-bit PPE though f_i's$(i = 1, \cdots, 9)$ of Figure 1 are independently chosen from an n-bit PPE.*

Proof. Let Ψ_{4n} be the set of all permutations over I_{4n} obtained from the three round simplified KASUMI. Consider a distinguisher $\mathcal{D}$ such as follows;

1. $\mathcal{D}$ chooses four $4n$-bit queries $x^{(1)}$, $x^{(2)}$, $x^{(3)}$, and $x^{(4)}$ such that

$$x^{(1)} = (0, 0, x_3, x_4) \ , \ \ x^{(2)} = (x_1, 0, x_3, x_4) \ ,$$

$$x^{(3)} = (0, x_2, x_3, x_4) \ , \ \ x^{(4)} = (x_1, x_2, x_3, x_4) \ ,$$

   where $x_1 \neq 0 \neq x_2$ and x_3, x_4 are fixed n-bit values.

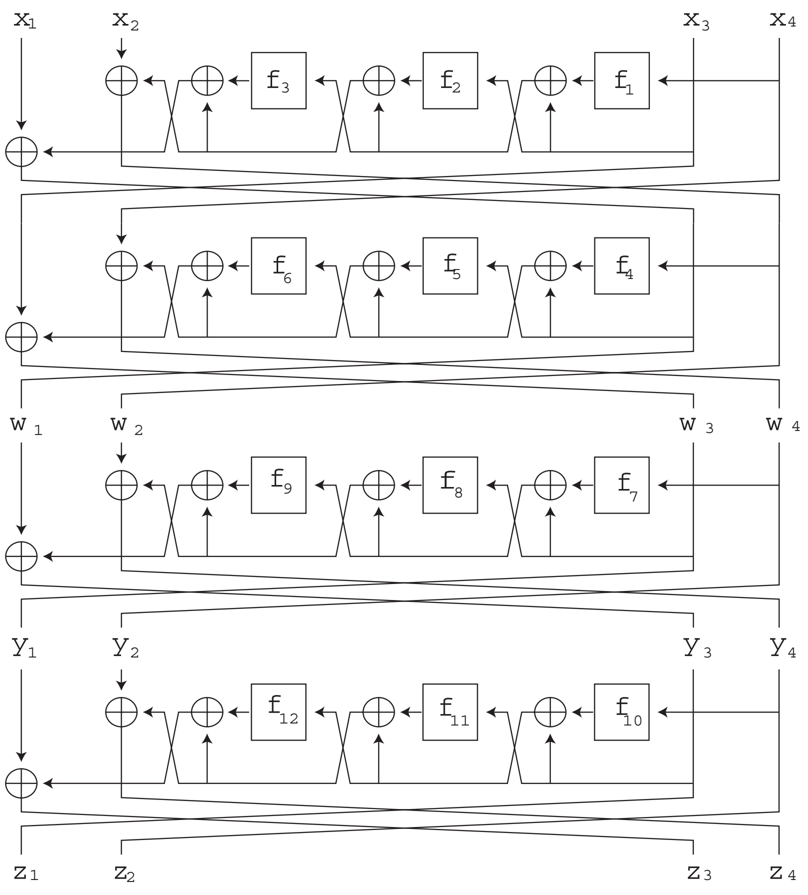


Fig. 1. Simplified four round KASUMI

2. $\mathcal{D}$ sends these four queries to the oracle $\mathcal{O}$ and receives the corresponding answers $(Y_1^{(i)}, Y_2^{(i)}, Y_3^{(i)}, Y_4^{(i)})(i = 1, 2, 3, 4)$ from the oracle.
3. $\mathcal{D}$ outputs 1 if and only if

$$Y_2^{(1)} \oplus Y_2^{(2)} \oplus Y_2^{(3)} \oplus Y_2^{(4)} = 0 \; .$$

If the oracle implements the TPE Ω_{4n}, then we obtain that

$$\begin{aligned} Pr(\mathcal{D} \text{ outputs } 1 \mid \mathcal{O} \leftarrow \Omega_{4n}) &\leq \frac{2^{4n}(2^{4n}-1)(2^{4n}-2)2^{3n}(2^{4n}-4)!}{2^{4n}!} \\ &= \frac{2^{3n}}{2^{4n}-3} \leq \frac{1}{2^{n-1}} \; . \end{aligned}$$

On the other hand, if $\mathcal{O}$ implements Ψ_{4n}, then for $x^{(1)} = (0, 0, x_3, x_4)$, $x^{(2)} = (x_1, 0, x_3, x_4)$, $x^{(3)} = (0, x_2, x_3, x_4)$, and $x^{(4)} = (x_1, x_2, x_3, x_4)$, we can see from Figure 1 that the corresponding $2n$-bit inputs of the second round are

$$(F_1(x_3, x_4)|_L, F_1(x_3, x_4)|_R) \ , \ \ (F_1(x_3, x_4)|_L, x_1 \oplus F_1(x_3, x_4)|_R) \ ,$$

$$(x_2 \oplus F_1(x_3, x_4)|_L, F_1(x_3, x_4)|_R) \ , \ \ (x_1 \oplus F_1(x_3, x_4)|_L, x_2 \oplus F_1(x_3, x_4)|_R)$$

respectively, where $F_1 = \mathbf{M}_{f_3} \circ \mathbf{M}_{f_2} \circ \mathbf{M}_{f_1}$ and $(x|_L, x|_R)$ denote the left and right n-bit block of $2n$-bit value x. Thus we obtain by the similar argument of Sakurai-Zheng[10] that

$$y_2^{(1)} \oplus y_2^{(2)} \oplus y_2^{(3)} \oplus y_2^{(4)} = 0$$

with probability 1.

Consequently we obtain that

$$Adv_{\mathcal{D}} = \left|p^{\Omega_{4n}} - p^{\Psi_{4n}}\right| \geq 1 - \frac{1}{2^{n-1}} \ ,$$

which is non-negligible. □

Theorem 7 *If f_i's($i = 1, 2, \cdots, 12$) in Figure 1 are independently chosen from an n-bit PPE, then the four round KASUMI is a $4n$-bit PPE.*

Proof. Assume that f_i's are independently chosen from the TPE Ω_n. Let Ψ_{4n} be the set of all permutations over I_{4n} obtained from the four round KASUMI. Suppose that the distinguisher $\mathcal{D}$ makes t calls to the oracle $\mathcal{O}$. In the i-th oracle call, $\mathcal{D}$ sends a $4n$-bit query $x^{(i)} = (x_1^{(i)}, x_2^{(i)}, x_3^{(i)}, x_4^{(i)})$ to $\mathcal{O}$ and receives the corresponding output

$$(Z_1^{(i)}, Z_2^{(i)}, Z_3^{(i)}, Z_4^{(i)}) = \pi((x_1^{(i)}, x_2^{(i)}, x_3^{(i)}, x_4^{(i)})) \ ,$$

where π is the randomly chosen permutation by $\mathcal{O}$ from Ω_{4n} or Ψ_{4n}.

Let A_{w_j} denote the event that the j-th block of the outputs of two round KASUMI $w_j^{(1)}, \cdots, w_j^{(t)}$ are all distinct for $j = 1, 2, 3, 4$(see Figure 1). If A_{w_3} occurs, then we can see that $z_2^{(1)}, \cdots, z_2^{(t)}$ are completely random since the outputs of f_8 are completely random. Furthermore we also see that $z_3^{(1)}, \cdots, z_3^{(t)}$ are completely random because the outputs of f_{10} and f_{12} are completely random. Similarly, if A_{w_4} occurs, then $z_1^{(1)}, \cdots, z_1^{(t)}$ and $z_4^{(1)}, \cdots, z_4^{(t)}$ are completely random due to $f_7(f_9)$ and f_{11}, respectively. Therefore, if A_{w_3} and A_{w_4} occur, then $Adv_{\mathcal{D}}$ is bounded above as follows;

$$Adv_{\mathcal{D}} \leq 1 - Pr(A_{w_3} \cap A_{w_4}) \leq \sum_{1 \leq i < j \leq t} Pr(w_3^{(i)} = w_3^{(j)}) + \sum_{1 \leq i < j \leq t} Pr(w_4^{(i)} = w_4^{(j)}) \ .$$

We estimate the summands $Pr(w_3^{(i)} = w_3^{(j)})$ and $Pr(w_4^{(i)} = w_4^{(j)})$ for any $1 \leq i < j \leq t$. Fix $x^{(i)} = (x_1^{(i)}, x_2^{(i)}, x_3^{(i)}, x_4^{(i)})$ and $x^{(j)} = (x_1^{(j)}, x_2^{(j)}, x_3^{(j)}, x_4^{(j)})$ arbitrarily. We separate the following four cases.

Case 1: $x_1^{(i)} \neq x_1^{(j)}$. Consider the path $x_1 \to f_5 \to w_4$. Then $w_4^{(i)}$ and $w_4^{(j)}$ behave randomly since f_5 is a truly random. So we obtain that $Pr(w_4^{(i)} = w_4^{(j)}) = \frac{1}{2^n}$. Similarly we also obtain that $Pr(w_3^{(i)} = w_3^{(j)}) = \frac{1}{2^n}$ by considering the path $x_1 \to f_5 \to w_3$.

Case 2: $x_2^{(i)} \neq x_2^{(j)}$. Consider the path $x_2 \to f_4 \to f_6 \to w_3$. Then $w_3^{(i)}$ and $w_3^{(j)}$ are completely random. Thus $Pr(w_3^{(i)} = w_3^{(j)}) = \frac{1}{2^n}$ holds. For w_4, consider the path $x_2 \to f_4 \to w_4$, then we obtain that $Pr(w_4^{(i)} = w_4^{(j)}) = \frac{1}{2^n}$.

Case 3: $x_3^{(i)} \neq x_3^{(j)}$. Consider the path $x_3 \to f_2 \to f_4 \to f_6 \to w_3$. Then we can see that $w_3^{(i)}$ and $w_3^{(j)}$ are completely random. By considering the path $x_3 \to f_2 \to f_5 \to w_4$, also we know that $w_4^{(i)}$ and $w_4^{(j)}$ are completely random. Hence we obtain that

$$Pr(w_3^{(i)} = w_3^{(j)}) = Pr(w_4^{(i)} = w_4^{(j)}) = \frac{1}{2^n} .$$

Case 4: $x_4^{(i)} \neq x_4^{(j)}$. In this case by considering the two paths $x_4 \to f_1 \to f_3 \to f_5 \to w_4$ and $x_4 \to f_1 \to f_4 \to f_6 \to w_3$, we obtain that

$$Pr(w_3^{(i)} = w_3^{(j)}) = Pr(w_4^{(i)} = w_4^{(j)}) = \frac{1}{2^n}$$

holds as the above three cases.

Therefore, for any case, we obtain that

$$Pr(w_3^{(i)} = w_3^{(j)}) = Pr(w_4^{(i)} = w_4^{(j)}) = \frac{1}{2^n} .$$

This implies that $Adv_{\mathcal{D}} \leq \frac{t(t-1)}{2^n}$, which is negligible since $t = poly(n)$. □

Since the pseudorandomness of the inverse transformation of Feistel-type is very similar to that of Feistel-type transformation, we get also the following fact.

Corollary 1. *If f_i's($i = 1, 2, \cdots, 12$) in Figure 1 are independently chosen from an n-bit PPE, then the four round KASUMI is a $4n$-bit SPPE for any non-adaptive distinguisher.*

5 Conclusion

We examined the pseudorandomness of the (unbalanced) MISTY-type and dual MISTY-type transformations, and by applying these results, also investigated the pseudorandomness of the 3GPP block cipher KASUMI. We showed that the four round (unbalanced) MISTY-type transformation is a pseudorandom permutation ensemble. We also proved that the three round KASUMI is not a pseudorandom permutation ensemble but the four round KASUMI is a pseudorandom permutation ensemble. In this paper we provided simplified probability-theoretic methods for non-adaptive distinguishers. By applying these proving methods to another block ciphers, we expect to obtain easily some useful results related to the pseudorandomness.

References

1. ETSI/SAGE, Specification of the 3GPP Confidentiality and Integrity Algorithms, available at http://www.etsi.org/dvbandca/3GPP/3gppspecs.htm
2. H. Gilbert and M. Minier, *New results on the pseudorandomness of some block cipher constructions*, Preproceedings of Fast Software Encryption workshop 2001, (2001, Yokohama), pp. 260-277.
3. T. Iwata and K. Kurosawa, *On the pseudorandomness of the AES finalists - RC6 and Serpent*, Preproceedings of Fast Software Encryption workshop 2000, (2000, New York).
4. T. Iwata, T. Yoshino, T. Yuasa, and K. Kurosawa, *Round security and super-pseudorandomness of MISTY type structure*, Preproceedings of Fast Software Encryption workshop 2001, (2001, Yokohama), pp. 245-259.
5. M. Luby and C. Rackoff, *How to construct pseudorandom permutations and pseudorandom functions*, SIAM J. Comput., Vol. 17, 1988, pp. 189-203.
6. M. Matsui, *New permutation of Block Ciphers with Provable Security against Differential and Linear Cryptalaysis*, Fast Software Encryption, LNCS 1039, Springer-Verlag, 1996, pp. 205-218.
7. M. Matsui, *New Block Encryption Algorithm MISTY*, Fast Software Encryption'97, LNCS 1267, Springer-Verlag, 1997, pp. 54-68.
8. U. M. Maurer, *A simplified and generalized treatment of Luby-Rackoff pseudorandom permutation generators*, Advances in Cryptology - Eurocrypt'92, LNCS 658, Springer-Verlag, 1992, pp. 239-255.
9. M. Naor and O. Reingold, *On the construction of pseudorandom permutations: Luby-Rackoff revisited*, J. Cryptology, Vol. 12, 1999, pp. 29-66.
10. K. Sakurai and Y. Zheng, *On non-pseudorandomness from block ciphers with provable immunity against linear cryptanaysis*, IEICE Trans. Fundamentals, Vol. E80-A, No. 1, 1997, pp. 19-24.

New Public-Key Cryptosystem Using Divisor Class Groups

Hwankoo Kim[1] and SangJae Moon[1,2]

[1] Mobile Network Security Technology Research Center, Kyungpook National University, Taegu 702-701, Korea.
hwankoo@knu.ac.kr

[2] School of Electronic and Electrical Eng., Kyungpook National University, Taegu 702-701, Korea.
sjmoon@knu.ac.kr

Abstract. We show how to use ideal arithmetic in the divisor class group of an affine normal subring of $K[X,Y]$ generated by monomials, where K is a field, to design new public-key cryptosystems, whose security is based on the difficulty of the discrete logarithm problem in the divisor class group of that integral domain.

1 Introduction

The security of many cryptographic systems has been based on the difficulty of several number theoretic problems. Prominent examples are the factoring problem for integers [17,16] and the discrete logarithm problem (DLP) in the multiplicative group of a finite field [15], in the class group of an order of a quadratic field [4], in the group of points on an elliptic curve over a finite field [12], in the group of points on a hyperelliptic curve over a finite field [13], and others. There is, however, no guarantee that those problems remain difficult to solve in the future. On the contrary, as the experience with the factoring problem shows, unexpected breakthroughs are always possible. Therefore, it is important to design cryptographic schemes in such a way that the underlying mathematical problem can easily be replaced with another one.

This paper shows how to use ideal arithmetic in the divisor class group of an affine normal subring of $K[X,Y]$ generated by monomials, where K is a field, to design new public-key cryptosystems, whose security is based on the difficulty of the discrete logarithm problem in the divisor class group of that integral domain. We believe that our DLP is much more difficult than that of the class group of an order of a quadratic number field.

2 Mathematical Background

Let R be an integral domain with quotient field K. If $\mathfrak{a}$ is an integral ideal of R, then any subset of K of the form $\frac{1}{d}\mathfrak{a}$, where d is a nonzero element of R, is called a *fractional* ideal of R. If $\mathfrak{a}$ and $\mathfrak{b}$ are fractional ideals, then their *product*

$$\mathfrak{a}\mathfrak{b} = \{\sum_{\text{finite}} \alpha\beta \mid \alpha \in \mathfrak{a}, \beta \in \mathfrak{b}\}$$

is again a fractional ideal of R. The set $\mathcal{F}(R)$ of all nonzero fractional ideals of R forms a commutative semigroup with identity R. A fractional ideal $\mathfrak{a}$ is said to be *invertible* if $\mathfrak{a}\mathfrak{b} = R$ for some fractional ideal $\mathfrak{b}$ of R. Clearly every nonzero principal fractional ideals are invertible. The (*residual*) *quotient* of $\mathfrak{a}$ over $\mathfrak{b}$

$$\mathfrak{a} : \mathfrak{b} = \{\alpha \in K \mid \alpha\mathfrak{b} \subseteq \mathfrak{a}\}$$

is a fractional ideal. For a fractional ideal $\mathfrak{a}$ of an integral domain R with quotient field K, $\mathfrak{a}_v$ is defined as the fractional ideal $R : (R : \mathfrak{a}) = (\mathfrak{a}^{-1})^{-1}$. A fractional ideal $\mathfrak{a}$ is called a *divisorial* ideal or v-ideal if $\mathfrak{a}_v = \mathfrak{a}$. The set $\mathcal{D}(R)$ of divisorial ideals is a commutative semigroup with identity R under the v-product $\mathfrak{a} * \mathfrak{b} = (\mathfrak{a}\mathfrak{b})_v$. Of course, $\mathcal{D}(R)$ is a group if and only if R is completely integrally closed. In this case, the quotient group $Cl(R) := \mathcal{D}(R)/\mathcal{P}(R)$ is called the *divisor class group* of R, where $\mathcal{P}(R)$ is the subgroup of $\mathcal{D}(R)$ which consists of all nonzero principal fractional ideals of R. Note that if R is a Dedekind domain (equivalently, a one-dimensional Krull domain), for example, a maximal order of a quadratic number field, then the definition of the divisor class group is equal to that of the usual class group. Elements of $Cl(R)$ will be denoted by $[\mathfrak{a}]$. Clearly, if $\mathfrak{b} \in [\mathfrak{a}]$, then $\mathfrak{b} = \alpha\mathfrak{a}$ for some $\alpha \in K$. In this case we say that $\mathfrak{a}$ and $\mathfrak{b}$ are *equivalent*, written $\mathfrak{a} \sim \mathfrak{b}$.

Let $\alpha_1, \ldots, \alpha_s$ be elements in an integral domain R. Then we set

$$\langle \alpha_1, \ldots, \alpha_s \rangle = \{\sum_{i=1}^{s} \beta_i\alpha_i \mid \beta_1, \ldots, \beta_s \in R\}.$$

Note that $\langle \alpha_1, \ldots, \alpha_s \rangle$ is an ideal of R. We will call $\langle \alpha_1, \ldots, \alpha_s \rangle$ the ideal generated by $\alpha_1, \ldots, \alpha_s$. We say that an ideal $\mathfrak{a}$ is *finitely generated* if there exist $\alpha_1, \ldots, \alpha_s \in R$ such that $\mathfrak{a} = \langle \alpha_1, \ldots, \alpha_s \rangle$, and we say that $\alpha_1, \ldots, \alpha_s$ is a basis of $\mathfrak{a}$. Recall that an integral domain R is said to be *Noetherian* if it satisfies the ascending chain condition on ideals. It is well-known that R is Noetherian if and only if every ideal of R is finitely generated. Any unexplained notation or terminology is standard like in [8].

The following results are easy to prove (or well-known) and we will use them frequently without mention.

Proposition 1. *Let R be an integral domain with quotient field K and let $\mathfrak{a}, \mathfrak{b}$, and $\mathfrak{c}$ be nonzero fractional ideals of R. Then*

(1) (i) $\langle\alpha\rangle_v = \langle\alpha\rangle$ for each $0 \neq \alpha \in K$; $\mathfrak{a} \subseteq \mathfrak{a}_v$.
(ii) If $\mathfrak{a} \subseteq \mathfrak{b}$, then $\mathfrak{a}_v \subseteq \mathfrak{b}_v$.
(iii) $(\alpha\mathfrak{a})_v = \alpha\mathfrak{a}_v$; $(\mathfrak{a}_v)_v = \mathfrak{a}_v$.
(2) $(\mathfrak{a}_v\mathfrak{b})_v = (\mathfrak{a}_v\mathfrak{b}_v)_v = (\mathfrak{a}\mathfrak{b})_v$.
(3) $(\mathfrak{a} : \mathfrak{b}) : \mathfrak{c} = \mathfrak{a} : \mathfrak{b}\mathfrak{c}$.
(4) If $\mathfrak{a}$ is divisorial, then $\mathfrak{a} : \mathfrak{b} = \mathfrak{a} : \mathfrak{b}_v$.
(5) If $\mathfrak{a}$ is invertible, then $\mathfrak{a}\mathfrak{b} : \mathfrak{c} = \mathfrak{a}(\mathfrak{b} : \mathfrak{c})$ and $\mathfrak{b} : \mathfrak{a}\mathfrak{c} = \mathfrak{a}^{-1}(\mathfrak{b} : \mathfrak{c})$.

Recall that an integral domain R is called a *Mori* domain if it satisfies the ascending chain condition on divisorial ideals. Clearly the class of Noetherian domains is contained in that of Mori domains.

Proposition 2. *Let R be an integral domain with quotient field K.*

(1) R is completely integrally closed if and only if $\mathfrak{a} : \mathfrak{a} = R$ for each nonzero fractional ideal $\mathfrak{a}$ of R.
(2) R is a Krull domain if and only if it is a completely integrally closed Mori domain.
(3) If R is completely integrally closed and $\mathfrak{a}$ and $\mathfrak{b}$ are divisorial, then $\mathfrak{a} \sim \mathfrak{b}$ if and only if $\mathfrak{a} : \mathfrak{b}$ is a principal fractional ideal.

Proof. (1) and (2) are well-known. (3) Assume that $\mathfrak{a} \sim \mathfrak{b}$. Then $\mathfrak{a} = \alpha\mathfrak{b}$ for some $\alpha \in K$. Thus $\mathfrak{a} : \mathfrak{b} = \alpha\mathfrak{b} : \mathfrak{b} = \alpha(\mathfrak{b} : \mathfrak{b}) = \alpha R$ since R is completely integrally closed. Conversely, assume that $\mathfrak{a} : \mathfrak{b}$ is fractional principal, say, $\mathfrak{a} : \mathfrak{b} = \beta R$, where $\beta \in K$. Then we have $\mathfrak{a} = \mathfrak{a} : R = \mathfrak{a} : (\mathfrak{b}\mathfrak{b}^{-1})_v = \mathfrak{a} : \mathfrak{b}\mathfrak{b}^{-1} = (\mathfrak{a} : \mathfrak{b}) : \mathfrak{b}^{-1} = \beta R : \mathfrak{b}^{-1} = \beta(R : \mathfrak{b}^{-1}) = \beta\mathfrak{b}_v = \beta\mathfrak{b}$, since R is completely integrally closed and $\mathfrak{a}$ is divisorial. Thus $\mathfrak{a} \sim \mathfrak{b}$.

3 Computational Aspects

Let K be any field and let $K[x_1, \ldots, x_m]$ be a polynomial ring. Denoted by

$$\mathbb{T}^m = \{x_1^{s_1} \cdots x_m^{s_m} \mid s_i \in \mathbb{N}, i = 1, \ldots, n\}$$

the set of power products. Sometimes we will denote $x_1^{s_1} \cdots x_m^{s_m}$ by $\mathbf{x}^{\mathbf{s}}$, where $\mathbf{s} = (s_1, \ldots, s_m) \in \mathbb{N}^m$. By a *term order* (or *monomial order*) on $\mathbb{T}^m$ we mean a total order $<$ on $\mathbb{T}^m$ satisfying the following two conditions:

(i) $1 < \mathbf{x}^{\mathbf{s}}$ for all $\mathbf{x}^{\mathbf{s}} \in \mathbb{T}^m, \mathbf{x}^{\mathbf{s}} \neq 1$;
(ii) if $\mathbf{x}^{\mathbf{s}} < \mathbf{x}^{\mathbf{t}}$, then $\mathbf{x}^{\mathbf{s}}\mathbf{x}^{\mathbf{u}} < \mathbf{x}^{\mathbf{t}}\mathbf{x}^{\mathbf{u}}$, for all $\mathbf{x}^{\mathbf{u}} \in \mathbb{T}^m$.

Note that every term order on $\mathbb{T}^m$ is a well-ordering [1, Theorem 1.4.6].

To introduce the concept of a Gröbner basis for the ideal, we fix some notation. First we choose a term order on $K[x_1, \ldots, x_m]$. Then for all $0 \neq f \in K[x_1, \ldots, x_m]$, we may write

$$f = a_1\mathbf{x}^{\mathbf{s_1}} + a_2\mathbf{x}^{\mathbf{s_2}} + \cdots + a_r\mathbf{x}^{\mathbf{s_r}},$$

where $0 \neq a_i \in K, \mathbf{x}^{\mathbf{s_i}} \in \mathbb{T}^m$, and $\mathbf{x}^{\mathbf{s_1}} > \mathbf{x}^{\mathbf{s_2}} > \cdots > \mathbf{x}^{\mathbf{s_r}}$. We will always try to write our polynomials in this way. We define:

- $\mathrm{lp}(f) = \mathbf{x}^{\mathbf{s_1}}$, the *leading power product* of f;
- $\mathrm{lc}(f) = a_1$, the *leading coefficient* of f.

We also define $\mathrm{lp}(0) = \mathrm{lc}(0) = 0$.

Definition 1. *A set of nonzero polynomials $G = \{g_1, \ldots, g_t\}$ contained in an ideal $\mathfrak{a}$ of the polynomial ring over a field, is called a Gröbner basis for $\mathfrak{a}$ if and only if for all $f \in \mathfrak{a}$ such that $f \neq 0$, there exists $i \in \{1, \ldots, t\}$ such that $lp(g_i)$ divides $lp(f)$.*

Buchberger's Algorithm to compute Gröbner bases is given in [1, Algorithm 1.7.1] and many computer algebra systems including *Macaulay 2* [9] implement a version of Buchberger's Algorithm for computing Gröbner bases. It is well-known that if $G = \{g_1, \ldots, g_t\}$ is a Gröbner basis for the ideal $\mathfrak{a}$, then $\mathfrak{a} = \langle g_1, \ldots, g_t \rangle$ [1, Corollary 1.6.3].

Definition 2. *A Gröbner basis $G = \{g_1, \ldots, g_t\}$ is said to be reduced if, for all i, $lc(g_i) = 1$ and no nonzero term in g_i is divisible by any $lp(g_j)$ for any $j \neq i$.*

Theorem 1. [1, Theorem 1.8.7] *Fix a term order. Then every nonzero ideal $\mathfrak{a}$ has a unique reduced Gröbner basis with respective to this term order.*

Let R be an integral domain and let $\mathfrak{a}$ and $\mathfrak{b}$ be integral ideals of R. Then

$$\mathfrak{a} :_R \mathfrak{b} = \{r \in R \mid r\mathfrak{b} \subseteq \mathfrak{a}\}$$

is also an ideal of R.

We show that if $R = K[x_1, \ldots, x_m]$ is a polynomial ring over a field K, a (Gröbner) basis for the ideal $\mathfrak{a} : \mathfrak{b}$ (resp., $\mathfrak{a}_v$) can be computed using a computer algebra system. The following useful proposition relates the quotient operation to the other operations:

Proposition 3. *Let $\mathfrak{a}, \mathfrak{a}_i, \mathfrak{b}, \mathfrak{b}_i$ and $\mathfrak{c}$ be ideals in an integral domain R for $1 \leq i \leq r$. Then*

$$\Big(\bigcap_{i=1}^{r} \mathfrak{a}_i\Big) :_R \mathfrak{b} = \bigcap_{i=1}^{r} (\mathfrak{a}_i :_R \mathfrak{b}), \tag{1}$$

$$\mathfrak{a} :_R \Big(\sum_{i=1}^{r} \mathfrak{b}_i\Big) = \bigcap_{i=1}^{r} (\mathfrak{a} :_R \mathfrak{b}_i), \tag{2}$$

$$(\mathfrak{a} :_R \mathfrak{b}) :_R \mathfrak{c} = \mathfrak{a} :_R \mathfrak{b}\mathfrak{c}. \tag{3}$$

The actual computation can be carried out as follows:

Proposition 4. [18, Proposition 2.2.1] *Let R be an integral domain with quotient field K and let $\mathfrak{a} = \langle a_1, \ldots, a_m \rangle$ and $\mathfrak{b}$ be integral ideals of R. Then:*

$$R : \mathfrak{a} = (\langle a_1 \rangle :_R \langle a_2, \ldots, a_m \rangle) a_1^{-1} \tag{4}$$

$$\mathfrak{b} : \mathfrak{a} = \Big(\bigcap_{i=1}^{m} \mathfrak{b}(a_1 \cdots \widehat{a_i} \cdots a_m)\Big)(a_1 \cdots a_m)^{-1}. \tag{5}$$

Let $R = K[x_1, \dots, x_n]$ be a polynomial ring over a field K. If $0 \neq f \in R$ and $\mathfrak{a}$ an ideal of R, we often write $\mathfrak{a} :_R f$ instead of $\mathfrak{a} :_R \langle f \rangle$. Note that a special case of equation (2) is that

$$\mathfrak{a} :_R \langle f_1, \dots, f_r \rangle = \bigcap_{i=1}^{r} (\mathfrak{a} :_R f_i). \tag{6}$$

We now turn to the question of how to compute generators of the ideal quotient $\mathfrak{a} :_R \mathfrak{b}$ given generators of $\mathfrak{a}$ and $\mathfrak{b}$. The following observation is the key step.

Theorem 2. [5, Theorem 4.4.11] *Let $\mathfrak{a}$ be an ideal and g an element of $R = K[x_1, \dots, x_n]$. If $\{h_1, \dots, h_p\}$ is a basis of the ideal $\mathfrak{a} \cap \langle g \rangle$, then $\{h_1/g, \dots, h_p/g\}$ is a basis of $\mathfrak{a} :_R \langle g \rangle$.*

• Theorem 2, together with an algorithm for computing intersections of ideals [1, Section 2.3] and equation (6), immediately leads to an algorithm for computing a basis of an ideal quotient $\mathfrak{a} :_R \mathfrak{b}$ and hence a basis of the divisorial closure $\mathfrak{a}_v$ of $\mathfrak{a}$ by equation (4).
• Equation (5), together with an algorithm for computing intersections of ideals, immediately leads to an algorithm for computing a basis of an ideal quotient $\mathfrak{b} : \mathfrak{a}$

Let R be an affine normal(= integrally closed) subring of $T = K[X, Y]$ generated by monomials with $R \subseteq T$ integral, where K is a field. Then R is isomorphic to either T or $R_{n,j} := K[X^n, XY^j, X^2Y^{\overline{2j}}, \dots, X^{n-1}Y^{\overline{(n-1)j}}, Y^n]$, where $0 < j < n$, $\gcd(j, n) = 1$, and $\overline{m}$ denotes the smallest representative in $\mathbb{N}$ of the congruence class of m modulo n [2, Theorem 2.5]. Note that $R_{n,j}$ is a two-dimensional Noetherian Krull domain which is $\mathbb{Z}^+ \times \mathbb{Z}^+$-graded by $\deg X^iY^j = (i, j)$. The following result is due to D. F. Anderson [2, Theorem 4.4].

Theorem 3. *Let $R_{n,j} := K[X^n, XY^j, X^2Y^{\overline{2j}}, \dots, X^{n-1}Y^{\overline{(n-1)j}}, Y^n]$, where K is a field, $0 < j < n$, $\gcd(j, n) = 1$, and $\overline{m}$ denotes the smallest representative in $\mathbb{N}$ of the congruence class of m modulo n. Then $Cl(R_{n,j}) \cong \mathbb{Z}/n\mathbb{Z}$.*

Let

$$\mathfrak{p}_1 = \langle X^n, XY^j, X^2Y^{\overline{2j}}, \dots, X^{n-1}Y^{\overline{(n-1)j}} \rangle$$

and

$$\mathfrak{p}_2 = \langle XY^j, X^2Y^{\overline{2j}}, \dots, X^{n-1}Y^{\overline{(n-1)j}}, Y^n \rangle.$$

Then from the proof of Theorem 3 we know that $Cl(R_{n,j})$ can be generated by either $[\mathfrak{p}_1]$ or $[\mathfrak{p}_2]$.

We do not know whether it is possible to compute a Gröbner basis of a nonzero ideal of $K[X^n, XY^j, X^2Y^{\overline{2j}}, \dots, X^{n-1}Y^{\overline{(n-1)j}}, Y^n]$. However, at least we can compute a (reduced) Gröbner basis of a nonzero ideal of the following special ring: $R_{n,1} = K[X^n, XY, Y^n]$, where K is a computable field, for example, $\mathbb{Q}$ or a finite field. Indeed, it is not difficult to see that $R_{n,1} \cong K[x, y, z]/\langle z^n - xy \rangle$, where x, y, and z are indeterminates. Since we can compute this isomorphism using a computer algebra system, we may assume that $R_{n,1} = K[x, y, z]/\langle z^n - xy \rangle$. An explicit method for computing the sum and product operations in the quotient ring $R_{n,1}$ is given in [5, section 5.3].

Since most algebraic properties of the ideal are easily (i.e., in polynomial time) deduced from a Gröbner basis, the complexity of Gröbner basis algorithm is an important problem. In [19], F. Winkler gave an upper bound for the degrees of the polynomials which appear during the computation of a Gröbner basis of an ideal in $K[x, y, z]$. This bound is $(8D+1)2^d$, where D (resp. d) is the maximal (resp. minimal) degree of the members of the initial system of generators. However, as mentioned in [19], this minimal degree usually drops during the Gröbner basis computation, thus giving better and better bounds for the actual result of the computation.

In the next section, we show that each divisor class of divisorial ideals of $R_{n,1}$ contains a "representative" ideal. Indeed, its proof immediately leads to an algorithm for finding such a representative ideal. This can then be used as the basis of our new public-key cryptosystems.

4 The New Cryptosystem

Let $\bar{\mathfrak{a}}$ be a nonprincipal divisorial ideal of $R_{n,1}$. Then $\bar{\mathfrak{a}} = \mathfrak{a}/\langle z^n - xy \rangle$ for some ideal $\mathfrak{a}$ of $K[x, y, z]$. Thus by Theorem 1 there exists a unique reduced Gröbner basis $\{g_1, \dots, g_m\}$. Now we can compute the greatest common divisor of $g_1, \dots, g_m$ as in [1, section 2.3]. Let $g = \gcd(g_1, \dots, g_m)$. Then for each i we have $g_i = gg'_i$ for some $g'_i \in K[x, y, z]$. Set $\mathfrak{a}' = \langle g'_1, \dots, g'_m \rangle$. Then $\{g'_1, \dots, g'_m\}$ is the unique reduced Gröbner basis for $\mathfrak{a}'$. Note that $\gcd(g'_1, \dots, g'_m) = 1$. We define $Red(\bar{\mathfrak{a}}) = \mathfrak{a}'/\langle z^n - xy \rangle$ and we say that $\bar{\mathfrak{a}}$ is *reduced* if $Red(\bar{\mathfrak{a}}) = \bar{\mathfrak{a}}$. Note that $Red(\bar{\mathfrak{a}})$ is a nonprincipal divisorial ideal of $R_{n,1}$ and $[Red(\bar{\mathfrak{a}})] = [\bar{\mathfrak{a}}]$. Now we show that there exists only one reduced ideal in every divisor class. To do this end, it suffices to show that if $\bar{\mathfrak{b}}$ is a nonprincipal divisorial ideal of $R_{n,1}$ such that $\bar{\mathfrak{b}} \in [\bar{\mathfrak{a}}]$, then $Red(\bar{\mathfrak{a}}) = Red(\bar{\mathfrak{b}})$. Let $\mathfrak{b} \subset K[x, y, z]$ be an ideal such that $\bar{\mathfrak{b}} = \mathfrak{b}/\langle z^n - xy \rangle$. Then again by Theorem 1 there exists a unique reduced Gröbner basis $\{h_1, \dots, h_l\}$ for the ideal $\mathfrak{b}$. Since $\bar{\mathfrak{b}} \in [\bar{\mathfrak{a}}]$, we have $\mathfrak{b} = f\mathfrak{a}$ for some f. Note that $\{fgg'_1, \dots, fgg'_m\}$ is also a reduced Gröbner basis for $\mathfrak{b}$. By the uniqueness of the reduced Gröbner basis for $\mathfrak{b}$, we have $l = m$. Let $h = \gcd(h_1, \dots, h_m)$. Then for each i, $h_i = hh'_i$, where each $h'_i \in K[x, y, z]$. Let $\mathfrak{b}' = \langle h'_1, \dots, h'_m \rangle$. Then $\gcd(h'_1, \dots, h'_m) = 1$ and $Red(\bar{\mathfrak{b}}) = \mathfrak{b}'/\langle z^n - xy \rangle$. Note that $\{hh'_1, \dots, hh'_m\}$ is also a reduced Gröbner basis for $\mathfrak{b}$. Since $\{fgg'_1, \dots, fgg'_m\} = \{hh'_1, \dots, hh'_m\}$ again by the uniqueness of the reduced Gröbner basis for $\mathfrak{b}$, we have $fg = h$ by the uniqueness of the greatest common divisor. Thus $\{g'_1, \dots, g'_m\} = \{h'_1, \dots, h'_m\}$ and so $\mathfrak{a}' = \mathfrak{b}'$. Hence $Red(\bar{\mathfrak{a}}) = Red(\bar{\mathfrak{b}})$.

We identify each divisor class of the divisor class group with the unique reduced ideal. Hence we define the operation in the class group as ideal v-multiplication followed by reduction, i.e., all arithmetic will be performed with reduced ideals.

The security of our proposed public-key cryptosystems is based on the difficulty of the DLP in the divisor class group of an affine normal subring of $K[X,Y]$ generated by monomials, where K is a field. We believe that this problem is much more difficult than the DLP in the class group of an order of a quadratic field, because the ideal $(\mathfrak{a}^x)_v$ is "masked" by the operation v. That is, it is also very difficult to calculate $\mathfrak{a}^x$ from $(\mathfrak{a}^x)_v$.

Discrete Logarithm Problem in Divisor Class Groups. Given divisorial ideals $\mathfrak{a}$ and $\mathfrak{b}$, compute $x \in \mathbb{N}$ such that

$$[\mathfrak{b}] = [\mathfrak{a}]^x \qquad (\text{i.e., } \mathfrak{b} \sim (\mathfrak{a}^x)_v),$$

if such an x exists.

Although we can obtain analogs of well-known public-key cryptosystems based on the (original) DLP, we present here only an analog of the ElGamal encryption and of the Diffie-Helman key exchange system employing the divisor class group of the integral domain $R_{n,1}$.

4.1 Analog of ElGamal

Let n be any positive integer such that the DLP in $Cl(R_{n,1})$ is intractable and let $\mathfrak{a}$ be a reduced ideal of $R_{n,1}$ such that $[\mathfrak{a}]$ is a primitive element of $Cl(R_{n,1})$. Let $\mathfrak{b} = Red\big((\mathfrak{a}^a)_v\big)$. Let $\mathfrak{m}$ be the plaintext, where $\mathfrak{m}$ is a reduced ideal of $R_{n,1}$. For a secret random integer k $(1 \leq k < n)$, define

$$E(\mathfrak{m}, k) = (\mathfrak{c}_1, \mathfrak{c}_2),$$

where

$$\mathfrak{c}_1 = Red\big((\mathfrak{a}^k)_v\big) \qquad \text{and} \qquad \mathfrak{c}_2 = Red\big((\mathfrak{m}\mathfrak{b}^k)_v\big).$$

For two reduced ideals $\mathfrak{c}_1$ and $\mathfrak{c}_2$, define

$$D(\mathfrak{c}_1, \mathfrak{c}_2) = Red\Big(\big(\mathfrak{c}_2(\mathfrak{c}_1^a)^{-1}\big)_v\Big).$$

Verification. The decryption of the above algorithm allows recovery of original plaintext $\mathfrak{m}$ since

$$\begin{aligned} DE(\mathfrak{m}, k) &= D\Big(Red\big((\mathfrak{a}^k)_v\big),\ Red\big((\mathfrak{m}\mathfrak{b}^k)_v\big)\Big) \\ &= Red\bigg(\Big(Red\big((\mathfrak{m}\mathfrak{b}^k)_v\big)\big(\big(Red\big((\mathfrak{a}^k)_v\big)\big)^a\big)^{-1}\Big)_v\bigg) \\ &= Red\bigg(\Big(\big(\mathfrak{m}\mathfrak{b}^k\big)_v\big(((\mathfrak{a}^k)_v)^a\big)^{-1}\Big)_v\bigg) \\ &= Red\bigg(\Big(\mathfrak{m}\big(\mathfrak{b}^k(\mathfrak{b}^k)^{-1}\big)_v\Big)_v\bigg) \\ &= Red\big((\mathfrak{m})_v\big) \\ &= Red(\mathfrak{m}) \\ &= \mathfrak{m}. \end{aligned}$$

4.2 Analog of the Diffie-Helman Key Exchange

We now set up a method similar to that of [7] for a secret key exchange. Two users Alice and Bob select a positive integer n such that the DLP on $Cl(R_{n,1})$ is intractable and a reduced ideal $\mathfrak{a}$ of $R_{n,1}$. The integer n, the integral domain $R_{n,1}$, and the ideal $\mathfrak{a}$ can be made public.

(1) Alice selects a random integer x and computes a reduced ideal $\mathfrak{b}$ such that

$$\mathfrak{b} \sim (\mathfrak{a}^x)_v.$$

Alice sends $\mathfrak{b}$ to Bob.

(2) Bob selects a random integer y and computes a reduced ideal $\mathfrak{c}$ such that

$$\mathfrak{c} \sim (\mathfrak{a}^y)_v.$$

Bob sends $\mathfrak{c}$ to Alice.

(3) Alice computes a reduced ideal $\mathfrak{k}_1 \sim (\mathfrak{c}^x)_v$ and Bob computes a reduced ideal $\mathfrak{k}_2 \sim (\mathfrak{b}^y)_v$.

Verification. Since $\mathfrak{k}_1 \sim (\mathfrak{c}^x)_v \sim (((\mathfrak{a}^y)_v)^x)_v = (((\mathfrak{a}^x)_v)^y)_v \sim (\mathfrak{b}^y)_v \sim \mathfrak{k}_2$, Alice and Bob have a common secret key $\mathfrak{k}_1 = \mathfrak{k}_2$.

4.3 Security Aspects

The security of our proposed public-key cryptosystems is based on the difficulty of the DLP in the divisor class group of the integral domain $R_{n,1}$. For our proposed public-key cryptosystems, we can choose any positive integer n and any computable field K as a coefficient ring such that the DLP in $Cl(R_{n,1})$ is intractable. Thus, to avoid brute force attack, we have to choose a sufficiently large positive integer n. We do not know the key size for a secure system yet.

In [14,10], a subexponential-time algorithm for computing class groups of imaginary quadratic orders in number fields was invented by J. L. Hafner and K. S. McCurley and it was shown how to use this algorithm and the index-calculus method to calculate discrete logarithms. The improved algorithms for computing class groups to simplify the index-calculus algorithm in class groups were presented in [3,11]. To the best of our knowledge, there is no subexponential-time algorithm for computing discrete logarithm in our divisor class groups.

Acknowledgements

This research is partially supported by the Mobile Network Security Technology Research Center. The authors thank the referee for encouraging criticism.

References

1. Adams, W., Loustaunau, P.: An introduction to Gröbner bases, Graduate Studies in Mathematics vol. 3, Amer. Math. Soc., Providence, 1994.
2. Anderson, D. F.: Subrings of $k[X, Y]$ generated by monomials. Canad. J. Math. **30** (1978) 215–224.
3. Buchmann, J., Düllmann, S.: On the computation of discrete logarithm in class groups, in *Advances in Cryptology - CRYPTO '90*, LNCS 537, Springer-Velag, Berlin, 1991, pp. 134–139.
4. Buchmann, J., Willams, H. C.: A key-exchange system based on imaginary quadratic fields. J. Cryptology **1** (1988) 107–118.
5. Cox, D., Little, J., O'Shea, D.: Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra, Springer-Velag, New York, 1997.
6. Denning, D. E.: Cryptography and Data Security, Addison-Wesley, Reading, Massachusetts, 1982.
7. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Trans. Inform. Theory **22** (1976) 472–492.
8. Gilmer, R.: Multiplicative Ideal Theory, Queen's Papers in Pure and Applied Mathematics, vol 90, Queen's University, Kingston, Ontario, 1992.
9. Grayson, D., Stillman, M.: Macaulay 2, 1996. Available via anonymous ftp from math.uiuc.edu.
10. Hafner, J. L., McCurley, K. S.: A rigorous subexponential algorithm for computation of class group. J. Amer. Math. Soc. **2** (1989) 837–850.
11. Jacobson Jr., M. J.: Computing discrete logarithms in quadratic orders. J. Cryptology **13** (2000) 473–492.
12. Koblitz, N.: Elliptic curve cryptosystems. Math. Comp. **48** (1987) 203–209.
13. Koblitz, N.: Hyperelliptic cryptosystems. J. Cryptology **1** (1989) 139–150.
14. McCurley, K. S.: Cryptographic key distribution and computation in class groups, in R. A. Mollin, editor, *Number Theory and Applications*, Kluwer Academic Publishers, 1989, pp. 459–479.
15. Odlyzko, A. M.: Discrete logarithms in finite fields and their cryptographic significance, *Advances in Cryptology - EUROCRYPT '84*, LNCS 209, Springer-Velag, Berlin, 1985, pp. 224–314.

16. Paulus, S., Takaki, T.: A new public-key cryptosystem over a quadratic order with quadratic decryption time. J. Cryptology **13** (2000) 263–272.
17. Rivest, R. L., Shamir, A., Adelman, L.: A method for abtaining digital signatures and public key cryptosystems. Communications of the ACM **21** (1978) 120–126.
18. Vasconcelos, W. V.: Computational Methods in Commutative Algebra and Algebraic Geometry, Algorithms and Computation in Mathematics, vol. 2, Springer-Velag, Berlin, 1998.
19. Winkler, F.: On the complexity of the Gröbner-bases algorithm over $K[x, y, z]$. *EUROSAM '84*, LNCS 174, Springer, Berlin, 1984, pp. 184–194.

First Implementation of Cryptographic Protocols Based on Algebraic Number Fields

Andreas Meyer, Stefan Neis, and Thomas Pfahler

Darmstadt University of Technology
Alexanderstr. 10
D-64283 Darmstadt, Germany
{amy,neis,pfahler}@cdc.informatik.tu-darmstadt.de

Abstract. We show for the first time how to implement cryptographic protocols based on class groups of algebraic number fields of degree > 2. We describe how the involved objects can be represented and how the arithmetic in class groups can be realized efficiently. To speed up the arithmetic we present our new method for multiplication of ideals. Furthermore we show how to generate cryptographically suitable algebraic number fields. Besides, we give a numerical example and analyse our run times.

1 Introduction

Succesful e-business requires secure authentication and binding communication. To reach this goal one uses digital signature schemes. Basically, the public key cryptosystems (including the signature schemes) used today in practice are based on the following two families of computational problems:

1. the integer factoring problem and the discrete log problem in finite fields (e.g. RSA or DSA)
2. the discrete log problem in the group of points of an elliptic curve over a finite field (e.g. ECDSA)

But it is absolutely unclear whether these problems remain difficult in the future. On the contrary, in the last 15 years there was very big progress regarding the development of efficient factoring algorithms and discrete log algorithms for finite fields ([21]). Furthermore, the crypto community found again and again algorithms that solve very efficiently the discrete log problem for families of elliptic curves over finite fields such that these are useless for cryptography ([27, 15, 36, 37]). **Therefore, a major task of today's public key cryptography is the search for new computational problems which can be used for the construction of secure and efficient public key cryptosystems.**

In [9] the discrete log problem in class groups of algebraic number fields (*NFDL*) was suggested. Recently the *root problem* was introduced as special case of NFDL [7]: Given a class group Cl of an order of an algebraic number field, a prime number p which does not divide the order of Cl, and a group element α, find the pth root of α. The NFDL and the root problem seem to be computational problems as desired: The best known algorithms require the solution of an index calculus problem and many shortest vector problems in lattices. Firstly the complexity is therefore subexponential in the binary length of the discriminant and exponentially in the degree of the number field (see [9]). Secondly solving NFDL hence is independent of the basic problems in public key cryptography used today.

In practice, however, many problems remained to be solved. Since no efficient algorithm is known for computing the class number (i.e. the order of the class group), the well known signature protocols (such as DSA, ElGamal, RSA) are not applicable over number fields. Very recently, modifications of these protocols resulted in protocols which can be used in number fields. They were introduced and analysed in [6, 7]. It was shown why the root problem in class groups of algebraic number fields proves difficult. Note that optimized implementations of signature schemes over imaginary quadratic number fields (degree of the number field is 2) are more efficient than implementations of the RSA signature scheme ([7]). In this paper we consider number fields of degree > 2. The root problem of these number fields is even harder than in the imaginary quadratic case (see [9]). But number fields of degree > 2 raise further problems: As far as the arithmetic is concerned, deciding equality of ideal classes is far from being trivial. Another difficulty arises from the generation of cryptographically suited class groups, since in general, class groups of number fields of degree > 2 are very small whereas we need large class groups.

Although approaches to these problems are known in theory, no implementation of a cryptosystem based on class groups of algebraic number fields of degree > 2 was done so far. In this paper, we describe our first implementation of such a cryptosystem (signature scheme). Besides, we explain our new method for the multiplication of ideals which leads to a faster signing and verification procedure.

This paper is organized as follows:

In section 2 we will explain our representation of the mathematical objects and the arithmetic in number fields and class groups. Particularly, we shall explain our new method for the multiplication of ideals and describe our implementation of equality decision in class groups.

In section 3, we shall discuss some requirements on cryptographically good orders of algebraic number fields. We will suggest instances of cryptographically good orders.

In section 4, we will present the RDSA signature scheme. Besides we shall give an explicit example and run times.

In a final section 5, we shall argue that cryptosystems based on algebraic number fields do have the potential to become practical in the future.

2 Efficient Arithmetic for Algebraic Number Fields

In the sequel we use the following notation: Let $\mathcal{K}$ be an algebraic number field (in the following only called "'number field"') of degree n with signature (r, s)

and with generating polynomial f. By $\sigma_1, \ldots, \sigma_r$ we denote the real embeddings and by $\sigma_{r+1}, \overline{\sigma_{r+1}}, \ldots, \sigma_{r+s}, \overline{\sigma_{r+s}}$ the nonreal embeddings of $\mathcal{K}$ into $\mathbb{C}$. Let $\mathcal{O}$ be the maximal order of $\mathcal{K}$. (Fractional) ideals are called $\mathfrak{a}, \mathfrak{b}, \mathfrak{c}, \ldots,$ for prime ideals we typically use $\mathfrak{p}_i$. By Cl and by $h = |\mathrm{Cl}|$ we denote the class group and the class number of $\mathcal{K}$. For an ideal $\mathfrak{a}$ we denote its equivalence class in the class group by $[\mathfrak{a}]$. Finally, let Δ be the discriminant of the number field $\mathcal{K}$. For a general introduction into the theory of number fields see for instance [2].

We explain in this section how to represent the mathematical objects that we use in our implementation of cryptographic protocols. Some of the material was already presented in [29]. Furthermore, we describe the basic algorithms we need for implementing the cryptographic protocols. Besides, we explain our new (faster) method for the multiplication of ideals and how we decide the equality of ideal classes.

2.1 Representation of the Objects

Representing Number Fields and Orders. The key for representing number fields and orders is the need to describe the multiplication of two algebraic numbers in the field or the order. So we represent the algebraic number field by its generating polynomial f. Computation in the number field then is the same as polynomial arithmetic mod f. Therefore, we can represent an algebraic number α by a polynomial mod f, i.e. we have $\alpha = \frac{1}{d} \sum_{i=0}^{n-1} a_i x^i$ with $d, a_i \in \mathbb{Z}$.

Moreover, this implicitely represents the order $\mathbb{Z}[x]/(f)$ consisting of those numbers with denominator $d = 1$. However, in general we do not want to compute in this specific order but rather in the maximal order, which cannot always be represented in this way. For this reason we additionally store a transformation matrix $T \in \mathbb{Z}^{n \times n}$ and a denominator d to describe the order with an $\mathbb{Z}$ basis $(\omega_0, \ldots, \omega_{n-1}) = (1, x, ..., x^{n-1}) \cdot \frac{1}{d} T$, where $\omega_i \in \mathbb{Z}[x]/(f)$. Arithmetic could then be done by using the matrices T and T^{-1} and doing polynomial arithmetic. However, we do this only if $T = I_n$, otherwise we use an additional multiplication table which describes how two elements of the basis are multiplied, i.e. we store $w_{i,j,k}, 0 \leq i, j, k < n$, with $\omega_i \cdot \omega_j = \sum_{k=0}^{n-1} w_{i,j,k} \omega_k$. Such a multiplication table is also sufficient to describe the order or number field, thus we can even omit the polynomial in this case.

Representing Algebraic Numbers. We store algebraic numbers as a coefficient vector with respect to a given $\mathbb{Z}$ basis and a denominator, where all coefficients and the denominator are numbers in $\mathbb{Z}$, i.e. $\alpha = \frac{1}{d} \sum_{i=0}^{n-1} a_i \omega_i$ is represented by the tuple $(d, (a_0, \ldots, a_{n-1}))$. Then addition and subtraction can be done componentwise (once we found the common denominator of the two numbers involved) and multiplication and division can be done using either polynomial arithmetic or the multiplication table mentioned above.

Representing Ideals. (Fractional) ideals of the maximal order $\mathcal{O}$ can be represented by an $\mathbb{Z}$ basis that contains n algebraic integers. Choosing the coefficient vector in $\mathbb{Q}^n$ to represent an algebraic integer and extracting the common denominator, one can represent an ideal $\mathfrak{a}$ by a $n \times n$ matrix A with integers entries whose absolute value is bounded by $|\Delta|$ and a denominator in $\mathbb{Z}$. This is the commonly used $\mathbb{Z}$ basis representation of ideals ([10]). We now explain our more efficient method: We can determine the exponent of such an ideal, i.e. the smallest positive number $e \in \mathbb{Z}$ with $e\mathcal{O} \subset \mathfrak{a}$. Then the ideal can be represented by e and $\bar{A}$, where $\bar{A}$ is a matrix with entries in $\mathbb{Z}/e\mathbb{Z}$ obtained from A by reducing each entry mod e. We store $\bar{A}$ as a reduced matrix – preferably with as few columns as possible as described in [29]. There is a way to represent $\mathfrak{a}$ by a uniquely determined $\bar{A}$ first described by Howell [18], but this method maximizes the number of columns of $\bar{A}$. Instead we use heuristics to try to have an $\bar{A}$ with fewer columns. This leads to a representation that is not unique. Moreover, computing e consumes some time and is not always needed, as any multiple of e will do as well in this representation, possibly at the cost of having more columns in $\bar{A}$ than necessary. In the following we shall call this representation the LiDIA representation of ideals ([23]).

Equality of two ideals is then decided by first determining the true exponent of both ideals and then computing the unique representation of the module generated by the columns of $\bar{A}$ as described by Howell.

Representing Prime Ideals. For efficiency reasons, prime ideals are represented in a different way, as we know that in our applications prime ideals are typically used to compute power products of prime ideals. Furthermore, when dealing with class groups, we typically have to handle many prime ideals. Thus we optimize for space efficiency and for an optimized computation of power products: We represent a prime ideal $\mathfrak{p}$ by a prime $p \in \mathbb{Z}$ and an algebraic integer π, such that $\mathfrak{p} = p\mathcal{O} + \pi\mathcal{O}$. Thus, an ideal $\mathfrak{a}$ can be multiplied by $\mathfrak{p}$ by adding $p\mathfrak{a}$ and $\pi\mathfrak{a}$, and powers of $\mathfrak{p}$ can also easily be computed ($\mathfrak{p}^k = p^{\lceil k/z \rceil}\mathcal{O} + \pi^k\mathcal{O}$, where z denotes the ramification index of p at $\mathfrak{p}$).

Representing Ideal Classes. Ideal classes are represented by any of the LLL-reduced ideals (see below), that are members of the class. Therefore, an ideal class is represented by $(e, \bar{A})$ where the positive integer e is the exponent of an LLL-reduced ideal in the given ideal class ($1 \leq e \leq |\Delta|$) and $\bar{A}$ is a $n \times r$ matrix ($1 \leq r \leq n$) with entries in $\mathbb{Z}/e\mathbb{Z}$. (The LLL-reduced ideals can roughly be seen as the equivalent of "small" remainders in computations modulo an integer.) Note that this representation is not unique.

2.2 Basic Algorithms in Number Fields

Let $\mathcal{K}$ be an number field, $\mathcal{O}$ an order of $\mathcal{K}$ and $\mathfrak{a}, \mathfrak{b}$ two ideals of $\mathcal{O}$.

The Group Operation. As explained in the previous subsection, we represent group elements (ideal classes) by one of its reduced representatives. Thus, we realize the group operation $[\mathfrak{a}] \cdot [\mathfrak{b}]$ by multiplying the representing ideals and LLL-reducing the resulting ideal $\mathfrak{ab}$.

Reducing an Ideal. Ideal reduction is done as follows: Determine Minkowski's embedding which leads to a lattice. Using the *LLL* algorithm calculate a short vector of this lattice. This is the representation of an *LLL*-reduced ideal in the ideal class $[\mathfrak{a}]$.

Ideal Multiplication. For ideal multiplication we use different algorithms depending on the type of ideal. If we multiply by an prime ideal we can use the method mentioned above; in general we need to determine a matrix whose columns generate the product. That matrix will have n^2 columns in the worst case and then needs to be reduced to a matrix with at most n columns as described in [29]. In the following section we will introduce a new faster multiplication for ideals which we will use in our signature scheme for signing a message.

Note that using the algorithm for group multiplication we can realize efficient exponentiation in the group [28].

Determining the Maximal Order. For computing the maximal order, even the simple Round 2 algorithm, as described e.g. in [10] is sufficient.

Choosing a Group Element at Random.

Theorem 1 ([1]). *Let $\mathcal{K}$ be an number field with discriminant Δ. Under the generalized Riemann hypothesis (GRH), the set of all prime ideals (more exactly, of their classes) with norm $\leq 12(ln\,|\Delta|)^2$ form a generating system of the class group of $\mathcal{K}$. Depending on the signature of the field there are better bounds for the norm known, e.g. the set of all prime ideals with norm $\leq 6(ln\,|\Delta|)^2$ in the case of imaginary quadratic number fields.*

Thus in this situation, we can choose an ideal class at random as follows:

1. Precomputation: Compute a generating system consisting of prime ideals due to Bach's theorem: $\{\mathfrak{p}_1, \ldots, \mathfrak{p}_k\}$
2. Choose an exponent vector at random: $(e_1, \ldots, e_k)$ where $1 \leq e_i \leq |\Delta|$.
3. Compute $\mathfrak{a} = \prod_{i=1}^{k} \mathfrak{p}_i^{e_i}$ and return $[\mathfrak{a}]$.

In practice we can use much smaller bounds than that given in Bach's theorem for determining a generating system. For example, Neis [30] succeeds in his experiments using norm bounds of size $O((ln\,|\Delta|)^{1.5})$ for class groups of number fields of degree $3, 4, 6$. Further experiments are needed: The larger the set the larger is the probability of getting a generating system. The smaller the set the more efficient the determination of a power product, but the less random the result.

Probability Distribution of the Pseudo–Random Element Choice Algorithm. We show that the described algorithm for pseudorandomly choosing an element of the class group of an number field leads to a distribution that is "almost" uniform.

Proposition 1. *(Theorem 5.2 of [22]) Consider the imaginary quadratic number field $\mathcal{K}$ with discriminant $\Delta \in \mathbb{Z}_{<0}$, its class group* Cl*, and its class number $h = |\mathrm{Cl}|$. Let $\mathcal{G}$ be a generating system of* Cl*, $[\mathfrak{a}] \in \mathrm{Cl}$. Then the number of vectors $r = (r([\mathfrak{p}]))_{[\mathfrak{p}]\in\mathcal{G}} \in \{1, 2, \ldots, |\Delta|\}^{|\mathcal{G}|}$ solving $\prod_{[\mathfrak{p}]\in\mathcal{G}}[\mathfrak{p}]^{r([\mathfrak{p}])} = [\mathfrak{a}]$ equals*

$$\frac{|\Delta|^{|G|}}{h} \cdot exp(\varepsilon) \ ,$$

where $\varepsilon \in \mathbb{R}$, $|\varepsilon| < \frac{h}{|\Delta|-h} < 1$. For sufficiently large $|\Delta|$ we have $|\varepsilon| \leq ln\, 2$.

We would have a uniform distribution, if the number of such exponent vectors r did not depend on the special choice of an ideal class $[\mathfrak{a}] \in \mathrm{Cl}$. The difference between our distribution and the uniform distribution is characterized by the constant $exp(\varepsilon)$ which depends on $[\mathfrak{a}]$. The closer $exp(\varepsilon)$ is to 1, the more uniform the resulting distribution is. For sufficiently large $|\Delta|$ we have $|\varepsilon| \leq ln\, 2$, i.e. $0.5 \leq exp(\varepsilon) \leq 2$.

By analogy with the proof of the proposition above one can prove:

Theorem 2. *Proposition 1 holds for all number fields $\mathcal{K}$.*

Thus as we have shown, the described algorithm for pseudorandomly choosing an element of the class group of an number field leads to a distribution that is "almost" uniform.

2.3 Advanced Algorithms in Number Fields

More Efficient Ideal Multiplication.

The Two-Element Representation of Ideals. Each fractional ideal $\mathfrak{a}$ of the maximal order $\mathcal{O}$ of $\mathcal{K}$ has a *two-element representation*

$$\mathfrak{a} = a\mathcal{O} + \alpha\mathcal{O}$$

where $a \in \mathbb{Z}_{>0} \cap \mathfrak{a}, \alpha \in \mathfrak{a}$ ([31]). Note that this representation of an ideal only needs $n+1$ integers instead of n^2 integers regarding the $\mathbb{Z}$ basis representation and $n \cdot r$ integers $(1 \leq r \leq n)$ regarding the LiDIA representation from the previous subsection.

Determining Two-Element Representations. Given an ideal $\mathfrak{a}$ in LiDIA representation we want to determine one of its two-element representations. We proceed as follows: We determine a $\mathbb{Z}$ basis representation of $\mathfrak{a}$. After, we choose as first generator the integer $a = N(\mathfrak{a})$. Then we test for all $\alpha \in \left\{\sum_{i=0}^{n-1} b_i\alpha_i : \ b_i \in \{-1, 0, 1\}\right\}$ whether $a\mathcal{O} + \alpha\mathcal{O} = \mathfrak{a}$. Since $a\mathcal{O} + \alpha\mathcal{O} \subseteq \mathfrak{a}$ it is sufficient to check whether $N(a\mathcal{O}+\alpha\mathcal{O}) = N(\mathfrak{a})$. In this case we actually have $a\mathcal{O} + \alpha\mathcal{O} = \mathfrak{a}$. Note that although this algorithm is probabilistic it works fine in practice.

Efficient Ideal Multiplication. Given an ideal $\mathfrak{a}$ in LiDIA representation $(e, \bar{A})$ and an ideal $\mathfrak{b} = b\mathcal{O} + \beta\mathcal{O}$ in two-element representation we can determine their product in LiDIA representation very efficiently using the equation

$$\mathfrak{a}\mathfrak{b} = \mathfrak{a} \cdot b + \mathfrak{a} \cdot \beta \,.$$

For determining $\mathfrak{a} \cdot b$ we multiply all entries of $\bar{A}$ and the module e with the integer b. For determining $\mathfrak{a} \cdot \beta$ we firstly multiply the algebraic integer β with all columns of the $\mathbb{Z}$ generating system $(\underbrace{\bar{A}}_{r} \mid \underbrace{e \cdot I_n}_{n})$ of $\mathfrak{a}$. Secondly, we reduce the resulting matrix of $n + r$ columns to a matrix with at most n columns using the Hermite normalform (HNF) computation (see [10, 40]). For adding $\mathfrak{a} \cdot b$ and $\mathfrak{a} \cdot \beta$ we substantially concatenate the columns of the representations of $\mathfrak{a} \cdot b$ and $\mathfrak{a} \cdot \beta$ and reduce the resulting matrix of at most $n + r$ columns to at most n columns using the HNF computation. For the details we refer to [29].

The Tables 1,2,3 compare the run times for the multiplication of two ideals in LiDIA representation with the run times for the multiplication of the same ideals where one is given in LiDIA representation, the other is given in two-element representation. We calculated in these experiments in *Stender fields* of degrees $n = 3, 4$, and 6:

$$\mathcal{K} = \mathbb{Q}(\sqrt[n]{D^n + 1}) \qquad D \in \mathbb{Z}_{\geq 10^{10}}$$

Stender fields shall prove to be suitable for cryptographic purposes, see section 3.2. We multiplied pseudo-random LLL-reduced ideals. The last column shows the speed-up factor of the new method for ideal multiplication ("LiDIA repres. $*$ Two-Elt repres.") compared with our usual method ("LiDIA repres. $*$ LiDIA repres."). All run times are given in milliseconds on an average (100 iterations). The run times were made on a Celeron 433 MHz processor. The run times show that our new method for the multiplication of ideals leads to a speed-up of factor $4.5, 8, 14$ for degrees of number fields $3, 4, 6$, resp., compared with the usual method. Note that the classical ideal multiplication using $\mathbb{Z}$ basis representations of ideals are even slower why we had used originally the LiDIA representation of ideals (see [30] for comparing run times).

Table 1. Run times for ideal multiplication, degree $n = 3$

D	**LiDIA $*$ LiDIA**	**LiDIA $*$ Two-Elt**	**factor**
$\sim 10^{10}$	6.205	1.54	4.02922
$\sim 10^{20}$	6.575	1.547	4.25016
$\sim 10^{30}$	6.630	1.364	4.8607
$\sim 10^{40}$	6.746	1.434	4.70432

We shall explain in section 4 how we can use this faster multiplication method for our cryptosystem.

Table 2. Run times for ideal multiplication, degree $n = 4$

D	LiDIA $*$ LiDIA	LiDIA $*$ Two-Elt	factor
$\sim 10^{10}$	18.387	2.911	6.31639
$\sim 10^{20}$	18.765	3.109	6.0357
$\sim 10^{30}$	20.188	2.264	8.91696
$\sim 10^{40}$	20.601	2.423	8.50227

Table 3. Run times for ideal multiplication, degree $n = 6$

D	LiDIA $*$ LiDIA	LiDIA $*$ Two-Elt	factor
$\sim 10^{10}$	104.850	7.797	13.4475
$\sim 10^{20}$	111.676	8.037	13.8952
$\sim 10^{30}$	116.265	9.427	12.3332
$\sim 10^{40}$	126.866	8.285	15.3127

Deciding Equality of Ideal Classes.
Let $\mathfrak{a}$ and $\mathfrak{b}$ be two ideals in the maximal order of a fixed number field $\mathcal{K}$ of degree n. We explain now how to decide whether or not $[\mathfrak{a}] = [\mathfrak{b}]$.

First, let $n = 2$. If $\mathcal{K}$ is an imaginary quadratic field, there exists exactly one reduced ideal in each ideal class, and there is a polynomial time reduction algorithm. Reduce the given ideals $\mathfrak{a}$ and $\mathfrak{b}$. We have $[\mathfrak{a}] = [\mathfrak{b}]$ if and only if the reduced ideals are equal. Alternatively, we can take the following approach and decide the equivalence of two ideals with a test for a principal ideal: $[\mathfrak{a}] = [\mathfrak{b}]$ if and only if $[\mathfrak{a}\mathfrak{b}^{-1}] = [\mathcal{O}]$, i.e. if $\mathfrak{a}\mathfrak{b}^{-1}$ is a principal ideal $\alpha\mathcal{O}$ with $\alpha \in \mathcal{K}$.

If $\mathcal{K}$ is a real quadratic field, the number of reduced ideals equivalent to $\mathcal{O}$ can in practice efficiently be computed, provided the regulator of $\mathcal{K}$ is small (see for instance [10]). In this case the principal ideal test consists of the reduction of $\mathfrak{a}\mathfrak{b}^{-1}$ followed by a table lookup.

A similar method is applied for number fields of degree larger than 2; however, determining all reduced principal ideals is much more complicated and is the most difficult problem when implementing cryptographic schemes in class groups of number fields of degree > 2.

We will explain the terms introduced above more precisely: If $\mathfrak{a}$ is a (fractional) ideal in $\mathcal{O}$, then we call a number $\mu \in \mathfrak{a}$ a *minimum* of $\mathfrak{a}$ if there is no element $\alpha \neq 0$ in $\mathfrak{a}$ such that $|\sigma_i(\alpha)| < |\sigma_i(\mu)|$ for $1 \leq i \leq m$. The ideal $\mathfrak{a}$ is called *reduced* if the smallest positive rational integer in $\mathfrak{a}$ is a minimum of $\mathfrak{a}$. The set of all reduced ideals in $[\mathfrak{a}]$ is called *cycle of reduced ideals* in the class of $\mathfrak{a}$ (despite the fact that its structure is in general much more complicated than in the case of real quadratic fields where one really has a "cycle"). Its cardinality is finite and called *period length* of $\mathfrak{a}$. It was proven in [5] that the period length is $O(\mathcal{R})$, where $\mathcal{R}$ denotes the regulator of $\mathcal{K}$. Indeed, the cycle of reduced ideals is only effectively computable if $\mathcal{R}$ is small. It is easy to see that those number fields whose maximal order has period length 1 are especially suited for our purposes. In the next chapter, we will give instances of such fields.

We have implemented an efficient version of an algorithm for computing all reduced principal ideals. The algorithm is a number geometric generalization of Lagrange's continued fraction algorithm and was presented by Buchmann [3]. It was primarily designed for computing fundamental units, but it also allows to compute the set of all reduced ideals in any given ideal class. Among the major difficulties when implementing the algorithm was the correct handling of the precision for approximations of real numbers, and the efficient calculation of short vectors in integer lattices. With the information computed by this algorithm, we are able to decide the equality of ideal classes in arbitrary number fields.

3 Cryptographically Good Orders of Number Fields

In this section we discuss in a first part conditions for orders whose class groups are to be used to implement the RDSA signature scheme from [7]. The statements hold for the cryptographic schemes presented in [6], too. Therefore, a lot of signature schemes could be implemented over algebraic number fields in the same way. In a second part of this section we present families of orders of number fields which fulfill the requirements listed before.

3.1 Requirements for Good Orders

The security of RDSA is based on the root problem. There are polynomial time reductions from the root problem to the order problem (find a non zero multiple of the order of a given group element) and from the order problem to the discrete logarithm problem [7]. Therefore, as necessary conditions for the security of RDSA those problems must remain intractable in our (class) group, particularly under the usage of the known algorithms solving those problems. Given a prime p and a finite abelian group extracting the p-th root of a group element α is to our knowledge only possible if a multiple of $ord\,\alpha$ is known, for example the group order. Therefore we must ensure that the known algorithms for determining the group order (class number) and discrete logarithms in class groups of orders of number fields will fail. Note that the calculation of the class number h is a very hard computational problem. Its complexity is subexponential in the binary length of the discriminant and exponential in the degree of the number field [9]. In the special case of imaginary quadratic number fields, computing the class number or computing discrete logarithms is at least as difficult as factoring integers [35].

Our analysis leads to the following necessary conditions for the class number h of such orders:

- *h must be large, i.e. the regulator R should be small.*
  This condition prevents the success of the following algorithms for determining the class number or discrete logarithms: The exhaustive search method,

Pollard's Rho method [33], Shanks' Baby-Step-Giant-Step-algorithm [10] including all variants (e.g. [8]), the Hafner-McCurley algorithm [26, 16], and the index-calculus-algorithms (e.g. COS [14] or NFS [41]).
From the Brauer–Siegel–Theorem (see for instance [19]) we know that for sufficiently large absolute values $|\Delta|$ of the discriminant the product of regulator and class number is of the order of magnitude of $\sqrt{|\Delta|}$. As our experiments show, we already have $hR \sim \sqrt{|\Delta|}$ for discriminants of a few hundred bits, which is the order of magnitude we are interested in. Unfortunately, the regulator is typically large and the class number is small (see [11, 12, 13]). There are, however, infinite families of number fields with small regulators, thus with large class numbers, as we will see in the next subsection.

– *h must have sufficiently large prime divisors.*
We have to prevent a Pohlig-Hellman attack for the following DL-problem: Given elements $\alpha, \beta \in \mathrm{Cl}$ chosen at random such that β is in the subgroup generated by α, determine an integer x such that $\alpha^x = \beta$.
If the class number can be determined, it is therefore necessary that the order of α has a sufficiently large prime divisor such that the discrete log algorithms mentioned above do not succeed in the cyclic subgroups of the group generated by α. This happens with very high probability as long as the class number has a sufficiently large prime divisor ([7, Th. 4]).
If the class number cannot be determined in practice (which is the typical case), it seems to be a (weaker) sufficient condition that the class number (thus with high probability the element order as well) contains several primes of medium size whose product is greater than some large bound (see the appendix for details). This prevents the success of the following discrete log algorithm: First determine a multiple of $ord\, \alpha$ using a method similar to Pollard's $p-1$ factorization method ([32], see also [20]). Then apply the Pohlig-Hellman algorithm. The first step does not succeed in our situation because of a combinatorical explosion since $ord\, \alpha$ has several primes of medium size. Namely, given a group element α, for finding a multiple of the B-smooth integer ord α one needs $O(B \cdot ln\, m / ln\, B)$ group operations, provided that an upper bound m for the element order ord α is known (see [20]).

3.2 Constructing Good Orders

We consider briefly the special case $n = 2$. Imaginary quadratic number fields always have regulator $R = 1$, thus large class numbers h (if the discriminant is large in absolute terms). In [6], it was shown why their maximal orders are suitable for cryptographic applications. Examples for suitable maximal orders of real quadratic number fields are given in [6], too.

In this paper we are interested in number fields of degree > 2 because we want to take advantage of the complexity of the NFDL and the root problem which is exponential in the degree n of the number field. In the following we present several examples for suitable families of orders of number fields.

Stender Fields. Stender [39] considers number fields

$$\mathcal{K} = \mathbb{Q}(\sqrt[n]{D^n \pm d}), \tag{1}$$

with defining polynomial $f(x) = x^n - (D^n \pm d)$ where $n \in \{3, 4, 6\}$, $D, d \in \mathbb{Z}_{>0}$, and d satisfies some further condition. We call them *Stender fields*. Stender explicitly determines a system of fundamental units of $\mathcal{K}$ which leads to the exact value for the regulator. The bounds of Table 4 (see [7, 25]) and the explicit examples from [7] show that the class numbers of Stender fields (at least in the special case $d = 1$) are large while their regulators are small. (Note that for $n = 3$ we assume in the table that $D^3 \pm 1 \geq 3 \cdot 10^4$ is cube-free). Furthermore, it can be deduced from the heuristics of Cohen and Martinet [12, 13] the class numbers seem to contain a large prime divisor. that the class numbers of number fields with small regulator have at least one large prime divisor. The experiments of Neis [30] confirm this.

Table 4. Bounds for regulator and class number of Stender fields, $d = 1, D > 16$, (for details about the constant c, see [38])

Degree	Upper bound for the regulator	Lower bound for the class number
$n = 3$	$R \leq \frac{2}{3} ln(3 \cdot (D^3 \pm 1))$	$h \geq \frac{1}{6}\sqrt{\frac{D^3 \pm 1}{ln^3(3 \cdot (D^3 \pm 1))}}$
$n = 4$	$R < 4 \cdot (\ln(\sqrt{3} \cdot D))^2$	$h > \frac{c}{136\pi} \cdot \frac{D^3}{(\ln(\sqrt{3} \cdot D))^2}$
$n = 6$	$R < 9324 \cdot \ln(2D)$	$h > \frac{c}{35812448 \cdot \pi^2} \cdot \frac{D^{10}}{\ln(2D)}$

Stender fields appear to have another advantage in the context of effective computations in class groups: For $n = 3$ and $d = 1$, with the exception of very few, but easily checkable cases, the period length of the maximal order is exactly 1 (i.e. $\mathcal{O}$ is the only reduced principal ideal, see section 2.3) if $D \not\equiv 0 \bmod 3$ [42]. For $n \in \{4, 6\}$ we conjecture that there are infinite classes of Stender fields with period length 1, too; our experiments seem to confirm this conjecture (more detailed results will be published in a subsequent paper). As this property greatly simplifies the equivalence test for ideals, such number fields might prove especially useful for cryptographic applications.

Buchmann's Number Fields of Degree 4. Let $\mathcal{K} = \mathbb{Q}(\sqrt[4]{-D})$, $D = 4k^4 + d$, $k \in \mathbb{Z}_{>0}$, $d \in \mathbb{Z}$ where $0 < |d| \leq 4k$. We consider the order $\mathcal{O} = \mathbb{Z}(\sqrt[4]{-D})$ of $\mathcal{K}$.

Depending on the values of d and k Buchmann [4] has determined the fundamental unit of $\mathcal{O}$ and the set of all minimas of $\mathcal{O}$. Buchmann's result is valuable for us for two reasons: Firstly, with the knowledge of the fundamental unit of $\mathcal{O}$

we can see that the regulator of $\mathcal{O}$ is very small, thus the class number is large. Secondly, the inverses of the minimas are the generators of the reduced principal ideals. According to [4] the number of reduced principal ideals is very small in all cases:

Case	# reduced principal ideals in $\mathcal{O}$
$d = 1$	1
$d > 1$	2
$k \geq 2$ and $d = -1$	2
$k \geq 2$ and $d < -1$ and $d \mid 4k$	4

Therefore, using the order $\mathcal{O}$ of the number field mentioned above we can easily compute the cycle of reduced ideals in $\mathcal{O}$ and, thus, decide equality of ideal classes. Obviously, the most efficient choice is $d = 1$.

Real Cubic Number Fields. Let ρ_l be the uniquely determined real root of the irreducible polynomial $f(X) = X^3 + lX - 1$ ($l \in \mathbb{Z}_{\geq 1}$) and let $\mathcal{K} = \mathbb{Q}(\rho_l)$. Then $\epsilon_l := \frac{1}{\rho_l}$ is a fundamental unit [25] and we have $l < \epsilon_l < l + 1$.

Choose l such that $3^2 \nmid l$ and $p^2 \nmid \Delta_l = 4l^3 + 27$ for all primes $p \geq 5$. (E.g. choose l such that the absolute value Δ_l of the discriminant of $\mathcal{K}$ is prime.) According to [25], the maximal order of $\mathcal{K}$ is $\mathcal{O} = \mathbb{Z}[\epsilon_l]$ and for $\Delta_l \geq 2 \cdot 10^5$ we have

$$h \geq \frac{\Delta_l}{20 ln^2 \Delta_l} \ .$$

Totally Imaginary Number Fields of Degree 4. Let $\mathcal{K} = \mathbb{Q}(i, \sqrt{\delta})$ where $\delta = m^2 + 4i$ is square-free in $\mathbb{Z}[i]$, $m = a + ib \in \mathbb{Z}[i]$ with $a > b \geq 0$, $a \not\equiv b \mod 2$. Let $D := |\delta|^2$. Then we obtain from [24]:

$$R \leq ln(\sqrt{D}) \text{ and } h \geq \frac{1}{17} \frac{\sqrt{D}}{ln^2 D} \quad \text{for } D \geq 2 \cdot 10^9 \ .$$

Note that $|\Delta| = 16D$. Therefore the class number grows sufficiently quickly.

Instances for such fields can be found for example by choosing $p > 2$ prime where $5 \nmid p^2 + 1$, $a = 2p$, $b = p$, $m = a + bi$, $\delta = m^2 + 4i$.

4 Computational Results

4.1 The Signature Scheme RDSA

Using the representations and algorithms of section 2 we implemented RDSA, a variant of the ElGamal signature scheme which does not require the knowledge of the group order. During the signature and verification process the exponents here are reduced modulo a large prime instead of the group order. This variant was described in [7]. Many well known signature schemes can be modified such that they can be implemented without knowledge of the group order, e.g. in class groups of number fields (see [6]). Here, we describe the RDSA signature scheme in terms of a multiplicatively written finite abelian group G. In the description A is the signer, B is the verifier, and $M \in \{0,1\}^*$ is the message to be signed. Moreover, for $x, y \in \{0,1\}^*$ we denote by $x \| y$ the concatenation of x and y. We also use a cryptographic hash function h which map strings in $\{0,1\}^*$ to $\{0, 1, \dots, p-1\}$ for some positive integer p. In our situation, G is the class group of one of the orders of number fields presented in section 3.2.

1. **Key generation**
   A randomly selects an element $\gamma \in G$ and a prime p;
   A randomly selects an integer a where $1 < a < p$ and computes $\alpha = \gamma^a$;
   A's public key is (G, γ, α, p), the private key is a.
2. **Signature**
   A randomly selects an integer k such that $0 \leq k < p$;
   A computes $\varrho = \gamma^k$;
   A computes $x = a + kh(M \| \varrho)$;
   A computes nonnegative integers s and ℓ such that $x = \ell p + s$ with $0 \leq s < p$;
   A computes $\lambda = \gamma^\ell$;
   the signature of the message M is $S = (s, \varrho, \lambda)$.
3. **Verification**
   B accepts if and only if $1 \leq s < p$ and $\gamma^s = \alpha \varrho^{h(M \| \varrho)} \lambda^{-p}$.

4.2 Example

To illustrate the algorithm, we give a small example:

Key generation. We use the number field $\mathcal{K} = \mathbb{Q}(\sqrt[3]{123^3+1})$. Its maximal order has period length 1, i.e. there is exactly one reduced principal ideal. We set $p = 731921033138277435612152393899$, and choose the ideal γ with $\mathbb{Z}$ basis

$$\begin{pmatrix} 115956 & 77304 & 110668 \\ 0 & 38652 & 20224 \\ 0 & 0 & 1 \end{pmatrix}$$

and $a = 458967366586392074529404946651$. Then we compute $\alpha = \gamma^a$, the public key, which is in the ideal class of the ideal with $\mathbb{Z}$ basis

$$\begin{pmatrix} 65564 & 0 & 9000 \\ 0 & 65564 & 63680 \\ 0 & 0 & 1 \end{pmatrix}.$$

Signature. We generate the signature for the message "`Hello World!`" by choosing $k = 485477408517287794924521196370$ and computing the ideal ρ as described above. In our implementation, we use the `SHA-1` algorithm for hashing; thus we yield $s = 267317238294110283157650658417$ and the ideal λ with an $\mathbb{Z}$ basis

$$\begin{pmatrix} 360587 & 0 & 323678 \\ 0 & 360587 & 328756 \\ 0 & 0 & 1 \end{pmatrix}.$$

Verification. Finally, in the verification step, both sides of the equation evaluate to

$$\begin{pmatrix} 302076 & 0 & 193186 \\ 0 & 151038 & 85726 \\ 0 & 0 & 1 \end{pmatrix}.$$

4.3 Implementation and Run Times

The Implementation. We implemented the RDSA signature scheme over class groups of orders of algebraic number fields using C++ and LiDIA [23]. Thereby, we used the `SHA-1` algorithm for hashing. We precomputed all required 16-powers of the base element (ideal class) γ and stored them in the two-element representation of LLL-reduced ideals. So, we could use our new fast method for multiplication of ideals (see section 2). We combined our new multiplication method with the *fixed base windowing exponentiation method* where we used the base 16 (see [28]). As LLL-reduction for ideals we used the variant of Schnorr-Euchner [34].

In practice, the users of the RDSA signature scheme would obtain the public key of the sender of a signed message as well as these precomputed 16-powers of γ. Note that each user would use the same class group G, base element γ, thus the same 16-powers for its own signatures.

Run Times. The run times of our implementation (see Table 5) were measured on a Celeron processor at 433MHz.

We remark that our implementations were completely general in the sense that it did not include any optimization for number fields of degree 3.

Table 5. Run times for signature generation

number field (defining polynomial)	$\log_2(\Delta)$	period length	signature	ideal mult.
$x^3 - (10^{25} + 1)^3 - 1$	500	1	2.5 s	10.5%
$x^3 - (10^{27} + 1)^3 - 1$	540	1	2.6 s	10.4%
$x^3 - (10^{30} + 1)^3 - 1$	600	1	3.0 s	10.3%
$x^3 - (10^{35} + 1)^3 - 1$	699	1	3.9 s	8.8%
$x^3 - (10^{40} + 1)^3 - 1$	799	1	4.6 s	9.2%

Complexity. Computing a RDSA signature requires two exponentiations in the class group. The verification takes three exponentiations, two multiplications and a table look-up for an equality check. For all exponentiations 160 bit exponents are used.

We now focus on the signature step. The two exponentiations using the base ideal class γ take almost 100% of the time we have to spend to generate a signature. We just had precomputed all required 16-powers of γ: $\gamma, \gamma^{16^1}, \gamma^{16^2}, \ldots$. Using the fixed base windowing exponentiation method with the base 16 we have to perform on average 50.5 group operations for a 160 bit exponentiation, in the worst case 53 group operations ([28]). For the precomputation we require memory for 40 group elements.
Remember that one group operation consists in our context of

1. the multiplication of two ideals (one of them given in two-element representation, the other one given in LiDIA representation).
2. LLL-reducing the resulting ideal.

Note that we optimized the multiplication step (see section 2.3), whereas no optimization was done concerning the reduction step. In the example of the Stender field of degree 3 with the defining polynomial $f(x) = x^3 - (10^{25}+1)^3 - 1$ and 500 bit discriminant (see above) we need 1900 milliseconds (msec) for one 160 bit exponentiation. Thereby the ideal multiplication steps take 10.5% (199 msec), whereas the ideal reduction steps take 89.5% (1701 msec). Detailed timings for other Stender fields confirm these time percentages (see Table 5).

Conclusion: From now on any speed-up of more than 10.5 % for the RDSA signature scheme must be done by optimizing the reduction step or by reducing the number of group operations to be performed.

Security Level. The size of the mathematical objects (ideal classes) is determined by the degree n of the number field and the discriminant Δ. Therefore, we can compare the role of n, Δ with the modulus in the RSA scheme. Table 6 shows a very pessimistic comparing of a RSA modulus and a RDSA discriminant Δ for a degree of number field $n > 2$ in order to get the same security levels for the RSA and RDSA signature schemes. We followed the argumentation of [17] where this table was determined for imaginary quadratic number fields ($n = 2$). If one takes into account that the complexity of the root problem (the problem to break RDSA) grows with the degree of the number field one will get even shorter binary lengths for the discriminant. We shall work out a corresponding more realistic comparing of the sizes of a RSA modulus with a RDSA discriminant in a subsequent paper.

5 Conclusions and Open Questions

From the theoretical point of view, cryptosystems based on the discrete log (NFDL) or on the root problem in number fields are alternatives for the cryptosystems used today. As the complexity of NFDL and the root problem seem to be exponential in the degree of the number field, it is interesting to use number fields of degree > 2. In this paper we have shown for the first time how to implement a cryptographic protocol over such fields. The run times of our

Table 6. Corresponding key lengths (bits) for the same security levels

RSA modulus	RDSA discriminant
675	500
768	540
850	600
1024	687
1044	699
1230	799
1536	958
2048	1208
3072	1665
4096	2084

first implementations show that the new signature scheme RDSA is much less efficient than the popular cryptosystems of today. Note that the cryptosystems used today in practice (e.g. RSA and elliptic curve cryptosystems) originally were also inefficient. The common research of a lot of people made these systems step by step efficient.

The improvement of the arithmetic in number fields of degree > 2 towards a very efficient implementation of cryptosystems like RDSA is now subject of further research. We have shown in this article in which area any optimization must be done for a significant speed-up of the RDSA scheme. In the case of imaginary quadratic fields ($n=2$) recent speed-ups have led to a RDSA implementation of the same efficiency as the RSA signature scheme ([7]). There are a lot of possibilities for a speed-up of number field based cryptosystems. Therefore, we believe cryptography based on algebraic number fields shall be one day also in practice an alternative for the cryptosystems used today. Currently, we are working on the following strategies:

- Better cryptosystems. We try to design cryptosystems that involve less group operations.
- More efficient ideal reduction. The bottle-neck regarding one group operation in class groups now is the reduction of ideals. We search for a speed-up.

Besides, this paper raises the following interesting research problems:

- Are there more infinite families of orders of number fields of degree > 2 with short period length, i.e. with small number (ideally: 1) of reduced principal ideals?
- How can we implement the RDSA scheme in non-maximal orders of number fields of degree > 2?

– How large must the discriminant Δ (depending on the degree n of the number field) be in order to get to the knowledge of today the same security as RSA 512 bit, 768 bit, 1024 bit, ... ?

References

[1] E. Bach. Explicit bounds for primality testing and related problems. *Math. Comp.*, 55:355–380, 1990.
[2] Z.I. Borevic and I.R. Safarevic. *Number theory.* Academic Press, New York, 1966.
[3] J. Buchmann. A generalization of Voronoi's unit algorithm I and II. *Journal of Number Theory*, 20:177–209, 1985.
[4] J. Buchmann. The computation of the fundamental unit of totally complex quartic orders. *Mathematics of Computation*, 48(177):39–54, January 1987.
[5] J. Buchmann. On the period length of the generalized Lagrange algorithm. *Journal of Number Theory*, 26:31–37, 1987.
[6] J. Buchmann, I. Biehl, S. Hamdy, and A. Meyer. Cryptographic protocols based on the intractability of extracting roots and computing discrete logarithms. Technical report No. TI-16/99, Darmstadt University of Technology, 1999.
[7] J. Buchmann, I. Biehl, S. Hamdy, and A. Meyer. A signature scheme based on the intractability of computing roots. *Design, Codes and Cryptography, to appear*, 2001.
[8] J. Buchmann, M.J. Jacobson, Jr., and E. Teske. On some computational problems in finite abelian groups. *Mathematics of Computation*, 66:1663–1687, 1997.
[9] J. Buchmann and S. Paulus. A one way function based on module arithmetic in number fields. In *Advances in Cryptology — CRYPTO '97*, number 1294 in LNCS, pages 385–394. Springer-Verlag, 1997.
[10] H. Cohen. *A course in computational algebraic number theory.* Springer, Heidelberg, 1995.
[11] H. Cohen and H.W. Lenstra, Jr. Heuristics on class groups of number fields. In *Number Theory, Lecture notes in Math.*, volume 1068, pages 33–62. Springer-Verlag, New York, 1983.
[12] H. Cohen and J. Martinet. Class groups of number fields: numerical heuristics. *Math. Comp.*, (48):123–137, 1987.
[13] H. Cohen and J. Martinet. Etude heuristique des groupes de classes des corps de nombres. *J. Reine Angew. Math.*, (404):39–76, 1990.
[14] D. Coppersmith, A.M. Odlyzko, and R. Schroeppel. Discrete logarithms in GF(p). *Algorithmica*, 1:1–15, 1986.
[15] G. Frey and H.-G. Rück. A remark concerning m-divisibility and the discrete logarithm problem in the divisor class group of curves. *Mathematics of Computation*, 62:865–874, 1994.
[16] J. L. Hafner and K. S. McCurley. A rigorous subexponential algorithm for computation of class groups. *J. Amer. Math. Soc.*, 2:839–850, 1989.
[17] S. Hamdy and B. Möller. Security of cryptosystems based on class groups of imaginary quadratic orders. In *Proc. of AISACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 234–247. Springer, 2000.
[18] J.A. Howell. Spans in the module $(\mathbb{Z}_m)^s$. *Lin. Mult. Alg.*, 19:67–77, 1986.
[19] S. Lang. *Algebraic number theory.* Springer, New York, 1994.
[20] A.K. Lenstra and H.W. Lenstra Jr. Algorithms in number theory. In J. van Leeuwen, editor, *Handbook of theoretical computer science. Volume A. Algorithms and Complexity*, chapter 12, pages 673–715. Elsevier, 1990.

[21] A.K. Lenstra and E.R. Verheul. Selecting cryptographic key sizes in commercial applications. *CCE Quarterley Journal*, 3:3–10, 1999. http://www.cryptosavvy.com.

[22] H.W. Lenstra Jr. and C. Pomerance. A rigorous time bound for factoring integers. *J. Amer. Math. Soc.*, 5:483–516, 1992.

[23] LiDIA Group. *LiDIA - A library for computational number theory*, 1994-2001. http://www.informatik.tu-darmstadt.de/TI/LiDIA.

[24] S. Louboutin. The exponent 2-class-group problem for non-galois- over $\mathbb{Q}$ quartic fields that are quadratic extensions of imaginary quadratic fields. *J. Number Theory*, 49:133–141, 1994.

[25] S. Louboutin. Class-number problems for cubic number fields. *Nagoya Math. J.*, 138:199–208, 1995.

[26] K. McCurley. Cryptographic key distribution and computation in class groups. In R.A. Mollin, editor, *Number Theory and Applications*, pages 459–479. Kluwer Academic Publishers, 1989.

[27] A. Menezes, T. Okamoto, and S. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. In *Proceedings of the 23rd Annual ACM Symposium on the Theory of Computing*, pages 80–89, 1991.

[28] A.J. Menezes, P.C. van Oorschot, and S.A.Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1997.

[29] S. Neis. Reducing ideal arithmetic to linear algebra problems. In *Proc. of Algorithmic Number Theory Symposium III (ANTS III)*, volume 1423 of *Lecture Notes in Computer Science.* Springer, 1998.

[30] S. Neis. *Berechnung von Klassengruppen.* PhD thesis, Technische Universität Darmstadt, Darmstadt, Germany, 2001. In work.

[31] M. Pohst and H. Zassenhaus. *Algorithmic algebraic number theory.* Cambridge University Press, Cambridge, 1989.

[32] J.M. Pollard. A Monte Carlo method for factorization. *BIT*, 15:331–334, 1975.

[33] J.M. Pollard. Monte Carlo methods for index computation $(mod\, p)$. *Math. Comp.*, 32:918–924, 1978.

[34] C.P. Schnorr and M. Euchner. Lattice basis reduction: improved practical algorithms and solving subset sum problems. In *FCT '91*, volume 529 of *Lecture Notes in Computer Science*, pages 68–85. Springer, 1991.

[35] R.J. Schoof. Quadratic fields and factorization. In H.W. Lenstra Jr. and R. Tijdeman, editors, *Computational methods in number theory*, pages 235–286. Mathematisch Centrum, 1982.

[36] N.P. Smart. The discrete logarithm problem on elliptic curves of trace one. *Journal of Cryptology*, 12/3:193–196, 1999.

[37] N.P. Smart. How secure are elliptic curves over composite extension fields? Technical report CSTR-00-017, Dept. of Computer Science, University of Bristol, 2000.

[38] H.M. Stark. Some effective cases of the Brauer-Siegel Theorem. *Inventiones math.*, 23:135–152, 1974.

[39] H.J. Stender. Eine Formel für Grundeinheiten in reinen algebraischen Zahlkörpern dritten, vierten und sechsten Grades. *Journal of Number Theory*, 7:235–250, 1975.

[40] P. Theobald. *Berechnung von Hermite-Normalformen.* PhD thesis, Technische Universität Darmstadt, Darmstadt, Germany, 2000.

[41] D. Weber. Computing discrete logarithms with the general number field sieve. In *Proc. of Algorithmic Number Theory Symposium II (ANTS II)*, volume 1122 of *Lecture Notes in Computer Science.* Springer, 1996.

[42] H.C. Williams. The period length of Voronoi's algorithm for certain cubic orders. *Pub. Math. Debrecen*, 37:245–265, 1990.

A Requirements for Good Orders (Appendix)

Let $(G, \cdot)$ be a finite abelian group with group order $|G| = \prod_p p^{e(p)}$. Let S be a set of prime numbers which divide $|G|$.

Conjecture. Let $\gamma \in G$ be chosen at random with equidistribution. If $|G|$ is unknown and can not be determined in practice and $\prod_{p \in S} p^{e(p)} \geq 10^{55}$ and $\sum_{p \in S} \frac{1}{p^{e(p)}} \leq 10^{-8}$ then the discrete log problem $\gamma^x = \delta$ (where γ, δ are given and the integer x was chosed at random) can not be solved in practice using a generic algorithm, i.e. an algorithm that works in an arbitrary finite abelian group.

Our conjecture is based on

Theorem 3. *Let $\gamma \in G$ be chosen at random with equidistribution. Then*

$$\Pr\left[\prod_{p \in S} p \mid \mathrm{ord}_G\,\gamma\right] \geq 1 - \sum_{p \in S} \frac{1}{p^{e(p)}}$$

Proof. Let F be the set of prime divisors of $|G|$. As finite abelian group G is the inner direct product of his p-Sylow groups S_p:

$$G = \prod_{p \in F} S_p$$

where for each $p \in F$ the set S_p consists of all group elements with prime power p^i as element order for some $i \in \{0, \ldots, e(p)\}$ and each $\gamma \in G$ has a unique representation $\gamma = \prod_{p \in F} \gamma_p$ where $\gamma_p \in S_p$. Note that $ord\,\gamma = lcm\{ord\,\gamma_p : p \in F\}$. We can choose a group element $\gamma \in G$ at random with equidistribution by choosing elements $\gamma_p \in S_p$ at random with equidistribution for all $p \in F$ and by building the product $\gamma = \prod_{p \in F} \gamma_p$. Therefore we have

$$\begin{aligned}
\Pr\left[\prod_{p \in S} p \mid ord\,\gamma\right] &= \Pr[p \mid ord\,\gamma \text{ for all } p \in S] \\
&= 1 - \Pr[p \nmid ord\,\gamma \text{ for at least one } p \in S] \\
&= \begin{cases} 1 - \sum_{p \in S} \Pr[p \nmid ord\,\gamma] & \text{if } |S| = 1 \\ 1 - \left(\sum_{p \in S} \Pr[p \nmid ord\,\gamma] - \Pr[p \nmid ord\,\gamma \text{ for all } p \in S]\right) & \text{if } |S| > 1 \end{cases}
\end{aligned}$$

For each $p \in S$ there exists exactly one element in S_p whose order is not divisible by p (the neutral element of the group). Thus we have

$$\Pr\left[\prod_{p \in S} p \mid ord\,\gamma\right] = \begin{cases} 1 - \sum_{p \in S} \frac{1}{p^{e(p)}} & \text{if } |S| = 1 \\ 1 - \sum_{p \in S} \frac{1}{p^{e(p)}} + \prod_{p \in S} \frac{1}{p^{e(p)}} & \text{if } |S| > 1 \end{cases}$$

which shows the claim.

Example.

1. Suppose that $|G|$ has a prime divisor p with $p \geq 2^{160}$. Then the probability for the order of a randomly chosen group element to be divisible by that prime is at least $1 - 2^{-160}$ which is almost 1.
2. Suppose that $|G|$ has prime divisors $p_1, p_2 \geq 10^9$ where $p_1 \cdot p_2 \geq 10^{55}$. Then the probability for the order of a randomly chosen group element to be divisible by $p_1 \cdot p_2$ is at least $1 - (\frac{1}{p_1} + \frac{1}{p_2}) \geq 1 - 2 \cdot 10^{-9}$ which is almost 1.

Practical Key Recovery Schemes

Sung-Ming Yen

Laboratory of Cryptography and Information Security (LCIS)
Department of Computer Science and Information Engineering
National Central University
Chung-Li, Taiwan 320, R.O.C.
yensm@csie.ncu.edu.tw

Abstract. In this paper, the Bell Labs key recovery scheme is extensively modified to enable a user to request on-line key recovery service when the file decryption key is forgotten or lost. New practical and important requirements of key recovery are also considered in the proposed schemes, for example, the key recovery server and any intruder over the communication channel should not learn the key to be reconstructed. Furthermore, the necessary authenticity and secrecy between a user and the key recovery server should be provided.

Keywords: Active attack detectability, Cryptography, Dictionary attack, Key escrow, Key recovery, Off-line attack, On-line attack.

1 Introduction

Since 1994, the topic of key escrowed encryption and communication has been widely noticed and studied [1,2,3,4,5]. However, this technique has never been widely employed because of the privacy issue. Until 1996, some researchers changed their attention on escrowed encryption into the commercial applications in an alternative scenario, say the commercial *key recovery* [6,7,8]. The topic of commercial key recovery is not only nonconflicting but also can be identified as a necessary component for the applications of data security service. Evidently, it is a very important and practical issue of how to get survive when using a hard to remember (and to guess) password in a real security system.

1.1 The Classification of Keys

Here we classify passwords (or keys) to be remembered by a person into three different types depending on the complexity required to recover them by the attacker and the difficulty required to remember them by the owner.

simple password : This category of passwords are more easy to remember by the owner, but they are still difficult to be guessed by nonprofessional attacker. However, a professional hacker may sometimes figure out the weak password P_a (if it is poorly chosen) with some probability via the *off-line dictionary attack* if a copy of $f(P_a)$ is accessible where $f()$ is any one-way function. In this paper, a simple password means a password that should at least be resistant with large probability against the guessing and matching attack from some collected dictionaries. Tools for preventing poorly chosen weak password from guessing attacks, i.e., to filter out fatal or inappropriate weak passwords, are available in [9,10,11]. In [9], it was reported that prior to the experiment, it is believed to find a large percentage of the collected passwords (selected by a novice) in the dictionaries and common word lists. However, surprisingly, only 1 out of 5 passwords were found in the dictionaries. The standard dictionary used in [9] has about or more than 25000 words. So, it is expected that each of the matched passwords needs $\frac{25000}{2}$ comparisons with words in the dictionary before been identified. Notice that while an off-line guessing approach may be feasible, an on-line approach however will cause an unreasonable delay and most of the time this attack will be easily detected. Evidently, the cost of on-line (and off-line) guessing attack on an already sieved (using the above mentioned tools) simple password will be extremely high since a much larger dictionary and more sophisticated manipulation will be necessary.

strong password : Theoretically, passwords of this category are selected to be random numbers. So, they are basically be quite difficult to be figure out by the attacker but they are also quite hard to remember by the owner.

pseudo strong password : A password "*pass*" for practical and high security requirement usage often falls into this category which can be considered as a mixture of the above two categories of passwords. It avoids or at least complexes greatly the off-line dictionary attack against professional hackers but it also brings the risk of forgetting the passwords to the owner. Another typical approach of selecting a pseudo strong password is by concatenating two or more simple passwords and it is sometimes called a passphrase.

1.2 Review of the Bell Labs Key Recovery Scheme

To compromise with the requirement of using more secure pseudo strong passwords or keys and the requirement of recovering any forgotten passwords or keys, Maher at Bell Labs developed a crypto key recovery scheme [6].

• The protocol

In the Bell Labs key recovery scheme [6], the key recovery server has its secret key x_s and the related public key $y_s = \alpha^{x_s} \bmod P$, where P is a large prime. Each user, say A, registers to this server through a physical manner and will be given the server's public key.

The working key generation and working key recovery protocol can be briefly reviewed as follows. Here the working key refers to the file encryption key or any login password used in a remote login procedure.

(1) A**:** Each time user A wishes to encrypt his important file, he computes the file encryption key K (it is assumed to be a *strong password*) as

$$K = h(\alpha^{pass} \bmod P || y_s^{pass} \bmod P)$$

where $h()$ is any secure one-way hash function, e.g., [12,13,14], and "*pass*" is the password (it is assumed to be a *pseudo strong password*) selected by the user. Besides the ciphertext $E_K(M)$, $\alpha^{pass} \bmod P$ is also stored. The encryption function $E()$ is assumed to be performed by using any symmetric-key cipher.

(2) $A \rightarrow S$**:** When the user A forgets his password, he tries to recover the file encryption key K by delivering $\alpha^{pass} \bmod P$ to the key recovery server.

(3) $A \leftarrow S$**:** The recovery server computes $T = (\alpha^{pass})^{x_s} \bmod P$ and sends the result T to the user.

(4) A**:** The file encryption key K can be recovered by the user as $h(\alpha^{pass} \bmod P||T)$.

• Some remarks on the Bell Labs protocol
Two important but overlooked issues of the above Bell Labs protocol are given below.

(a) In the step (2), the value $\alpha^{pass} \bmod P$ should be delivered to the recovery server by A through an *authenticated* channel.

(b) In the step (3), the value T should be sent back to user A via a *secure* channel.

However, no solution has been provided in the original protocol to meet the above two important and necessary requirements. Otherwise, a physical face-to-face approach will be necessary for each query and this reduces the practicality of commercial key recovery.

In the Bell Labs protocol, the recovery server will learn the password or key K to be reconstructed. Therefore, multiple recovery servers will be necessary and the file encryption key K can be computed as a combination of many subkeys K_i (e.g., $K = K_1 \oplus K_2 \oplus K_3$ for $i = 1, 2, 3$) and each of which should be recovered when the user forgets his password. The above development aims to prevent any single server or a subset of servers in collusion to obtain the key K. To have a satisfying security level, the number of servers should at least be three. There are two drawbacks for this arrangement. First, system performance may be worse than its original scheme. The second drawback, the more critical issue for the case of using multiple key recovery servers, is that the applicability of the key recovery will be reduced. Based on the above construction, when one of the servers is not accessible or becomes faulty, then the forgotten key K can theoretically never be reconstructed.

Another issue to be pointed out is that in the step (1) of file encryption key generation procedure, K can be generated as either $h(\alpha^{pass} \bmod P)$ or $h(y_s^{pass} \bmod P)$ when $y_s^{pass} \bmod P$ or $\alpha^{pass} \bmod P$, respectively will be stored with the ciphertext $E_K(M)$.

2 The Model of a Practical Key Recovery

Since the problem of how to recover a forgotten password or key is basically a practical issue. Therefore, a practical consideration of what a key recovery scheme should provide is necessary. From the view point of practicality, the requirements of a good key recovery scheme are identified and listed in the following.

(1) The key recovery protocol should be performed at the user's location through an on-line process interacted with the recovery server. For this purpose, the on-line process must also provide both secrecy and authenticity.
(2) The key recovery server should not know the exact passwords or keys to be recovered by the user. This implies that a simple key backup approach does not match the requirement of a good key recovery.
(3) An attacker cannot try to impersonate to be a specific valid user without being detected and recovers that user's passwords or keys via the assistance from the key recovery server which acting as an oracle. In another word, any on-line impersonating and guessing attack should be detectable by the recovery server.
(4) In the real world of using digital systems, any user may have many passwords or keys to remember, however the user does not have to keep a cleartext backup of them in order to prevent forgetfulness.
(5) Even if the user loses his local copy of the most important personal secret information, the key recovery scheme should also enable the server to assist the user to recover his password or key. Although, in this situation, it maybe requires the user to return back to the recovery server physically and performs the recovery process.

The requirement (1) implies that the user can request a key recovery service through an on-line process performed from a remote location instead of returning back to the recovery server physically. To realize this useful functionality, a key recovery scheme itself should also provide both authenticity and secrecy in order to avoid impersonation (to be discussed in the requirement (3)) and to protect the recovered passwords or keys.

As to the requirement (2), since the secret information to be recovered may be a password to log into a computer system connected to the Internet. It is not reasonable to enable the server to know the exact secret information.

The requirement (3) is crucial because that the stored information (e.g., the value of $\alpha^{pass} \bmod P$ in the Bell Labs scheme) in the user's machine used to recover the forgotten password or keys may be accessible to an internal attacker in some situations. However, it is not reasonable for an attacker with access to the above stored information to recover the passwords or keys via the assistance from the trusted recovery server. This will open a backdoor of a key recovery system. Therefore, this suggests that the stored information for key recovery to be user specific or under the protection by using an important password or key of the specific user in order to avoid impersonation. Furthermore, since a remote key recovery process will be performed, any on-line guessing attack if happen should be detectable by the recovery server. The key recovery scheme should guarantee that the recovery server will not be used as an oracle by an attacker impersonating to be a specific valid user without being detected and trying to recover that user's passwords or keys.

The requirement (4) says that key recovery server should provide promising service to its users to help them to reconstruct important information. Otherwise, a trivial solution that each user keeps a backup of all his passwords or keys in a *secret* (although it is difficult to define precisly) place is enough.

The requirement (5) is to make the scheme be robust enough. After registering to the recovery server, each user may select or receive some long term personal secret information which will be used to help recovering the working passwords or keys. However, it is not reasonable that the passwords or keys cannot be recovered forever if that long term personal information is lost.

Some issues of chosing and using passwords and keys are described in the following. Usually it is strongly suggested that a person should not keep only a single password or key and uses this same password for every system that he has access. Often, it is suggested that a lift cycle of a password should not exceed three or four months, especially for remote login passwords used to enter a system that does not have strong intrusion detection measures. At the same time, people are educated to avoid of using poorly chosen weak passwords. For the problem of file protection, people sometimes use different passwords or keys for files of different security classifications. Therefore, each person will have a moderately amount of *pseudo strong* passwords or keys to remember. In this paper, we focus our attention on considering the problem of how to recover any of such pseudo strong passwords or keys if they will be forgotten under the above mentioned practical environment.

3 The Proposed Key Recovery Scheme – KRS-1

In this section, the first proposed key recovery scheme, named the KRS-1, is given to modify the Bell Labs scheme in order to enhance both security and functionary.

3.1 The Protocol of KRS-1

The key recovery server S has its secret key x_s and the related public key $y_s = \alpha^{x_s} \bmod P$, where P is a large prime and α is a primitive root of P. For security reasons, P is often selected to be a *safe prime* of the form $P = 2q + 1$ where q is also a large prime. A prime where $P - 1$ has only small factors is called smooth. Smooth primes should be avoided because they allow a much faster discrete logarithm computation [15].

When each user, say A, registers to the server, he selects a personal long term password P_a and gives it to the server S. Noticeably, the password P_a is assumed to be a *simple password*. This will enable the user to remember his long term password in a more easy way and makes the key recovery scheme be robust enough. The server stores each user's identity and the related personal long term password in a secure table and gives the above server's public key to the user. The server's public key and its certificate can also be retrieved through the network when required.

The working key (it can be a file encryption key or a login secret to access a remote machine) generation and recovery protocol goes as follows.

(1) A**:** Each time user A wishes to encrypt his important file, he computes the file encryption key K (it is assumed to be a *strong password*) as

$$K = h(y_s^{pass} \bmod P)$$

where $h()$ is any secure one-way hash function, e.g., [12,13,14], and "*pass*" is a file accessing password selected by the user for a specific classification of files.
As previously described, *pass* should be a *pseudo strong password* so that an off-line dictionary attack is infeasible or at least with excessive cost that will make the attack meaningless. Besides the ciphertext $E_K(M)$,

$$R = E_{P_a}(\alpha^{pass} \bmod P)$$

(or just as $R = (\alpha^{pass} \bmod P) \oplus P_a$) is also stored. Of course, if two or more files share a common encryption key K (or file accessing password *pass*), then the value of R can also be shared.

(2) $A \rightarrow S$**:** When the user A forgets his file accessing password *pass*, he tries to recover the file encryption key K by retrieving $\alpha^{pass} \bmod P$ from R using his long term personal password P_a and computes

$$V = \alpha^{pass} \cdot \alpha^{r_1} \bmod P$$

where r_1 is a random integer selected by user A. User A then computes

$$\begin{cases} c_1 = \alpha^{r_2} \bmod P \\ c_2 = y_s^{r_2} \cdot V \bmod P \\ h(V, P_a) \end{cases}$$

where r_2 is a random integer selected by A and $\{c_1, c_2\}$ are the ciphertext of V produced by using the ElGamal encryption scheme [16]. Finally, the user sends c_1, c_2, and $h(V, P_a)$ along with his identity ID_a to the recovery server.

(3) $A \leftarrow S$**:** The key recovery server first decrypts V from $\{c_1, c_2\}$ by using its secret key x_s and checks the data integrity and originality via the assistance of $h(V, P_a)$. If the above verification is correct, then the key recovery server computes

$$\begin{aligned} T &= V^{x_s} \bmod P \\ &= (\alpha^{pass} \cdot \alpha^{r_1})^{x_s} \bmod P \\ &= y_s^{pass} \cdot y_s^{r_1} \bmod P. \end{aligned}$$

The server then returns T to the user.

(4) A**:** Since A knows the random integer r_1, he can recover

$$y_s^{pass} \equiv T \cdot y_s^{-r_1} \pmod{P},$$

then the file encryption key K can be recovered as $h(T \cdot y_s^{-r_1} \bmod P)$.

3.2 Security Analysis of the KRS-1 Protocol

The reason of sending V in the ciphertext version instead of the cleartext version is to counteract the following off-line *verifiable text attack* on the long term password P_a (it is a simple password). If the passive attacker can intercept both V and $h(V, P_a)$, then he can perform an *off-line* dictionary attack trying to recover P_a if P_a is poorly chosen. In the above protocol, the attacker can intercept $T = V^{x_s} \bmod P$ from the step (3), however to derive V from the intercepted value T requires the server's secret key x_s.

An interesting problem for the above protocol is that if the user A still remembers his long term password P_a, then why not just storing the encrypted version of a working key as $E_{P_a}(K)$ or $E_{P_a}(pass)$ using P_a as the protection key? Therefore, the file encryption key K or file accessing password $pass$ can be recovered when required without the assistance from the recovery server.

Recall that P_a is a simple password. If the above approach is employed, then the attacker who has access to both the ciphertext $E_K(M)$ and the encrypted version of key $E_{P_a}(K)$ can conduct an *off-line* exhaustive search and test on the possible long term password P_a.

On the other hand, in the proposed key recovery protocol, when an active attacker has both the knowledge of $E_K(M)$ and $R = (\alpha^{pass} \bmod P) \oplus P_a$ he has to conduct the following *on-line* password guessing attack. However, the attack can be detected and can be prevented by some precautions. The attacker should try a guessed long term password P'_a and computes

$$G = R \oplus P'_a = (\alpha^{pass} \bmod P) \oplus P_a \oplus P'_a.$$

The attacker then computes $V = G \cdot \alpha^r \bmod P$ and its ciphertext $\{c_1, c_2\}$ and the keyed hash $h(V, P'_a)$. Finally, the attacker sends ID_a, $\{c_1, c_2\}$, and $h(V, P'_a)$ to the recovery server. If the guessed P'_a is identical to P_a, then the integrity check will be correct and the server will return T, otherwise the active attack will be detected and the recovery server will take some suitable countermeasures. Possible countermeasures include: (1) to keep a log file and to delay the following attempts; (2) to advise the legal user to change his P_a. It should be emphasized that for a simple password based protocol, the on-line password guessing attacks are always possible. The main concern is that the protocol should be active attack detectable.

In fact, the above on-line guessing attack can be modified to send ID_t, $\{c_1, c_2\}$, and $h(V, P_t)$ to the recovery server where ID_t and P_t are the identity and the personal password, respectively of the active attacker. The server will not detect the existence of an attack. If the recovered $h(T \cdot y_s^{-r} \bmod P)$ is equivalent to $h(y_s^{pass})$ and can be used to decrypt $E_K(M)$ correctly, then the guessed P_a' is correct and now the recovery server acts like an oracle.

However, the above scenario is not exactly the case of a usual on-line guessing attack. First, the attacker now conducting an unusual (in terms of its high frequency) service request will reveal its identity. Different application environments may take their own appropriate countermeasures, e.g., to reveal the attacker's identity or to restrict the number of service provided within a predetermined period of time. Second, a large amount of service request (each itself will cost the attacker some service charge) will be necessary in order to figure out the possible P_a. Recall that even a password chosen by a novice it can only be identified with a probability of about 0.2 [9] and it will take a huge amount of on-line test, say on average $\frac{25000}{2}$. As suggested previously, P_a should be sieved by some tools to rule out poorly chosen passwords. This precaution may complex the on-line guessing attack extensively. Therefore, such kind of on-line attack will be extremely unreasonable for commercial key recovery since the total amount of cost (paid to the recovery server) may exceed the profit of the attack or the cost to find *pass* via an exhaustive search.

4 The Key Recovery Scheme Based on RSA – KRS-2

In this section, the second proposed key recovery scheme, named the KRS-2, is given which is based on the blinding of the RSA system.

4.1 The Protocol of KRS-2

The key recovery server S selects two large secret primes p and q, and publishes $n = p \cdot q$. Also, the server chooses a base number α with order $\phi(n)$. The server publishes its RSA [17] public encryption exponent e and keeps privately the decryption exponent d such that $e \cdot d \equiv 1 \pmod{\phi(n)}$ where $\phi(n) = (p-1) \cdot (q-1)$. When each user, say A, registers to the server, he selects a personal long term password P_a and his identity ID_a such that $\gcd(ID_a, \phi(n)) = 1$. When encoded into an integer, the identity should be unique. Alternatively, a random number (often called a salt) is chosen for each identity and an extended identity is computed by using a cryptographic hash function on the identity and the salt to obtain a unique ID_a. The details of generating unique identity numbers are out of the scope of this paper. It is the same as in the KRS-1 that the password P_a is assumed to be a simple password. The server stores each user's identity and the personal long term password in a secure table.

The working key generation and recovery protocol goes as follows.

(1) ***A*:** Each time user A wishes to encrypt his important files, he computes the file encryption key K as

$$K = h(\alpha^{pass} \bmod n)$$

where *pass* is a pseudo strong file accessing password selected by the user. Besides the ciphertext $E_K(M)$,

$$R = \alpha^{pass \cdot ID_a} \bmod n$$

is also stored. Of course, if two or more files share a common encryption key K (or file accessing password *pass*), then the value of R can also be shared.

(2) $A \rightarrow S$**:** When the user A forgets his file accessing password *pass*, he computes

$$V = \alpha^{pass \cdot ID_a} \cdot r_1^{ID_a} \bmod n$$

where r_1 is a random integer in $[1, n-1]$ selected by user A so that $\gcd(r_1, n) = 1$. The user then computes

$$\begin{cases} r_2^e \bmod n \\ h(V, r_2, P_a) \end{cases}$$

where r_2 is a random integer in $[1, n-1]$ selected by the user A. User A then sends V, $r_2^e \bmod n$, and $h(V, r_2, P_a)$ along with his identity ID_a to the recovery server.

(3) $A \leftarrow S$**:** The recovery server first decrypts r_2 using the decryption key d and checks the integrity and data originality via $h(V, r_2, P_a)$. If the above checking is correct, then the key recovery server computes

$$\begin{aligned} T &= V^{ID_a^{-1}} \bmod n \\ &= (\alpha^{pass \cdot ID_a} \cdot r_1^{ID_a})^{ID_a^{-1}} \bmod n \\ &= (\alpha^{pass} \cdot r_1) \bmod n \end{aligned}$$

where ID_a^{-1} is the multiplicative inverse of ID_a modulo $\phi(n)$. The server then returns T to the user.

(4) A**:** Since A knows the random integer r_1, he can recover

$$\alpha^{pass} \equiv T \cdot r_1^{-1} \pmod{n},$$

where r_1^{-1} is the multiplicative inverse of r_1 modulo n. Then, the file encryption key K can be recovered as $h(T \cdot r_1^{-1} \bmod n)$.

4.2 Security Analysis of the KRS-2 Protocol

One major difference to the previous KRS-1 protocol is that in the KRS-2 protocol the value of V is delivered to the recovery server in the cleartext version. The reason of not sending $V^e \bmod n$ is that V can be easily obtained via $V = T^{ID_a} \bmod n$, where T can be intercepted from the step (3).

If the protocol is modified to remove the inclusion of r_2 such that V is sent in the ciphertext version as $V^e \bmod n$, the integrity and originality check value $h(V, P_a)$ is sent in the step (2), and T is protected by $t = (T \cdot P_a) \bmod n$ when received from the recovery server in the step (3). The protection of T is to avoid the direct derivation of V by computing $V = T^{ID_a} \bmod n$. However, it can be easily verified that an off-line *verifiable text attack* still applies to the above modified protocol. The attacker tries a guessed long term password P'_a and computes $T = (t \cdot P'^{-1}_a) \bmod n$. Based on this T, the attacker obtains $V = T^{ID_a} \bmod n$, then the intercepted integrity and originality check value $h(V, P_a)$ can be employed to verify the correctness of P'_a.

Alternatively, in the KRS-2 protocol, the inclusion of random integer r_2 is to prevent the possible verifiable text attack. Because the random integer r_2 selected by user A is sent in the ciphertext version, therefore it disables the attacker to guess and verify a possible P_a via the assistance of intercepted $h(V, r_2, P_a)$. To have a successful guess, the attacker should try both r_2 and P_a at the same time. However, since r_2 is a random integer in $[1, n-1]$ and this makes the off-line guessing attack infeasible.

The situation of having the recovery server acting as an oracle in order to derive other person's long term password P_a (in fact, it has already been described on how to prevent this drawback) can be avoided in the KRS-2 protocol. In the KRS-2 protocol, in order to recover the working key $K = h(\alpha^{pass} \bmod n)$ from the value $R = \alpha^{pass \cdot ID_a} \bmod n$, the active attacker can only pretend to be user A by sending ID_a, V, $r_2^e \bmod n$, and $h(V, r_2, P'_a)$ to the recovery server. This will however enable the server to identify a possible attack if $h(V, r_2, P'_a)$ is not correctly computed because of an incorrect P'_a.

5 Conclusions

From the view point of security engineering, commercial and even non-commercial key recovery is a necessary key management function in order to provide a complete and sound security application environment. Without a satisfying and practical key recovery solution, any research of strong cryptography can be in vain for many situations. This is especially true for the cases where tamper proof hardware for storing passwords or keys are not accessible. Furthermore, access of the tamper proof hardware also needs passwords. Because strong and secure cryptography always needs strong passwords or keys to apply and this implies a high risk of losing anything valuable if the keys are lost or forgotten. This situation applies to both individual requirement and organization requirement.

Since key recovery is considered to be a practical issue. Practical and important requirements of key recovery are first pointed out in this paper. Then, two commercial key recovery schemes based on these identified requirements are proposed. In the future research, other key recovery requirements for more complex environments or largely different applications will be considered. Key recovery schemes based on these different models will also be developed.

6 Acknowledgments

This research was supported in part by the National Science Council of the Republic of China under contract NSC 89-2213-E-008-055. The author wish to thank the anonymous referees for their many valuable and constructive suggestions to improve both the technical contents and presentation skills. Some suggestions will be considered in the full version of this article.

References

1. NIST, "Escrowed Encryption Standard," *Federal Information Processing Standards Publication (FIPS PUB) 185*, 1994.
2. D.E. Denning, "The US key escrow encryption technology," *Computer Communication Magazine*, vol. 17, no. 7, pp. 453–457, July 1994.
3. D.E. Denning and M. Smid, "Key escrowing today," *IEEE Communication Magazine*, pp. 58–68, Sept. 1994.
4. D.E. Denning, "Key Escrow Encryption – the third paradigm," *Computer Security Journal*, vol. 11, no. 1, pp. 43–52, 1995.
5. D.E. Denning and D.K. Branstad, "A taxonomy for key escrow encryption systems," *Commun. ACM*, vol. 39, no. 3, pp. 34–40, 1996.
6. D.P. Maher, "Crypto backup and key escrow," *Commun. ACM*, vol. 39, no. 3, pp. 48–53, 1996.
7. S.T. Walker, S.B. Lipner, C.M. Ellison, and D.M. Balenson, "Commercial key recovery," *Commun. ACM*, vol. 39, no. 3, pp. 41–47, 1996.
8. S.T. Walker, S.B. Lipner, C.M. Ellison, D.K. Branstad, and D.M. Balenson, "Commercial key escrow: Something for everyone, now and for the future," *TISR #541*, Trusted Information Systems, Apr. 16, 1995.
9. E.H. Spafford, "Observing reusable password choices," *Purdue University Technical Report CSD–TR 92–049*, 31 July 1992.
10. E.H. Spafford, "Observations on reusable password choices," In *Proc. of the 3rd Security Symposium*, Usenix, September 1992.
11. E.H. Spafford, "OPUS: Preventing weak password choices," *Computers & Security*, vol. 11, no. 3, pp. 273–278, 1992.
12. R. Rivest, "The MD5 message digest algorithm," *RFC 1321*, Apr. 1992.
13. FIPS 180-1, "Secure Hash Standard," NIST, US Department of Commerce, Washington D.C., April 1995.
14. Y. Zheng and J. Pieprzyk and J. Seberry, "HAVAL - a one-way hashing algorithm with variable length of output," *Advances in Cryptology – AUSCRYPT'92*, Lecture Notes in Computer Science, Vol.718, Springer-Verlag, pp.83–104, 1993.
15. S.C. Pohlig and M.E. Hellman, "An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance," *IEEE Trans. on Inform. Theory*, vol. 24, no. 1, pp. 106–110, January 1978.
16. T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. on Inform. Theory*, vol. 31, no. 4, pp. 469–472, July 1985.
17. R.L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystem," *Commun. of ACM*, vol. 21, no. 2, pp. 120–126, 1978.

Non-deterministic Processors

David May, Henk L. Muller, and Nigel P. Smart

Department of Computer Science,
Woodland Road,
University of Bristol,
BS8 1UB,
UK
{dave,henkm,nigel}@cs.bris.ac.uk

Abstract. New techniques have been discovered to find the secret keys stored in smart-cards. These techniques have caused concern for they can allow people to recharge their smartcards (in effect printing money), or illegally use phone or digital TV services. We propose a new processor design which will counteract these techniques. By randomising the instruction stream being executed by the processor we can hide the secret key stored in a smartcard. The extension we propose can be added to existing processors, and is transparent to the algorithm.

1 Background

Modern cryptography is about ensuring the integrity, confidentiality and authenticity of digital communications. As such it has a large number of applications from e-commerce on the Internet through to charging mechanisms for pay–per-view-TV. As more and more devices become network aware they also become potential weak links in the chain. Hence cryptographic techniques are now being embedded into devices such as smart cards, mobile phones and PDA's. This poses a number of problems since the cryptographic modules are no longer maintained in secure vaults inside large corporations. For a cryptographic system to remain secure it is imperative that the secret keys used to perform the required security services are not revealed in any way.

The fact that secret keys are now embedded into a number of devices means that the hardware becomes an attractive target for hackers. For example if one could determine the keys which encrypt the digital television transmissions, then one could create decoders and sell them on the black market. On a more serious front if one could determine the keys which protect a number of store valued smart cards, which hold an electronic representation of cash, then one could essentially print money.

Since cryptographic algorithms themselves have been studied for a long time by a large number of experts, hackers are more likely to try to attack the hardware and system within which the cryptographic unit is housed. A particularly worrying attack has been developed in the last few years by P. Kocher and colleagues at *Cryptography Research Inc.*, [12,13]. In these attacks a number of physical measurements of the cryptographic unit are made which include power consumption, computing time or EMF radiations. These measurements are made over a large number of encryption or signature operations and then, using statistical techniques, the secret key embedded inside the cryptographic unit is uncovered.

These attacks work because there is a correlation between the physical measurements taken at different points during the computation and the internal state of the processing device, which is itself related to the secret key. For example, when data is loaded from memory, the memory bus will have to carry the value of the data, which will take a certain amount of power depending on the data value. Since the load instruction always happens at the same time one can produce correlations between various runs of the application, eventually giving away the secret of the smart card. The three main techniques developed by Kocher et. al. are timing attacks, simple power analysis (SPA) and differential power analysis (DPA). It is DPA which provides the most powerful method of attack which can be mounted using very cheap resources.

Following Kocher's papers a number of people have started to examine this problem and propose solutions, see [3], [4], [8,14]. Goubin and Patarin [8] give three possible general strategies to combat DPA type attacks:

1. Introduce random timing shifts so as to decorrelate the output traces on individual runs.
2. Replace critical assembler instructions with ones whose signature is hard to analyse, or reengineer the crucial circuitry which performs the arithmetic operations or memory transfers.
3. Make algorithmic changes to the cryptographic primitives under consideration.

The last of these approaches has been proposed in a number of papers, and various examples have been given. For example [8] suggests essentially splitting the operands into two and duplicating the work load. However, this leads to at least a doubling in the computing resources needed to perform the cryptographic operation. This is similar to the defence proposed by Chari et.al [3], who propose to mask the internal bits by splitting them up and processing the bit shares in such a way that on recombination we obtain the correct result. In this way the target bits for the DPA selector function are not exposed internally to the processor and so will hopefully have no effect on the power trace.

Kocher et.al [13] recommends using a level of blinding, especially when applied to algorithms such as RSA. This again increases the computing time needed to implement the operation and also modifies the original cryptographic primitive in ways which could lead to other weaknesses being uncovered. This is a popular approach which is mentioned by a number of authors and in private communications.

The second approach has been studied, for example in [12], where the application of balanced architectures is described. They balance the Hamming weights of the operands, and propose physical shielding or adding noise circuitry.

It is the first approach which we consider to be the most promising, but one which the current literature has only considered in a passing way. We end this section by noting that we are not addressing the problem of making the hardware or software tamper resistant. This is an important area which also needs to be taken into consideration to produce secure microprocessors or micro-controllers, see [1] and [2].

2 Prior Work

Essentially the defence to DPA we have in mind is an instance of what Kocher et.al call "temporal misalignment of traces". This is a way to introduce noise which can prevent the use of DPA type techniques.

Kömmerling and Kuhn [14] mention various techniques that introduce a certain amount of non-determinism into the processor. An example of this is randomised clocking which puts an element of non-determinism into the instruction cycle. However, they state that this does not provide enough of a defence, since attacks can use cross correlation techniques to remove the effect of the randomised clock.

Kocher et.al [13] mention that randomising execution order can help defeat DPA, but can also lead to other problems if not done carefully. Kömmerling and Kuhn mention the idea of randomised multi-threading at an instruction level. They describe this with a set of essentially *shadow* registers. The auxiliary threads could then execute random encryptions, hence hoping to mask the correct encryption operation. This has its draw back as the processor is required to perform tasks which are in addition to the desired computation, hence increasing computational costs considerably.

Chari et.al [3] mention a number of counter measures to DPA type attacks. Including the creation of balanced hardware, clock cycles of varying lengths and a randomised execution sequence. They mention that for randomised execution sequence to be effective then the randomisation needs to be done extensively. For example they mention that if only the XORs in each DES [6] round are randomised then one can still perform DPA by taking around eight times as much data. In addition no mechanism is provided which would enable aggressive randomised execution.

Hence for randomised execution order to work it needs to be done in a highly aggressive manner which would preclude the type of local randomisation implied by the descriptions above. In addition this cannot be achieved in software since a software randomiser would work at too high a level of abstraction. The randomised multi-threading idea is close to a solution but suffers from increased instruction count and requires a more complex processor with separate banks of registers, one for each thread.

We have designed simple additions to a processor with either single or multiple execution units which allow for aggressive randomised execution of instructions on an instruction by instruction basis, with the added bonus that every instruction executed is required by the algorithm. No extra execution time or power are required. In addition our randomisation process requires no alteration to the source code, since the randomisation is done by the processor itself. Hence we require no modifications to the basic cryptographic primitives.

In addition we have introduced two assembly instructions which allow for even greater levels of randomised execution, especially when combined with our basic hardware modifications. We have called this processor architecture NDISC for *Non-Deterministic Instruction Stream Computer*.

Our new architecture will make optimal use of algorithms that have high instruction-level parallelism. Clapp [5] gives an analysis of some cryptographic algorithms from this perspective. Hence, algorithms that are optimised for use on current processors with multiple concurrent execution paths, as found in superscalar [9,10,19], parallel and VLIW architectures [7], will be particularly suitable for our new processor.

In the following discussions we shall focus on examples such as DES [6] and integer multiplication, which is used in RSA [17] and EC-DSA [11] [16]. However, it is clear that these techniques can be applied to any cryptographic algorithm. Or indeed to any algorithm where it is desirable to limit the environmental impact of the processor, for example in reducing resonances in small computing devices.

3 Non Deterministic Processors

In order to prevent attacks based on correlating data, we have designed a simple addition to standard processors that randomises instruction issuing.

Crucially, an attack works because two runs of the same program give comparable results; everything compares bar the data that is the part where the attack comes in. By changing the data even slightly the attacker will get a knowingly different trace, and by correlating the traces, one builds a picture of what is happening inside.

Our protection scheme removes correlation between runs, thereby making the attack much harder. Our observation is that a conventional processor executes a sequence of instructions deterministically; it may execute instructions *out-of-order*, but it will always execute instructions out-of-order in the same way. If the same program is run twice in a smart card, then the same instruction trace will be executed. By allowing the processor at run time choose a random instruction ordering, we get multiple possible traces that are executed.

Naively viewed, if there are 10 instructions without dependencies, then there are $10! = 3628800$ different ways of executing those instructions. Of course not all instructions are independent, however our experiments indicate that there are sufficient execution traces to efficiently hide the data trace. In addition, we can decrease the number of dependencies using the techniques described below in Section 3.2.

Relating this to existing processor design: like superscalar processors, we select a number of independent instructions that can be executed in any order,

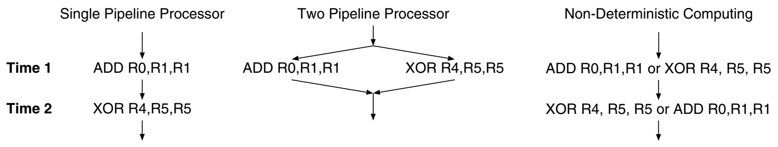

Single Pipeline Processor
Two Pipeline Processor
Non-Deterministic Computing
Time 1
Time 2
ADD R0,R1,R1
XOR R4,R5,R5
ADD R0,R1,R1
XOR R4,R5,R5
ADD R0,R1,R1 or XOR R4, R5, R5
XOR R4, R5, R5 or ADD R0,R1,R1

Fig. 1. Simple comparison of how a Non-deterministic processor executes two instructions as opposed to other processors

and then randomly select instructions to be executed. Unlike superscalar processors, we do *not* execute the instructions in parallel. Instead we use available parallelism to increase non determinism.

In addition, there are instruction sequences that cannot be executed in parallel on a superscalar, but that can be executed in a non-deterministic manner. For example, the instruction sequence `ADD R0, R7, R7 ; ADD R1, R7, R7` (which first adds R0 to R7 and then adds R1 to R7) can be executed either order; even though the instructions cannot be executed in parallel on a superscalar.

3.1 Random Issuing

In single pipeline processors a sequence of instructions is executed in the order in which they are fetched by the processor. There is a little out-of-order execution to help with branch prediction but this all occurs on a very small scale. On multiple pipeline processors there are a number of *execution units* through which independent instructions can be passed in parallel. For example, if a processor has a logic pipeline and an integer-arithmetic pipeline, then the following two instructions

```
ADD R0, R1, R1
XOR R4, R5, R5
```

may be executed in parallel in the two pipelines. One pipeline will execute the ADD, the other will execute the XOR.

Our idea is the following: like a superscalar we identify instructions that can be issued independently, but instead of using this information to issue instructions in parallel, we use this information to execute instructions out-of-order, where the processor makes a random choice as to issue order. We call this process *Instruction Descheduling.* This creates a level of non-determinism in the internal workings of the processor. This is illustrated in Figure 1.

If such a system introduces large amounts of non-determinism then this could produce a significant breakthrough in reducing the effectiveness of DPA. The reduction in the effectiveness of DPA results from the fact that the power trace from one run will be almost completely uncorrelated with the power trace from a second run, since on the two runs different execution sequences are used to produce the same result. For example, a program that adds the values found in four memory locations may consist of the following 8 instructions:

```
I0:   LOAD [R1], R8
I1:   LOAD [R2], R9
I2:   ADD R8, R9, R10
I3:   LOAD [R3], R11
I4:   LOAD [R4], R12
I5:   ADD R11, R12, R13
I6:   ADD R10, R13, R14
I7:   STORE R14, [R5]
```

The instruction `LOAD [R1], R8` executes by first reading the value of `R1`, using that as an index into memory to read a value, X, and writing X into `R8`. The ADD instruction sums the values found in the first two operands into the third operand; the `STORE` operation stores a register-value at a specified memory location.

One way of executing this program is by executing instructions [I0, I1, ..., I7] in order, but another, equally valid execution path will execute [I1, I0, I3, I4, I5, I2, I6, I7]. Indeed, there are 80 different ways of executing these 8 instructions that will all produce the same result. Instruction descheduling means that at run time the processor will select, at random, an instruction to execute, thereby randomising the instruction stream, and randomising the access pattern to memory caused by both data and instruction streams.

Clearly at the start and end of the program there is no non-determinism, It is the execution of the program which will be non-deterministic and not the final output of the program.

Random Instruction Selection The random instruction selection unit selects instructions from the instruction stream that are executable. That is, the instruction does not depend on any result that is not yet available, and the instruction does not overwrite any data that is still to be used by other instructions that are not yet executed, or instructions that are in execution in the pipeline.

The implementation of this closely follows the implementation of multi-issue processors. There is a block of logic that determines conflicts between instructions, resulting in a set of instructions that is executable. From this set we select an instruction at random. Given a random number generator, which will normally be constructed from a pseudo random number generator that is reseeded regularly with some entropy, we select one of the executable instructions and schedule it for execution.

Conditional Branches As with superscalar processors, conditional branches cause a problem. The non-determinism in the code is reduced if the issue unit is drained on a conditional branch and filled up immediately after the branch is taken. As with superscalars, one solution is to employ branch prediction and speculative execution.

A neater solution is to split the branch instruction into two instructions: `settarget` and `leap` [18]. The `settarget` instruction (conditionally) sets the

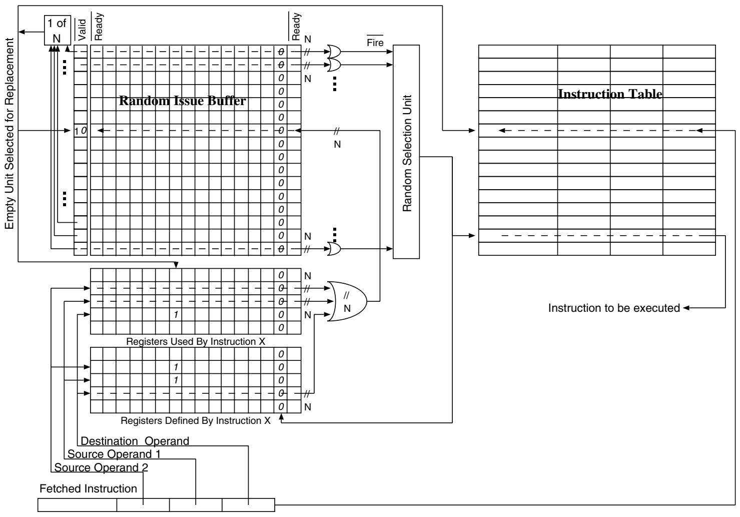


Fig. 2. Sample implementation of random issue unit

target address for the `leap` instruction. As soon as a `leap` instruction is loaded into the random-issue unit, the random-issue unit executes the instruction by loading the prefetch program counter with the target address.

Memory Accesses Memory accesses can be scheduled out-of-order, unless they interfere. Loads can be scheduled completely out-of-order if they access memory rather than I/O devices. Stores can be issued out of order provided that memory consistency is preserved. Again, this is a problem addressed in previous research on load-store units for superscalar processors.

Example Implementation of a Random-Issue Unit A possible implementation of a random-issue unit is shown in Figure 2. The instructions are read and stored in the instruction prefetch register. The operands of the instruction are decoded (we have assumed three operand instructions, although the scheme will also work with two and one operand instructions), and the dependencies of the operands are analysed using bitmasks to record dependencies:

– A bit-mask stores the use-dependencies of each register. Each register that this instruction needs to read is looked up, and the bits are or-ed together.
– A bit-mask stores the define-dependencies of each register. Each register that this instruction needs to write is looked up, and the bits are or-ed together with the previous bit-mask.
– The bit-mask that is created is stored in a free slot of the random issue buffer, together with the instruction.

In parallel, a slot of the random issue buffer for which all dependency bits are zero is selected at random. This instruction can be executed because it does

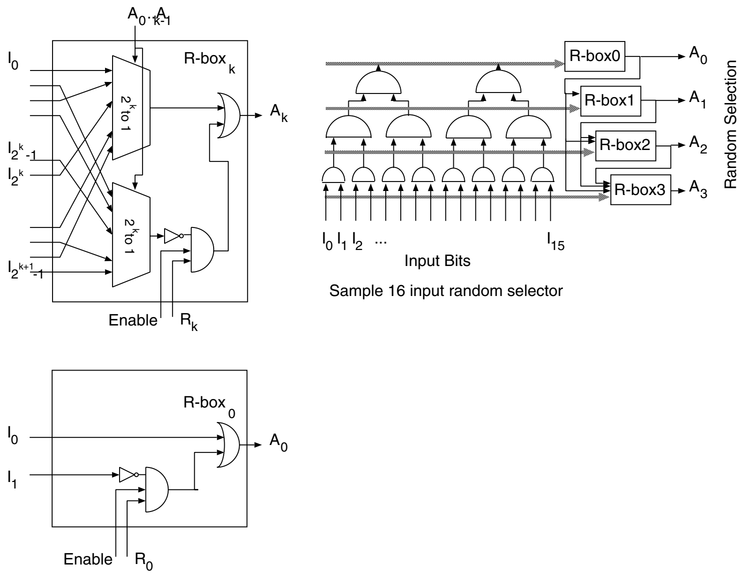


Fig. 3. Sample implementation of random selection unit

not have any define- or use-dependencies. Then all dependencies for the selected instruction are cleared so that dependent instructions become executable.

A possible implementation of a random selection unit is given in Figure 3 [15]. This unit only selects one from the 2^k bits that have a value 0. Instructions progress as follows through this implementation of the random-issue unit:

– An instruction is read into the instruction register and split into its components including the registers used in the operands.
– Each operand is looked up in the defined-by and used-by tables. Bit-masks are read indicating whether source operand registers to this operation are the result of a previous instruction, and if the destination register may overwrite a value used by previous instructions.
– The values of these bit-masks are or-ed, resulting in a new bit-mask that specifies the instructions on which this instruction depends. This bit-mask is stored in an empty slot of the random issue table, and the instruction is stored in an associated slot in the instruction table.

Instructions are selected for execution as follows:

– For each instruction, all bits in the dependency mask are or-ed, resulting in a '0' if there are no dependencies for this instruction.
– From all instructions where the or results in a '0', one is selected at random. This selected instruction is sent off into the execution pipeline. For a multi-issue machine 2 or more ready instructions can be chosen and executed.
The *enable signal* of the random-issue unit is fed from a *mode-bit* which allows the programmer to disable random-issue so that non-determinism

can be switched off to debug a program. For production systems, the enable input should be hard-wired to 1.
– The dependency-column for that particular instruction is erased; indicating that any instruction that was waiting for this instruction can now be executed.

This process is repeated ad infinitum.

Random Issuing in Superscalar Processors Random issuing works both on pipelined and superscalar processors. For the sake of simplicity we only describe it in terms of single pipeline processors. In order to illustrate the effects of non-determinism on a superscalar machine, consider the following instructions:

```
ADD R0, R1, R1
ADD R2, R3, R3
XOR R4, R5, R5
XOR R6, R7, R7
```

On a 2 processor with 2 execution unit (logic + arithmetic) these instructions are normally executed as follows. In the first clock-cycle the first ADD and the first XOR are executed; in the second clock-cycle the second ADD and second XOR are executed. Although the instructions are independent, it is still completely deterministic in that the *ADD* instructions will pass through the integer unit one after the other, and the *XOR* instructions will pass through the logic unit. A non-deterministic version of this superscalar processor would in execute one of the ADD operations and one of the XOR operations in the first clock-cycle, and the remaining ADD and XOR in the next clock-cycle. This way, there are four possible execution traces.

A non superscalar random issue processor would take twice as long to execute this program, but would execute one of 12 possible execution paths. There is a trade-off between the number of possible execution paths and the amount of parallelism exploited. For maximum non-determinism, the processor should execute instructions in a single pipeline.

3.2 Techniques for Increasing Non-determinism

Non-deterministic processing opens up new challenges in processor and instruction set design. In the following sections we examine a number of these.

In many common cases code is generated which contains unnecessary restrictions on the order in which instructions can be issued. For example, consider that we want to XOR 4 numbers, $R1 \oplus R2 \oplus R3 \oplus R4$. The fastest way to perform this on a machine which can XOR two integers in parallel is by computing $(R1 \oplus R2) \oplus (R3 \oplus R4)$:

```
I0: XOR      R1,R2,R5
I1: XOR      R3,R4,R6
I2: XOR      R5,R6,R5
```

In a non-deterministic machine with a single pipeline there would be two legal execution paths: [I0,I1,I2] and [I1,I0,I2]. However, there are actually more ways of computing this result, for example: $(R1 \oplus R3) \oplus (R2 \oplus R4)$, which generates another power trace. We discuss two assembly instructions to increase non-determinism, multi-source reduction operations and Ignore/Depend instructions.

Multi-source Reduction Operation Many operations have a number of possible ways of executing them. This is mainly because they involve executing a single operation on a set of data. An easy, and often used example, is when the operation involved is both associative and commutative. This happens in the case of addition, multiplication and *XOR*.

For example, consider XORing four values in registers $R1, \ldots, R4$ This could be done in a number of ways, all of which give the same result, but all of which would have different power outputs.

$$\begin{aligned}((R1 \oplus R2) \oplus R3) \oplus R4 &= (R1 \oplus R2) \oplus (R3 \oplus R4)\\ &= ((R1 \oplus R3) \oplus R4) \oplus R2\\ &\quad \ldots\ldots\\ &= (R3 \oplus R2) \oplus (R1 \oplus R4)\\ &= ((R4 \oplus R3) \oplus R2) \oplus R1.\end{aligned}$$

In standard assembly language on standard computers this would be executed in only one way, using a sequence such as

```
XOR       R1,R2,R5
XOR       R3,R4,R6
XOR       R5,R6,R5
```

which corresponds to the first of the descriptions above.

The crucial point about these multi source operations is that although the input and output are fixed the actual calculation steps are non-deterministic. There are several way in which this non-determinism can be achieved. The instruction can be interpreted in microcode randomly picking registers to add, the instruction can be translated by the compiler into a sequence of instructions that can be reordered at run time, or the instruction can be translated at run-time into a sequence of instructions.

A reduction instruction could be of the form

```
XOR R1,R2,R3,R4, R5
```

The disadvantage of this kind of instruction is that we introduce a different addressing scheme, with, in this case, 5 address, or even N address operations.

Ignore Depend Another way to increase parallelism without having to add extra operands to instructions it to introduce two extra instructions, `IGNORE` and `DEPEND`, which inform the instruction issue unit that a reduction sequence is to be processed. As an example, `IGNORE` and `DEPEND` will be used as follows to allow non-deterministic execution of $R1 \oplus R2 \oplus R3 \oplus R4$:

```
LOAD      #0,R5
IGNORE    R5
XOR       R5,R1,R5
XOR       R5,R2,R5
XOR       R5,R3,R5
XOR       R5,R4,R5
DEPEND    R5
```

In such a block the Instruction Descheduler is told to ignore the dependencies on `R5` between the three `XOR` instructions. This allows the Instruction Descheduler to execute the three `XOR`s in any order. The advantage of this solution is that we can use a conventional three address instruction set, and just add two operations `IGNORE` and `DEPEND`.

Once an `IGNORE` instruction on a particular register has been fetched, the processor will start delaying dependencies on that register; it will effectively ignore any dependencies, but store them for future reference in a second bit mask. When the `DEPEND` instruction is fetched, the dependency mask used will consist of all the delayed dependencies. The `DEPEND` instruction will therefore not complete until the dependencies have been completed.

Note that we defined `IGNORE` here in such a way that any number of `IGNORE` instructions can be outstanding; one per register in the architecture. Also, instructions not related to the `IGNORE` register can be issued between an `IGNORE`/ `DEPEND` pair, which will be executed as normal.

3.3 Compiler Techniques

Standard compiler techniques to increase concurrency by minimising dependencies between instructions can be employed to increase non-determinism. Random issue processors will have a limited window to look for instructions that can be executed non-deterministically. The window of instructions may be large (eg 16) but it will by its nature be limited. The size of the window may limit non-deterministic execution. Consider the following example code:

```
LOAD     I, R0
ADD      #1, R0, R0
STORE    R0, I
LOAD     J, R1
ADD      #1, R1, R1
STORE    R1, J
```

If the random issue window is two instructions wide, then the first two instructions executed will be:

`LOAD R0, I` and `ADD R0, R0, #1`

In this case the first three instructions depend on each other. Only when the window advances to the second LOAD instruction can the processor choose between the LOAD and STORE instruction. The number of execution paths is therefore limited to 4.

However, if we resequence the original program, then the number of execution paths can be increased to 13 even with the 2 instruction issue window:

```
LOAD     I, R0
LOAD     J, R1
ADD      #1, R0, R0
ADD      #1, R1, R1
STORE    R0, I
STORE    R1, J
```

Hardware register renaming [15] can remove more dependencies. Instruction resequencing does not compromise performance or power consumption. Note that compilers for superscalars do not perform all necessary reorderings: they only perform the reorderings needed to increase instruction level parallelism. However, there are segments of code where no parallelism can be gained but where extra non-determinism is available. An example are reduction operations: even though there is only limited parallelism in a reduction operation, any order of operations produces an equally valid result.

4 Experiments

In order to verify that a non-deterministic processor makes a DPA attack more difficult, we have built a simulator that outputs a power trace. We emulate a Sparc-like processor at instruction level, and produce a power trace. The power trace is based on the operands of the instructions, the type of instructions, and the memory/register addresses involved. We split the power consumption into static and dynamic component [20]:

- The Hamming weight of all addresses, operands, and instructions involved (which models the static power consumption)
- The Hamming weight of the changes in registers, busses, and the ALU (this models the dynamic power consumption)

This model closely follows the model constructed by researchers performing DPA on real processors [3,4,13].

The power trace that we produce is, obviously, not a real trace but it is sufficient to show whether we can perform differential power analysis. Our trace shows a worst-case situation where the power trace of a real processor contains considerable background noise caused by, for example, prefetching logic or random cache-replacement. Our trace is worst case in that it contains all the information that allows for a DPA attack, and none of the noise. Therefore, if it is difficult to perform a DPA attack on our trace, it will be even more difficult on the real processor.

With this simulator we have executed a DES program, and performed DPA on both the deterministic and non-deterministic versions. We used DES as a test example since this is the easiest algorithm to break using DPA.

In Figure 4 we show the simulated DPA output using four guesses for a certain six-bit subkey in the DES algorithm (there are 64 guesses, we show only 4 not to clutter the paper). The large peak on the bottom right hand graph corresponds to the correct subkey being chosen.

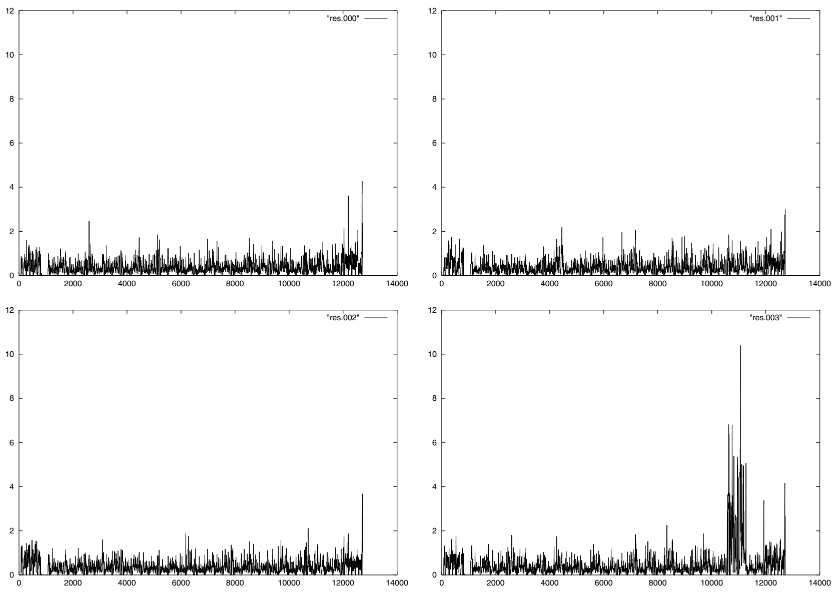


Fig. 4. Standard processor:DPA attack on DES

Using exactly the same DES source code and power model on a processor with our method of random issuing of instructions produces the DPA outputs shown in Figure 5. The peaks have disappeared in the noise, because there is little correlation left. The noise contains many peaks, but none of them flags the correct subkey.

From simulation data we have also calculated the total number of execution paths through the program. The total number of execution paths in our DES program is approximately 10^{176}, however, only the last (or first) of the 16 rounds is susceptible to attack, which leaves 10^{11} different execution traces. We think that we can further increase the number of paths by introducing more compiler analysis.

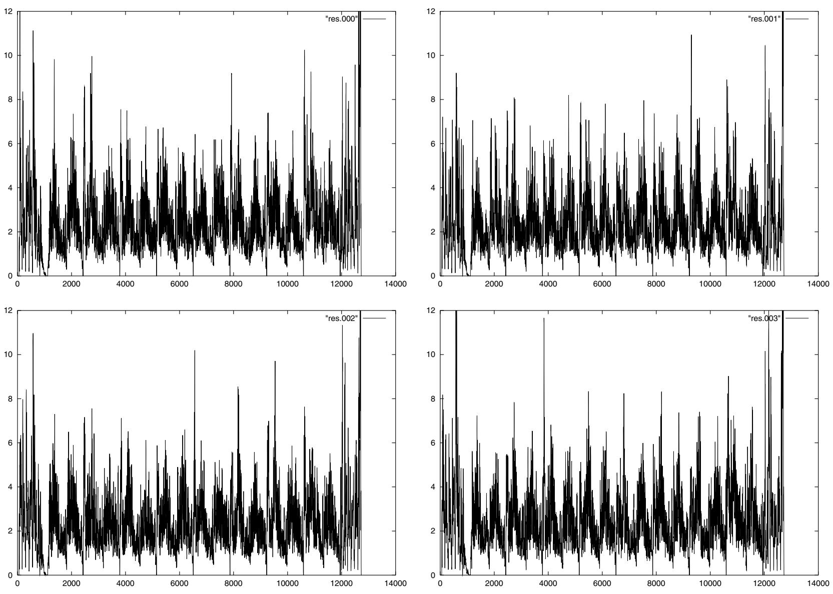


Fig. 5. NDISC processor:DPA attack on DES

5 Conclusion and Future Work

We have shown how one can use ideas from superscalar architectures to produce a randomised *non-deterministic* processor. This idea essentially allows a single instruction stream to correspond to one of an exponential number of possibly executed programs, all of which produce the same output. Since attacks such as differential power analysis rely on the correlation of data across many execution runs of the same program, the idea of a non-deterministic processor can help to defeat such attacks.

Further research still needs to be carried out, yet we feel that the proposed solution to differential power analysis gives a number of advantages; such as the fact that program code does not need to be modified and neither speed nor power consumption are compromised. The simulated results show that the DPA attack is spoiled.

References

1. R. Anderson and M. Kuhn. Tamper Resistance - a Cautionary Note. *The Second USENIX Workshop on Electronic Commerce Proceedings* pp 1–11, Oakland, California, November 18-21, 1996.
2. R. Anderson and M. Kuhn. Low Cost Attacks on Tamper Resistant Devices. *Security Protocols*, Springer LNCS 1361, pp 125–136, 1997.

3. S. Chari, C.S. Jutla, J.R. Rao and P. Rohatgi. Towards sound approaches to counteract power-analysis attacks. *Advances in Cryptology, CRYPTO '99*, Springer LNCS 1666, pp 398–412, 1999.
4. S. Chari, C.S. Jutla, J.R. Rao and P. Rohatgi. A cautionary note regarding evaluation of AES candidates on Smart-Cards. *Second Advanced Encryption Standard Candidate Conference*, Rome March 1999.
5. C. Clapp. Instruction level parallelism in AES Candidates. *Second Advanced Encryption Standard Candidate Conference*, Rome March 1999.
6. FIPS 46. Data Encryption Standard. *NIST*, 1977 Revised as FIPS 46-1:1988; FIPS 46-2:1993
7. J.A. Fisher, J.R. Ellis, J.C. Ruttenberg and A. Nicolau. Parallel Processing: A Smart Compiler and a Dumb Machine. *SIGPLAN84*, 37–47, ACM, 1984.
8. L. Goubin and J. Patarin. DES and differential power analysis. The "duplication method". *Cryptographic Hardware and Embedded Systems*, Springer LNCS 1717, pp 158–172, 1999.
9. J.L. Hennessy and D.A. Patterson. *Computer architecture: a quantitative approach.* Morgan Kaufmann Publishers, Palo Alto, California, 1990.
10. N. P. Jouppi and D. W. Wall. *Available instruction-level parallelism for superscalar and superpipelined machines.* ASPLOS-III, 272–282, 1989.
11. N. Koblitz. Elliptic curve cryptosystems. *Math. Comp.*, **48**, 203–209, 1987.
12. P. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS and other systems. *Advances in Cryptology, CRYPTO '96*, Springer LNCS 1109, pp 104–113, 1996.
13. P. Kocher, J. Jaffe and B. Jun. Differential Power Analysis. *Advances in Cryptology, CRYPTO '99*, Springer LNCS 1666, pp 388–397, 1999.
14. O. Kömmerling and M. Kuhn. Design Principles for Tamper-Resistant Smartcard Processors. *USENIX Workshop on Smartcard Technology*, Chicago, Illinois, USA, May 10-11, 1999.
15. D. May, H.L. Muller and N.P. Smart. Random Register Renaming to Foil DPA. To appear *Crytographic Hardware and Embedded Systems - CHES 2001.*
16. V. Miller. Use of elliptic curves in cryptography. *Advances in Cryptology, CRYPTO - '85*, Springer-Verlag LNCS 218, 47–426, 1986.
17. R. Rivest, A. Shamir and L. Adleman. Cryptographic communications system and method. *US Patent 4,405,829*, 1983.
18. N. Sidwell. A computer system for executing branch instructions. European 0 689 131 A1, US 08/493103.
19. D Sima, T Foutain and P Kacsuk. *Advanced Computer Architectures.* Addison Wesley, 1997.
20. N. Weste and K. Eshraghian. *Principles of CMOS VLSI design.* Addison Wesley, ISBN 0-201-53376-6, 1993.

Personal Secure Booting

Naomaru Itoi[1], William A. Arbaugh[2], Samuela J. Pollack[3], and Daniel M. Reeves[3]

[1] Center for Information Technology Integration
University of Michigan
itoi@eecs.umich.edu

[2] Department of Computer Science
University of Maryland, College Park
waa@cs.umd.edu

[3] Electrical Engineering and Computer Science Department
University of Michigan
pollack@engin.umich.edu, dreeves@eecs.umich.edu

Abstract. With the majority of security breaches coming from inside of organizations, and with the number of public computing sites, where users do not know the system administrators, increasing, it is dangerous to blindly trust system administrators to manage computers appropriately. However, most current security systems are vulnerable to malicious software modification by administrators. To solve this problem, we have developed a system called sAEGIS, which embraces a smartcard as personal secure storage for computer component hashes, and uses the hashes in a secure booting process to ensure the integrity of the computer components.

1 Introduction

With the rapid integration of information technology into society, the demand for computer system security is soaring. Despite decades of extensive research on information security, computer systems remain vulnerable to malicious modifications. This trend reflects prevalent, but inaccurate, assumptions about computer systems: that they are trustworthy. For the purpose of this paper, we define a trusted computer as "a computer system that behaves as its users intend, without damaging or leaking the resource or information".[1] A major problem today is that modern commodity computers are not trustworthy because (1) they tend to overlook or ignore physical security issues, and (2) they are vulnerable to the exploitation of software bugs. Once an adversary compromises a computer by one of the above two methods, he can install a malicious modification that defeats any security mechanism on the computer. For example, consider a Kerberos client that steals a user's password [4], an SSL client that leaks plain text

[1] This definition is narrower than the one used in the US *Trusted Computer Security Evaluation Criteria* [19], in which the word "trusted" includes access control, covert channel analysis, etc. Our definition is closer to the ones used by Neumann [18], Brewer et al. [5], Goldberg et al. [9], and Loscocco et al. [16]

packets [5, 16], or a loadable kernel module that redirects system calls to fool a system integrity checker [1, 10].

Conventionally, the problem of trusted computing has been tackled by approaches such as access control mechanisms [6], layered architecture [17], sandboxing [9, 23], and application-level integrity checking [12]. However, all of these approaches trust the underlying hardware and operating system kernels, and are of little use if any of these components are compromised. Furthermore, many of the approaches require custom operating systems, which increases management and operational problems.

To counter this problem, Arbaugh et al. have developed a high assurance bootstrap process called *AEGIS* [2, 3]. AEGIS ensures that a valid and authorized operating system kernel is started by verifying the integrity and authorization of every component that comprises the bootstrap process through the use of digital signatures and authenticity certificates. When it boots an operating system, AEGIS guarantees that the boot process takes a valid path (in terms of integrity and authorization) from the initial power-on event to the login prompt through an inductive process [2].

Although AEGIS significantly improves the security of personal computers, it has drawbacks. First, users must trust their system administrator to authorize, i.e., digitally sign, the trusted operating systems and applications. However, because (1) security threats often come from inside of organizations, and (2) in public computing sites, such as Internet cafes and libraries, system administrators are unknown, the user may choose not to trust the administrators. Second, AEGIS is inflexible: it is difficult to change the hardware configuration of a host, and it can boot only FreeBSD.

To solve these problems, we have developed *sAEGIS*, which integrates a smartcard into the bootstrap process. In sAEGIS, we use a smartcard to store the set of component hashes that the holder of the smartcard authorizes, pushing control over the selection of approved components from the system administrator to the user. We also have ported AEGIS to support GRUB [8], a free and flexible boot loader, which supports a larger set of operating systems.

The remainder of the paper is structured as follows. First, we provide a brief review of AEGIS. Next, we present the design of sAEGIS and analyze its security. Then, we describe the implementation and provide performance benchmarks for sAEGIS. Finally, we conclude the paper and provide details of our future work.

2 Background: AEGIS Secure Bootstrap Process

Here we review AEGIS to provide background for understanding sAEGIS.

AEGIS is a secure bootstrap process, whose goal is to provide a trusted foundation on a computer system. As described in Section 1, a modern computer system cannot usually be trusted because of the lack of physical security, and an untrusted initialization process. One way of addressing this problem is to ensure the *integrity* of a computer system. A system is said to possess integrity if no unauthorized modification has been made to it. Denning defines integrity similarly for communication [7]. AEGIS assures integrity of a personal computer at boot time, through a process called *chaining layered integrity checks*, which uses induction and digital certificates.

AEGIS works as follows:

1. A system administrator, or other authorized party generates a hash, H, of a bootstrap component, and creates a certificate, C, which includes a unique component identifier, an expiration date, and H.
2. The authorized party signs C with her private key.
3. C is then stored in the component if possible, and, if not, then in a data block of the flash memory device on the host's motherboard.
4. Execution control is passed to the component *if and only if:*
   (a) The certificate, C, has not expired.
   (b) The signature of C is valid, and
   (c) The hash value stored in the certificate matches the computed value of the component under consideration.

It is important to note that AEGIS provides integrity guarantees only for *starting* a system. Once a system is running, AEGIS does not provide any guarantees that the integrity of the OS remains valid.

As described in Section 1, AEGIS has two problems. First, the user is forced to trust the system administrator because the certificates are stored in the component or the BIOS, both of which are controlled by the administrator. The administrator can create and install malicious software by simply creating and signing a component certificate. AEGIS cannot detect the malicious software because the component passes all of the validity tests described earlier. The second problem is the lack of flexibility. The bootloader used in AEGIS can boot only FreeBSD. Furthermore, hardware re-configuration on AEGIS requires the creation and installation of new device certificates. Because of its size limitation, BIOS (which is in a flash memory chip) cannot store file system drivers, and is unable to access data on the hard disk.

Readers interested in the further details of AEGIS are advised to refer to articles [2, 3].

3 Design

3.1 Design Goals

The goal of sAEGIS is as follows:

– Personalization
  In AEGIS, it is a system administrator's responsibility to manage certificates and MACs. By embracing a smartcard as a personal storage of MACs, sAEGIS hands the control to the users.

- Authentication
  In AEGIS, a user who attempts to boot a computer is not authenticated. That is, anyone who can invoke the boot process, for example, by hitting the reset button, may boot it. sAEGIS boots an operating system only if a correct smartcard and associated PIN are presented by a user. This two-factor authentication (what-you-have and what-you-know) makes theft of a mobile computer less threatening, as the thief cannot use the computer.
- Operating System Flexibility
  The only operating system the AEGIS prototype is able to boot is FreeBSD. In contrast, sAEGIS employs a free, flexible boot loader called *GRUB* [8] to boot several operating systems, namely, Linux, FreeBSD, NetBSD, OpenBSD, Windows 9*, NT, and 2000.
- Hardware Configuration Flexibility
  In AEGIS, the certificates are stored in a flash memory chip, which is hard to configure. In sAEGIS, because the smartcard access library is small enough to fit into the flash chip, the hardware configuration information, certificates, and MACs can be moved to the smartcard, which is more easily configured than the flash chip.

In the above four goals, the first three were achieved in our sAEGIS prototype. The reason why the last goal was not achieved is discussed in Section 7.

3.2 Design Overview

In a nutshell, sAEGIS = AEGIS + GRUB + smartcard + `verify`. That is, (1) sAEGIS relies on AEGIS to boot GRUB securely, (2) GRUB boots an operating system kernel securely using a smartcard for verification, and (3) the kernel checks the integrity of daemons with an application called `verify`.

The basic idea behind sAEGIS is as follows: *if a lower layer verifies the integrity of all higher layers before booting them, the system integrity is ensured.* Therefore, to comprehend the design of sAEGIS, it is essential to understand which component verifies and boots which, and how. The bootstrap process of sAEGIS is summarized in the following events, in chronological order.

1. Power on Self Test (*POST*). The processor checks itself.
   POST is invoked by either applying power to the computer, hardware reset, warm boot (*ctrl-alt-del* under DOS), or jump to the processor reset vector invoked by software. This starts the bootstrap process.
2. BIOS section 1 verifies itself and BIOS section 2, and boots section 2.
   In sAEGIS, BIOS is divided into two parts, section 1 and section 2. The former contains the bare essentials needed for integrity verification, such as a cryptographic hash function (MD5 and SHA1), a public key function (RSA), and the public key certificate of a trusted third party. The integrity of this part is assumed, i.e., it is assumed to never be modified. Discussion about this assumption is in Section 4.3.

BIOS section 1 reads the certificate of itself from the flash chip, and verifies itself.

BIOS section 1 reads the binary and certificate of BIOS section 2 from the flash chip, and verifies the binary. If the check goes through, it boots section 2.

3. BIOS section 2 verifies the ROM of extension cards, and executes them.

   BIOS section 2 reads the programs stored in the ROM of extension cards, reads the associated certificates from the flash chip, and verifies the programs. If the check goes though, it executes them.
4. BIOS section 2 verifies GRUB stage 1, and boots it.

   GRUB is divided into two parts, stage 1 and stage 2, because an Intel-compatible personal computer requires a primary boot loader to be no more than 512 bytes long. Stage 1 is booted by BIOS section 2; and stage 2 is booted by stage 1.

   BIOS section 2 reads the binary of GRUB stage 1 from a floppy disk, reads the certificate from the flash chip, verifies the binary, and boots it.
5. GRUB stage 1 verifies GRUB stage 2, and boots it.

   GRUB stage 1 reads the binary and certificate of GRUB stage 2 from a floppy disk, verifies the binary, and boots it.
6. GRUB stage 2 verifies the kernel and the verification tools, and boots the kernel.

   GRUB stage 2 mounts the file system (typically on a hard disk) that stores a kernel, `verify`, and a shell script that invokes `verify` (e.g., `/etc/rc.d/init.d/inet` on UNIX). It reads these files from the file system, reads the MACs from a smartcard, and verifies the files. If the check goes through, it boots the kernel.
7. The kernel uses the `verify` application to verify the important files, and starts the system daemons that pass the check.

   `verify` is invoked by the kernel at boot to check important files. If the check fails, the kernel does not start the related daemons. The important files are system daemons (e.g., `login`, `logind`, `ssh`, and `sshd` should be verified on UNIX to detect a password sniffer), configuration files (e.g., `SYSTEM.INI` should be verified on Windows to detect a Trojan horse), and shared libraries (e.g., `GINA.DLL` should be verified on Windows NT / 2000 to detect a password sniffer).

The bootstrap process is depicted in Figure 1.

3.3 Smartcard Communication Protocol

In step 6 of the list presented above, a workstation and a smartcard carry out a protocol to (1) authenticate the smartcard and (2) verify the hash presented by the workstation. The protocol is shown in Figure 2, and is described as follows.

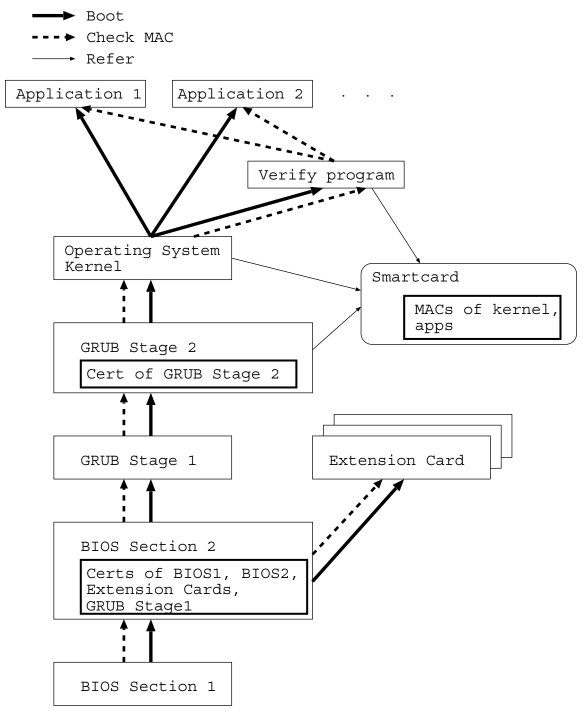


Fig. 1. Bootstrap Process

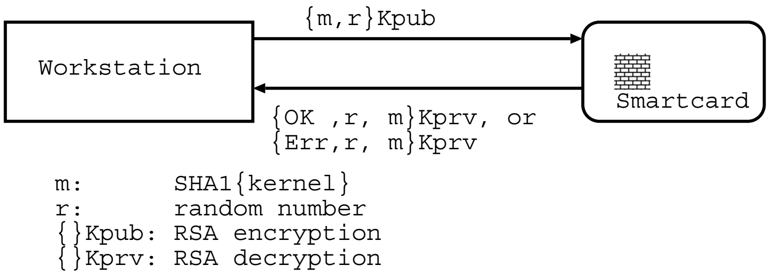


Fig. 2. Smartcard - Workstation Communication Protocol

Workstation:

- obtain PIN from the user
- compute the hash of the kernel : m = SHA1{kernel}
- generate a random challenge : r
- encrypt $\{m, r\}$ with public key : $\{m, r\}K_{\text{pub}}$
- send $\{m, r\}K_{\text{pub}}$ to the smartcard, along with the PIN

Smartcard:

- check PIN; if the PIN does not match, set ANSWER to ERR
- decrypt $\{m, r\}$ with private key
- compare m to the stored hash, and set ANSWER to OK or ERR
- sign $\{\text{ANSWER}, r, m\}$ with Kprv : $\{ANSWER, r, m\}K_{\text{prv}}$
- send it to the workstation

Workstation:

- encrypt $\{ANSWER, r, m\}K_{\text{prv}}$ with K_{pub}
- make sure it is signed by the smartcard.
- if (ANSWER == OK and r == original r and m == original m) continue with boot, otherwise, halt the boot process

4 Security Consideration

In this section, we discuss the security of our design.

4.1 Model

We start with constructing a model of the system. The model consists of the following participants:

Alice (A) A legitimate user who wants to boot and use a personal computer. She owns a smartcard.

Smartcard Alice's smartcard. It stores a private key, K_{prv}, and MACs. It is PIN protected, i.e., a secret number must be presented before it is used. It blocks itself if a wrong PIN is typed for n consecutive times.

Mallory (M) An adversary.

Personal Computer (PC) An Intel-compatible personal computer to be verified and booted. It consists of BIOS section 1 and 2, extension cards, GRUB boot loader stage 1 and 2, an operating system kernel, `verify`, and the other files.

4.2 Claims

Here we claim the security properties of sAEGIS.

System integrity after boot
: When a PC is booted using sAEGIS, the integrity of the following components of the PC are ensured; BIOS, extension cards, GRUB boot loader, operating system, and the other files that are verified.

User authentication
: When a PC is booted using sAEGIS, it has been booted by a legitimate user.

4.3 Assumptions

We make the following assumptions in our model.

1. BIOS section 1 is integral.
   We assume that the BIOS section 1 is not modified. This guarantees that section 1 starts up the sAEGIS bootstrap process every time the PC is booted.
   The security property of the entire sAEGIS system relies on this assumption because BIOS section 1 is the base of the secure bootstrap. If BIOS section 1 is modified maliciously, BIOS section 2 may not be verified correctly, resulting in a compromised section 2. This leads to a compromised GRUB stage 1, stage 2, and finally, a compromised operating system kernel. This defeats the goal of sAEGIS.
   We believe this assumption is reasonable. A portion of Intel's latest generation of flash ROM can be write-protected by setting one of the PINs (RP#) to high [11]. Although this protection can still be compromised by setting one of the jumper switches on a chip set, this attack can be countered by storing BIOS in ROM, prohibiting any modification.
2. Mallory can read anything in the PC, but nothing in the smartcard.
   Mallory can read any data stored in the PC. However, she cannot read any data in the smartcard. This is a reasonable assumption because it is usually easy to physically open a PC and access data storage in it. In contrast, a smartcard is tamper-resistant. While a smartcard suffers from newly developed attacks [13, 14], we ignore such attacks in this paper because (1) a smartcard is still much harder to compromise than a PC, and (2) smartcard developers are devising countermeasures to the new attacks.
3. Mallory can write anything in the PC except in BIOS section 1. She cannot write anything in the smartcard.
   Similarly to Assumption 2, Mallory can write anything in the PC except in the protected region. However, she cannot write anything in the smartcard.
4. Cryptographic functions are strong.
   We assume that cryptographic hash functions (MD5 used in BIOS, and SHA1 used in GRUB stage 2) are collision-free. We also assume that the random number generator used in the protocol given in Section 3.3 is unpredictable. Finally, we assume that our principal cipher, RSA, is impossible to compromise in a reasonable amount of time.

5. Mallory does not know Alice's private key.
6. Mallory can snoop and modify messages on the serial port in which the PC and the smartcard are communicating.

4.4 Attacks

Modification to PC's Components By Assumption 3, Mallory can modify anything she wants in the host except the BIOS section 1. However, if she does, Alice will notice it at the next boot because sAEGIS verifies every byte of code executed during the bootstrap process. By Assumption 1, a correct bootstrap process will be invoked every time Alice boots the PC. By Assumption 4, Mallory cannot forge a certificate or a MAC without knowing Alice's private key, and this does not happen, by Assumption 5.

Modification to PC's Components after Boot Being a secure bootstrap system, sAEGIS makes no attempt to protect the PC after it is booted. Mallory can modify the system maliciously, e.g., install Trojan horse or a sniffer. However, Alice can always restore the integrity the PC by rebooting it.

Unauthorized Boot Attempt Mallory may steal the PC and try to use it. This is impossible unless Mallory obtains Alice's smartcard and PIN, as the authentication protocol presented in Section 3.3 prevents such an attempt. Without knowing Alice's private key, K_{prv} (Assumption 2 and 5), Mallory cannot produce $\{$OK, r, m$\}K_{\text{prv}}$, because the random number generator is strong (Assumption 4).

Mallory may try to replay an OK message $\{$OK, r, m$\}K_{\text{prv}}$, but this does not work either because of the random nonce, r.

Mallory may try a man-in-the-middle attack, i.e., modifying the kernel and replacing the message from the host, $\{$m', r$\}K_{\text{pub}}$, with $\{$m, r$\}K_{\text{pub}}$. The smartcard, not knowing the hash value was altered, sends an OK message. However, the workstation notices the attack because the hash values m and m' do not match.

Serial Cable Wiretapping By Assumption 6, Mallory can read and write messages on the serial cable connecting the PC and the smartcard. However, she cannot produce $\{$OK, r$\}K_{\text{prv}}$.

PIN Theft Mallory may obtain Alice's PIN by breaking into the PC, or by sniffing the serial cable. This is a common problem for today's smartcard systems because a PIN is entered on the keyboard of the PC, and is transmitted to the smartcard through a serial cable. This problem can be addressed by a smartcard reader with a built-in PIN pad. For example, SPYRUS produces such a smartcard reader [22]. Another approach to this problem is to use a one-time pad for PINs, thus making replay of a PIN meaningless.

Mallory as System Administrator Mallory may be Alice's malicious system administrator, and may try to compromise her secrets. For example, consider a case in which Mallory tries to read Alice's e-mail. Alice may encrypt her e-mail with a secure mail tool, e.g., PGP. However, without a system like sAEGIS, Mallory can modify the executable code of PGP to leak information. sAEGIS prevents this by detecting such modifications. If the operating system and application software vendors publish the signatures of their software, Alice can store the signatures in her smartcard, and can check the system.

It is still unclear whether we can counter all the possible attacks mounted by system administrators because security software usually is written with the assumption that system administrators are trustworthy, and attacks by system administrators have not been well studied. However, we believe that sAEGIS is the first step to counter such attacks.

5 Implementation

We describe the sAEGIS prototype, which is an implementation of the design described in Section 3. It is implemented on an ASUS P55T2P4 Pentium motherboard, running a 233 MHz AMD K6 processor.

The prototype is based on the AEGIS prototype by Arbaugh et al. We do not go into the details of the AEGIS implementation. sAEGIS uses GNU GRUB 0.5.93.1. Interested readers should consult with GRUB's website [8] for details.

5.1 GRUB Stage 1

GRUB stage 1 is modified to verify GRUB stage 2 before jumping to it. Stage 1 tells AEGIS where stage 2 starts (`0x800:0`) and how large it is, and calls the AEGIS interrupt (`0xc2`).

5.2 GRUB Stage 2

GRUB stage 2 is modified to carry out the protocol described in Section 3.3.

First, to communicate with a smartcard through a serial port, the smartcard communication library is implemented by replacing the system-dependent part of the sc7816 library [21] with modified serial console access routines in OpenBSD-2.4 (`/usr/src/sys/dev/ic/com.c`).

Then, it needs some cryptographic functions. SHA1 routines in GRUB are ported from Kerberos version 5-1.0.5 distributed by MIT. RSA routines are taken from PGP 2.6.2.

In this prototype, random number generation is not implemented. It is replaced with a constant.

The `kernel` command in the GRUB user interface loads a kernel from a file system to main memory. This command is modified to invoke the verification protocol before letting GRUB boot the kernel. Another command, `updatehash`, is added to update the SHA1 hash so that files can be verified in addition to the kernel.

5.3 `verify`

`verify` is a C program that reads a given file, computes its hash, verifies it with a hash stored in a file, and returns the result of verification. An example use of `verify` is as follows. In this example, `verify` makes sure `inetd` is not modified before it is started.

```
/etc/rc.d/init.d/inet:

 /boot/verify /usr/sbin/inetd
   /boot/hash-table.txt &&
   daemon /usr/sbin/inetd
```

In future implementation, `verify` should use hashes stored in a smartcard.

5.4 Smartcard-Side Code

The program in the smartcard is implemented in a Schlumberger Cyberflex Access smartcard with Java. Cyberflex Access is the only smartcard we know that offers both programmability and cryptographic functions (DES, RSA, and SHA1).

The smartcard reads 128 byte input from GRUB, decrypts it with the RSA private key. It then compares the hash value with the one previously stored in its memory and determines whether the kernel image is unmodified. It concatenates its reply (0x8080808080808080 if OK, 0x4040404040404040 if not) with the random key and signs the resulting string with the RSA private key. Finally, it sends the result to GRUB.

In this prototype, the kernel hash is not included in the message sent from the smartcard to the host because the necessity for checking this value was identified after the prototype was implemented. In addition, a smartcard can hold only one SHA1 hash value. This should be improved to allow more flexibility.

6 Performance Evaluation

To evaluate the efficiency of sAEGIS, the boot process is timed. The following is the amount of time elapsed from the time that a PC is powered up until an operating system starts the last system daemon. In addition, the smartcard access time (the time spent in the protocol in Section 3.3) is measured, as it is one of the most expensive components.

Measurement was carried out on Linux 2.2 (RedHat 6.2) with a 233 MHz AMD K6 processor. We used the RDTSC instruction to obtain the number of ticks after the processor powers up. All the numbers are in seconds, and are averages of 5 trials each. Variance is small.

	time (sec)
boot with sAEGIS	69.55
boot without sAEGIS	57.88
difference	11.67

	time (sec)
smartcard access	5.54

The result shows that sAEGIS adds 11.67 seconds to the bootstrap process. About half of the added cost is for accessing the smartcard. The other half includes the following:

– Code checking, which involves MD5 hashing and RSA operations. More details about this are available [2].
– Loading GRUB, which is 77KB, from a floppy disk, takes more time than loading the much smaller (4.5KB) Linux boot loader, LILO, from a hard disk.

Adding 11.67 seconds to the bootstrap process, which already takes 1 minute, is acceptable in many environments.

7 Discussion

7.1 Key Management

To use sAEGIS effectively, it is essential to manage the private key in the smartcard appropriately. We describe two ways of managing private keys.

First, if the computer to be protected is personal, e.g., a laptop computer, one computer is associated with one owner. Therefore, the private key should be unique, and should be known only to the owner of the computer (i.e., should be only in the owner's smartcard). sAEGIS can prevent an adversary from booting the computer, thus discouraging theft of the computer. This approach may cause a problem when a smartcard is lost, broken, or stolen because the associated computer is no longer usable. Some kind of key escrow system is needed to address this problem.

Second, if the computer to be protected is public, e.g., in a library, or in an Internet cafe, one computer is associated with many users. The current sAEGIS prototype cannot provide such multi-user authentication because it has only one key pair between the smartcard and the computer. To achieve this, some multi-user authentication mechanism is necessary, e.g., a certificate-based mechanism with revocation, or a symmetric key-based mechanism such as Kerberos. An alternative to this is not to authenticate users at boot time, let anyone boot the computer, and rely on application level authentication. sAEGIS can achieve this by assigning the same private key for multiple users. The trade-offs between these two approaches are under discussion.

7.2 Future Direction

Fix Implementation Limitations Four implementation limitations described in Section 5 should be fixed, namely, (1) no random number generator, (2) `verify` does not use a hash in the smartcard, (3) kernel hash, m, is not included in the message the smartcard sends to the workstation, and (4) the smartcard holds only one hash.

Smartcard Access from BIOS To achieve Goal 4 described in Section 3.1, it is necessary to move the smartcard access library into BIOS. The library is 11 KB, so the size should not be a problem for the 1M flash BIOS.

Unfortunately, one of the authors who was responsible for smartcard programming did not have permission to access the BIOS source code. Instead of working out licensing issues, we decided to implement a prototype, and to wait until open-source BIOS projects are mature enough to be used as the next platform [15, 20].

8 Conclusion

We have implemented a personal, secure bootstrap process, sAEGIS, which is an extension to AEGIS. Advantages of sAEGIS over AEGIS are: (1) the smartcard lets users control what they use, (2) the smartcard serves as an authentication token, and (3) it is more flexible than AEGIS.

The following two aspects highlight the value of this work.

– Improvement to important software
  As attacks that modify an operating system itself are becoming more common, secure bootstrap, such as AEGIS, is strongly demanded. One of the problems of AEGIS is the lack of flexibility: it can only boot the FreeBSD kernel, and it requires reprogramming of a flash chip when the hardware configuration is changed. We solved the former problem, and proposed a solution to the latter.
– Idea of personalization
  sAEGIS suggests a system in which the user does not have to trust system administrators. We believe it is a huge security gain because many attacks come from inside organizations.

Acknowledgments

We thank Jim Rees at the University of Michigan for directing us about serial communication in a boot loader. We thank Professor Peter Honeyman and Professor Brian Noble at the University of Michigan for their advice.

This work was partially supported by a research grant from Schlumberger, Inc.

References

[1] Rootkit homepage. http:// www.rootkit.com/.

[2] William A. Arbaugh. *Chaining Layered Integrity Checks.* PhD thesis, University of Pennsylvania, 1999.

[3] William A. Arbaugh, David J. Farber, and Jonathan M. Smith. A secure and reliable bootstrap architecture. In *1997 IEEE Symposium on Security and Privacy*, Oakland, CA, May 1997.

[4] S. M. Bellovin and M. Merritt. Limitations of the Kerberos authentication system. In *Proceedings of the Winter 1991 Usenix Conference*, January 1991. ftp://research.att.com/dist/internet_security/ kerblimit.usenix.ps.

[5] Eric Brewer, Paul Gauthier, Ian Goldberg, and David Wagner. Basic flaws in internet security and commerce, 1995. http:// www.ao.net/ netnigga/ endpoint-security.html.

[6] A. Dearle, R. di, J. Farrow, F. Henskens, D. Hulse, A. Lindstrm, S. Norris, J. Rosenberg, and F. Vaughan. Protection in the grasshopper operating system, 1994.

[7] Dorothy Denning. *Cryptography and Data Security.* Addison-Wesley, 1983.

[8] Free Software Foundation. Gnu grub, 1999. http://www.gnu.org/ software/ grub/ grub.html.

[9] Ian Goldberg, David Wagner, Randi Thomas, and Eric Brewer. A secure environment for untrusted helper applications. In *Proceedings of 6th USENIX Unix Security Symposium*, July 1996.

[10] halflife. Bypassing integrity checking systems. Phrack Magazine, September 1997. Volume 7, Issue 51, Article 9 of 17.

[11] Peter Hazen. Flash memory boot block architecture for safe firmware updates. Technical Report AB-57, Intel, 1995. http://developer.intel.com/ design/flcomp/ applnots/292130.htm.

[12] Gene H. Kim and Eugene H. Spafford. The design and implementation of tripwire: A file system integrity checker. Technical report, Purdue University, 1995. CSD-TR-93-071.

[13] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Introduction to differential power analysis and related attacks. Cryptography Research, 1998. http://www.cryptography.com / dpa / technical / index.html.

[14] Oliver Kommerling and Markus G. Kuhn. Design principles for tamper-resistant smartcard processors. In *Proceedings of USENIX Workshop on Smartcard Technology*, Chicago, May 1999.

[15] Linux bios. http://www.acl.lanl.gov/ linuxbios/.

[16] Peter A. Loscocco, Stephen D. Smalley, Patrick A. Muckelbauer, Ruth C. Taylor, S. Jeff Turner, and John F. Farrell. The inevitability of failure: The flawed assumption of security in modern computing environments. In *21st National Information Systems Security Conference*, Crystal City, Virginia, October 1998. National Security Agency, NISSC. http://www.jya.com / paperF1.htm.

[17] H. Nag, R. Gotfried, D. Greenberg, C. Kim, B. Maccabe, T. Stallcup, G. Ladd, L. Shuler, S. Wheat, and D. van Dresser. Prose: Parallel real-time operating system for secure environments, 1996.

[18] Peter G. Neumann. Architectures and formal representations for secure systems, 1996. Technical Report SRI-CSL-96-05, Computer Science Laboratory, SRI International.

[19] Department of Defense. Trusted computer system evaluation criteria, December 1985. http:// www.radium.ncsc.mil/ tpep/library/ rainbow/5200.28-STD.html.
[20] Openbios. http://www.freiburg.linux.de/ OpenBIOS/.
[21] Jim Rees. Iso 7816 library, 1997. http://www.citi.umich.edu / projects / sinciti / smartcard / sc7816.html.
[22] Spyrus. http:// www.spyrus.com/.
[23] R. Wahbe, S. Lucco, T. Anderson, and S. Graham. client software-based fault isolation, 1993.

Evaluation of Tamper-Resistant Software Deviating from Structured Programming Rules

Hideaki Goto, Masahiro Mambo, Hiroki Shizuya, and Yasuyoshi Watanabe

Graduate School of Information Sciences, Tohoku University
Kawauchi Aoba Sendai, 980-8576 Japan

Abstract. Recently the demand to make software resistant to manipulation is increasing. Similarly the demand to hide operation of software or to hide secret used in software is increasing. Software possessing such properties is called tamper-resistant software. One of methods to realize tamper-resistant software is obfuscation of software, and evaluating such software objectively and quantitatively has been an important research subject. One of the known objective and quantitative methods is the method using a parse tree of a compiler proposed in [GMMS00]. This method takes into account the complexity in one module of software but not the complexity originated from relationships among modules. We propose at first several obfuscation methods to create a complicated module structure which violates the structured programming rules. Then, we propose a new evaluation method which can measure the difficulty caused by complicated structure among modules. Its effectiveness is proven through experiments. One of experiments shows the grades obtained by the proposed evaluation well reflects the actual reading time required by analysts.

1 Introduction

Tamper-resistance is a property such that secret object hidden inside is hardly observed or modified from the outside. Software/hardware with such attribute is called tamper-resistant software/hardware. Tamper-resistant hardware intrinsically requires a special physical device so that there are problems of cost and handling. In contrast, tamper-resistant software [Auc96, MTT97, MMO98] is expected to require less production cost. Also, due to no physical limitation, it can be delivered through electronic network. If we can create promising tamper-resistant software, we can replace a certain type of tamper-resistant hardware with its software version.

There is high demand for tamper-resistant software in the electronic commerce systems and agent systems. For example, a bank wants to prevent customers from modifying its software for handling electronic money. Customers succeeding in the modification may be able to cheat merchants as well as the bank. Similarly, mobile agents should not be modified in a remote place. If tamper-resistant software has enough strength against analysis and manipulation, its users have no choice but to obey the process designated by the software.

Obfuscation is one of approaches to generating tamper-resistant software. In this approach the description of software is converted into another one which analysts cannot easily read. Analysts who cannot understand the algorithm of software fail to properly modify the software. We can consider obfuscation in different levels of language, e.g. assembly language and high-level language like C. Software is often distributed in a binary form, but it is sometimes distributed in source code. One can imagine free application software for UNIX and codes written in script languages like Perl and Java Script for such distribution. Meanwhile, even software distributed in a binary form may be transformed into source code by reverse engineering. Therefore, obfuscation of source code has its own importance.

There are several known methods for making software hard to read. For example, several basic operations such as dummy code insertion, code replacement and code shuffling are proposed for the assembly language in [MMO98]. Modification of class files into a complicated form is proposed for Java in [CTL97, KM00]. Modification of the structure of loop into a complicated form and separating source code into modules are proposed for the C language in [MTT97] and in [TOM97], respectively.

In order to produce reliable tamper-resistant software, it is necessary to evaluate the difficulty of reading tamper-resistant software. So far the following evaluation methods are known. In [MTT97] a subject is requested to read tamper-resistant source code of C language and its reading time is counted. Without doubt this method is affected by the skill and subjectivity of each analyst. Thus an alternative objective and quantitative evaluation method should be established. There are several evaluation methods which are regarded to be objective and quantitative. In [MMO98] the distribution of opcodes is observed for evaluating the assembly language. In [GMMS00] the depth and weights of a parse tree created by a compiler is counted for evaluating the high-level language. There is another approach of [AM00] which tries to evaluate the complexity of finding out a secret hidden inside tamper-resistant software. In this method data of a block cipher appearing in memory is observed and time required for identifying a secret key out of the data is counted.

In this paper we seek to objectively and quantitatively evaluate the difficulty of reading tamper-resistant software written by a high-level language. As explained above, there is a proposal of [GMMS00] for such evaluation. However, the method proposed in [GMMS00] solely evaluates the complexity of the internal structure of a module, and does not take into account the complexity originated from relationship among modules. Therefore, we examine i) how to create a complicated structure among modules and ii) how to evaluate the complexity originated from relationship among modules. Regarding the second subject we give experimental results on the validity of our measure in comparison with the actual reading time required by analysts. Such a comparison was not examined in [GMMS00].

This paper is organized as follows. After the introduction, we explain in Sect.2 notations, definitions and the evaluation method used in [GMMS00], which is also used in our paper. In Sect.3, we explain new obfuscation methods. Then in Sect.4, we propose an evaluation method which can measure the difficulty caused by the obfuscation method in the previous section. In Sect.5, we conduct experiments and show evidence on the effect of our evaluation. Finally, conclusions are given in Sect.6.

2 Preliminaries

2.1 Notations and Definitions

Since we improve the evaluation method proposed in [GMMS00], we use their notations and definitions.

An algorithm T to generate tamper-resistant software converts a source code $c(f)$ of an algorithm f into another source code $TRC(f)$ of tamper-resistant software of f, where TRC is the acronym of Tamper-Resistant Code.

Given parameters (t, s), (t, s)-*tamper-resistant software* satisfies the following conditions.

1. Let P_c and P_{TRC} be an executable program of $c(f)$ and $TRC(f)$, respectively. Let t_c and t_{TRC} be computational time of P_c and P_{TRC}, respectively, and s_c and s_{TRC} be program size of $c(f)$ and $TRC(f)$, respectively. For given parameters (t, s), parameters t_c, t_{TRC}, s_c, and s_{TRC} satisfy
$$\frac{t_{TRC}}{t_c} < t, \quad \frac{s_{TRC}}{s_c} < s.$$
2. P_c and P_{TRC} output the same value for the same input. In other words, P_c and P_{TRC} are software performing in the same way.

Although the definition given in [GMMS00] sets a condition on memory, description on memory is omitted in the above definition. We do not use it in our analysis.

2.2 Evaluation Using Parse Tree

Computer uses a compiler for translating high-level language like FORTRAN, PASCAL and C into machine language which computer can directly execute. Conceptually, a compiler operates in the following phases one by one: lexical analysis, syntax analysis, semantic analysis, intermediate code generation, code optimization and code generation. The translation of a compiler is regarded as a sequential operation of reading, analyzing and understanding a source language. Especially, the compiler analyzes and understands a source language syntactically in the syntax-analysis phase. Such an operation is exactly what a human performs in case of reading source code. Therefore, a parse tree obtained in the syntax-analysis phase is used in [GMMS00] for evaluating the difficulty of reading tamper-resistant software. In the parse tree, the root of a parse tree, each leaf and each interior node are labeled by a *start* symbol, a *terminal* symbol, and a *nonterminal* symbol, respectively.

The evaluation rules used in [GMMS00] are as follows:

Rule 1: Weigh edges of a parse tree by the following sub-rules.

Rule 1.1: Set an initial weight into all edges of parse trees both for original source code $c(f)$ and tamper-resistant code $TRC(f)$.

Rule 1.2: Only for tamper-resistant code $TRC(f)$, change weight of edges of its parse tree depending on the algorithm used for generating $TRC(f)$.

Rule 2: Output the maximum weight, called *points*, among all sums of weight from the root to each leaf of a parse tree.

The *grades* of a tamper-resistant code is defined as the difference between the points of the tamper-resistant code and the points of the original source code. In order to assess the grade of a conversion algorithm to generate tamper-resistant software, such grades is computed for each of multiple source codes. At last, the grades of a conversion algorithm is computed by processing a set of grades by some statistical method like arithmetic mean.

From the experimental results shown in [GMMS00] modification of loop contributes to the obfuscation more than dummy code insertion and replacement of function do. Modification of loop increases the depth of nest so that a parse tree of a converted code becomes deeper. Hence modification of loop marks high grades.

However, the method described above only evaluates the complexity originated from the internal structure of a module, and fails to evaluate the complexity originated from relationship among modules. This is because functions are dealt with as a terminal symbol in the parse tree. Since the relationship among modules does contribute to the difficulty of reading software, the evaluation method should measure such complexity.

3 Proposal of Obfuscation Methods

Structured programming rules are famous programming rules allowing easy analysis and maintenance of programs. Such property conversely implies that we can obtain a complicated program by destroying the structured programming rules. In this section we explain at first structured programming rules proposed by Dijkstra et. al. [DDH72]. Then we show two obfuscation algorithms, decomposition algorithm and composition algorithm, which destroy the structured programming rules.

3.1 Structured Programming Rules

In order to allow a programmer to easily analyze and maintain programs, Dijkstra et. al. have proposed in [DDH72] the following structured programming rules:

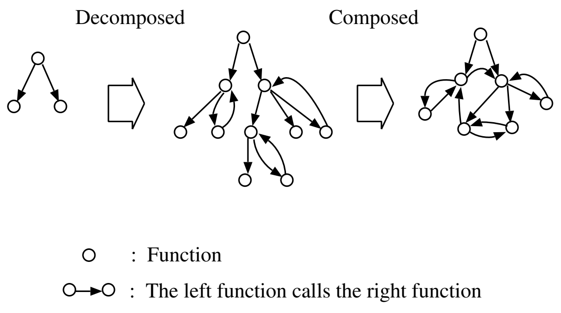


Fig. 1. Change of the structure among functions

1. A program is composed of three basic structures, concatenation, selection and repeat. Here concatenation means a sequence of statements. Selection means "**if** condition **then** statement 1 **else** statement 2" and "case-of." Repeat means "**while** condition **do** statement" and "**repeat** statement **until** condition." In other words, a program should be go-to less.
2. A program is composed of modules which can be programmed independently and revised with no, or reasonably few, implications for the rest of the system.
3. Modules are designed by the stepwise refinement.

Program design is conducted in two steps. The first step is to divide functionalities of a program. Each corresponding piece of program is called module. A good module satisfies the second rule described above. There are several ways to create modules. A top-down design is one of them. In this design functionality of program is refined stepwise, which is described in the third rule.

The second step is the design of the inside of modules. Structured programming rules are particularly useful in this regards. The first rule contributes to expressing flow of a program clearly, and the third rule contributes to giving a designer a method to think in a structured way and to reducing the risk of including errors in programs.

When a large system is designed, one should follows the following rule.

4. In a large system the division process is executed step by step, and a divided functionality is further divided afterwards. So a hierarchy of functionalities should be created.

Essentially, the fourth rule can be achieved by the stepwise refinement of the third rule.

3.2 Idea of the Proposed Obfuscation Method

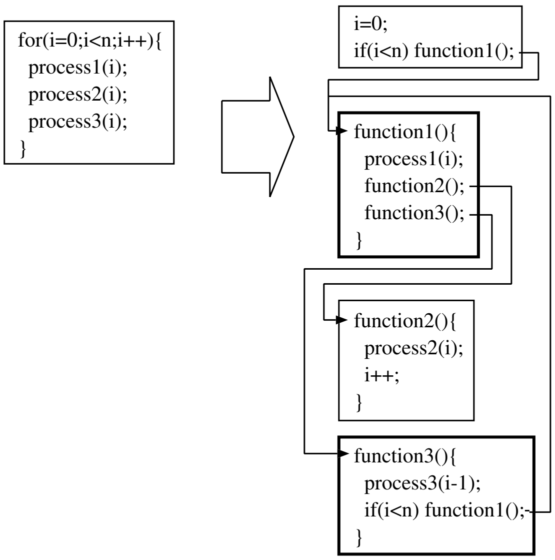
In the C language a program is a set of functions, and a module is represented by a function. Among the structured programming rules,

(a) the first rule and a part of the third rule are set as rules for dealing with one function, and
(b) the second rule, a part of the third rule and the fourth rule are set as rules for dealing with relationship among functions.

We design a program difficult to read by destroying the structured programming rules. Rules for the case (a) can be destroyed by frequently using go-to statement. Rules for the case (b) can be destroyed by making the structure among functions very complicated. Related to the latter method, we propose two obfuscation algorithms, decomposition algorithm and composition algorithm.

Before explaining these algorithms, we give an example of changing the structure among functions in Fig.1. In this figure a directed graph represents relationship among functions. A function is shown as a vertex and a functional call is shown as an edge from a calling function to a called function. The directed graph on the left is a graph of the original program. The directed graph in the middle is a graph of a program converted by the decomposition algorithm. The directed graph on the right is a graph of a program further converted by the composition algorithm.

3.3 Obfuscation through Decomposition


```
for(i=0;i<n;i++){
  process1(i);
  process2(i);
  process3(i);
}
```

```
i=0;
if(i<n) function1();

function1(){
  process1(i);
  function2();
  function3();
}

function2(){
  process2(i);
  i++;
}

function3(){
  process3(i-1);
  if(i<n) function1();
}
```

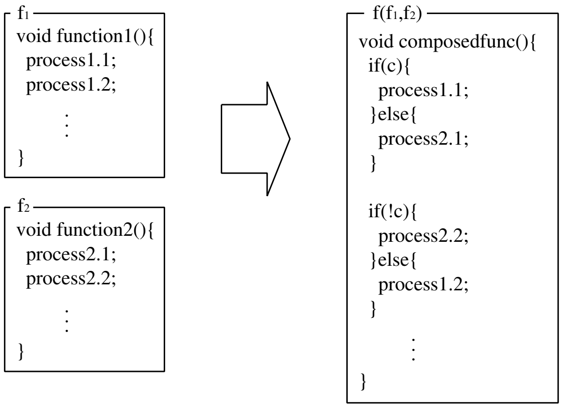
Fig. 2. Example of decomposition

A program can be obfuscated by a decomposition algorithm. The decomposition algorithm replaces loops generated by **for** statement and **while** statement with a cycle of functions, which is composed of **if** statement and multiple component functions. In this conversion, at first, data and variables used in a loop are defined externally. Then processes in a loop are divided into multiple parts, and each part is represented by a function. Finally, each of created functions is designed to call some of other functions in such a way that the loop can be replaced with created functions. When a condition of **if** statement is not satisfied, a functional call is stopped and the cycle ends. Obviously the decomposition algorithm deviates from the structured programming rules, especially the third condition, more precisely the fourth condition. The decomposition is expected to contribute to the obfuscation. An example of the decomposition is shown in Fig.2.

Conducting this conversion needs care for the value of variables. If the value of a variable in a function are changed outside the function, the value of the variable should be accordingly changed inside the function. In Fig.2 a variable i of function 3 is added outside function 3, i.e. in function 2. So, $i - 1$ is used in function 3 instead of i.

The increase of the number of functional calls, substitution and other operations leads to speed down of the execution. Therefore, it is better to adopt a conversion which does not increase the number of these operations very much.

3.4 Obfuscation through Composition



```
void function1(){
  process1.1;
  process1.2;
      ⋮
}

void function2(){
  process2.1;
  process2.2;
      ⋮
}

void composedfunc(){
 if(c){
   process1.1;
 }else{
   process2.1;
 }

 if(!c){
   process2.2;
 }else{
   process1.2;
 }
      ⋮
}
```

Fig. 3. Example of composition

A program can be obfuscated by a composition algorithm. The composition algorithm combines multiple functions performing different processes into a single function. In this conversion, at first, more than two functions having the same type for parameters and also for returned value are randomly selected. Then selection statements like **if** statement and **switch** statement are used for exclusively executing one of functions. In this way, a generated function is composed of selection statements and selected functions. Since the composition algorithm deviates from the structured programming rules, especially the second rule, it is expected to contribute to the obfuscation.

An example of the composition is shown in Fig.3. Two functions f_1 and f_2 are combined into one combined function $f(f_1, f_2)$. In the combined function $f(f_1, f_2)$ processes of one out of f_1 and f_2 are selected based on the **if** statement concerning a variable c. Naturally speaking we can further increase the difficulty by changing the condition on c into a more complicated one.

3.5 Decreasing Slow Down

The decomposition and composition algorithms sometimes introduce overhead and leads to slow down. This is because these algorithms increase the number of calling functions after the creation of cycles of functions, and also because the composition algorithm particularly increases the number of selection statements. In order to avoid unacceptable slow down in generated codes, we should adopt the following strategy for using these obfuscation algorithms. As explained above subsection, the decomposition and composition algorithms modify the syntactical structure of algorithms, and the difficulty of a code generated by these algorithms is not affected by the number of repetitions of a loop existing in the program. Since the overhead introduced by these algorithms is accumulated in every repetition of loop, we should apply the obfuscation algorithms to loops with less repetitions.

4 New Evaluation Method

As explained in subsection 2.2, The evaluation method proposed in [GMMS00] does not take into account the complexity originated from relationship among modules. So it cannot properly measure the difficulty of programs created by the decomposition and composition algorithms described in subsections 3.3 and 3.4. Therefore, we propose a new evaluation method which can deal with such complexity. Although we can consider different algorithms which produce the same output but perform in a different way, we examine only the difficulty originated from difference of representation of the same algorithm as done in [GMMS00].

Cycles of functions created by the conversion algorithms described in subsections 3.3 and 3.4 violate the fourth rule, which similarly means the third rule, of the structured programming rules. Just as a loop creates a nest in a parse tree and contributes to the obfuscation, we can regard cycles of functions as a kind of nest which is effective for the obfuscation.

We define by equivalent cycles multiple cycles containing exactly the same functions in the same calling order among them.

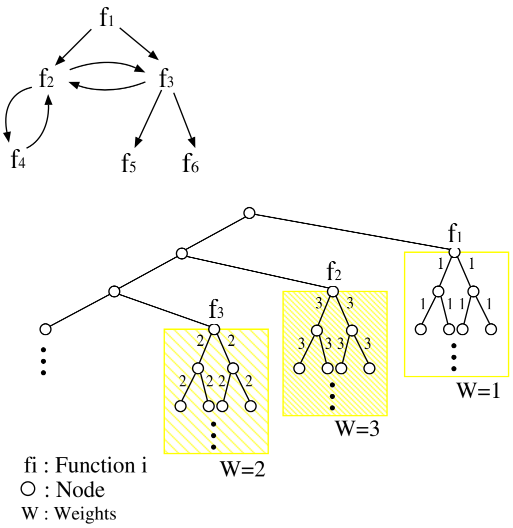


Fig. 4. Application of the new rule

Additional rule:

Step 1: Draw a directed graph on the structure among functions and find out all cycles containing more than two functions. If there are equivalent cycles, only one cycle is used out of all equivalent cycles.

Step 2: In each of the found cycles, add one to the weights of all edges of a parse tree of each function contained in the cycle.

A drawing method of the directed graph is explained in subsection 3.2.

In place of the cycle, we may be able to use two different values for the evaluation: the number of edges in the directed graph or the number of vertices in the directed graph. However, these values are easily increased just by adding functions. That means we can obtain a higher grades simply by refining functions in a stepwise way based on the structured programming rules.

On the other hand, let an upper function be a function lying in the upper layer in the hierarchical structure of functions. For instance, when f_1 calls f_2 and f_2 calls f_3, f_1 and f_2 lies in upper layers of f_3. We may be able to increase only the weights of edges of the upper function in place of all functions contained in a cycle as defined in our evaluation rule. However, functions other than the upper function are ignored in this evaluation. Since analysis of any function contained in the cycle needs the knowledge on all of other functions contained in the cycle, it is not appropriate to increase only the weights of edges of the upper function.

The new evaluation rule is exemplified in Fig.4. In the figure the tree on the bottom is a parse tree of a program and the graph depicted above the tree is a directed graph of the same program. There are two cycles, one between f_2 and

Table 1. The grades of tamper-resistant codes of program 1 (The points of the original source code is 25.)

	Alone	T_1	T_2	T_3	T_4	T_5
T_1	26(1) (1.03,1.21)	27(2) (1.00,1.48)	28(3) (1.03,2.02)	39(14) (1.03,1.33)	42(17) (1.08,1.80)	31(5) (1.05,1.71)
T_2	29(4) (1.02,1.91)	30(5) (1.04,2.19)	31(6) (1.04,$\underline{2.75}$)	43(18) (1.04,2.02)	45(20) (1.14,2.49)	35(10) (1.13,2.45)
T_3	38(13) (1.02,1.12)	41(16) (1.03,1.66)	43(18) (1.04,2.02)	52(27) (1.02,1.26)	44(19) (1.07,1.60)	53(28) (1.04,1.66)
T_4	40(15) (1.08,1.59)	42(17) (1.07,1.80)	45(20) (1.14,2.49)	57(32) (1.08,1.75)	51(26) (2.14,2.13)	56(31) ($\underline{2.17}$,1.95)
T_5	30(5) (1.05,1.54)	31(6) (1.03,1.83)	35(10) (1.13,2.45)	44(19) (1.09,1.69)	60(35) (1.09,1.97)	46(21) (1.11,1.81)

f_3 and the other between f_2 and f_4. Therefore, the weights W of all edges of a parse tree of f_2 becomes 3 and that of f_3 and f_4 becomes 2.

5 Experimental Results

We have conducted experiments for confirming the validity of the proposed evaluation. Conversion algorithms used are as follows.

1. Dummy code insertion, T_1,
2. Replacement of function, T_2,
3. Modification of loop, T_3,
4. Decomposition of functions, T_4,
5. Composition of functions, T_5.

Each of or two out of these algorithms have been applied to three programs, program 1, program 2, and program 3. Program 1 is a program for factoring. Program 2 is a program for computing the greatest common divisor. Program 3 is a program for the shell sort. These programs are relatively small programs with about 30 lines.

When we apply T_i at first and then T_j to a program p_k a corresponding algorithm of this sequential operation is expressed as T_jT_i, and a generated tamper-resistant code is expressed as $TRC_jTRC_i(p_k)$. When the same conversion algorithm T_i is applied n times, we express T_i^n and $TRC_i^n(p_k)$ for a corresponding algorithm and a generated tamper-resistant code.

Using a lexical analyzer lex and a syntax analyzer yacc, we have implemented an evaluation program.

5.1 The Grades of Tamper-Resistant Software

Table 1 shows the grades of tamper-resistant code of program 1 generated by combinations of 5 conversion algorithms. An algorithm in the raw is applied at first and then an algorithm in the column is applied. In this case the points of the original source code is 25 and the grades expressed between parentheses in the upper raw is derived by adding -25 to the points expressed without parentheses.

The execution time and the program size of the original source code of program 1 are 0.1544 seconds and 182 bytes, respectively. The execution time is computed by arithmetic mean after 100 trials under Soralis 7, Sun Ultra 10, Ultra SPARC-IIi/333MHz. File size does not count return and the space. A pair of values shown in the lower raw represents parameters (t, s) mentioned in subsection 2.1. The maximum value $(2.17, 2.75)$ of (t, s) is the underlined in Table 1. That means the results for 25 tamper-resistant codes in Table 1 can be considered as the results for $(2.17, 2.75)$-tamper-resistant software.

In Fig.5 we show the structure among functions of program 1. A directed graph in the middle represents a graph of tamper-resistant code converted by T_4 twice. A directed graph on the right represents a graph of tamper-resistant code generated by the experiment in the next subsection. It is converted by T_4 twice and then by T_5 twice.

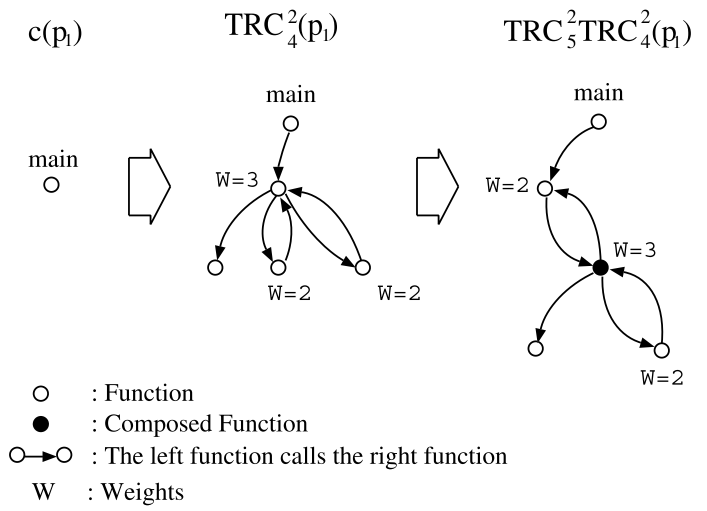


Fig. 5. Modifying the structure among functions into a complicated form

Figure 6 shows the grades of conversion algorithms with respect to program 1, 2 and 3. This figure implies that conversion algorithms composed of modification of loop T_3, decomposition of functions T_4 and composition of functions T_5 mark high grades. As mentioned in subsection 2.2, the original evaluation method in [GMMS00] can evaluate only the difficulty originated from modification of loop. The result shown in Fig.6 indicates that the proposed improvement is effective for evaluating the complexity originated from the structure among functions, either.

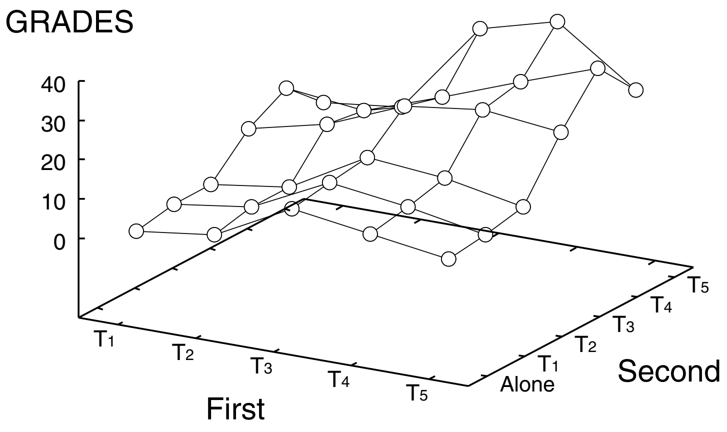


Fig. 6. The grades of tamper-resistant software with respect to program1,2,3

5.2 Relationship between the Grades and the Reading Time

Since the proposed evaluation method does not involve analysts, it is considered to be objective. However, we do not know how the computed grades relates to the actual reading time of analysts. In order to obtain an evidence of the validity of our evaluation, we have conducted experiments for clarifying relationship between the grades obtained by the proposed evaluation and the actual reading time required by analysts.

Let p_k be a program k for $k = \{1, 2, 3\}$. For a program k, an original source code $c(p_k)$, tamper-resistant codes $TRC_i^2(p_k)$ for $i \in \{1, 2, 3, 4\}$ and $TRC_5^2 TRC_4^2(p_k)$ are evaluated. In the evaluation, a subject is given a source code and data, and answers what is the output of the program. If the answer is wrong, the subject continues to read it until the subject reaches the right answer. The time until the subject answers correctly is counted.

The number of subjects is 6. The following assignment of source codes follows the idea adopted in [MTT97]. One source code is selected from each of three categories, $c(p_k)$ and $TRC_1^2(p_k)$, $TRC_2^2(p_k)$ and $TRC_3^2(p_k)$, and $TRC_4^2(p_k)$ and $TRC_5^2 TRC_4^2(p_k)$. Note that there are 6 source codes in each of these categories. Selected 3 source codes are analyzed by a subject. With this assignment, subjects do not read multiple source codes generated by the same conversion algorithm. So they do not become familiar with the program. At the same time, source codes converted from the same original program p_k are not assigned to the same subject. With this assignment, subjects do not become familiar with the program, either.

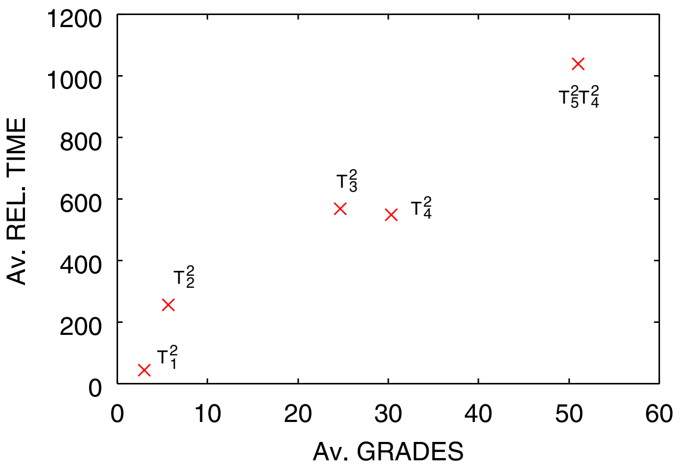


Fig. 7. Relationship between the average grades and the average relative time for analysis

The relationship between the average grades and the average relative time for the analysis is shown in Fig.7. Relative time means difference between the actual reading time of a tamper-resistant code and that of the original source code. We can observe that source codes possessing high grades require longer time for analysis. It is fair to say that the grades reflects the actual reading time of analysts.

From Fig.7 the grades of $T_5^2T_4^2$ is the highest among all examined conversion algorithms. Its rough reason in case of program 1 would be as follows. The directed graph of a tamper-resistant code of program 1 generated by $T_5^2T_4^2$ is shown in Fig.5. A composition function belongs to two cycles of functions and the weights of edges of its parse tree is increased to 3. Such increase provides high grades. Moreover, the increase of the number of selection statements after the conversion by T_5 results in the increase of nests, which also provides high grades. We can observe the similar property in cases of program 2 and 3.

From the results of experiments in the previous subsection and this subsection, we can conclude our evaluation method is effective in evaluating the complexity originated from relationship among modules.

6 Conclusions

Based on the idea to make a program deviate from the structured programming rules, we have developed two obfuscation methods to make the module structure complicated. Two obfuscation algorithms are the decomposition algorithm and the composition algorithm. The decomposition algorithm replaces loops with a cycle of decomposed functions. The composition algorithm combines multiple functions performing different processes into a single function.

On the other hand, in order to overcome the incompleteness of the evaluation method proposed in [GMMS00], we have proposed a new objective and quantitative evaluation method which can measure the difficulty of programs caused by complicated structure among modules. Relationship among modules is shown by a directed graph, and we have estimated that cycles appearing in the graph contribute to the obfuscation, and used them for evaluation. Experimental results show that the difficulty originated from the structure among modules is evaluated in the proposed method. We have also examined the relationship between the grades and the actual reading time required by analysts. A corresponding result tells that the grades obtained by the proposed method well reflects the actual reading time.

References

[AM00] Kenichiro AKAI, Tsutomu MATSUMOTO, "Brute force method of tamper resistance evaluation for block cipher software," Computer Security Symposium 2000, CSS 2000, IPSJ Symposium series, Vol.2000, No.12, pp.205-210, 2000. [in Japanese]

[Auc96] David Aucsmith, "Tamper Resistant Software: An Implementation," Information Hiding, IH '96, Lecture Notes in Computer Science Vol.1174, Springer-Verlag, pp.317-333, 1996.

[CTL97] Christian Collberg, Clark Thomborson, Douglas Low, "A Taxonomy of Obfuscating Transformations," Technical Report 148, Department of Computer Science, The University of Auckland, 1997.

[DDH72] Ole-Johan Dahl, Edsger Dijkstra, C.A.R. Hoare, Structured Programming, A.P.I.C. Studies in Data Processing, No.8, Academic Press, 1972.

[GMMS00] Hideaki Goto, Masahiro Mambo, Kenjiro Matsumura, Hiroki Shizuya, "An Approach to the Objective and Quantitative Evaluation of Tamper-Resistant Software," Information Security Workshop, ISW2000, Lecture Notes in Computer Science Vol.1975, Springer-Verlag, pp.82-96, 2000.

[KM00] Jun KUWAHARA, Tsutomu MATSUMOTO, "A Method of Constructing Tamper-Resistant Java Classfiles," Proc. of 2000 Symposium on Cryptography and Information Security, SCIS2000-D10, Jan. 2000. [in Japanese]

[MMO98] Masahiro MAMBO, Takanori MURAYAMA and Eiji OKAMOTO, "A Tentative Approach to Constructing Tamper-Resistant Software," 1997 New Security Paradigms Workshop, ACM Press, pp.23-33 1998.

[MTT97] Akito MONDEN, Yoshihiro TAKADA and Koji TORII, "Methods for Scrambling Programs Containing Loops," Trans. of IEICE, Vol.J80-D-I, No.7, pp.644-652, 1997. [in Japanese]

[TOM97] Eisaku Teranishi, Eiji Okamoto and Masahiro Mambo, "A Proposal of Copyright Protection Scheme for Software Programs," Proc. of 1997 Symposium on Cryptography and Information Security, SCIS97-10B, Jan. 1997. [in Japanese]

A Strategy for MLS Workflow*

Vlad Ingar Wietrzyk[1], Makoto Takizawa[2], and Vijay Varadharajan[3]

[1] School of Computing, University of Western Sydney
Australia
v.wietrzyk@uws.edu.au
[2] Department of Computures and Systems Engineering
Tokyo Denki University, Japan
[3] Department of Computing, Macquarie University - Sydney, Australia

Abstract. This paper describes the design of a model as well as an architecture to provide support for distributed advanced workflow transactions. We discuss the application of transaction concepts to activities that involve integrated execution of multiple tasks over different processes. This kind of applications are described as transactional workflows. The classical commit protocol, used in many commercial systems, is not suitable for use in multilevel secure distributed workflow database systems that use a locking protocol for concurrency control. We choose to develop formal framework for secure distributed workflow architecture since we are actively involved in building a prototype of such a system. We strive to develop a practical logical characterization of multilevel secure (MLS) distributed workflow for the first time using the inherently difficult concept of non-monotonic reasoning.

1 Introduction

Many technical and nontechnical issues hinder enterprise-wide workflow management. Because workflow types cannot always be fully predefined, they often need to be adjusted or extended during operation. Distributed workflow execution across functional domains is necessary, but distribution transparency is currently impossible because, different types of Workflow-Management-Systems (WFMSs) implement different WFMS metamodels.

One possible way to enable distributed workflow execution is to build a workflow-management infrastructure integrating different and heterogeneous workflows. Users would have access to total funcionality because they access the workflow-management underlying infrastructure, not individual WFMSs. The resulting architecture is general and can accommodate as many WFMSs as required.

Transaction concepts have begun to be applied to support applications or activities that involve multiple tasks of possibly different types - including, but not limited to transactions, and executed over different types of entities - including

* Research supported by the **UWS, Versant Technology Corporation** and **Intel Corporation**

DBMSs. Generally we will refer to such applications as multi-system transactional workflows.

The recent trend to distribute workflow executions requires an even more advanced transaction support system that is able to handle distribition. Workflow applications are long-duration applications since the duration of a workflow can range from a few hours to a few months.

To summarize, the new aspects of our approach to security in distributed workflow database management systems include the following researh contributions. The novel approach to the development of a practical logical characterization of multilevel secure (MLS) distributed workflow for the first time using the inherently difficult concept of non-monotonic reasoning. Distinguishing feature of the workflow transaction support system proposed is the ability to manage the arbitrary distribution of business processes over multiple workflow management systems. We also derived general theorem which must be active when classifying every item of information.

1.1 Outline of the Paper

We have planned the presentation of the current research as follows. We first present a brief introduction to work on workflow transaction models and discuss extended – relaxed approach to handle workflow transactions in section 2. Section 3 covers related aspects of workflow distribution and heterogeneity. A number of relaxed transaction models in workflow contexts that have been defined recently permitting a controlled relaxation of the transaction isolation and atomicity to better match the requirements of various workflow applications are discussed in section 4. In section 5 we develop a formal model and some axioms related to multilevel secure distributed workflow object-relational model are given from which theorems regarding secure workflow database models are derived. Section 6 concludes the paper with a summary and a short discussion of future research.

2 Related Work

The traditional transactions are usually characterized by the atomicity, consistensy, isolation and durability requirements, called the ACID properties of transactions. Some known examples of `extended transaction models` include nested and multi-level transactions. Some examples of extended – relaxed transaction models are reported in [1, 2].

In the WIDE project [3], a workflow is supported at two transaction levels: global and local. At the global level, the SAGA - based model offers relaxed atomicity through compensation and relaxed isolation by limiting the isolation to the SAGA steps. Some researchers in workflow systems have proposed the notion of transactional workflow [4]. In transactional workflow environment, additional correctness requirements can be specified on top of traditional workflow specifications.

The Workflow Management Coalition has specified a standard interface to facilitate the interoperability between different WFMSs [5]. However, they do not address transactional issues with the exception of writing an audit log.

The transaction model used in the Exotica project [6] is based on the SAGA model, but relies on statically computed compensation patterns. As a result, its functionality is limited compared to the work presented in this research paper.

Finally, most commercial products are designed around a centralized database. This database and the workflow engine attached to it — in most cases there is a single workflow engine are a single point of failure which quickly become a bottleneck and are not capable of providing a sufficient degree of fault tolerance.

Very often, a WFMS processes data for which high standards must be set with respect to privacy and data security. Most of the workflow transaction management theory for multilevel secure database systems has been developed for workflow transactions that act within a single security class. In our research work, we look at workflow transactions that act across security classes, that is, the workflow transaction is a multilevel sequence of database commands, which more closely resemble user expectations. We propose a formal model and semantics for interpreting security issues in a workflow architecture which can incooperate a multilevel deductive database.

3 Workflow Distribution and Heterogeneity

Workflow distribution introduces additional level of requirements. Because distributed workflow execution across heterogeneous WFMSs is currently not possible in a transparent way, we must to consider the problem of workflow funcionality isolation.

A workflow is distributed when at least two of its objects reside in two different WFMS installations. This is relevant to workflow definitions as well as workflow instances. An often-cited situation is subworkflow distribution, where subworkflows are subject to excution on remote WFMSs. Some variants are possible, such as executing a subworkflow synchronously or asynchronously to the invoking workfow. One of the typical variant involves executing some part of a workflow on one WFMS, and continuing on another (see Fig.1).

If the associated WFMSs, do not know about each other, it's indirect distribution. In this case, the WFMSs do not implement distribution natively, and system designer must attach distribution functionality to the associated WFMSs. A recognised way is to establish communication buffers between the WFMSs, such as a database or persistent file stores. Fig. 2 shows an example workflow definition with one distribution task. The distribution task invokes an application for buffer communication. Typically, workflow types can be distributed, too.

3.1 An Architecture for Multilevel Secure Workflow Interoperability

Global information management strategies based on a sound distributed architecture are the foundation for effective distribution of complex applications

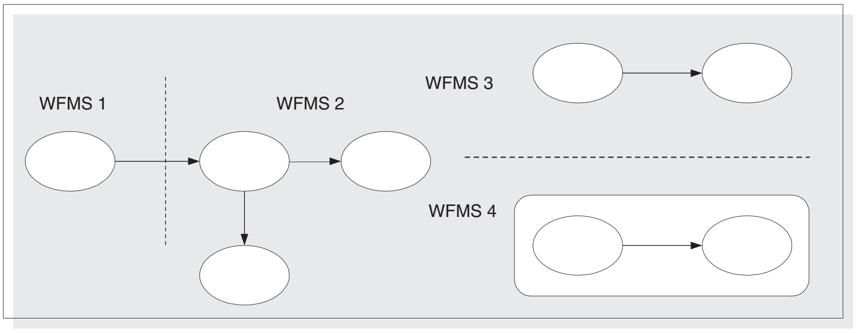


Fig. 1. Workflows Division across different WFMSs

that are needed to support ever changing operational conditions across security boundaries. What we need is a new MLS distributed computing paradigm that can assist users at different locations and at different security levels to cooperate.

We present the fully distributed architecture for implementing a Workflow Management System (WFMS). An MLS workflow distributed database consists of a set $\mathcal{N}$ of sites, where each site $N \in \mathcal{N}$ is an MLS database. The sites in the workflow system are interconnected via communication links over which they can communicate. The WFMS architecture operates on top of a Common Object Request Broker Architecture (CORBA) implementation. A CORBA's Interface Definition Language (IDL) is used to provide a means of specifying workflows. Also we assume that communication links are secure — possibly using encryption. This distributed workflow transaction processing model describes mainly those components necessary for the distribution of a transaction on different domains.

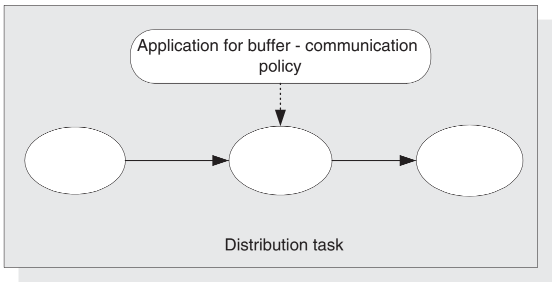


Fig. 2. The Distribution Task Invokes an Application for Buffer Communication

Domain is a unit of autonomy that owns a collection of flow procedures and their instances. In practical terms, a domain might define the scope of a department or division in an organization. Therefore, flows are grouped by domains, and each domain also manages a set of flow procedures installed in the domain. A domain is not defined or limited by networks, processors, or peripherals. The manager of resources can, however, be designed in any fashion, they are exclusively responsible for the ACID properties on their data records. Solely the interface to the components of the distributed workflow model must exist.

If a transaction should be dstributed on several domains — a global transaction, in every domain there must exist the following components, (see Fig.3).

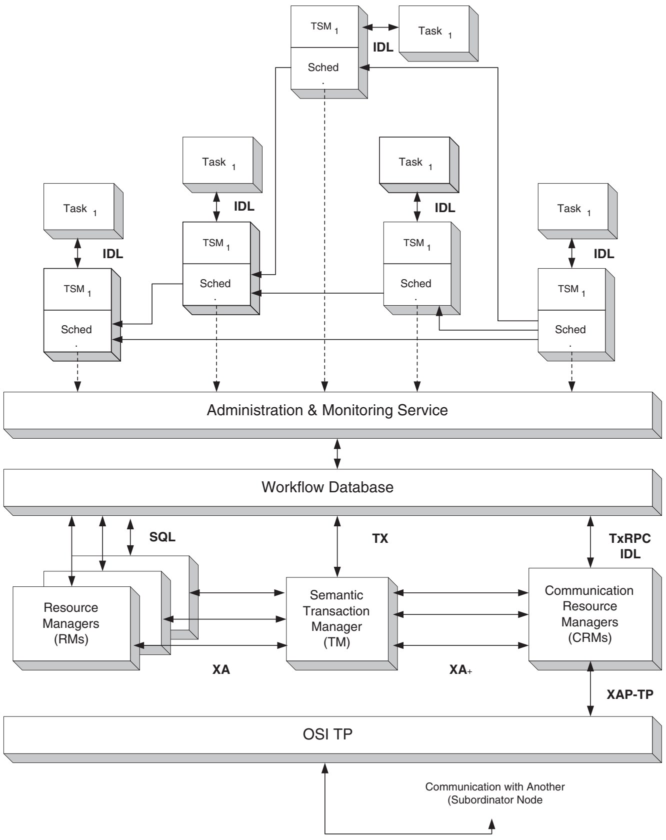

TSM 1
Sched
Task 1
IDL
Administration & Monitoring Service
Workflow Database
SQL
TX
TxRPC
IDL
Resource Managers (RMs)
Semantic Transaction Manager (TM)
Communication Resource Managers (CRMs)
XA
XA+
XAP-TP
OSI TP
Communication with Another (Subordinator Node

Fig. 3. Distributed Workflow Architecture

– TM - Transaction--Manager. The transaction manager plays the role of the coordinator in the respective domain. If a transaction is initiated in this domain, the TM assigns a globally unique identifier for it. The TM monitors all actions from applications and resource managers in its domain. In every domain involved in the distributed workflow transaction environment there exists exactly one TM.

- CRM - Communication--Resource--Manager. Multiple applications in the same domain talk with each other via the CRM. This module is used by applications but also other management components for inter-domain communication. CRM is the most important module with respect to the transactional support for distributed workflow executions. Our model specifies the T*RPC as a communication model, which supports a remote procedure call (RPC) in the transactional environment.
- RM - Resource--Manager. An accountable performer of work. A resource can be a person, a computer process, or machine that plays a role in the workflow system. This module controls the access to one or more resources like files, printers or databases. The RM is responsible for the ACID properties on its data records. A resource has a name and various attributes defining its characteristics. Typical examples of these attributes are job code, skill set, organization unit, and availability.
- AMS - Administration--Monitoring--Service. The monitoring manager is used to control the workflow execution. In our approach, there is no centralized scheduler. In the figue, each Task Manager - designated as TSM, is equipped with a conditional fragment of code which determines if and when a given task is due to start execution. The scheduler communicates with task managers using CORBA's asynchronous Interface Definition Language(IDL) interfaces. Task managers communicate with tasks using synchronous IDL interfaces as well. AMS module is also responsible for the coordination of the different sites in case of an abort that involves multiple sites. Individual task managers communicate to monitoring manager their internal states, as well as data object references - for possible recovery.

The distributed architecture suits the inherent distributional character of workflow adequately in a natural way. This approach also eliminates the bottleneck of task managers having to communicate with a remote centralized scheduler during the execution of the workflow. This architecture also posseses high resiliency to failures — if any one node crashes, only a part of the workflow is affected.

4 Relaxed Transaction Models in Workflow Contexts

A number of `relaxed transaction models` have been defined recently that permit a controlled relaxation of the tranaction isolation and atomicity to better match the requirements of various workflow applications. Usually, we will refer to such applications as multi-system transactional workflow. This area has been also influenced by the concept of long running activities.

As has been pointed out in [7], WFMSs lack the ability to ensure the correctness and reliability of the workflow execution in the presence of concurrency and failures.

4.1 Transactional Workflows

Support for workflow applications has been addressed by researchers focusing on workflow systems and transaction systems. Our approach falls in the category of transactional workflows [4] where additional correctness requirements can be specified on top of traditional workflows specifications. `Flexible transactions` work in the context of heterogeneous distributed multidatabase workflow environments [8]. In such workflow environments, each database acts independently from the others. Because a local database can unilaterally abort a transaction, it is not possible to enforce the commit semantics of global transactions. Therefore, flexible transaction were designed to address this problem.

4.2 The Functionality of Flexible Transactions in Workflow Systems

A flexible transaction is specified by providing: the precondition of the global transaction, a set of subtransactions, the externally visible states of each subtransaction and the possible transitions among these externally visible states, preconditions and postconditions for the possible transitions of each subtransaction, and the postcondition of the global transaction.

To better support workflow operational environment, the flexible transaction model relaxed the isolation and atomicity properties. This approach is the direct result of our believe, that tying a workflow system to a particular transaction model, will result in major restrictions that will limit its applicaility and usefulness as a workflow tool.

4.3 A Formal Model of Flexible Transactions

From a user's point of view, a `transaction` is a sequence of actions performed on data items in a database. Flexible transaction model proposed for the distributed workflow environment will increase the failure resiliency of global transactions by allowing alternate subtransactions to be executed when a local database fails or a subtransaction aborts. The approach supports the concept of varied transactions allowing compensatable and noncompenstable subtransactions to coexist within a single global transaction. The concurrency control of global transactions require, that each global transaction has at most one subtransaction at each local site [9]. Following [8, 10], the definition of flexible transactions takes the form of a high-level specification. The flexible transaction model supports flexible execution control flow by specifying two kinds of dependencies among the subtransactions of a global transaction:

- Execution ordering dependencies between two subtransactions.
- Alternative dependencies between two subsets of subtransactions.

In what follows, we shall formally describe the flexible execution control in the flexible transaction model.

Let $\Omega = \{t_1, t_2, \ldots, t_n\}$ be a collection of subtransactions and $\Pi(\Omega)$ the collection of all subsets of Ω. Let $t_i, t_j \in \Omega$ and $T_i, T_j \in \Pi(\Omega)$. Two types of control flow relations are defined on the subsets of Ω and on $\Pi(\Omega)$, namely:

– precedence $t_i \prec t_j$ if t_i precedes t_j $(i \neq j)$;
– preference $T_i \rhd T_j$ if T_i is preferred to T_j $(i \neq j)$. If $T_i \rhd T_j$, we also declare that T_j is an alternative to T_i.

Both of the above relations, precedence and preference are irreflexive and transitive or more formally, for each $t_i \in \Omega$, $\neg(t_i \prec t_i)$; and for each $T_i \in \Pi(\Omega)$, $\neg(T_i \rhd T_j)$. If $t_i \prec t_j$ and $t_j \prec t_k$, then $t_i \prec t_k$; if $T_i \rhd T_j$ and $T_j \rhd T_k$, then $T_i \rhd T_k$.

From he above definitions, we can see than, the precedence relations determines the correct parallel and sequential execution ordering dependencies among the subtransactions, while the preference relation determines the priority dependencies among alternate sets of subtransactions for selecting in completing the execution of Ω.

Now a flexible transaction can be defined as follows:

Definition 1. Flexible transaction *A flexible transaction Ω is a set of related subtransactions on which the precedence $(\prec)$ and preference $(\rhd)$ relations are defined.*

The semantics of the precedence relation refers to the execution order of subtransactions. For example, $t_1 \prec t_2$ may imply that t_2 cannot start before t_1 finishes or that t_2 cannot finish before t_1 finishes. By the same token, the preference relation defines alternative choices and their priority. For example, $\{t_i\} \rhd \{t_j, t_k\}$ may imply that t_j and t_k must abort when t_i commits or that t_j and t_k should not be executed if t_i commits. In this environment, $\{t_i\}$ is of higher priority than $\{t_j, t_k\}$ to be chosen for execution.

We consider that a workflow database state is consistent if it preserves workflow database integrity constraints. As it is the case for traditional transactions, the execution of a flexible transaction as a single unit should map one consistent multidatabase workflow state to another. We designate relation $(T_i, \prec_i)$ as a partial order of subtransactions. $(T_i, \prec_i)$ is a representative partial order, if the execution of subtransactions in T_i represents the execution of the entire flexible transaction Ω. From the above it is clear that, if $(T_i, \prec_i)$ is a representative partial order, then there are no subsets T_{i1} and T_{i2} of T_i such that $T_{i1} \rhd T_{i2}$. Because each global transaction has at most one subtransaction at a local site, each representative partial order of a flexible transaction must have at most one subtransaction at a local site. In our workflow execution environment, for flexible transactions, the above definition of consistency requires that the execution of subtransactions in each representative partial order must map one consistent workflow multidatabase state to another.

4.4 Scheduling of Flexible Transactions

Since the flexible transaction model was proposed, much research has been devoted to its application. The availability of visible prepare–to–commit states in local database systems is the basic assumption underlying this work. Also, time used in conjunction with subtransaction and global transaction can be exploited in transaction scheduling.

A schedulable subtransaction may be submitted for execution to the transaction module. The scheduler first has to check for satisfaction of the preconditions for execution of each subtransaction — it determines whether a subtransaction is schedulable. This entails the specification of the execution dependency among the subtransactions of a global transaction. Execution dependency [4], is a relationship among subtransactions of a global transaction which determines the legal execution order of the subtransactions. To support the specification of the execution dependency, we define a transaction execution state as follows:

Definition 2. *The transaction execution state x for a global transaction T with m subtransactions, is an $m-tuple$ $(x_1, x_2, \dots, x_m)$ where:*

$$x_i = \begin{cases} E & \textit{if } t_i \textit{ is currently being executed;} \\ N & \textit{if subtransaction } t_i \textit{ has not been} \\ & \textit{submitted for execution;} \\ S & \textit{if } t_i \textit{ has successfully completed;} \\ F & \textit{if } t_i \textit{ has failed or completed without} \\ & \textit{achieving its objective;} \end{cases}$$

Under normal operational circumstnces transaction execution state is used to keep track of the execution of the workflow subtransactions. It is also used to determine if a global workflow transaction has achieved its objectives. When a subtransaction t_i complete the corresponding execution state, x_i is set to S if the subtransaction has achieved its objective, and to F, therwise. At a certain point of execution, the objectives of the global workflow transaction may be achieved. At that point, the global transaction is considered to be successfully completed and can be committed.

A number of approaches can be used to assure global serializability which constitutes a satisfactory correctness criterion for concurrent execution of multidatabase workflow transactions, if there is a lack of additional information about their semantics. The objective of concurrency control is to assure that the serialization order of multidatabase workflow transactions should be the same, at all sites they execute. It was shown in [8, 11], that the above condition is sufficient to assure global serializability. However, in our workflow operational environment this requirement can be relaxed to require that the relative serialization order of Workflow Transactions should be the same only at those nodes where they conflict. This would lead to a weaker notion of serializability; called WT-serializability, which will be used as our correctness criterion for concurrent execution of Workflow Transactions. We define `conflict` among workflow transactions if they execute at the same (local) site, and they are not commutative. The conflict relation is transitive, and therefore determines a set of equivalence classes, which can be named as conflict classes. In our workflow environment they are used to determine the granularity of locking. In order to define workflow transaction serializability; WT-serializability, let us consider two workflow flexible transactions WT_α and WT_β, and conflict classes, i and j. A global schedule is WT-serializable if for any subtransactions ST_i^α and $ST_j^\alpha \in WT_\alpha$, and ST_i^β and $ST_j^\beta \in WT_\beta$ such that conflict (ST_i^α, ST_i^β) and conflict (ST_j^α, ST_j^β), $ST_i^\alpha \prec ST_i^\beta \Rightarrow ST_j^\alpha \prec ST_j^\beta$, at all sites they conflict. In our workflow environment the $\prec$ relationship is defined in terms of local serializability. WT-serializability establishes a partial order among all workflow flexible transactions. The submission order at each system, can be used to determine the execution and, consequently, the serialization order at each site. Therefore, the concurrency control mechanism of the local system will assure that the transactions that are submitted to the local system will be executed correctly with respect to the local concurrency control. As a result, the lock held by a subtransation can be released as soon as the subtransaction completes its submission phase. Therefore, we will have several transactions that are executing concurrently at each local site.

5 A Formal Approach to Support Workflow Security

An MLS distributed workflow management system should support functionality equivalent to a single-level workflow management system from the perspective of MLS distributed workflow users who design, implement and utilize multilevel secure distributed workflows.

A number of models for secure workflow have been proposed. These models differ in many respects. Despite heavy interest in building model of secure workflow management systems, there is no clear understanding regarding what a multilevel secure data model exactly is.

5.1 A Logic – Based Semantics for Multilevel Secure Workflow

In a multilevel secure workflow database management system users cleared to different security levels access and share a database consisting of data items at different sensitivity levels.

As a part of our research work, we introduce a belief-based semantics for multilevel secure workflow databases that supports the notion of declarative belief and belief reasoning in multilevel security scheme (MLS) in a meaningful way. We strive to develop a practical logical characterization of MLS workflow databses for the first time using the inherently difficult concept of non-monotonic reasoning.

Recent research shows that users in the MLS workflow model have a ambiguous view and confusing belief of data [12]. Multilevel security implements the policy of mandatory protection defined in [13] and interpreted for computerized systems by Bell and LaPadula [14]. In this research paper we assume the representation and execution of MLS rules obey the Bell-LaPadula "no read up, no write down" principles. Many multilevel data models have been proposed in the literature, just to mention a few: SeaView [15, 16]; also models proposed by Sandhu-Jajodia [18, 19]; and by Smith-Winslett [20] and many others. Some of these models has its strong points (e.g., the belief-based semantics of the Smith-Winslett model, etc.). However, we argue that most of these proposals are not completely satisfactory, in particular, if the workflow database may be polyinstantiated.

5.2 Multilevel Workflow Database

The majority of proposals for multilevel workflow secure relational (MLS) databases have utilized various syntactic integrity properties to control problems that arise under very strict security, such as polyinstantiation and proliferation of tuples resulting from updates, with only some partial success. We propose modal logic as a natural vehicle for reasoning about security. Because much security is dependent on the concept of what a subject knows, logic allows us to reason about knowledge, one of the fundamental concept of computer security.

We are interested in our research in workflow databases which enforce the multilevel security policy. Lets designate by *Level* a finite set of security levels. The set *Level* is assumed to be a lattice associated with a partial order relation denoted by $<$. This directly implies that, the *least upper bound* and *greatest lower bound* are determined. To describe that, we shall employ two functions *lub* and *glb*. Assuming that l_1 and l_2 are two security levels, then $lub\,(l_1,\, l_2)$ and $glb\,(l_1,\, l_2)$ are respectively the upper bound and greatest lower bound of l_1 and l_2. There are also two distinctive levels, the one which is lower than all other levels, designated by $\perp$ and the other level which is higher than all other levels, designated by $\top$. We view the global multilevel database as a set of partitions, where each partition accomodates single–level database associated with one particular security level. We can formally represent this as follows. A multilevel database DB is represented by a set of databases $\{DB_i,\, i \in Level\}$. Every DB_i is a partition containing a finite set of propositional formulae whose classifications are equal to i and which are satisfiable but not necessarily complete. We assume that the integrity constraints are classified at level $\perp$ because there is a single set of integrity constraints which is common to every single–level database DB_i, $i \in Level$. We wish to remove this restriction, therefore we have to consider that we partition the global set of integrity constraints into subsets I_i associated with each single–level database DB_i. For example, let us assume that the following integrity constraint i_1 is stored at the unclassified level:

– $\forall_x, \forall_y, Emp(x) \wedge Earn(x,y) \rightarrow y \leq 80,000$

i.e. an employee must not earn more than $\$80,000$.

However, let us assume that there are employees who can earn up to $\$99,000$ but this data must be kept secret. Inductively, we can proclaim the following integrity constraint i_2 at the secret level:

– $\forall_x, \forall_y, Emp(x) \wedge Earn(x,y) \rightarrow y \leq 99,000$

However, two different sets of integrity constraints I_i and I_j may be conflicting, i.e. $I_i^* \cap I_j^* = \emptyset$, therefore we might suggest using so called [23] the trusted approach. We need to observe that data stored in each single–level workflow database generally correspond to a partial view of the universe by users at the corresponding security level. This is induced from our assumption that each single–level workflow database DB_l only contains data classified at level l. Therefore, in the trusted approach, the view at a given level l is obtained by merging the single–level workflow database at level l with all the lower single–level workflow databases. For example, if a workflow database at level l_{k-1} is consistent with a workflow database at level l_k, then $DB_{l_{k-1}}$ can completely flow to level l_k — as in the additive approach [23]. Lets describe, $View\_at\_level\_l$ as the view of the multilevel workflow database for users at level l. Therefore, we can use the trusted approach to derive the set of integrity constraints $Integrity\_at\_level\_l_k$ which apply to the security level l_k:

- $(Integrity\_at\_level\_l_1)^* = I_{l_1}^*$
- $(Integrity\_at\_level\_l_k)^* =$
  $I_{l_k}^* \triangleright (Integrity\_at\_level\_l_{k-1})^*$

To be realistic, we shall assume that the global workflow multilevel database may be polyinstantiated. We define this as follows: a workflow multilevel database DB is polyinstantiated if and anly if there are two security levels i and j such that $DB_i^* \cap DB_j^* = \emptyset$.

Formally, a multilevel relation consists of two parts: scheme and instances, defined below.

Definition 3. <u>Relation Scheme</u> *Let $A_1, \dots, A_n$ be data attribute names over domain D_i, each C_i is a* classification attribute *for A_i and TC is the* tuple-class attribute. *The domain of C_i is specified by a range $[L_i, H_i]$ which defined a sublattice of access classes ranging from L_i to H_i. Let the domain of TC be the range $[lub^1\{L_i : i = 1, \dots, n\}, lub\{H_i : i = 1, \dots, n\}]$.*

Definition 4. <u>Relation Instances</u> *Let $R(A_1, C_1, A_2, C_2, \dots, A_n, C_n, TC)$ be a multilevel relation scheme. This collection of state-dependent* relation instances *one for each access class c in the given lattice is designated by R_c. Then each instance of a multilevel relation is a set of distinct and ordered tuples of the form $(a_1, c_1, a_2, c_2, \dots, a_n, c_n, t_c)$ where each $a_i \in D_i$ or $a_i = null$, and $t_c = lub\{c_i : i = 1, \dots, n\}$. If $a_i \neq \perp$ (null value) then $c_i \in [L_i, H_i]$. We also require that c_i be defined even if a_i is null - a classification attribute cannot be null or more formally, $c_i \neq \perp$ for $\forall a_i$.*

Similarly to classical relations, multilevel workflow relations are required to satisfy several integrity properties. Since multilevel workflow relations have different instances at different access classes, the definition of keys becomes unclear because a relation instance is now a collection of sets of tuples rather than a single set of tuples.

[1] Least upper bound

5.3 The Necessity for Semantics in Secure Workflow Databases

The problem of polyinstantiation arises because of different views of a single entity in the real world at different security levels by two subjects. Also the above problem generally occurs through the avoidance of a covert channel. If for example a user inserts a relation instance—tuple with key K_1, a user from a lower security level cannot be prevented from inserting a different tuple with key K_1 later on, as rejecting the later insertion would open a covert channel. As a direct result of this operation, MLS workflow relations can contain multiple tuples with the same key value — polyinstantiated tuples. This problem has been indicated in some previous models by means of syntactic integrity properties, which control the extent and nature of polyinstantiation—e.g., Jajodia and Sandhu [18, 19] and Jukic and Vrbsky [12].

Our contention is that both these models of asserting user beliefs about security are incomplete and somewhat stringent.

The Jukic-Vrbsky model is too restrictive and has only fixed interpretations. On the other hnd, Jajodia-Sandhu model is too basic where users are left to discover the truth. Users in these frameworks really do not have any reasoning capabilities as the interpretations are already given.

The paucity of attempts aimed at developing a logical characterization for MLS models evidences that MLS workflow deductive databases are realy at their embryonic state. While there were proposals such as [17] that addressed the general issue of authorization in a deductive framework, only Cuppens addressed the issue of querying MLS deductive databases [21].

We believe a middle ground is warranted where the user is given the choice to reason and theorize about the beliefs of others and decide how he wants to believe information visible to him. To support that approach, we assert that users should be given linguistic tools to view data as well as to construct meaning of the visible data. In such environment, the user may take a firm view of the data and insist that whatever is created at his security level only are correct and believable data. Thus lower level data are of no value.

5.4 Inference Control Theorems of MLS Workflow Database

We argue that any proposed model of MLS workflow database, under either discretionary or mandatory security, should incorporate at least the following elemants:

- A formally defined model of the MLS including all the security propeties that databases under this model will possess.
- Classification of any piece of information at any given classification level, should be enforced by powerful inference control rules
- A formal definition—semantics for databases under the proposed model, which can represent the beliefs about the state of the world held by the users at a chosen security level

The axiomatics of the language $\mathcal{L}$, which we consider is based on classical axiom schemas of first order logic with equality, augmented with appropriate axioms of our theory related to the multilevel workflow object–relational database. The subset of our language $\mathcal{L}$ is universally consistent with any language based on first order logic with equality [21]. What follows is a set of some axioms, which are relevant to a set of integrity constraints to be enforced by the multilevel workflow object-relational database:

– If a is an attribute of the object o then o is an object.
  $\forall_a \forall_o, OA(o,a) \rightarrow Object(o)$ **($\mathcal{A}$)**
– If m is a method of the class c then c is a class.
  $\forall_m \forall_c, Method(c,m) \rightarrow Class(c)$ **($\mathcal{B}$)**
– If a is an attribute of the class c then c is a class.
  $\forall_a \forall_c, CA(c,a) \rightarrow Class(c)$ **($\mathcal{C}$)**
– Any object attribute has a value.
  $\forall_a \forall_o, OA(o,a) \leftrightarrow \exists_v, Val(o,a,v)$ **($\mathcal{D}$)**
– The value of an object attribute is unique.
  $\forall_a \forall_o \forall_v \forall_{v'}, Val(o,a,v) \wedge Val(o,a,v') \rightarrow$
  $(v = v')$ **($\mathcal{E}$)**
– Any object is instance of at least one class.
  $\forall_o, Object(o) \rightarrow \exists c, Instance(o,c)$ **($\mathcal{F}$)**
– If o is an instance of c then o is an object and c is a class.
  $\forall_o \forall_c, Instance(o,c) \rightarrow Object(o)$
  $\wedge\, Class(c)$ **($\mathcal{G}$)**

In this section we also present the general constraints that should be enforced when classifying the workflow database content. Those constraints must be satisfied when classifying $Class - c$, containing objects o and attributes a at level l and $Class(c)$ at level $\hat{l}$. The language that we propose to represent the multilevel workflow database is an extension of the above defined language combined with the acclaimed Datalog language which is also augmented with the predicates of the Logic Data Language – LDL, resulting in powerful combination of the expressive power of a high-level, logic-based language (such as Prolog) with the nonnavigational style of relational query language, where the system is expected to devise an efficient execution strategy for it. For each predicate P of an arbitrary n used to represent the non-protected workflow database content, there is a predicate $\hat{P}$ of arity $(n+1)$ used to represent the MLS workflow database.

It is generally acknowledged that when classifying any piece of information at a given level, the following inference control rule must be active:

Definition 5. Rule - 1 *Let $x_1, \ldots, x_n$ be tuples of variables consecutively compatible with the arity of predicates $P_1, \ldots, P_n$. Let y be another tuple of variables compatible with the arity of Q. For simplicity we assume that each variable in tuple y appears in at least one of the tuples $x_1, \ldots, x_n$. therefore if:*

$\forall x_1, \ldots, \forall x_n, P_1(x_1) \wedge, \ldots, \wedge P_n(x_n) \rightarrow Q(y)$

is an axiom *of the non-secure object oriented database, then by following the similar approach as in [21], we can derive the following theorem in relation to the multilevel workflow object oriented database:*

$$\forall x_1 \ldots \forall x_n \forall l_1 \ldots \forall l_n \forall l, \hat{P}_1(x_1, l_1) \wedge \ldots$$
$$\wedge \hat{P}_n(x_n, l_n) \wedge \hat{Q}(y, l) \rightarrow l \leq lub(l_1, l_2, \ldots, l_n)$$

If the above rule 1 is not complied with, then a subject cleared at level lub($l_1, \ldots, l_n$) can access every $P_i(x_i)$ and use the above defined axiom to derive $Q(y)$. On the other hand if the classification of $Q(y)$ is not lower or equal to lub($l_1, \ldots, l_n$), then an inference passage enabling prohibited information to be disclosed is opend. By combining the above derived rule 1 with some more axiomatic of our language, we can derive more useful theorems[2].

For example by combining rule - 1 with axiom ($\mathcal{D}$), we can derive the following theorem:

– $\forall_a \forall_o \forall_v \forall_l \forall_l', Val'(o, a, v, l)$
$\wedge OA'(o, a, l') \rightarrow (l')$ **($\mathcal{H}$)**

Which can be described as follows: the sensitivity of "v is a value of the attribute a in object o" dominates the sensitivity of "a is an attribute of object o".

This model includes the possibility to hide some parts of the multilevel workflow database schema and to deal with rules in the database. Therefore, it may also be used as a formal semantics for multilevel workflow deductive databases.

When classifying any data of information at a given sensitivity level [22], the following control rule must be operational if one wants to protect the existence of secure information:

Definition 6. *Rule - 2 Let $x_1, \ldots, x_n$ and $y_1, \ldots, y_p$ be tuples of variables consecutively compatible with the arity of predicates $P_1, \ldots, P_n$ and $Q_1, \ldots, Q_p$ and let y be another tuple of variables. For simplicity we assume that each variable in tuple y appears in at least one of the tuples $y_1, \ldots, y_p$ and each variable in tuples $y_1, \ldots y_p$ appears in at least one of the tuples $x_1, \ldots x_n, y$. If:*

– $\forall x_1 \ldots \forall x_n, P_1(x_1) \wedge \ldots \wedge P_n(x_n) \rightarrow \exists y,$
$Q(y_1) \wedge \ldots \wedge Q(y_p)$ **($\mathcal{L}$)**

is an axiom of the non–protected workflow object–relational database, then, the following theorem can be derived related to the workflow multilevel object–relational database:

– $\forall x_1 \ldots \forall x_n \forall l_1 \ldots \forall l_n, P_1'(x_1, l_1)$
$\wedge \ldots \wedge P_n'(x_n, l_n) \rightarrow \exists y \exists l_1' \ldots \exists l_p', Q'(y_1, l_1')$
$\wedge \ldots \wedge Q'(y_p, l_p') \wedge lub(l_1', \ldots, l_p')$
$\leq lub(l_1, \ldots, l_n)$ **($\mathcal{M}$)**

[2] Detailed demonstration on how similar theorems can be established can be found in [23]

In case, when Rule - 2 is not satisfied, then a subject cleared at level $lub(l_1, \ldots, l_n)$ can access every $P_i(x_i)$ and use the axiom $(\mathcal{L})$ to derive the existence of the secure data (facts) $Q(y_1), \ldots, Q(y_p)$ some of them being classified higher than $lub(l_1, \ldots, l_n)$. As the result, effectively a signaling channel is created, which enables the existence of prohibited information within the workflow repository to be disclosed.

6 Conclusion

The impetus for our current research is the need to provide an adequate framework for belief reasoning about security in MLS distributed workflow management systems. The notions of correctness for transaction processing that are usually proposed for multiuser databases are not necessarily suitable when these databases are parts of a multilevel secure workflow systems. We believe, that the best approach will depend upon the characteristics of the multilevel secure workflow database and the applications. It is incumbent upon those who develop multilevel secure database systems to ensure that the user's needs and expectations are met to avoid misunderstandings about the system's functionality.

The insight developed in the current research serves as the basis for a complete logical synthesis of SecureLog, the language which we are currently developing as an orthogonal extension of the work contained in this paper in the direction of F-logic [24].

We choose to develop formal framework for secure distributed workflow architecture since we are actively involved in building a prototype of such a system. We strive to develop a practical logical characterization of MLS distributed workflow for the first time using the inherently difficult concept of non-monotonic reasoning. We also derived general theorem which must be active when classifying every item of information.

References

[1] V. Wietrzyk, Mehmet A. Orgun. A Foundation for High Performance Object Database Systems. In *Databases for the Millennium 2000 in proceedings of the 9th International Conference on Management of Data*, Hyderabad, December, 1998.

[2] A. Elmagarmid. Transaction Models for Advanced Database Applications. *Morgan-Kaufmann*, February 1992.

[3] G. Grefen, B. Pernici, G. Sanchez. Database Support for Workflow Mnagement - The WIDE Project. In *Kluwer Academic Publishers*, August 1999.

[4] M.Rusinkiewicz and A. Sheth. On transactional Workflows. *Bulletin of the Technical Committee on Data Engineering.* (June 1993).

[5] The Workflow Management Coalition Interoperability Abstract Specification. *The Workflow Management Coalition* , June 1996

[6] G. Alonso and D. Agrawal Advanced transaction Models in Workflow Contexts. *Procs. Int. Conf. on data Engineering.* 1996.

[7] D. Georgakopoulos, M. Hornick, and A. Sheth. An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. *Distributed and Parallel Databases*, 3(2):119-153, April 1999.

[8] A. K. Elmagarmid, Y. Leu, W. Litwin, and M. E. Rusinkiewicz. A Multidatabase Model for Interbase. In *Proc. of the 16th VLDB Conference*, August 1990.

[9] V. Gligor and R. Popescu-Zeletin. Transaction Maagement in Distributed Heterogeneous Database Management Systems. In *Information Systems*, 11(4), 1986.

[10] A. Zhang, M. Nodine, B. Bhargava, O. Bukhres Scheduling with Compensation in Multidatabase Systems. In *CSD-TR-93-063*, 11(4), October 1993.

[11] M. Ansari, M. Rusinkiewicz, L. Ness, A. Sheth Executing Multidatabase Systems. In *TM-TSV-019450*, July 1991.

[12] N. A. Jukic and S. V. Vrbsky. Asserting beliefs in mls relational models. In *Sigmod Record*, pages 30-35, Ithaca, NY, 1997. ACM Press.

[13] Department of Defense, National Computer Security Center. department of Defense Trusted Computer System Evaluation Criteria, 1985. *DOD 5200.28-STD.*

[14] D. E. Bell and L. J. LaPadula. Secure Computer Systems: Mathematical Foundations and Model. In *Technical Report*, MITRE Corporation, 1974.

[15] D. Denning, T. Lunt, R. Heckman, W. Shockley. A Multilevel relational data Model. In *Proceedings of the IEEE Symposium on research in Security and Privacy*, Oakland, April. IEEE, New York, 220-234.

[16] T. Lunt. Multilevel Security for Object-Oriented Databases. In D. L. Spooner and C. Landwehr, editors,*Database Security, III*, pages 199-209. North-Holland, Amsterdam, 1990.

[17] K. S. Candan, S. Jajodia and V. S. Subrahmanian. Secure Mediated Databases. In *Proceedings: ICDE*, pages 35–55, 1996.

[18] S. Jajodia and R. Sandhu. Toward a Multilevel Secure Data Model. In *Proceedings: ACM SIGMOD*, Denver, Colo., May. ACM, New York, 50-59.

[19] S. Jajodia and R. Sandhu. Polyinstantiation Integrity in Multilevel Relations. In *Proceedings: IEEE Symposium on Research in Security and Privacy*, Oakland, May. IEEE, New York, 104-115.

[20] M. Winslett and K. Smith Entity Modeling in the MLS Relational Model. In *Proceedings of the 18th International Conference on VLDB.* VLDB Endowment, 199-210.

[21] F. Cuppens Querying a Multilevel Database: a logical Analysis. In *Proceedings of the 22nd VLDB Conference.* VLDB Endowment, 1996.

[22] N. Boulahia-Cuppens, F. Cuppens, A. Gabillon, and K. Yazdanian Decomposition of Multilevel Objects in an Object-Oriented Database . In *Proceedings of the European Symposium on research in computer security.* Brighton, UK, 1994. Springer Verlag.

[23] A. Gabillon *Sécurité Multi-Niveaux dans les Bases de Données à Objects.* ENSAE, 1995.

[24] M. Kifer, G. Lausen, and J. Wu Logical Foundations for Object-Oriented and Frame-Based Languages. *Journal of the Association of Computing Machinery.* 42(3):741:843, July 1995.

Condition-Driven Integration of Security Services
Keynote Lecture

Clifford Neuman

Information Sciences Institute
University of Southern California, USA

Abstract. Deploying security services is hard. Security services are more readily integrated when they can be added at a single point in a network or at a single layer in the protocol stack. Most of today's widely deployed security tools are deployed in this manner. Unfortunately this kind of deployment significantly limits the kinds of security policies that can be enforced.

The end-goal of security is to control access to information. Many applications require that access be controlled to pieces of information that are only delineated at the application layer. Enforcement of these policies requires application cognizance of security, and today this means that applications and application protocols must be modified.

This talk advocates extending authorization policy enforcement mechanisms with a means for integrating security services. A simple API for authorization will be described that allows application developers to focus on only the aspect of security that matters to them - whether access should be granted. This allows security service policies (i.e. which security mechanisms are to be used for authentication, payment, audit, etc.) to be enforced through the API without specific knowledge or understanding by the application programmer. As new security services become available, this also allows the new services to be integrated by changing policy, rather than by rewriting the application.

Dr. Neuman will additionally suggest that the policies themselves adapt to perceived network threat conditions, possibly affected by the receipt of audit data at other processes. The use of such policies can assist in detecting and responding to intrusion and misuse and lead to more efficient utilization of all security services.

SKETHIC: Secure Kernel Extension against Trojan Horses with Information-Carrying Codes[⋆]

Eun-Sun Cho[1], Sunho Hong[1], Sechang Oh[1], Hong-Jin Yeh[1], Manpyo Hong[1], Cheol-Won Lee[2], Hyundong Park[2], and Chun-Sik Park[2]

[1] Graduate School of Information and Communication
Ajou University
San 5 Wonchon-dong Paldal-gu
Suwon 442-749, Korea
{eschough,sunhong,sechang,hjyeh,mphong}@ajou.ac.kr
[2] National Security Research Institute (NSRI)
161 Kajong-dong, Yusong-gu
Taejon 305-350, Korea
{cheolee,hdpark,cspark}@etri.re.kr

Abstract. Trojan-horses are hard to detect since they pretend normal programs [14]. This paper proposes 'SKETHIC (Secure Kernel Extension against Trojan Horses with Information-carrying Codes)', an anti-Trojan method based on resource access information attached to codes. This information serves as criteria for users' decision on installation of programs and forms access control policies for the runtime monitoring system. Compared to the previous approaches, SKETHIC introduces a way of reducing the users' decision-making overhead. To show clearly how it keeps a host secure from Trojans, we describe the mechanism in a formal way.

1 Introduction

A 'Trojan horse' program, or a 'Trojan' is a program that pretends to be a normal code but does something unwanted, like stealing passwords and destroying files [14]. One can have Trojan-horses usually through the Internet and E-mails, and due to hackers' intrusion. It is easy for a dishonest developer to deceive users to accept a Trojan with a reasonable name and documents.

This paper introduces a new anti-Trojan mechanism called 'SKETHIC (Secure Kernel Extension against Trojan horses with Information-carrying Codes)'. To detect unknown Trojans, SKETHIC focuses on the gap between users' expectations for a code and its actual behaviors. However, in contrast to existing anti-Trojan monitoring tools, SKETHIC allows users free from describing expected access rights necessary for an unacquainted code. Instead, they just examine the information attached to the code, called a resource access list, to decide whether

⋆ This work is supported by Brain Korea 21 project and by National Security Research Institute (NSRI).

to accept it or not. Execution is permitted only for the code looking honest. During the execution of the code, the monitoring system is watching the process, to check if it follows its resource access list. When the process tries to access a resource beyond the list, appropriate responses will be given.

The rest of the paper is organized as follows. In section 2, we review the previous anti-Trojan approaches, and we show an abstract view of SKETHIC in section 3. In section 4, the mechanism used in SKETHIC is described in a formal way. Also we show how it keeps a host secure from Trojan horses. In section 5, the proposed mechanism is compared with other approaches and discussed in detail. In section 6, we give a conclusion and future works.

2 Previous Anti-Trojan Approaches

Conventional anti-Trojan tools are classified into three categories – static code scanners, runtime monitoring systems and integrity checkers.

In the *static scanning approach*, a code is deemed to be a Trojan if it has the features of a known Trojan horse. For example, some tools scan suspicious codes to find out any signature of the known Trojan horse [5,13]. With its tight dependence on information about the known Trojans, this approach does not seem to be promising in the current situation where lots of malicious codes appear.

In *runtime monitoring systems*, which SKETHIC also follows, a code is executed in an environment with confined resources, called a 'sandbox' [1,2,3,7,12]. Similarly to the mechanism for Java applet security [11], a monitoring tool audits and controls the processes based on a policy. However, sandboxes defined by policy specification need to be fine-grained enough to cover the various kinds of Trojans with different properties [1]. This induces the access control models for the anti-Trojan policies to be based on concrete behaviors of the codes, for instance, the lists of allowed system calls [2,7] and state transition machines [3,12]. The main advantage of this approach is the ability to deal with unknown Trojans. However, because of the complexity and the number of codes on a host, it would be hard for users to develop suitable behavioral description especially for unacquainted codes [1].

Finally *integrity checkers* are helpful to detect the Trojan horses generated by modification of normal programs. They let users know whether an important code has been changed or not, by comparing the actual integrity data on the code with the original value kept in a database [4,6].

3 The Suggested Approach

As mentioned in the previous section, SKETHIC adopts the runtime monitoring system approach, which enables to detect unknown Trojan. However, not to burden users with describing the access rights for unacquainted codes, code developers are supposed to attach resource access information to their codes. This requires cooperation of the three subjects – developers, users and the reference monitor. Things that each of them has to do in SKETHIC are listed as follows.

- *A Developer:* He/she distributes his/her code together with the information on possible resource accesses by the code, called the *'resource access list'*. A program is defined as $p = \langle m, c, l \rangle$, where $m \in M$ is the identifier of the program such as a name, $c \in C$ is the code, and $l \in L$ is the resource access list.
- *A User and a System Administrator:* He/she decides on the acceptance of a given program $\langle m, c, l \rangle$, by examining the resource access list l. If l contains suspicious operations that do not meet user's expectation for the code, the program is deemed a Trojan and simply discarded. Otherwise, the program is accepted to be executable.
- *The Monitoring System:* During the execution of the code c, all the attempts to access a resource are monitored. If any attempt to access beyond the list l is detected, appropriate responses will be given such as terminating the process, removing the program from the system and notifying to users what is happened. Note that it is not the user, but the developer who described the list l the policy for monitoring the process behaviors.

The proposed mechanism of SKETHIC is depicted in Figure 1. The developer distributes a program accompanied with a resource access list. Based on this list, the user makes decision whether to install the program. During the execution, the monitoring system intercepts the system calls of the process asking for system resources. As long as the request follows the resource access list, the monitoring system considers it safe and enables the operation to proceed.

One of the main advantages of the proposed system is the ability to detect unknown Trojans, without users' burden of developing access policies. Let us consider a malicious code like navidad.exe[8]. Users would not accept the code if it is delivered with the resource access list implying file destruction. With a legitimate but dishonest list, however, the users would be easily deceived. But at runtime, the monitoring system would detect the pretence during the execution.

4 Formal Description

Formal description is useful for viewing the advantages and the limitations of a mechanism [14]. In this section, we start by clarifying the terms *'Trojans'* and *'safety from Trojans'*, and then describe the proposed mechanism. Finally, we show how SKETHIC keeps the host safe from Trojans.

4.1 Definition of the Problem

Let U and R denote the set of users and the set of the states of the system resource, respectively. O represents the set of operations, and Q means the set of *'operation execution sequences'*, or *'execution sequences'*, indicating possible

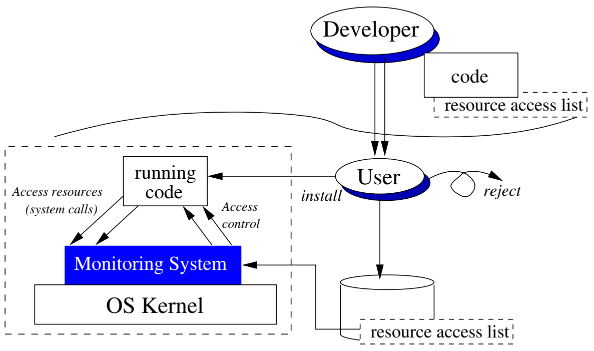


Fig. 1. Anti-Trojan Mechanism in SKETHIC

sequences of operations performed while a program is running. Note that a program may have more than one execution sequence. For the programs without termination, the maximum length of the execution sequence is assumed to be infinite.

Definition 1. *An operation execution sequence, or execution sequence* $q = o_1;\ \ldots\ ;o_n \in Q$ *s.t.* $o_i \in O$ *and* $0 < n \leq \infty$ *is defined as an ordered list of operations performed in sequence while a program is running.*

A *code* is defined the set of all the possible execution sequences of a program. For the endless programs, the maximum number of execution sequences is infinite as above. We use the term *'code'* and *'program'* mixed in Section 4.1 where the meanings of them make no big differences.

Definition 2. *A code* c *is* $\{q_1, q_2, \ldots, q_n\} \in C$ *such that* $q_i \in Q$ *and* $0 < n \leq \infty$.

The function *operations* : $C \mapsto \wp(O)$ maps each code $c$to the set of all the operations possibly performed during the execution of c. That is, $operations(c) = \{o \mid \exists q \in c \ s.t.\ o \in_L q\ \}$, where '$\in_L$' is used as the list-inclusion symbol. A *system* s is defined as $\langle\{c_1, \ldots, c_n\}, r\rangle \in S$ such that $c_1, \ldots, c_n \in C$ and $r \in R$, that is, the pair of a set of programs and a state of system resources.

All operations in a code fall into two groups: the ones making effects on the system and the remains. What an '*effect*' here means changing the system state or stealing information from the host.

Definition 3. *An effective operation* $e \in E$ *is the operation which changes the system state, or steals information from the host.*

In this paper, the only interesting kinds of operations are those that possibly influence security. We consider that effective operations and security holes as such operations, and call them *'problematic'*.

Definition 4. *A problematic operation is an effective operation or a security hole.*

Further classification of operations is given by the following functions. Note that they take the user u as well as the code c as their inputs, since users' expectation is crucial to the decision on Trojans.

Definition 5. *Function overt* : $U \times C \mapsto \wp(O)$ *maps a user* u *and a code* c *to the set of operations that belong to operations*(c) *and* u *also expects as such.*

Definition 6. *Function hcovert* : $U \times C \mapsto \wp(O)$ *maps a user* u *and a code* c *to the set of problematic operations that belong to operations*(c) *whereas* u *does not expect them for* c*. (hcovert means a harmful covert operation) .*

Definition 7. *Function hlcovert* : $U \times C \mapsto \wp(O)$ *maps a user* u *and a code* c *to the set of non-problematic operations that belong to operations*(c) *whereas* u *does not expect them for* c*. (hlcovert means a harmless covert operation) .*

*hcover*t and *hlcovert* are collectively called '*covert*', for the operations performed beyond u's expectation. Note that *overt* is not divided into somethings like '*hovert*' and '*hlovert*'. We assume that all the operations that the user expects to be performed are harmless. Then a Trojan-horse is defined as the code c with harmful operations, $hcovert(u, c)$.

Definition 8. *Given a user* u*, a Trojan-horse or Trojan is the code* c *such that* $hcovert(u, c) \neq \emptyset$*.*

A 'safe' system from Trojans is the one free from Trojans. In addition, one can maintain a system safe from Trojans by disallowing new installation of Trojans. We can formalize these as below.

Definition 9. *The system* $s = \langle cs, r\rangle$ *is safe from Trojans when for every user* u*, there is no Trojan* t *in* cs*.*

Theorem 1. *For a non-Trojan* $c \notin cs$ *and an initial system* $s = \langle cs, r\rangle$ *safe from Trojan, the system* $\langle cs \cup \{c\}, r\rangle$ *is also safe from Trojans.*

Proof. *It is clear by Definition 8 and Definition 9.* □

4.2 Definition of SKETHIC

Here, we give a formal description of the proposed anti-Trojan mechanism. A program in SKETHIC is defined as a triple of the identifier, the code and the resource access list as mentioned in the previous section. This also requires redefinition of the term *'Trojan horse'.*

Definition 10. *A program* p *is* $\langle m, c, l\rangle \in P = M \times C \times L$*, where* M *denotes the set of identifiers for programs.*

Definition 11. *A Trojan horse is a program* $p = \langle m, c, l\rangle$ *such that* $hcovert(u, c) \neq \emptyset$ *for some user* u.

An operation is *implied* by a resource access list if the list indicates the possibility of the execution of the operation. We use the symbol '$\Rightarrow$' for this implication.

Definition 12. *For* $l \in L$ *and* $o \in O$*, a resource access list* l *implies* o*, or* '$l \Rightarrow o$'*, if* l *indicates that the operation* o *might be executed at runtime. In addition, for* $l \in L$ *and* $os \in \wp(O)$*,* '$l \Rightarrow os$' *means that* $l \Rightarrow o$ *for every* $o \in os$*. We assume that the operations irrelevant to resource access are implied by every resource access list in* L*.*

It is desirable for a resource access list to imply all the operations in *operations*(c). In addition, a good resource access list should not imply incorrect and harmful behaviors. This motivates following definition.

Definition 13. *For a code* $c \in C$*, a correct resource access list* $l \in L$ *is defined as follows.*

- $l \Rightarrow operations(c)$ *and*
- *There is no* $o \in O$ *such that* $l \Rightarrow o$ *and* $o \in hcovert(u, c)$ *but* $o \notin operations(c)$*.*

We assume that every resource access list from a well-intended developer is always correct, whereas it does not hold for Trojans.

The proposed mechanism is described by states and operations. A SKETHIC state t_i is defined as $\langle u, d, k, s, f_{mc}, f_{ml}, x_1; \ldots; x_n\rangle \in T = U \times D \times K \times S \times F_{mc} \times F_{ml} \times List(X)$. Here, u, d, k and s mean a user, a developer, a monitoring system, and a system, respectively. $f_{mc} \in F_{mc} : M \mapsto C$ and $f_{ml} \in F_{ml} : M \mapsto L$ are functions mapping a program identifier to a related code and a related resource access list, respectively. $List(X)$ is the set of lists of SKETHIC operations in X.

A SKETHIC operation $x \in X = \{INSERT, DELETE, EXECUTE, Run\}$ changes the SKETHIC state, or performs code operations in O. $INSERT(u, p)$ means that a user u inserts a new program p into the system. $DELETE(u, m)$ represents removing the program identified by m from the host. $EXECUTE(u, m)$ denotes the execution of the code of m. $Run(k, o)$ performs a code operation o in the kernel k. Currently, we assume that a user can run only the codes installed by him/herself, but believe that the description in this paper is easily extended to general cases. The meaning of the SKETHIC operations is described as rules explaining their ways of changing the SKETHIC states, as follows;

$$\langle previous\ state\rangle \rhd \langle next\ state\rangle\ if\ \langle conditions\rangle$$

It reads that, under the given $\langle conditions\rangle$, $\langle previous\ state\rangle$ is changed into $\langle next\ state\rangle$ after executing the head of the SKETHIC operation list in $\langle\ previous\ \ state\rangle$.

SKETHIC prohibits installation of a new program $p = \langle m, c, l\rangle$ if l indicates any execution of operations in $hcovert(u, c)$ (see [INSERT III] below). Otherwise, p is installed with appropriated changes on cs, f_{mc} and f_{ml} of the state $\langle u, d, k, \langle cs, r\rangle, f_{mc}, f_{ml}, INSERT(u, p); Ops\rangle$ (see [INSERT I] and [INSERT II] below). The symbol '$f[y/x]$' means substituting or expanding the function f with the value y for x. *Ops*is a list of SKETHIC operations like $x_1; \ldots; x_n$. [INSERT II] is for the case that a pre-existing program already has the identifier m.

$$\begin{array}{lll}
\text{[INSERT I]} & \langle u,d,k, & \langle cs,r\rangle, f_{mc}, f_{ml}, INSERT(u,\langle m,c,l\rangle); Ops\rangle \\
& \rhd\langle u,d,k, & \langle cs \cup \{c\}, r\rangle, f_{mc}[c/m], f_{ml}[l/m], Ops\rangle \\
& & if\ \neg(l \Rightarrow hcovert(u,c)) \wedge \neg(\exists c^{'}\ s.t.\ \langle m,c^{'}\rangle \in f_{mc}) \\
\text{[INSERT II]} & \langle u,d,k, & s, fmc, fml, INSERT(u,\langle m,c,l\rangle); Ops\rangle \\
& \rhd\langle u,d,k, & \langle cs - \{c^{'}\} \cup \{c\}, r\rangle, f_{mc}[c/m], f_{ml}[l/m], Ops\rangle \\
& & if\ \neg(l \Rightarrow hcovert(u,c)) \wedge \exists c^{'}\ s.t.\ \langle m,c^{'}\rangle \in f_{mc} \\
\text{[INSERT III]} & \langle u,d,k, & s, f_{mc}, f_{ml}, INSERT(u,\langle m,c,l\rangle); Ops\rangle \\
& \rhd\langle u,d,k, & s, f_{mc}, f_{ml}, Ops\rangle \\
& & if\ \ l \Rightarrow hcovert(u,c)
\end{array}$$

The execution of a program will be completed only when if it exists in the system and each operation making up the code is implied by its resource access list (see [EXECUTE I] below). Since the execution sequence to be performed varies according to the resource states of the system, we use the function $selected(s, c)$ for the execution sequence of the code c under the system s. $Run(k, o)$ denotes performing each code operation in O by the kernel k. The meaning of Run, the semantics of k, is beyond this paper. We only assume that Run does not change a SKETHIC state. $EXECUTE(u, m)$ cannot be completed, either when the runtime monitoring system detects an operation not implied by the corresponding resource access list (see [EXECUTE II] below), or when it cannot find a program identified by m in the current system (see [EXECUTE III] below).

$$\begin{array}{llll}
\text{[EXECUTE I]} & \langle u,d,k, & s, & f_{mc}, f_{ml}, EXECUTE(u,m); Ops\rangle \\
& \rhd\langle u,d,k, & s, & f_{mc}, f_{ml}, Run(k,o_1); Run(k,on); Ops\rangle \\
& & if & f_{mc}(m) = c \wedge selected(s,c) = \langle o_1,,on\rangle \wedge \\
& & & f_{ml}(m) \Rightarrow o_i\ for\ all\ o_i\ s.t.\ 1 \le i \le n \\
\text{[EXECUTE II]} & \langle u,d,k, & s, & f_{mc}, f_{ml}, EXECUTE(u,m); Ops\rangle \\
& \rhd\langle u,d,k, & s, & f_{mc}, f_{ml}, Run(k,o_1); Run(k,o_{I-1}); \\
& & & DELETE(u,m); Ops\rangle \\
& & if & f_{mc}(m) = c \wedge selected(s,c) = \langle o_1,,o_n\rangle \wedge \\
& & & \exists o_i.1 \le i \le n\ s.t.\ \neg(f_{ml}(m) \Rightarrow o_i) \wedge \\
& & & I = min\{j : 1 \le j \le n \wedge \neg(f_{ml}(m) \Rightarrow o_j)\} \\
\text{[EXECUTE III]} & \langle u,d,k, & s, & f_{mc}, f_{ml}, EXECUTE(u,m); Ops\rangle \\
& \rhd\langle u,d,k, & s, & f_{mc}, f_{ml}, DELETE(u,m); Ops\rangle \\
& & if & \neg(\exists c\ s.t.\ f_{mc}(m) = c) \vee \neg(\exists l\ s.t.\ f_{ml}(m) = l)
\end{array}$$

For the moment that a program execution is failed, SKETHIC needs the operation $DELETE(u, m)$, which removes information on m from the SKETHIC state (see [DELETE I] below). If there is no code identified by m, the $DELETE(u, m)$ is simply ignored (see [DELETE II] below).

$$\begin{array}{lll}
\text{[DELETE I]} & \langle u, d, k, & \langle cs, r\rangle, f_{mc}, f_{ml}, DELETE(u, m); Ops\rangle \\
& \rhd\langle u, d, k, & \langle cs - \{c\}, r\rangle, f_{mc} - \{\langle m, c\rangle\}, f_{ml} - \{\langle m, l\rangle\}, Ops\rangle \\
& & if\ \exists c\ s.t.\ \langle m, c\rangle \in f_{mc} \wedge \exists l\ s.t.\ \langle m, l\rangle \in f_{ml} \\
\text{[DELETE II]} & \langle u, d, k, & s, f_{mc}, f_{ml}, DELETE(u, m); Ops\rangle \\
& \rhd\langle u, d, k, & s, f_{mc}, f_{ml}, Ops\rangle \\
& & if\ \neg(\exists c\ s.t.\ \langle m, c\rangle \in f_{mc}) \vee \neg(\exists l\ s.t.\ \langle m, l\rangle \in f_{ml})
\end{array}$$

The sequence of execution of SKETHIC operations is denoted by '$\rhd\rhd$', the reflexive and transitive state-transition based on '$\rhd$'. The inductive definition is provided as follows.

Definition 14. *For SKETHIC states t_1 and t_2, '$t1 \rhd \rhd t_2$' if any of followings is satisfied;*

(i) $t_1 = t_2$
(ii) $t_1 \rhd t_2$
(iii) $t_1 \rhd \rhd t3$ *and* $t3 \rhd t_2$, *for some* $t3 \in T$

4.3 Safety

In 4.1, a safe system from Trojans is defined as a Trojan-free one. According to this definition, a safe system needs to exclude Trojan horses before program installation, but it is too hard especially for unknown Trojans to check binary codes. Here, we give a less strict definition of safety from Trojans as follows;

Definition 15. *A state $\langle u, d, k, \langle cs, r\rangle, f_{mc}, f_{ml}, Ops\rangle$ is safe from Trojans when, if there exists a Trojan $\langle m, t, l\rangle$ in cs according to a user u, none of $o \in hcovert(u, t)$ can be executed.*

Now, let us show how SKETHIC keeps the host safe from Trojans. Before proceeding further, we show two useful properties of SKETHIC, in the following lemmas.

Lemma 1. *1 If a Trojan $\langle m, t, l\rangle$ exists in the SKETHIC state, then l does not imply any $o \in hcovert(u, t)$.*

Proof. Suppose that there is a Trojan $\langle m, t, l\rangle$ such that an operation $o \in hcovert(u, t)$ is implied by l. It must be installed by the operation '$INSERT(u, \langle m, t, l\rangle)$', since $INSERT$ is the only operation inserting a program into a SKETHIC state. Due to the existence of $o \in hcovert(u, t)$ such that $l \Rightarrow o$, $INSERT(u, \langle m, t, l\rangle)$ follows the rule [INSERT III], which, however, results in the failure of insertion. Thus, it is impossible for a SKETHIC state to have the Trojan $\langle m, t, l\rangle$ with an operation in $hcovert(u, t)$ implied by l. This completes the proof. □

Lemma 2. *Let $t_1 = \langle u, d, k, \langle cs, r\rangle, f_{mc}, f_{ml}, Ops\rangle$ and $t_2 = \langle u, d, k, \langle cs^{'}, r^{'}\rangle, f^{'}_{mc}, f^{'}_{ml}, Ops^{'}\rangle$. For all m, if $r = r^{'}$, $f_{mc}(m) = f^{'}_{mc}(m)$ and $f_{ml}(m) = f^{'}_{ml}(m)$, then the execution of $EXECUTE(u, m)$ at t_2 is exactly the same as that at t_1.*

Proof. *According to [EXECUTE I], [EXECUTE II] and [EXECUTE III], among the elements composing t_2, only $m, k, f^{'}_{mc}(m), f^{'}_{ml}(m)$, and $r^{'}$ make effects on the execution of $EXECUTE(u, m)$ at t_2. These elements are same in t_1 and in t_2, so is the execution of $EXECUTE(u, m)$.* □

Now we show that a safe system remains safe after insertion of a non-Trojan program. The proof is based on the fact that inserting a new program is almost independent of the existing programs.

Lemma 3. *Suppose $t_1 = \langle u, d, k, \langle cs, r\rangle, f_{mc}, f_{ml}, Ops\rangle$ is changed to $t_2 = \langle u, d, k, \langle cs^{'}, r^{'}\rangle, f^{'}_{mc}, f^{'}_{ml}, Ops^{'}\rangle$ by the execution of $INSERT(u, \langle m_1, c, l\rangle)$. Then for all m_2 in M such that $m_2 \neq m_1$, the execution of $EXECUTE(u, m_2)$ at t_2 is just the same as that at t_1.*

Proof. *According to the semantic rules, $INSERT(u, \langle m_1, c, l\rangle)$ changes only $cs, f_{mc}(m_1)$ and $f_{ml}(m_1)$ which results in that $r = r^{'}$, $f_{mc}(m_2) = f^{'}_{mc}(m_2)$ and $f_{ml}(m_2) = f^{'}_{ml}(m_2)$. Thus, by Lemma 2, the proof is completed.* □

Lemma 4. *(Safety after Insertion of a Non-Trojan) If the state $t_1 = \langle u, d, k, s, f_{mc}, f_{ml}, Ops\rangle$ is safe from Trojans, then so is t_2, the state right after a non-Trojan $p = \langle m, c, l\rangle$ is inserted at t_1.*

Proof. *When t_2 has no Trojans this lemma is clearly hold. Suppose that t_2 has a Trojan $p_t = \langle m_t, t, l_t\rangle$. We know that $p_t \neq p$, since p_t is a Trojan while p is not. That is, p_t is one of the programs that have already been in the system before the insertion of p. Since t_1 is safe from Trojan, execution of $o \in hcovert(u, t)$ will fail at t_1 by Definition 15. We want to show that execution of $o \in hcovert(u, t)$ will also fail at t_2. There are two possible cases;*

(i) If t_1 has no program identified by m : This implies that the identifier of p_t is not m, that is, $m_t \neq m$. We know that, by Lemma 3 the execution of $EXECUTE(u, m_t)$ at t_2 is equivalent to that at the safe state t_1. This means that the execution of $o \in hcovert(u, t)$ will fail at t_2.

(ii) If t_1 has a program identified by m : If $m_t = m$, $INSERT(u, p)$ follows the rule [INSERT II], which changes $s = \langle cs, r\rangle$ and f_{mc} into $\langle cs - \{t\} \cup \{c\}, r\rangle$ and $f_{mc}[c/m_t]$, respectively. Thus, there remains no way to perform any execution sequences of t at t_2, and moreover the operations in $hcovert(u, t)$. If $m_t \neq m$, the proof is similar to that of above (i).

By (i) and (ii), we have shown that execution of $o \in hcovert(u, t)$ will also fail at t_2, if t_2 has a Trojan $p_t = \langle m_t, t, l_t\rangle$. By Definition 15, this completes the proof. □

Even after insertion of a Trojan program, a safe system remains safe. Before proving this, we show that if a Trojan $\langle m, t, l\rangle$ is installed at a safe state, then none of the operations in $hcovert(u, t)$ can be performed.

Lemma 5. *Let* $t_1 = \langle u, d, k, s, f_{mc}, f_{ml}, Ops\rangle$ *be a safe state and* $t_2 = \langle u, d, k, s', f'_{mc}, f'_{ml}, Ops'\rangle$ *be the state right after a Trojan* $p = \langle m_t, t, l\rangle$ *is installed at* t_1. *Then,* $EXECUTE(u, m_t)$ *at* t_2 *cannot perform any operation* $o \in hcovert(u, t)$.

Proof. Clearly, Lemma 5 holds for the case that $selected(s', f'_{mc}(m_t)) = \langle o_1, , on\rangle$ *does not include any* $o_i \in hcovert(u, t)$. *Suppose that* $selected(s', f'_{mc}(m_t)) = \langle o_1, , o_n\rangle$ *has* $o_i \in hcovert(u, t)$. *We know that* $\neg(l \Rightarrow o_i)$ *by the Lemma 1, which leads* $EXECUTE(u, m_t)$ *at* t_2 *to follow the rule [EXECUTE II]. This implies that only* o_1 *and* o_{I-1} *are performed by Run, where* $I = min\{j : 1 \leq j \leq n \wedge \neg(l \Rightarrow o_j)\}$. *Because* $I \leq i$, o_i *cannot be performed at* t_2 *and this completes the proof.* □

Lemma 6. *(Safety after Insertion of a Trojan) If the state* $t_1 = \langle u, d, k, s, f_{mc}, f_{ml}, Ops\rangle$ *is safe from Trojans, then so is* t_2, *the state right after a Trojan* $p = \langle m, c, l\rangle$ *is installed at* t_1.

Proof. It is clearly hold if t_2 *has no Trojans. Let us suppose that* t_2 *has a Trojan* $p_t = \langle m_t, t, l_t\rangle$. *If* $p_t = p$, $o \in hcovert(u, t)$ *cannot be executed at* t_2, *by Lemma 5. If* $p_t \neq p$, *the proof is similar to that of Lemma 4.* □

Deletion of a program is almost independent of other programs. Based on this property, we show that a safe system remains safe after deletion of a program.

Lemma 7. *Suppose that* $t_1 = \langle u, d, k, \langle cs, r\rangle, f_{mc}, f_{ml}, Ops\rangle$ *is changed to* t_2 *by the execution of* $DELETE(u, m_1)$. *Then, for all* m_2 *in* M *such that* $m_2 \neq m_1$, *the execution of* $EXECUTE(u, m_2)$ *at* t_2 *is the same as that at* t_1.

Proof. According to the semantic rules, $DELETE(u, m_1)$ *changes only cs,* $f_{mc}(m_1)$ *and* $f_{ml}(m_1)$. *Thus, by Lemma 2, the proof is done.* □

Lemma 8. $(SafetyafterDeletion)$ *If the state* $t_1 = \langle u, d, k, s, f_{mc}, f_{ml}, Ops\rangle$ *is safe from Trojans, then so is* t_2, *the state right after a program p identified by* m *is deleted from* t_1.

Proof. It is clearly true if t_2 *has no Trojans. Suppose that* t_2 *has a Trojan* $p_t = \langle m_t, t, lt\rangle$.

(i) If t_1 *has no program identified by* m *: By the definition,* $DELETE(u, m)$ *does not change the state. Thus, by Lemma 2, the execution of* $o \in hcovert(u, t)$ *will fail at* t_2.

(ii) If t_1 *has a program identified by* m *: Since the information on* m *is deleted by* $DELETE(u, m)$, m_t *must not be* m. *We know that by Lemma 7, the execution of* $o \in hcovert(u, t)$ *will fail at* t_2.

By (i) and (ii), we have shown for each case that if t_2 *has a Trojan* $p_t = \langle m_t, t, l_t \rangle$*, the execution of* $o \in hcovert(u, t)$ *will fail at* t_2*. Thus, by Definition 15, the proof is done.* □

Now we are prepared to show that SKETHIC keeps the host safe from Trojans.

Theorem 2. *2 (Safety after an Operation) A safe SKETHIC state remains safe from Trojans, after any SKETHIC operation execution.*

Proof. We prove this Theorem for each SKETHIC operation. Suppose $t_1 = \langle u, d, k, \langle cs, r \rangle, f_{mc}, f_{ml}, Ops \rangle$ *is a safe state from Trojans and* t_2 *is the state right after a SKETHIC operations performed at* t_1*. By Lemma 6 and Lemma 8,* t_2 *is safe from Trojans after* $INSERT$ *or* $DELETE$ *at* t_1*. Since for any Trojan* $p_t = \langle m_t, t, lt \rangle$ $EXECUTE$ *and Run do not change* r, f_{mc} *or* f_{ml}*, their execution at* t_2 *is not different from those at* t_1 *either by Lemma 2. This implies that no operation* $o \in hcovert(u, t)$ *can be executed after* $EXECUTE$ *and after Run, which completes the proof by Definition 15.* □

Theorem 3. *(Safety Maintained by SKETHIC) If an initial state is safe from Trojans, SKETHIC maintains it always safe from Trojans.*

Proof. The proof is easily done by Theorem 2 and induction on the number of '$\triangleright$*'s involved from the initial state to a given state.* □

5 Comparisons and Discussions

It is a well-known approach to attach security information to codes. Developers' signatures and integrity values are commonly delivered along with mobile codes [11]. With a proof-carrying code (PCC) [9], correctness of accompanied proof is checked mechanically before running the codes [9]. In contrast, SKETHIC ensures the correctness of a resource access list by the runtime monitoring system.

Note that SKETHIC also examines a resource access list before execution, but for the agreement with the user's expectation for the code, not for its correctness. This concept is similar to the approaches in some mobile agent systems, which check the accompanied data before the execution of the codes to see if expected resource consumption agrees with the capability of the local host [10].

SKETHIC transfers the burden of generating policies from users to program developers. We believe that developers' overhead will be relatively small, since they know their own programs better than the users. In addition, automatic extraction of information from source codes is easier than that from binary codes delivered to hosts [5,15].

It is possible that a well-intended code is mistaken for a Trojan by SKETHIC with an incorrect resource access list. We expect that it can be avoided by support of intelligent tools helping the extraction of resource access information from codes. We also hope that the proposed mechanism encourages developers to use minimal system resources, making it easier to discriminate between honest programs and Trojans.

6 Summary and Future Works

We proposed SKETHIC (Secure Kernel Extension against Trojan Horses with Information-carrying Codes), an anti-Trojan mechanism based on the data called the 'resource access list', attached to the codes. Before the execution the user checks the resource access list to ensure that a code is not a Trojan. During the execution the monitoring system at runtime checks the correctness of the attached data.

One of the main advantages of SKETHIC is to reduce the users' burden of developing access policies for codes. This paper also formalizes the SKETHIC mechanism and shows how SKETHIC keeps a system safe from Trojans.

Currently, we are developing the prototype on Windows 2000, and studying on resource access list models. Especially, the proposed mechanism can employ rather complex models of access policies without difficulties, since access policies are not specified by users but by program developers who we believe have more intelligence. We are also interested in developing tools supporting automatic extraction of information from codes or helping users decide on acceptance of codes.

References

1. A. Acharya and M. Raje. MAPBox : Using parameterized behavior classes to confine applications. Technical Report TRCS99-15, Dept. of Computer Science University of California Santa Barbara, 1999.
2. EROS: The extremely reliable operating system. http://www.eros-os.org.
3. C. Ko, G. Fink, and K. Levitt. Automated detection of vulnerabilities in privileged programs by execution monitoring. In *Proc. of the 10th Annual Computer Security Applications Conference*, pages 134–144, Orlando FL, 1994.
4. Trojan horses,
http://www.ladysharrow.ndirect.co.uk/Maximum%20Security/trojans.htm, 2001.
5. R.W. Lo, K.N. Levitt, and R.A. Olsson. MCF: a Malicious Code Filter. *Computers and Security*, 14(6):541–566, 1995.
6. S. Mann and E. L. Mitchell. *Linux System Security : An Administrator's Guide to Open Source Security Tools.* Prentice Hall PTR, 2000.
7. T. Mitchem, R. Lu, and R. O'Brien. Using kernel hypervisors to secure applications. In *Proc. of the Annual Computer Security Application Conference (ACSAC97)*, 1997.
8. Navidad.exe, http://www.symantec.com/avcenter/vinfodb.html, 2000.
9. G.C. Necula and P. Lee. Safe kernel extensions without run-time checking. In *Proc. of the Second Symposium on Operating Systems Design and Implementation (OSDI'96)*, 1996.
10. R. De Nicola, G. Ferrari, and R. Pugliese. Types as specifications of access polices. In J. Vitek and C. Jensen, editors, *Secure Internet Programming: Security Issues for Distributed and Mobile Objects*, LNCS 1603, pages 117–146. Springer Verag, 1999.
11. S. Oaks. *Java Security.* O'Reilly, 1998.
12. F. Schneider. Enforceable security policies. Technical Report TR98-1664, Dept of Computer Science Cornell University, 1998.

13. Symantectm. http://www.norton.com.
14. H. Thimbleby, S. Anderson, and P. Cairns. A framework for modeling trojans and computer virus infection. *Computer Journal*, 41(7):444–458, 1999.
15. J. Viega, J.T. Bloch, T. Kohno, and G. McGraw. ITS4: A static vulnerability scanner for C and C++ code. In *Proc. of the 16th Annual Computer Security Applications Conference (ACSAC'00)*, 2000.

Secure and Private Distribution of Online Video and Some Related Cryptographic Issues

Feng Bao, Robert Deng, Peirong Feng, Yan Guo, Hongjun Wu

Kent Ridge Digital Labs
21 Heng Mui Keng Terrace, Singpore 119613
{baofeng, deng, pfeng, yguo, hongjun}@krdl.org.sg

Abstract. With the rapid growth of broadband infrastructure, it is thought that the bottleneck for video-on-demand service through Internet is being cleared. However, digital video content protection and consumers privacy protection emerge as new major obstacles. In this paper we propose an online video distribution system with strong content security and privacy protection. We mainly focus on the study of security and privacy problems related to the system. Besides presenting the new system, we intensively discuss some relevant cryptographic issues, such as content protection, private information retrieval, super-speed encryption/decryption for video, and PKC with fast decryption etc. The paper can be viewed as one that proposes practical solutions to real life problems, as well as one that presents applied cryptography research.

1 Introduction

Television has been elected as one of the greatest inventions in the last century. Public demand on video-based communication, entertainment and education has been the driving force for many technologies, such as broadband network and video compression. Nowadays, people are no longer satisfied with the fixed TV programs. They want to watch what they love to watch, and pay for that, i.e. personalized video service like the services provided in restaurants. To meet this need, Video-on-Demand (VoD) has been studied for many years. [Minoli] is a good reference for the academic and industrial effort for VoD technologies. Researchers have been focusing on how to stream the video to an online Internet consumer without dropping of critical frames. SMIL is ironed out to serve as a standard for synchronized integration of multimedia streams by W3C. It is claimed in [Jai99] that industries have even moved far ahead of academies in this field to step into the new frontier.

With the rapid growth of broadband infrastructure, it is thought that the bottleneck for video-on-demand service through Internet is cleared. Digital content security emerges as a new challenge. Up to now, online video consumers (OVCs) have very limited choice of online video contents, as video content providers (VCPs) hesitate to put their contents in digital format in the network. VCPs are not comfortable with the level of content security provided by the current technology [GMDS98]. On the other hand, online consumers also concern about their privacy being disclosed.

In this paper, we propose an online video distribution scheme that protects VCP s video contents and the consumer s privacy simultaneously. The content protection in our scheme is based on the public key cryptography implemented in a tamper-resistant hardware, which is not a new idea. But we focus more on security discussion and analysis. We also study some cryptographic issues arising from the scheme. The privacy protection is based on a simple PIR(private information retrieval) scheme.

The organization of the paper is as follows. In Section 2 we describe our online video distribution system. In Section 3 we discuss the advantages of using public key cryptography in tamper-resistant hardware for content protection. In Section 4 we study the privacy protection issue in our online video system. The system features are displayed in Section 5. In Section 6, we propose a general method to construct the symmetric key ciphers that have super-speed for video encryption. In Section 7 we propose a public key cryptosystem with fast decryption, which is motivated by implementing decryption in a hardware device with cheap processors. We present the design and analysis of two concrete super-high speed ciphers for video encryption in the Appendix, which can be excluded from the paper.

2 System Description

2.1 Outline of the System

In our online video system there are four parties.

VCP---Video Content Provider,
OVW---Online Video Warehouse,
OVC---Online Video Consumer,
THM---Tamper-resistant Hardware Manufacture

An OVW is an online storage service provider that may support several VCPs. A VCP encrypts its different videos by different secret keys and puts the encrypted videos at an OVW. An OVC can freely download the encrypted videos in his/her favor. The OVC can only watch the video after he/she pays the VCP for the secret key to decrypt the video.

However, the secret key should not be given to OVC plainly for content protection. That key should be given to OVC in the encrypted form and be decrypted in a tamper-resistant hardware device (produce by THM) as described in the following.

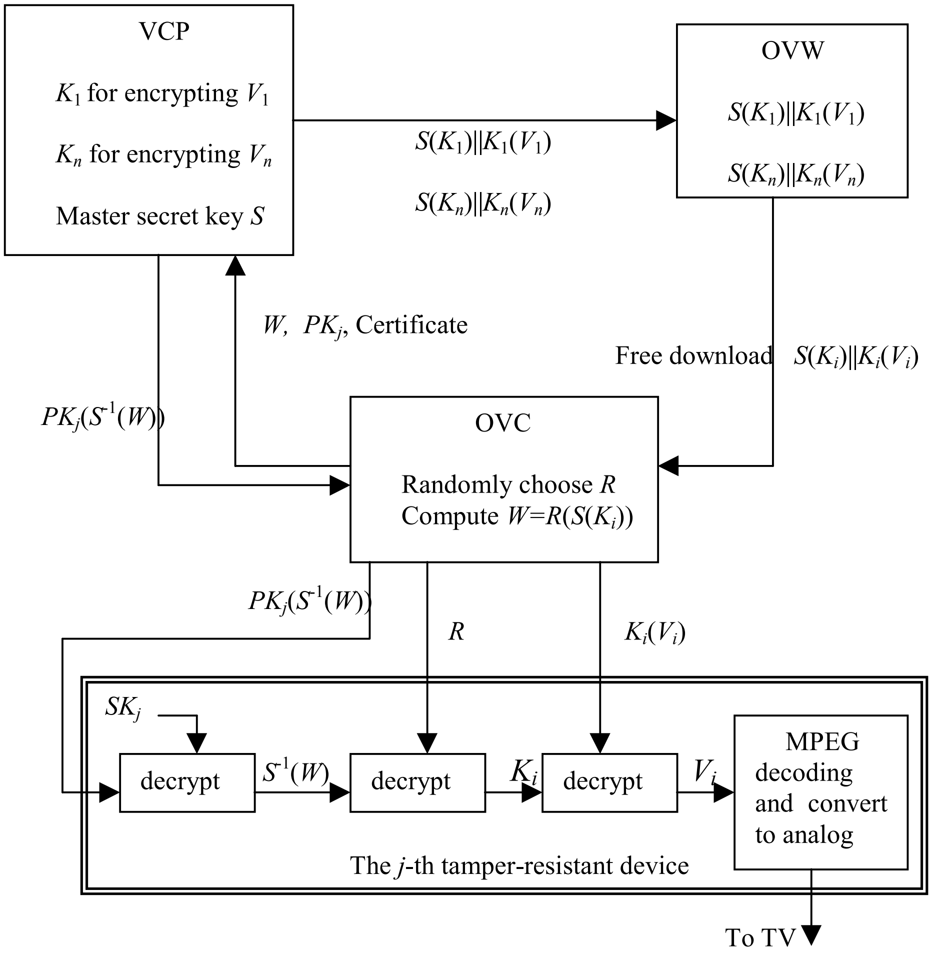
2.2 System Description in Detail

The system has three encryption algorithms:

1. Symmetric Key Cryptosystem I (SKC I)---SKC I is a fast cipher as studied in Section 6.
2. Symmetric Key Cryptosystem II (SKC II)---SKC II is a commutative cipher as studied in Section 4 and Appendix A.
3. Public Key Cryptosystem (PKC)---PKC can be any public key cryptosystem. A PKC with fast decryption as presented in Section 7 may be favored for this application.

System Description:

1. A VCP has n videos V_1, V_2, $,V_n$. The VCP chooses n secret keys K_1, K_2, $,K_n$ and encrypts V_1, V_2, $,V_n$ with SKC I, respectively. Denote the n ciphertexts by $K_1(V_1)$, $K_2(V_2)$, $, K_n(V_n)$.
2. The VCP also chooses a key S and encrypts K_1, K_2, $,K_n$ by S with SKC II. Denote the ciphertexts by $S(K_1)$, $S(K_2)$, $, S(K_n)$.
3. Suppose an OVC wants to watch V_i. He downloads $S(K_i)\|\ K_i(V_i)$ and chooses a key R for SKC II and encrypts $S(K_i)$. Denote the ciphertext by $W=R(S(K_i))$.
4. Decryption of W by key S is denoted by $S^{-1}(W)$.
5. PK_j/SK_j is a pair of public/private key generated by THM. SK_j is embedded into the j-th tamper-resistant hardware. PK_j is certified by THM and is given together with the certificate to the OVC who buys the hardware device.
6. When a VCP receives a PK_j, the VCP should check whether the PK_j is legal.



3 Content Protection

3.1 A Brief Review of Content Protection Technologies

Content protection is the key security issue for e-commerce of digital goods, no matter the transacted digital object is a picture, a video, an audio or a piece of news. It is commonly recognized that a digital content provider is hard to survive without certain means of protection. In online distribution of video, the content protection is the issue about how to prevent the illegal users (who do not pay) from watching a video. Content protection for digital goods is a very difficult problem from the technology angle. So far no fully satisfactory solution exists. Available technologies for content protection include follows.

Watermarking Technology

Watermarking technology has been considered to be a key technology for multimedia content protection. There have been so many research papers addressing watermarking technology in the past several years. Readers are referred to [CL97, CMY96, ZK95] and the references therein.

There are two sorts of watermarking. The first one is for ownership. The second one is for tracing illegal users. The technique of the second sort is also called fingerprinting in some references. The first sort of watermarking is to embed an identical watermark into every copy of the digital object. Hence, it cannot be used to distinguish who is the user who has distributed the illegal copy. The technology is to deter the large-scale resale. There are a lot of research publications in this area.

The second sort is to embed different watermarks into different copies. It can be used to trace the illegal users. But this sort of watermarking has certain difficulties. One is how to efficiently resist colluding attacks [BS95]. Another one is, as pointed out in [PS96], that there is actually no lawful basis for the content provider to sue the illegal user. This is because the provider himself possesses the watermarked digital object. Hence there is no way to distinguish who actually disclosed the copy. Asymmetric fingerprinting was proposed to solve this problem, see [PS99] and the references therein. However, it seems that the technique is not ready for practical use due to its complicated and interactive implementation.

Tamper-Resistant Software

This technology is to prevent the decryption party from accessing the decryption key in software. Combining with other techniques, the technology can be used to prevent making illegal copies. This is advantageous over watermarking technology at the point that watermarking is used to *catch* illegal copy while the tamper-resistant technology is used to *prevent* illegal copy. Tamper-resistant software, in principle, is to hide some secret information in a software program. It is based on anti-reverse-engineering. The current status of the technology is more like know-how and the technology is more studied within industry community than within academic community. There are quite a number of patents but rare publications in this area. It seems that there is no solid theoretical foundation for this technology.

Tamper-Resistant Hardware

Tamper-resistant hardware has been studied for many years. This technology has already been used in many realistic applications such as cable TV and DVD etc. In this paper, we take tamper-resistant hardware as our basis for content protection. The tamper-resistant hardware in our system contains a private key that is used to decrypt the ciphertext of the secret key of a video.

There have been various attacks against tamper-resistant hardware devices, such as fault-differential attack [BDL97, JQBD97], timing attack, differential power attack [Kocher], and probing attack [AK97, HPS99], etc. Researchers find that any information leakage in the procedure of the computation may lead to the secret key compromised. On the other hand, various counter measures have been proposed. Counter measures against fault-differential attack can be found in [BDL97]. Methods to resist differential power attack are presented in [Cor99, Kocher]. [WBYD00] presents some counter measures against probing attack. But in general, the attitude toward tamper-resistant hardware from academic is negative. This is because the fact that it might be hard to absolutely prevent the leakage of side-channel information [CKN01], which would cause key-compromising.

Industry, whereas, has the different view on tamper-resistant devices, which have been running well in the reality. One example is cable TV box, which is insecure from whatever angle in researchers eyes. But it does make good business. There is a big gap between academic and industry in the recognition of security. The former tends to consider absolute security based on complexity assumptions while the latter usually concerns more about relative security with respect to the costs. We do not believe that tamper-resistant devices could be relied upon as the security basis for military or government secrecy. But we think it should be qualified for small-valued business. In addition, it is commonly recognized that tamper-resistant hardware is much more reliable than tamper-resistant software.

3.2 The Content Protection Based on Public Key Cryptosystem

Why Use Public Key Cryptosystem The most important and essential discipline for a content protection system is that component-compromising must not cause the whole system crashing. If we only use symmetric key cryptosystem in the tamper-resistant hardware devices, we have two choices. First, we can install a master secret key into every tamper-resistant hardware device. This choice is apparently not secure since breaking one device may cause the master key compromised, and therefore, the whole system is broken. The second choice is to install different secret keys into different devices. In this case all the VCPs must know all the secret keys (otherwise they cannot do encryption). This is also dangerous since once a VCP is compromised, all other content providers are exposed beyond any protection. The whole system crashes. Using public key cryptosystem perfectly solves above problems. The system proposed in this paper meets the discipline that component-compromising does not cause the whole system crashing.

Protection of Private Keys The private key installed in each tamper-resistant hardware device is very important. The manufacturer of the hardware (THM) must be very cautious on these private keys. A suggestion is to destroy the keys once they are installed into the hardware devices. The manufacturer is a trusted party like the CA in PKI. Actually, the manufacturer is required to maintain a revocation list as done by a CA. Once a device is found to be broken, its serial number should be put into the list to prevent its use any more.

Tamper-Resistant Technology In this paper we do not discuss how to build up tamper-resistant hardware devices. There has been research on this technique for many years. What we want to emphasize here is that in our system the tamper-resistant technique can be focused on the private key. It is the critical clue. Once the private key is destroyed, the device is completely useless. So the guideline to build tamper-resistant property should be that *once the device is tampered or opened, the private key is automatically erased or modified***.** Protection techniques may include, for example, hiding a photoelectric cell inside the device, which is touched off (once the device is opened) to erase/change the private key. Another technique is a kind of careful wiring from inside so that the device is hard to be opened without breaking off the wire, which would also cause the private key erased. Of course, there must be multiple levels of protection.

Business Consideration In the proposed system, every OVC must buy a hardware device. This is the disadvantage of hardware solutions compared with software solutions. But from another angle, hardware solutions are not excluded here since a video has a comparatively high value. A DVD movie usually costs about 20 US dollars, while online video may cost much less as long as the content is perfectly protected. If an online video costs only one dollar for example, the attraction for a customer to buy a hardware device is considerably large. More specifically, such a tamper-resistant hardware device is not expensive since the processors to conduct decryption and D to A converting are not very expensive. Another choice is to build such hardware device into home appliances like VCD/DVD players. Then there is only a small additional cost while the player has a new function used together with the home PC. That is very alluring.

4 Privacy Protection

A VCP may provide a large number of videos with various categories. This is also the attractive point of online video. Therefore, privacy is another concerned issue. A customer may not like to let others know what video he/she is watching or is in favor. Such privacy should be guaranteed as long as the online video is a charged service. From another viewpoint, if two VCPs are providing same video at same price, the one who guarantees privacy is more competitive.To retrieve a message from a database without revealing which message is actually being retrieved has been theoretically studied under the term PIR (private information retrieval) [CGKS95, CG97, KO97, CMS99]. However, the computational costs of these solutions are very large due to their bit-by-bit processing manner. All those schemes need at least computation of $O(N)$ operations for retrieving only one bit, where N is the number of bits of the whole database. All the previous PIR schemes are aiming at reducing communication complexity. The scheme of [CMS99] can even achieve a communication complexity of poly(logN). However, those schemes can hardly be accepted for practical use. In this section we propose a simple and efficient scheme for privacy protection in the online video system. The scheme is not a PIR scheme from a strict viewpoint. We describe the scheme in the way as describing a PIR scheme for simplicity.

A Simple PIR Scheme

In Appendix A, we will present our scheme in detail and give cryptanalysis. Here only its principle is given. Let ***E*** be a symmetric-key encryption algorithm that has *commutative property*, i.e., for any pair of keys K_1, K_2 and for any message m, we have

$$\boldsymbol{E}(K_1, \boldsymbol{E}(K_2, m))=\boldsymbol{E}(K_2, \boldsymbol{E}(K_1, m))$$

Denote the decryption algorithm of ***E*** by ***D***. Suppose that the Database has n files denoted by M_1, M_2, , M_n (possibly with different lengths) and the User wants M_s. By the following scheme the User can get M_s without Database knowing what s is.

Database **User**

Randomly choose n keys K_1, , K_n
for a symmetric encryption, say , DES.
Encrypt M_i by K_i, C_i=DES(K_i, M_i)
Randomly choose a key r for ***E***,
Encrypt K_i by r, $h_i=\boldsymbol{E}(r, K_i)$

$$\xrightarrow{h_1||C_1,\ h_2||C_2,\ \ ,\ h_n||C_n}$$

Randomly choose key w for ***E***.
Compute $W=\boldsymbol{E}(w, h_s)$

$$\xleftarrow{W}$$

Compute $U=\boldsymbol{D}(r, W)$

$$\xrightarrow{U}$$

$K_s=\boldsymbol{D}(w, U)$
$M_s=\mathrm{DES}^{-1}(K_s, C_s)$

The Database has no way to know which message the User can get, no matter how maliciously Database performs. Meanwhile the User can only get one message by implementing the scheme once. In [BDF00], a concrete ***E*** and the security analysis of the protocol were presented.

It is easy to see that the above PIR scheme processes messages file-by-file. It does not get communication complexity reduced if it is regarded as a PIR scheme. But it fits our online video scheme very well because of the following reasons.

The customer s downloading can be anonymous. When the customer downloads the encrypted video from OVW, he need not show his personal information such as membership or credit card number etc. If the download is through some specific proxy, the customer's IP address can be hidden. Or if the download is through dial-up, the IP address changes every time. Some companies, such as www.zeroknowledge.com and www.anonymizer.com, provide service for anonymous download. On the other hand, the communication between the customer and the VCP cannot be anonymous since the VCP must know whom he is dealing with. When the VCP decrypts the secret key for the customer, the service is a charged service. Either the membership authentication or a payment is needed, which would disclose certain information about the customer.

5 System Features and Discussions

The system proposed in this paper has many good features.

1. The system is flexible. There may be multiple VCPs and OVWs. An OVW can support multiple VCPs. Each OVC can enjoy services from multiple VCPs with only one hardware device.
2. No VCP holds any secret of any OVC (the secret of his hardware device); therefore, if a VCP is compromised or becomes malicious, the other VCPs are not effected.
3. The OVCs' privacy is guaranteed no matter how malicious a VCP performs. At most a VCP can let an OVC receive no service, but can never get to know which video the OVC is trying to watch. On the other hand, VCP can still get statistical data on frequency of videos being downloaded from OVW (this seems necessary for business).
4. Low requirement on download speed. Unlike streaming VoD, where the network speed must be faster than the speed of video playing, in this system an OVC can download the encrypted video at the speed slower than that of video playing. The download can also be in off-peak hours.
5. Cheap computations. The system exploits some cryptosystems. But the crypto operations required in the system are light.

Compared with DVD The DVD encryption scheme is not robust: all the videos are encrypted by one secret key (for each zone) and the secret key is stored in all DVD players. Disclosing the secret key causes the whole scheme cracked. Our video distribution scheme is designed to be robust. In our scheme, the private keys in different hardware devices are independent from each other. In case one hardware player is cracked, the other hardware devices are not affected. Even if the hacker makes the cracked key public, the damage would be limited: the VCPs just refuse to provide service to that device any more.

Payment Choice and Privacy There are two ways of payment for online video. The first one is like membership. An OVC can subscribe to a VCP and the VCP will always serve the OVC (there may be a limit on number of videos for the OVC per month). For this payment manner, the OVC's privacy is perfectly protected. The second payment way is pay-per-video. For this payment manner, the privacy can only be guaranteed among all the videos with the same price. In this case the system needs a slight modification. The master key S in Section 2.2 should be replaced with a set of master keys, each key for one group of videos with same price.

Flexible Distribution Means In reality there may be more means to distribute the encrypted videos. The VCPs encourage the distribution of the encrypted videos among video fans. Another possibility is by CD. The CD with huge storage capability is going to emerge in a few years. We believe that the storage media is much cheaper than the stored content. A CD containing many encrypted movies can be very cheap in the future.

6 Fast Symmetric Key Encryption Scheme

It is well known that symmetric key encryption schemes are much faster than PKC schemes. For example, DES can achieve speed 20-30 Mb/s on a 233 MHz Pentium II Processor [Dai]. That speed is sufficient for video play. However, the decryption of the video may be conducted on a resource-limited chip. In our system, the speed of the processor in the tamper-resistant hardware device may be much slower than a 233 MHz Pentium II Processor. So it is better if we have faster encryption algorithms. We show that there is a large room to increase the speed of a symmetric key encryption algorithm while maintaining its security as long as the encrypted file is very large.

It is widely believed that there is a tradeoff between the speed and security strength of a cipher. It is a big challenge to design a very strong cipher that has very fast speed. But if we consider the situation of encrypting large files, it is possible to design a cipher with both very fast speed and very strong security. The reason is that we can combine a very strong but slow cipher with a very fast but weaker cipher such that the combined cipher is very fast and very strong. The reason behind the construction is that a weaker cipher may be a strong one if it is used in the way that each key is used to encrypt a limited amount of messages only. Just looking at those powerful cryptanalysis techniques, such as differential attack [BS91] and linear attack [Mat93], large amount of chosen/known plaintext/ciphertext pairs are always the pre-condition.

In our scheme we combine fast stream ciphers with secure block ciphers.

Let ***SE*** denote a strong encryption algorithm, such as AES, and ***FE*** be a weaker but very fast encryption algorithm, such as some fast stream-cipher. Denote a plaintext by $M_1M_2M_3 \quad M_n$ where M_i is a block of size same as that of ***SE***. Let K be a key, the encryption can be done as follows

$$\begin{aligned}\text{Ciphertext} = {} & \boldsymbol{SE}(K, M_1)\|\boldsymbol{FE}(K_1, M_2M_3 \quad M_k)\| \\ & \boldsymbol{SE}(K, M_{k+1})\|\boldsymbol{FE}(K_2, M_{k+2}\, M_{k+3} \quad M_{2k})\| \\ & \qquad \| \\ & \boldsymbol{SE}(K, M_{tk+1})\|\boldsymbol{FE}(K_{t+1}, M_{tk+2}M_{tk+3} \quad M_n)\end{aligned}$$

where K_{i+1}=$\boldsymbol{SE}(K, \boldsymbol{SE}(K, M_{ik+1}))$ (i=0,1, ,t) are called segment keys. The k (segment size) is the value determined by ***FE*** such that ***FE*** is strong enough if one key is used to encrypt at most k blocks forever.

It is obvious that such combinations have speed advantage only for large files. If the plaintext consists of only a few blocks, the speed is close to that of ***SE***. But if the plaintext is large and the k is fairly large, the speed is close to that of ***FE***. In analogy this is like to construct a door with steel frame and plastic filling pieces such that the door is as light as a plastic door while as strong as a steel door. The k is like the size of the grid. The smaller it is, the more secure the scheme.

Dividing the video into segments is also needed for fast-preview. The video can be played from any segment. In Appendix, we show two concrete ciphers with speed 300 Mb/s, and 1,500 Mb/s on a 233 MHz Pentium II Processor.

We have seen some research papers, such as [MS95, QNT97, Tan96 etc] on increasing video encryption speed by exploiting the structures of MPEG. But none of them can compare with our solution. Ours is very much faster as long as the encrypted file is large.

7 PKC with Fast Decryption

In our system, a tamper-resistant hardware device contains a private key of a PKC (public key cryptosystem). The PKC decryption is conducted in the device. Although any PKC can be used in our system, a PKC with fast decryption is favored for lowering the cost of the device. It is well known that RSA can be made fast for encryption. But PKC with fast decryption has rarely been studied. In this section we make an effort to design a PKC with fast decryption. We propose a PKC that is much faster in decryption than RSA and at least ten times faster than MultiPrime, while the security strength is the same. The PKC proposed here is similar to Shamir s unbalanced RSA except that we have a small *d*. In RSA a small *d* is dangerous. We show that our scheme is immune to small *d* attack. To our knowledge, this is the first PKC design for fast decryption.

Algorithm Description

Private key: primes p, q (better p, q are safe primes) and an odd number d.

Public key: $n(n=pq)$, $e(ed\equiv 1 \bmod q-1)$.

Encryption: $c=m^e \bmod n$ where m $(0<m<q)$ is the plaintext, c is the ciphertext.

Decryption: $m=c^d \bmod q$

It is easy to verify that the decryption is correct. The scheme is different from RSA at the point there is an expansion from plaintext to ciphertext.

Fast Decryption

We take $|n|=1024$, $|q|=341$ and $|d|=120$. The decryption speed of this scheme is apparently much higher than 1024 bit RSA. But the most important issue is the security. It is dangerous to take small *d* in RSA. In our algorithm, however, a small *d* is conjectured to be safe.

Security Analysis

Small *d*. It is shown in [Wie90] that if *d* is small, say $|d|<|n|/4$, then the RSA scheme can be broken. The attack is very simple and beautiful:

> In number theory we have: if η/ξ is an approximation of a known number c within $1/\xi^2$, i.e., $|c-\eta/\xi|<1/(2\xi^2)$, then η and ξ can be efficiently computed out by continuous fraction. Since $ed\equiv 1 \bmod \varphi(n)$, we have $ed=k\varphi(n)+1$ for some k, $|k|\le|d|$. Then $|e/n-k/d|=|(kp+kq-k+1)/(nd)|<1/(2d^2)$ due to $|d|<|n|/4$. Therefore, k and d can be quickly computed from e and n.

In [BD99], the result is improved to breaking RSA for $|d|<0.292|n|$ by lattice method, which can be regarded as the generalization of approximation in multiple dimensions. In both [Wie90] and [BD99], the e satisfying $ed\equiv 1 \bmod \varphi(n)$ is the key point. But in our scheme, the e satifies $ed\equiv 1 \bmod q-1$ instead of mod $\varphi(n)$. If we target at the d' such that $ed'\equiv 1 \bmod \varphi(n)$ for public key e, the d' must be very large. Another attack to small exponent d is the meet-in-the-middle attack that is similar to the birthday attack but requires FFT technique. The complexity of that attack is $O((\log d)^2\sqrt{d})$. Therefore taking d 120 bits gives a security level of $2^{70\sim 80}$.

Chosen Ciphertext Attack. The scheme is fragile to chosen ciphertext attack. An attacker can choose a $M>q$ and set $c=M^e \bmod n$. The decryption $m=c^d \bmod q$ satisfies $\gcd(n, M-m)=q$. However this attack does not cause any problem if we carefully choose a mapping format before encryption, as done in [BR94] and [OU98], which provide provable security. Besides, the application of the scheme in our video system prevents the chosen ciphertext attack since the decrypted value never goes out of the tamper-resistant hardware device. The decrypted value is the key to encrypt video. So chosen ciphertext attack does not apply.

Factorization. In our scheme, n is a composite of two primes with different sizes. For the situation where n has 1024 bits and the smaller prime factor has 341 bit, the current factoring techniques cannot provide better performance than factoring 1024-bit n with two equal-size primes. This is because the most efficient number field sieve algorithm has complexity $L_n(1/3, c)$, which is dependent on size of n. Elliptic curve factoring algorithm is dependent on the size of the smaller prime q, but it has complexity $L_q(1/2, c)$. So currently available factoring techniques do not make factoring our n easier. The same argument is taken in [OU98], where $n=p^2q$ has 1024 bits, and in MultiPrime [Compaq] where n ($|n|$=1024) is a composite of three different primes.

References

[AK97] R. Anderson and M. Kuhn, "Low cost attacks on tamper resistant devices", in Security protocols: International Workshop'97, LNCS 1361, Springer-Verlag, pp.125-136, 1997.

[BDF00] F. Bao, R. Deng, P. Feng, An efficient and practical scheme for privacy protection in e-commerce of digital goods , Pre-Proceedings of The 3rd International Conference on Information Security and Cryptology (ICISC00), pp. 167-176, 2000.

[BR94] M. Bellare and P. Rogaway, "Optimal asymmetric encryption", Eurocrypt'94, LNCS, Springer-Verlag, 1995.

[BS91] E. Biham and A. Shamir, "Differential cryptanalysi of DES-like cryptosystems", Journal of Cryptology, Vol. 4, No. 1, pp. 3-72, 1991.

[BAK] E. Biham, R. Anderson and L. Knudsen, "Serpent: a proposal for the advanced encryption standard", http://www.cl.cam.ac.uk/~rja14/serpent.html.

[BD99] D. Boneh and G. Durfee, "Cryptanalysis of RSA with private key d less than $N^{0.292}$", Eurocrypt'99, pp. 1-11, Springer-Verlag, 1999.

[BDL97] D. Boneh, R. DeMillo, and R. Lipton, "On the importantance of checking cryptographic protocols for faults", in Proc of Eurocrypt'97, LNCS 1233, Springer-Verlag, pp. 37-51, 1997.

[BS95] D. Boneh and J. Shaw, "Collusion-secure fingerprinting for digital data", Crypto'95, LNCS, pp. 452-465, Springer-Verlag, 1995.

[Cor99] J. Coron, Risistance against differential power analysis for elliptic curve cryptosystems , Proc. Of CHES 99, LNCS 1717, Springer-Verlag, pp. 292-302, 1999.

[CG97] B. Chor and N. Gilboa, "Computational private information retrieval", Proc. of 29th STOC, pp. 304-313, 1997.

[CGKS95] B. Chor, O. Goldreich, E. Kushilevita, and M. Sudan, "Private information retrieval", Proc. of 36th FOCS, pp. 41-50, 1995.

[CKN01] J. Coron, P. Kocher, and D. Naccache, Statistics and secret leaksge , to appear in the Proceedings of Financial Cryptography 01, LNCS, Springer-Verlag.

[CL97] I.J.Cox, J.P.M.G.Linnartz, "Some general methods for tampering with watermarks", in IEEE international Conference on Image Processing", 1997.

[CMS99] C. Cachin, S. Micali, and M. Stadler, "Computationally Private Information Retrieval with Polylogrithmic Communication", in Proceedings of Eurocrypt'99, LNCS, Springer-Verlag, pp. 402-414, 1999.

[CMY96] S.Craver, N. Memon, B. Yeo, M. Yeung, "Can invisible watermarks resolve rightful ownership", IBM Research Report, RC 20509, July 25, 1996.

[CNS99] J. Coron, D. Naccache and J. Stern, "On the security of RSA padding", Crypto'99, pp. 1-18, Springer-Verlag, 1999.

[Compaq] http://www.tandem.com/brfs_wps/esscpttb/esscpttb.htm

[CWSK98] D. Coppersmith, D. Wagner, B. Schneier and J. Kelsey, "Cryptanalysis of TWOPRIMES", Proceedings of FSE'98, LNCS, Springer-Verlag, 1998.

[Dai] W. Dai, "Speed benchmarks of various ciphers and hash functions", http://www.eskimo.com/~weidai/benchmarks.html.

[GIKM98] Y. Gertner, Y. Ishai, E. Kushilevita and T. Malkin, "Protecting data privacy in private information retrieval schemes", Proc. of 30th STOC, 1998.

[GMDS98] C. Griwodz, O. Merkel, J. Dittmann, R. Steinmetz, Protecting VoD the Easier Way , ACM Multimedia 98, pp. 21-28, Bristol, UK, 1998.

[HPS99] H. Handschun, P. Paillier, and J. Stern, Probing attacks on tamper-resisyant devices , Proc. Of CHES 99, LNCS 1717, Springer-Verlag, pp. 303-315, 1999.

[Jai99] R. Jain, "The convergence of PCs and TV", IEEE Multimedia, October/December 1999.

[JQBD97] M. Joye, J.-J. Quisquater, F. Bao, and R.H. Deng, "RSA-type signatures in the presence of transient faults", In M. Darnell, editor, Cryptography and Coding, Vol. 1355 of Lecture Notes in Computer Science, pp. 155--160, Springer-Verlag, 1997.

[Knu98] L. R. Knudsen, "The block cipher lounge---AES", http://www.ii.uib.no /~larsr/aes.html.

[KO97] E. Kushilevita and R. Ostrovsky, "Single-database computationally private information retrieval", Proc. Of 38th FOCS, 1997.

[Kocher] P. Kocher, http://www.cryptography.com/resources/

[Mat93] M. Matsui, "Linear cryptanalysis method for DES cipher", Proceedings of Eurocrypt'93, LNCS 765, Springer-Verlag, pp. 386-397, 1994.

[Minoli] Video Dialtone Technology, McGraw-Hill, 1995.

[MQ95] B. M. Macq and J-J Quisquater, Cryptology for digital TV broadcasting , Proceedings of the IEEE, Vol. 83, No, 6, pp. 944-957, 1995.

[MS95] T. Maples and G. Spanos, "Performance study of a selective Encryption scheme for security of networked, real-time video", Proc. of the 4th International Conference on Computer and Communications and Networks, Las Vegas, Nevada, Sept, 1995.

[OU98] T. Okamoto and S. Uchiyama, "A new public-key cryptosystem as secure as factoring", Proceedings of Eurocrypt 98, LNCS, Springer-Verlag, 1998.

[PS96] B. Pfitzmann and M. Shunter, "Asymmetric fingerprinting", Eurocrypt'96, LNCS 1070, pp. 84-95, Springer-Verlag, 1996.

[PS99] B. Pfitzmann and A. Sadeghi, "Coin-based anonymous fingerprinting", Eurocrypt'99, pp. 150-164, Springer-Verlag, 1999.

[QNT97] L. Qiao, K. Nahrstedt and I. Tam, "Is MPEG encryption using random lists instead of Zig Zag Order", IEEE International Symposium on Consumer Electronics, Dec, 1997.

[Rue86] R. A. Rueppel, Analysis and Design of Stream Ciphers, Springer-Verlag, 1986.

[Tan96] L. Tang, "Methods for Encrypting and decrypting MPEG video data efficiently", Proc. of the 4th ACM Multimedia Conference, Boston, MA, November, 1996.

[Wie90] M. Wiener, "Cryptanalysis of short RSA secret exponents", IEEE Transactions on Information Theory, Vol. 36, No. 3, pp. 553-558, 1990.

[WBYD00] H. Wu, F. Bao, D. Ye, R. Deng, "Cryptoanalysis of the m-permutation protection schemes", Proc. of ACISP2000, LNCS 1841, Springer-Verlag, pp. 97-111, 2000.

[ZK95] J. Zhao and E. Koch, "Embedding robust label into images for copyright protection", Proceedings of the International Conference on Intellectual Property Rights for Specialized Information, Knowledge and New Technologies, Austria, Aug. 21-25, 1995.

Appendix

Fast Encryption Scheme I

Now we introduce the first scheme of our fast encryption framework. AES is supposed to be the encryption standard for this whole century. It is regarded unbroken unless some impossible breakthroughs in math take place. Therefore, we take ***SE***() as AES (Rijndael). The stream cipher ***FE***() is given as follows. The whole picture of the scheme is described in Section 6.

Description of the Stream Cipher *FE*()

The stream cipher has a 128-bit key size and operates on 32-bit plaintext strings $b_1 b_2 \cdots b_m \cdots$. Denote the 128-bit key as $k = k_1 k_2 k_3 k_4$, where k_i s are 32-bit strings. Define

$$F(k,x) = ((((x + k_1) \oplus k_2) \times k_3) \oplus k_4) \lrcorner$$

where x is a 32-bit string, $\oplus$ is the bit-wise XOR, + and $\times$ are mod 2^{32} addition and multiplication respectively, and $\lrcorner$ is to reverse the 32 bits into opposite ranking. Encryption of the plaintext strings $b_1 b_2 \cdots b_m \cdots$ is then given by

$$d_i = b_i \oplus F(k, F(k, F(k, d_{i-1}) \oplus b_{i-1}) \oplus d_{i-2})$$

where $d_1 d_2 \cdots d_m \cdots$ are the corresponding ciphertext strings and where d_{-1}, d_0, b_0 are set to k_2, k_3, k_4, respectively.

We implemented the encryption scheme on a 233MHz Pentium-II/MMX processor (The encryption speed of Serpent on the same processor is about 25.8 Mbit/s). The experiment are given in the table below.

Table 1. Experiment Results of Encryption Scheme I

Total Data Size (bits)	Segment Size (bits)	Test 1 (Mbit/s)	Test 2 (Mbit/s)	Test 3 (Mbit/s)	Test 4 (Mbit/s)	Average Speed (Mbit/s)
5,242,880,000	32,768	297.0	296.1	297.0	297.0	296.7
5,242,880,000	65,536	304.0	302.9	304.0	304.0	303.7
5,242,880,000	131,072	307.0	307.0	307.0	308.1	307.3
5,242,880,000	262,144	309.1	309.1	309.1	309.3	309.2

Security Discussion

Security of the secret key: The secret key K is protected by ***SE***() which by our assumption, is secure against all known attacks.

Meet-in-the-middle attack to segment keys: This is a type of brute force attack. By meeting one or more bits in the middle, the attacker exhaustively search the key bits relevant to these middle bits. Since we take 3 rounds of F, the meet-in-the-middle attack does not work. This is because at least one of the two sides of the middle bits goes through two rounds of F; therefore, at least 96 bits of the key effect one middle bit.

Chosen ciphertext attack to segment keys: It is well known that all stream ciphers that have ciphertext feedback are vulnerable to chosen ciphertext attacks. Suppose our stream cipher was defined as

$$d_i = b_i \oplus F(k, F(k, F(k, d_{i-1}) \oplus d_{i-2}) \oplus d_{i-3}).$$

By letting $d_{i-1} = d'_{i-1}$, $d_{i-2} = d'_{i-2}$ and d_{i-3} and d_{i-3} differing in only one bit, an attacker can ask for the decryption of d_i and d_i by applying the **differential attack**. However, our stream cipher is defined by

$$d_i = b_i \oplus F(k, F(k, F(k, d_{i-1}) \oplus b_{i-1}) \oplus d_{i-2})$$

which has both ciphertext and plaintext feedback. In this case, if the attacker chooses both plaintext and ciphertext, the decrypted plaintext from the chosen ciphertext will have a very small chance to match the chosen plaintext. On the other hand, if the attacker tries to find such match from known plaintext/ciphertext (instead of chosen plaintext/ciphertext), the required number of known plaintext/ciphertext pairs is around 2^{48} blocks (like the birthday attack to $2^{3\times 32} = 2^{96}$). However, this amount of plaintext/ciphertext pairs will not be available to the attacker since our segment size can never be so large.

Fast Encryption Scheme II

This encryption scheme is identical to Scheme I except that it uses another very fast stream cipher ***FE*()**, which is given below.

Description of the Stream Cipher *FE*()

This stream cipher is used to expand a 128-bit key into a key stream of a plaintext segment size. Before illustrating its detailed design, we introduce the notations below:

T:	a table containing 19 elements, with each element consisting of 32 bits.
T_i:	the ith element of the table T
k:	the 128-bit secret key, consisting of four 32-bit words: k_0, k_1, k_2 and k_3.
c_i:	a 32-bit constant generated from the constant e as $c_i = (e \times 2^{32(i+1)})$ & 0xFFFFFFFF, $i = 0$ to 18.
r'_i:	a constant between 3 and 14. It is generated from the constant π as $r'_i = ((\pi \times 2^{8(i+1)})\ \&\ \text{0xFF}) \bmod 12 + 3$, $i = 0$ to 18.

We use the standard notations &, $\oplus$ and >>> to represent bit-wise AND, bit-wise XOR and right rotation, respectively. In addition, we define a feedback function F and an output function G below.

Definition of *F*. The input to F is the table T and a rotation constant r. The output of F is given as

$$f = ((T_0 \oplus T_4) + T_{13}) >>> r) \oplus T_{14}$$

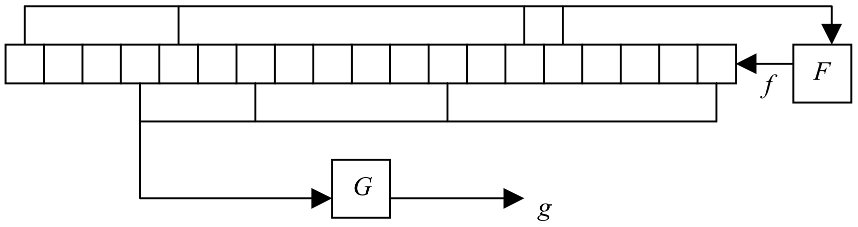
Definition of G. The input to G is the table T and the output is given as

$$g = ((T_{18} + T_{11}) \oplus T_6) + T_3 .$$

The operation of this stream cipher consists of two stages: the initial setup stage and the output stage or the main algorithm.

The initial setup stage

1. Let $T_i = c_i + k_{i \bmod 4}$, for $i = 0$ to 18.
2. Let $r_i = r_i' + ((k_0 >> 4i) \,\&\, 0xF)$ for $i = 0$ to 7;
   $r_i = r_i' + ((k_1 >> (4 \times (i-8))) \,\&\, 0xF)$ for $i = 8$ to 15;
   $r_i = r_i' + ((k_2 >> (4 \times (i-16))) \,\&\, 0xF)$ for $i = 16$ to 18.
3. Run the main algorithm below for 38 cycles and prepare for the output.



The main algorithm: For the ith cycle

1. Run the F function with $r = r_{i \bmod 19}$ to obtain the value of f.
2. Let $T_j = T_{j+1}, j = 0$ to 17, and let $T_{18} = f$.
3. Run the function G and generate the output g .

We implemented the encryption scheme described on a 233MHz Pentium-II/MMX processor. The encryption speed of Serpent on the same processor is about 25.8 Mbit/s. The experiment results are given in the table below.

Table 2. Experiment Results of Encryption Scheme II

Total Data Size (bits)	Segment Size (bits)	Test 1 (Mbit/s)	Test 2 (Mbit/s)	Test 3 (Mbit/s)	Test 4 (Mbit/s)	Average Speed (Mbit/s)
24,903,680,000	38,912	1234.9	1236.2	1234.9	1235.5	1235.4
24,903,680,000	79,824	1370.5	1370.5	1369.7	1370.5	1370.3
24,903,680,000	159,648	1448.8	1447.8	1451.8	1453.8	1449.8
24,903,680,000	319,296	1493.1	1492.1	1491.2	1493.1	1492.4

Security Discussion of Encryption Scheme II

First of all, the secret encryption key K is protected by ***SE***. Therefore, attacking the key is as hard as attacking AES, which is supposed to be absolutely secure against all attacks. Second, each segment key generated by ***SE*** is used to encrypt a plaintext segment of very limited length by the stream cipher. For known ciphertext attack, our stream cipher can resist a large number of known ciphertexts.

The security of this stream cipher greatly depends on the feature that those 19 elements of the table *T* are updated in a non-linear way as the encryption goes on. With the carefully chosen parameters of function *F*, we can show that any two outputs of *F* are generated from at most one of the same elements of *T*. We note that each updated element, which is the output of *F*, is the non-linear combination of four elements of *T*. The key-related rotation amount in *F* strengthened the cipher further. With these unknown rotation amounts, we believe that it would be very difficult to find linear relationship among the elements of *T*.

The key stream is also generated from the elements of *T* in a non-linear way. The parameters of function *G* are carefully chosen so that any two outputs of *G* are generated from at most one of the same elements of *T*, and any output of *G* is generated from at most one of the same elements of *T* as any output of *F* (the updated element). Thus recovering the continuously updating elements of *T* from the output of *G* or revealing the linear relationship among the generated key stream becomes almost infeasible.

In this stream cipher, the elements of *T* are modified in a non-linear way. Thus it is not possible to compute the period of the generated key stream cipher. However, the period would not be a problem here. There are 19 32-bit elements of *T*. It is very unlikely that those elements will come back to their initial values even in the process of generating a 2^{128}-bit key stream. Furthermore, the stream cipher is used to encrypt only one package. The period of the output key stream is believed to be far larger than the size of a package.

Private Information Retrieval Based on the Subgroup Membership Problem

Akihiro Yamamura[1] and Taiichi Saito[2]

[1] Communications Research Laboratory,
4-2-1, Nukui-Kitamachi, Koganei, Tokyo 184-8795, Japan
aki@crl.go.jp
[2] NTT Laboratories,
1-1, Hikarinooka, Yokosuka, Kanagawa 239-0847, Japan
taiichi@sucaba.isl.ntt.co.jp

Abstract. Many algorithmic problems, which are used to prove the security of a cryptographic system, are shown to be characterized as the subgroup membership problem. We then apply the subgroup membership problem to private information retrieval schemes following the method by Kushilevitz and Ostrovsky. The resulting scheme has the same communication complexity as that of Kushilevitz and Ostrovsky.

1 Private Information Retrieval

Chor, Goldreich, Kushilevitz and Sudan [3] introduced the *private information retrieval scheme* for remote database access, in which the user can retrieve the data of user's choice without revealing it. Their scheme attains *information theoretic security*, however, the database must be replicated in several locations where the managers are not allowed to communicate each other. The *computational private information retrieval scheme* was introduced by Chor and Gilboa [4]. Their scheme attains more efficient communication than Chor, Goldreich, Kushilevitz and Sudan's model by sacrificing the information theoretic security, nevertheless, their scheme enjoys computational security by assuming the existence of pseudorandom generators. However, their scheme still needs replication of the database. Kushilevitz and Ostrovsky [6] introduced a computational private information retrieval scheme in which only one database is needed. Their scheme depends on the intractability of the quadratic residue problem. More efficiency, polylogarithmic communication complexity, is attained by Cachin, Micali and Stadler [2]. They assume a number theoretic hypothesis, which they call the Φ assumption, and sacrifice one-round communication and then obtain polylogarithmic communication complexity. However, a rigorous proof of the intractability of the Φ assumption or its equivalence to a widely used assumption like the quadratic residue assumption or the integer factorization is not given in [2]. We summarize the known results on private information retrievals in Table 1 below.

We briefly review the general scheme of a private information retrieval (PIR for short) scheme. A computational PIR scheme with a single database is a protocol for two players, a user $\mathcal{U}$ and a database manager $\mathcal{DB}$. Both are able to perform only probabilistic polynomial time computation. The database manager $\mathcal{DB}$ maintains a database, which is a binary sequence $X = x_0 x_1 x_2 \cdots x_{n-1}$. The goal of the protocol is to allow $\mathcal{U}$ to obtain the ith bit x_{i+1} of X without leaking any information on x_i to $\mathcal{DB}$. The protocol runs as follows:

Step 1 $\mathcal{U}$ computes a query Query(i) using his random tape (coin toss), which $\mathcal{U}$ keeps secret. Then he sends Query(i) to $\mathcal{DB}$.

Step 2 $\mathcal{DB}$ receives Query(i). He performs a polynomial-time computation for the input X, Query(i) and his random tape. The computation yields the answer Answer(Query(i)). He sends Answer(Query(i)) back to $\mathcal{U}$.

Step 3 $\mathcal{U}$ receives Answer(Query(i)). He performs a polynomial-time computation using the answer Answer(Query(i)) and his private information (his random tape). The computation yields the ith bit x_{i+1} of the database.

Correctness
For any database sequence X and for any query Query(i) for ith bit of X, $\mathcal{U}$ obtains x_i at the end.

Privacy
$\mathcal{DB}$ cannot distinguish a query for the ith bit and a query for the jth bit for all i and j by a polynomial-time (probabilistic) computation with non-negligible probability. Formally, for all constants c, for all database of length n, for any two $1 \leq i, j \leq n$, and all polynomial-size family of circuits C_k, there exists an integer K such that for all $k > K$ we have

$$|\mathbf{Prob}(C_k(\mathrm{Query}(i)) = 1) - \mathbf{Prob}(C_k(\mathrm{Query}(j)) = 1)| < \sigma \ , \qquad (1.1)$$

where k is the security parameter of the protocol and $\sigma = \frac{1}{(\mathrm{Max}(k,n))^c}$.

Computation
Computations of both $\mathcal{DB}$ and $\mathcal{U}$ are bounded above by a polynomial in the size n of the database and the security parameter k.

2 Subgroup Membership Problem

The quadratic residue (QR for short) problem and the decision Diffie-Hellman (DDH for short) problem have numerous applications in cryptography, and hence, they have been studied in detail. Our aim of this paper is to generalize and formalize them as the *subgroup membership problem* and to show many other algorithmic problems, which are used in public key cryptography, are characterized as the subgroup membership problem as well. Such a unification of algorithmic problems used in cryptography has not been appeared up to date as far as the authors are concerned. Widely used assumptions in cryptography are divided into two groups: the algorithmic assumptions related to the integer factoring (and the QR) and the algorithmic assumptions related to the discrete logarithm problem (and the DDH). The first is originated from the RSA cryptosystem and

Table 1. Several Private Information Retrieval Schemes

Scheme	Round Number	Security Assumption	Communication Complexity	Number of DBs
Chor, Goldreich, Kushilevitz, Sudan [3]	1	Information Theoretical	$O(n^{1/3})$	≥ 2
Ambainis [1]	1	Information Theoretical	$O(n^{1/2k-1})$ for $k(>1)$ DBs	≥ 2
Chor and Gilboa [4]	1	Existence of Pseudo Number Generators	$O(n^c)$ for $c > 0$	≥ 2
Kushilevitz and Ostrovsky [6]	1	Quadratic Residue Problem Assumption	$O(n^c)$ for $c > 0$	1
Ostrovsky and Shoup [9]	Multiple	Reduction to Read only scheme		
Cachin, Micali and Stadler [2]	2	Φ Assumption	Polylogarithmic	1
Proposed Scheme	1	Subgroup Membership Assumption (e.g. DDH assumption)	$O(n^c)$ for $c > 0$	1

the second from the Diffie-Hellman key exchange protocol. These two look different and are usually discussed separately. The unified approach to the integer factoring problem and the discrete logarithm problem shed light on the fundamental properties of algorithms required to provide the security. Therefore, we can get better understanding of the algorithmic problems by unified treatment of subgroup membership problems.

Once we prove that the subgroup membership problem is applicable to a certain scheme in general, then any primitive based on the subgroup membership problem concerning a specific group is applicable to the scheme in principle. As an example, in this paper, we show that any subgroup membership problem can be employed to construct a computational PIR system by constructing a PIR system using the subgroup membership problem in a general manner.

2.1 Subgroup Membership Assumption

Determining the membership of a given element of a certain group in its subgroup is not always easy. As a matter of fact, the membership problem of a subgroup in a finitely presented group is not recursive in general. To apply the membership problem to cryptographic schemes such as asymmetric cryptosystems, we require the efficiency of computation for legal participants and the existence of a trapdoor. In this section we consider the subgroup membership problem with a trapdoor, and show that several problems widely used in cryptography are characterized as the subgroup membership problem.

Let G be a group, and let H be its subgroup. The membership problem is to decide whether or not a given element $g \in G$ belongs to H. We suppose that every element in G has a binary representation of size k, where k is the security parameter. The membership can be decided within polynomial time in k if a certain information, called a *trapdoor*, is provided. The membership of an element $g \in G$ in H can be decided provided the trapdoor, however, the membership cannot be decided with a probability substantially larger than $\frac{1}{2}$ without the trapdoor. We now formalize the subgroup membership problem.

Let k be the security parameter. For the input 1^k, a probabilistic polynomial time algorithm $\mathcal{IG}$ outputs the description of a group G, the description of a subgroup $H \subset G$ and the trapdoor that provides a fast algorithm for the subgroup membership problem of H in G. The algorithm $\mathcal{IG}$ is called the *instance generator*. Every element of G is represented as a binary sequence of length k. Computation of the multiplication in G is performed in polynomial time in k.

The predicate for the membership of a subgroup is denoted by Mem, that is, Mem is defined as follows:

$$\mathrm{Mem}(G, H, x) = \begin{cases} 1 & \text{if } x \in H \\ 0 & \text{if } x \in S \end{cases} ,$$

where $\mathcal{IG}$ outputs the pair (G, H) for 1^k, x is in G, and $S = G \setminus H$. The *subgroup membership problem* is to compute Mem in polynomial time in k when we inputs 1^k and obtain a pair of groups (G, H) and an element g in G, which is uniformly and randomly chosen from H or G according to the coin toss $b \overset{R}{\leftarrow} \{0, 1\}$. If there does not exist a probabilistic polynomial time algorithm that computes Mem with a probability substantially larger than $\frac{1}{2}$, then we say that the membership problem is *intractable*. We also assume that one can choose uniformly and randomly an element from both H and G. This is significant to apply to cryptographic schemes.

The following is trivial, however, it is useful for the construction of an PIR system based on the subgroup membership problem.

Proposition 1. *Let G be a group, and let H be a subgroup of G. For any $g \in G$ and $h \in H$, we have $gh \in H$ if and only if $g \in H$.* □

Subgroup Membership Assumption I
For every constant c, and every family $\{C_k \mid k \in \mathbb{N}\}$ of circuits of polynomial size in k, there is an integer K such that for all $k > K$ we have

$$\mathbf{Prob}(C_k(G, H, g) = \mathrm{Mem}(G, H, g)) < \frac{1}{2} + \frac{1}{k^c} , \tag{2.1}$$

where the probability is taken over $(G, H) \leftarrow \mathcal{IG}(1^k)$, $b \overset{R}{\leftarrow} \{0, 1\}$, $g \overset{R}{\leftarrow} H$ if $b = 1$, $g \overset{R}{\leftarrow} S$ if $b = 0$.

The assumption claims that there exists no polynomial size circuit family to compute the predicate Mem. The following is equivalent to the assumption above.

Subgroup membership assumption II
For every constant c, and every family $\{C_k \mid k \in \mathbb{N}\}$ of circuits of polynomial size in k, there is an integer K such that for all $k > K$ we have

$$|\mathbf{P}_H - \mathbf{P}_S| < \frac{1}{k^c} \ , \tag{2.2}$$

where the probabilities $\mathbf{P}_H$ and $\mathbf{P}_S$ are defined as follows;

$$\mathbf{P}_H = \mathbf{Prob}_{(G,H)\leftarrow \mathcal{IG}(1^k)\ ;\ g \overset{R}{\leftarrow} H}(C_k(G,H,g)=1) \ ,$$

and

$$\mathbf{P}_S = \mathbf{Prob}_{(G,H)\leftarrow \mathcal{IG}(1^k)\ ;\ g \overset{R}{\leftarrow} S}(C_k(G,H,g)=1) \ .$$

2.2 Examples

We exhibit several subgroup membership problems: the DDH problem, the QR problem, the rth residue (RR for short) problem studied by Kurosawa and Tsujii [7], the p-subgroup (PSUB for short) problem introduced by Okamoto and Uchiyama [10] and the decisional composite residuosity (DCR for short) problem introduced by Paillier [11]. Recall that the assumption that the QR problem is intractable (QR assumption) is employed to prove the semantic security of Goldwasser-Micali cryptosystem [5], and the assumption that the DDH problem is intractable (DDH assumption) is employed to prove the semantic security of ElGamal cryptosystem. These two have many other applications. The assumption that one of problems above is intractable is employed to prove the semantic security of the corresponding cryptosystem [7], [10], [11], respectively. We also note that the security of the cryptosystem introduced by Naccache and Stern [8] depends on the PSUB assumption as well.

Quadratic Residue Problem
Let p, q be primes. Set $N = pq$. The primes p and q are trapdoor information for the quadratic residue problem, on the other hand, the number N is public information. Let G be the subgroup of $(\mathbb{Z}/(N))^*$ consisting of the elements whose Jacobi symbol is 1, and let H be the subgroup of G consisting of quadratic residues of G, that is, $H = \{x \in G \mid x = y^2 \bmod N \text{ for } y \in (\mathbb{Z}/(N))^*\}$. The quadratic residue problem of H in G is to decide whether or not, a given element $g \in G$, g belongs to H. We can effectively determine the membership of g in H provided that the information p and q are available. No polynomial time algorithm is known for the membership of a randomly chosen element of G in H without the information p and q. Hence, if we define an instance generator for the QR problem as a probabilistic algorithm that outputs two primes p and q of size k and a quadratic non-residue h whose Jacobi symbol is 1 for the input 1^k, then the QR problem is considered as a subgroup membership problem. Note that we can obtain a quadratic non-residue h with Jacobi symbol 1 by using p, q, and that it is possible to uniformly and randomly choose elements from H without the trapdoor information provided h is given.

Decision Diffie-Hellman Problem
Let C be a cyclic group of prime order p. The group C may be a multiplication group of a finite field or a group of rational points of an elliptic curve. Let g be a generator of C. The decision Diffie-Hellman problem is to decide whether or not $h_2 = g_2^a$ for the given quadruple (g_1, h_1, g_2, h_2) of elements in C with $h_1 = g_1^a$ for some $1 \leq a \leq p-1$. If so, we say that (g_1, h_1, g_2, h_2) is a Diffie-Hellman quadruple. The integer a is the trapdoor of the decision Diffie-Hellman problem. Knowing the trapdoor a, we can efficiently decide whether or not $h_2 = g_2^a$.

We show that the DDH problem can be characterized as a subgroup membership problem for a certain group. We set G to be the direct product $C \times C$. Then the input to the DDH problem is (x, y) where $x, y \in G$, that is, $x = (g_1, h_1)$ and $y = (g_2, h_2)$. It is obvious that (g_1, h_1, g_2, h_2) is a Diffie-Hellman quadruple if and only if y belongs to the subgroup $< x >$ of G generated by x. It follows that the DDH problem for the cyclic group C is equivalent to the subgroup membership problem of the group $H =< x >$, where $x = (g_1, g_1^a)$, in the group $G = C \times C =< g_1 > \times < g_1 >$. Note that, when a generator x of H is given, it is possible to choose uniformly and randomly elements from H without the trapdoor information.

Rth Residue Problem
The RR problem is a natural extension of the QR problem defined as follows. Let p, q be primes, and let e_1, e_2 be odd integers dividing $p-1$ and $q-1$, respectively, such that e_1 is prime to $q-1$ and e_2 is prime to $p-1$. Set $N = pq$ and $r = e_1 e_2$. The primes p and q are the trapdoor information for the RR problem, on the other hand, the number N and r are the public information. Let G be the group $(\mathbb{Z}/(N))^*$, and let H be the subgroup consisting of rth residues of G, that is, $H = \{x \in G \mid x = y^r \bmod N \text{ for } y \in G\}$. The RR problem of H in G is to decide whether or not, a given element $g \in G$, g belongs to H. Thus, the RR is a subgroup membership problem of H in G. We can effectively determine the membership of g in H provided that the information p and q are available. No polynomial time algorithm is known for the membership of a randomly chosen element of G in H without the information p and q. Note that we can obtain an element h such that $h^i \notin \{x^r \bmod N : x \in (\mathbb{Z}/(N))^*\}$ for any $1 \leq i \leq r-1$ by using the trapdoor information, and that we can uniformly and randomly choose an element from H provided h is given.

P-Subgroup Problem
Let p, q be primes such that p does not divide $q-1$. Set $N = p^2 q$ and let g be a random element in $(\mathbb{Z}/(N))^*$ such that the order of $g^{p-1} \bmod p^2$ is p. The primes p and q are trapdoor information for the PSUB problem, on the other hand, the number N, g, k are public information. Let G be a group defined by $G = \{x \mid x = g^m y^N \bmod N \text{ for } m \in \mathbb{Z}/(p) \text{ and } y \in (\mathbb{Z}/(N))^*\}$, and let H be the subgroup defined by $H = \{x \mid x = y^N \bmod N \text{ for } y \in G\}$. The PSUB problem of H in G is to decide whether or not, a given element $g \in G$, g belongs to H. Thus, the PSUB is the membership problem of H in G. We can efficiently determine the membership of g in H provided that the information p and q are available. No

polynomial time algorithm is known for the membership of a randomly chosen element of G in H without the information p and q. Note that our description of PSUB is slightly diffrent from Okamoto-Uchiyama [10], where the PSUB is introduced as a variant of the *coset indistinguishability problem*, which we will present in Section 2.3. Naccache and Stern [8] implicitly used PSUB problem in their scheme. Paillier introduces the *decisional composite residuosity* (DCR for short). This is a generalization of [10] and also characterized as a subgroup membership problem.

For other plausible applications of the subgroup membership problem, the reader is also referred to [12] in which the DDH assumption is applied to the cryptographic schemes which only known method to construct is to base on the QR assumption. We summarize the examples above in Table 2, however, the table is not exhaustive at all.

Table 2. Subgroup Membership Problems

	Related Problem	Group / Subgroup	Applications
DDH	DLP	$C \times C$: Direct Product of Cyclic Groups	ElGamal
	DH	$\langle (g,h) \rangle$: Subgroup Generated by (g,h)	
QR	FACT(pq)	$\{x \in \mathbb{Z}_N^* \mid (\frac{x}{N}) = 1\}$	Goldwasser-Micali [5]
		$\{x^2 \bmod N \mid x \in \mathbb{Z}_N^*\}$	
RR	FACT(pq)	$\mathbb{Z}_N^*$	Kurosawa-Tsujii [7]
		$\{x^r \bmod N \mid x \in \mathbb{Z}_N^*\}$	
PSUB	FACT(p^2q)	$\{x \mid x = g^m y^N \bmod N$ for $m \in \mathbb{Z}/(p),\ y \in (\mathbb{Z}/(N))^*\}$	Okamoto-Uchiyama [10]
		$\{y^N \bmod N \mid y \in \mathbb{Z}_N^*\}$	Naccache-Stern [8]
DCR	FACT(pq)	$\{x \mid x = g^m y^N \bmod N^2$ $m \in \mathbb{Z}/(N), y \in (\mathbb{Z}/(N^2))^*\}$	Paillier [11]
		$\{y^N \bmod N^2 \mid y \in (\mathbb{Z}/(N^2))^*\}$	

2.3 Equivalent Problems

We examine several algorithmic problems equivalent to the subgroup membership problem. Suppose that $\mathcal{IG}$ is an instance generator of a family of groups, and that $\mathcal{IG}$ outputs (G,H) for the input 1^k. We set $S = G \setminus H$. Suppose that t is an integer bounded above by a polynomial in k. Let K_i be the direct product of $t-1$ H's and S, where all jth position $(j \neq i)$ is occupied by H except for ith position, that is, $K_i = H \times H \times \cdots \times \overset{i}{S} \times \cdots \times H$ for every $i = 1, 2, \ldots, t$. Let L be the union of K_1, K_2, $\cdots$, K_t, that is, $L = K_1 \bigcup K_2 \bigcup \cdots \bigcup K_t$.

Pattern Indistinguishability Assumption
The *pattern indistinguishability assumption* is to assume the following holds: for every constant c, every family $\{C_k \mid k \in \mathbb{N}\}$ of circuits of polynomial size in k and all i, j such that $1 \leq i, j \leq n$ there is an integer K such that for all $k > K$ we have

$$|\mathbf{P}_i - \mathbf{P}_j| < \frac{1}{k^c} \ . \tag{2.3}$$

Here the probabilities $\mathbf{P}_i$ and $\mathbf{P}_j$ are defined as follows;

$$\mathbf{P}_i = \mathbf{Prob}_{(G,H) \leftarrow \mathcal{IG}(1^k)\ ;\ (g_1, g_2 \ldots, g_t) \overset{R}{\leftarrow} K_i} (C_k(G, H, i, g_1, g_2 \ldots, g_t) = 1) \ ,$$

$$\mathbf{P}_j = \mathbf{Prob}_{(G,H) \leftarrow \mathcal{IG}(1^k)\ ;\ (g_1, g_2 \ldots, g_t) \overset{R}{\leftarrow} K_j} (C_k(G, H, i, g_1, g_2 \ldots, g_t) = 1) \ .$$

General Pattern Indistinguishability Assumption
The *general pattern indistinguishability assumption* is to assume the following holds: for every constant c, every family $\{C_k \mid k \in \mathbb{N}\}$ of circuits of polynomial size in k and all $(i_1, i_2, \ldots, i_u)$ and $(j_1, j_2, \ldots, j_u)$, there is an integer K such that for all $k > K$ we have

$$|\mathbf{P}_{(i_1, i_2, \ldots, i_u)} - \mathbf{P}_{(j_1, j_2, \ldots, j_u)}| < \frac{1}{k^c} \ . \tag{2.4}$$

Here the probabilities $\mathbf{P}_{(i_1, i_2, \ldots, i_u)}$ and $\mathbf{P}_{(j_1, j_2, \ldots, j_u)}$ are defined by

$$\mathbf{P}_{(i_1, i_2, \ldots, i_u)} = \mathbf{Prob}(C_k(G, H, x_1, x_2 \ldots, x_u) = 1) \ ,$$

where the probability is taken over $(G, H) \leftarrow \mathcal{IG}(1^k)$ and $(x_1, x_2 \ldots, x_u) \overset{R}{\leftarrow} K_{i_1} \times K_{i_2} \times \cdots \times K_{i_u}$ and

$$\mathbf{P}_{(j_1, j_2, \ldots, j_u)} = \mathbf{Prob}(C_k(G, H, x_1, x_2 \ldots, x_u) = 1) \ ,$$

where the probability is taken over $(G, H) \leftarrow \mathcal{IG}(1^k)$ and $(x_1, x_2 \ldots, x_u) \overset{R}{\leftarrow} K_{j_1} \times K_{j_2} \times \cdots \times K_{j_u}$.

Coset Indistinguishability Assumption
The *coset indistinguishability assumption* is to assume the following holds: for every constant c, every family $\{C_k \mid k \in \mathbb{N}\}$ of circuits of polynomial size in k and every algorithm F that on input (G, H) outputs a pair of elements in G, there is an integer K such that for all $k > K$ we have

$$\mathbf{Prob}(C_k(G, H, g_0, g_1, g) = b) < \frac{1}{2} + \frac{1}{k^c} \ , \tag{2.5}$$

where the probability is taken over $(G, H) \leftarrow \mathcal{IG}(1^k)$, $(g_0, g_1) \leftarrow F(G, H)$, $b \overset{R}{\leftarrow} \{0, 1\}$ and $g \overset{R}{\leftarrow} g_b H$.

Theorem 1. *The following are equivalent.*
(1) The subgroup membership assumption I.
(2) The subgroup membership assumption II.
(3) The pattern indistinguishability assumption.
(4) The general pattern indistinguishability assumption.
(5) The coset indistinguishability assumption.

Proof. We show the equivalence among (1), (2), (3). Note that (1) clearly implies (3). The proof for the equivalence among (1), (4) and (5) is omitted.

(2) implies (1): Suppose that there exists a constant c and that for every K, there is $k \geq K$ such that the circuit C_k does not satisfy (2.1). Note that $\mathbf{Prob}(C_k(G,H,g) = \mathrm{Mem}(G,H,g)) = \frac{1}{2}\mathbf{P}_H + \frac{1}{2}(1-\mathbf{P}_S)$. Since (2.1) does not hold, we have $\frac{1}{2}(\mathbf{P}_H - \mathbf{P}_S + 1) > \frac{1}{2} + \frac{1}{k^c}$. Therefore we have $|\mathbf{P}_H - \mathbf{P}_S| > \frac{2}{k^c}$.

(1) implies (2): Suppose that there exists a constant c and that for every k, there is $k \geq K$ such that the circuit C_k does not satisfy (2.2). For the circuit C_k, we have $\mathbf{Prob}(C_k(G,H,g) = \mathrm{Mem}(G,H,g)) = \frac{1}{2}\mathbf{P}_H + \frac{1}{2}(1-\mathbf{P}_S) = \frac{1}{2}(1+\mathbf{P}_H - \mathbf{P}_S) > \frac{1}{2} + \frac{1}{k^c}$.

(3) implies (2): Suppose that there exists a constant c and that for every k, there is $k \geq K$ such that the circuit C_k does not satisfy (2.3). Construct a circuit C'_k as follows. Given (G,H) and $g \in G$, we choose uniformly and randomly $x_1, x_2, \ldots, x_{t-2}$ form H. We also choose uniformly and randomly y from H. We toss a coin, say, $b \xleftarrow{R} \{0,1\}$. If $b = 0$, then we input $(G, H, x_1, x_2, \ldots, \overset{i}{y}, \ldots, \overset{j}{g}, \ldots, x_{t-2})$, and the circuit C'_k returns the output of C_k. If $b = 1$, then we input $(G, H, x_1, x_2, \ldots, \overset{i}{g}, \ldots, \overset{j}{y}, \ldots, x_{t-2})$, and the circuit C'_k returns the negation of the output of C_k. If $g \in S$, then we have $\mathbf{Prob}(C'_k(G,H,g) = 1 : g \leftarrow S) = \frac{1}{2}\mathbf{P}_i + \frac{1}{2}(1-\mathbf{P}_j)$. If $g \in H$, then we have $\mathbf{Prob}(C'_k(G,H,g) = 1 : g \leftarrow H) = \frac{1}{2}\theta + \frac{1}{2}(1-\theta)$, where $\theta = \mathbf{Prob}(C_k(G,H,g_1,g_2,\ldots,g_t))$ and the probability is taken over $g_1, g_2, \ldots, g_t$ are taken uniformly and randomly from H. It follows that $|\mathbf{P}_H - \mathbf{P}_S| > \frac{1}{2}|\mathbf{P}_i - \mathbf{P}_j| > \frac{1}{2k^c}$. □

3 PIR Based on the Subgroup Membership Problem

We show that the subgroup membership problem can be applied to a PIR scheme by modifying Kushilevitz and Ostrovsky's scheme [6]. The proposed scheme has the same communication complexity as Kushilevitz and Ostrovsky's scheme whose security depends on the QR assumption. On the other hand, the security of the private information retrieval scheme proposed in this paper is based on the subgroup membership assumption. Therefore, we can construct a private information retrieval scheme based on any algorithmic problems in Section 2.2, in particular, we can use groups of rational points on elliptic curves or multiplicative groups of finite fields under the corresponding DDH assumption. We should remark that all the private information retrieval schemes proposed so far depend on either the existence of pseudorandom number generators or intractability assumption related to the integer factorization. No private information retrieval scheme based on the DDH has been proposed, yet as far as the authors are concerned. Modifying [6], we construct a PIR scheme based on the subgroup membership problem.

3.1 Basic Idea

First of all, we explain the basic idea of the scheme by a simple model. Suppose $\mathcal{DB}$ has the database $X = x_0x_1x_2\cdots x_{n-1}$ and that $\mathcal{U}$ wishes to know the ith bit x_{i-1}. $\mathcal{U}$ chooses group elements $g_0, g_1, g_2, \ldots, g_{i-1}, \ldots, g_{n-1}$ so that $g_j \in H$ for $j \neq i-1$ and $g_{i-1} \in S = G \setminus H$. Then $\mathcal{U}$ sends them all to $\mathcal{DB}$. $\mathcal{DB}$ computes the group element $g = g_0^{x_0} g_1^{x_1} g_2^{x_2} \cdots g_{i-1}^{x_{i-1}} \cdots g_{n-1}^{x_{n-1}}$ and sends it back to $\mathcal{U}$. $\mathcal{DB}$ cannot get to know which of $g_0, g_1, g_2, \ldots, g_{i-1}, \ldots, g_{n-1}$ comes from S if the subgroup membership problem of H in G is intractable. Since $\mathcal{U}$ possesses the trapdoor, he can determine whether or not g lies in H. By Proposition 1, g lies in H if and only if $x_{i-1} = 0$. Therefore, $\mathcal{U}$ can obtain the ith bit x_{i-1}. This simple model illustrates the idea of using the subgroup membership problem, but the communication complexity is still large. We need the trick by [6] to reduce the communication complexity.

3.2 Scheme

Step 0 The user $\mathcal{U}$ inputs 1^k to the instance generator $\mathcal{IG}$ and then gets a pair (G, H) of groups and the trapdoor for the subgroup membership problem of H in G, where k is the security parameter and every element of G is represented by a binary sequence of length k. We assume the subgroup membership assumption of H in G. The group G is shared by both $\mathcal{DB}$ and $\mathcal{U}$. On the other hand, $\mathcal{U}$ keeps the trapdoor information for the subgroup membership problem of H secret. Computations of both $\mathcal{DB}$ and $\mathcal{U}$ are performed in the group G. Let X be the database managed by $\mathcal{DB}$. We suppose that $X = x_0x_1x_2\cdots x_{n-1}$, where $x_i \in \{0, 1\}$, and that $n = t^l$, where t, l are positive integers.

Step 1 $\mathcal{U}$ computes a query Query(i) for his desired bit x_{i-1}, where $1 \le i \le n$, in the following manner. First, $\mathcal{U}$ computes the t-adic expansion of i. Let $i = \alpha_0$. Then the t-adic expansion of i is $\beta_l\beta_{l-1}\cdots\beta_2\beta_1$, where

$$\begin{array}{rl}
\alpha_0 = \alpha_1 t + \beta_1 & 0 \le \alpha_0 \le t^{l-1} - 1, \text{ and } \ 0 \le \beta_1 \le t-1 \\
\alpha_1 = \alpha_2 t + \beta_2 & 0 \le \alpha_1 \le t^{l-2} - 1, \text{ and } \ 0 \le \beta_2 \le t-1 \\
\alpha_2 = \alpha_3 t + \beta_3 & 0 \le \alpha_2 \le t^{l-3} - 1, \text{ and } \ 0 \le \beta_3 \le t-1 \\
 & \cdots\cdots \\
\alpha_{l-2} = \alpha_{l-1} t + \beta_{l-1} & 0 \le \alpha_{l-2} \le t - 1, \text{ and } \ 0 \le \beta_{l-1} \le t-1 \\
 & 0 \le \alpha_{l-1} = \beta_l \le t-1 \text{ and } \alpha_l = 0 \ .
\end{array} \tag{3.1}$$

For each u $(1 \leq u \leq l)$, $\mathcal{U}$ chooses uniformly and randomly $t-1$ elements $g_{(u,0)}$, $g_{(u,1)}$, $\ldots$, $g_{(u,\beta_u-1)}$, $g_{(u,\beta_u+1)}$, $\ldots$, $g_{(u,t-1)}$ from H. He also chooses uniformly and randomly $g_{(u,\beta_u)}$ from $S = G \setminus H$. $\mathcal{U}$ defines $Q(u)$ by

$$\left(g_{(u,0)}, g_{(u,1)}, \cdots, g_{(u,\beta_u-1)}, g_{(u,\beta_u)}, g_{(u,\beta_u+1)}, \cdots, g_{(u,t-1)}\right) , \tag{3.2}$$

that is, $Q(u)$ is a sequence of group elements of G such that the β_uth component is uniformly and randomly chosen from $S = G \setminus H$ and the others are uniformly and randomly chosen from H. Then, $Q(1)$, $Q(2)$, $\ldots$, $Q(l)$ comprise a query (denoted by Query(i)) for the ith bit x_{i-1} of X, and $\mathcal{U}$ sends Query(i) to $\mathcal{DB}$. Since each $Q(u)$ consists of t group elements from G, $Q(u)$ is represented by $k \times t$ bits. Thus, Query(i) consists of $k \times t \times l$ bits.

Step 2 Receiving Query(i), $\mathcal{DB}$ constructs child databases recursively from the original database X. We regard X as the $t^{l-1} \times t$ binary matrix

$$D(0, \lambda) = \begin{pmatrix} x_0 & x_1 & x_2 & \cdots & x_{t-1} \\ x_t & x_{t+1} & x_{t+2} & \cdots & x_{2t-1} \\ & & \cdots & & \\ x_{t^l-t} & x_{t^l-t+1} & \cdots & \cdots & x_{t^l-1} \end{pmatrix} ,$$

where λ denotes the empty sequence in $\{0, 1, 2, \ldots, k-1\}^*$. We note that the target bit x_{i-1} is the (α_1, β_1) entry of $D(0, \lambda)$ (α_1 and β_1 are obtained in (3.1)). Denote it by Target($D(0, \lambda)$).

We recursively define child databases $D(u, s)$, where $1 \leq u \leq l$ and $s \in \{0, 1, 2, \ldots, k-1\}^u$. Suppose that we have defined the databases $D(u, s)$ and their target bits Target($D(u, s)$) and $s \in \{0, 1, 2, \ldots, k-1\}^u$ for $0 \leq u < l-1$. Then we define the databases $D(u+1, s0)$, $D(u+1, s1)$, $\ldots$, $D(u+1, s(k-1))$.

The database $D(u, s)$ is a binary sequence of length t^{l-u}. We regard $D(u, s)$ as a $t^{l-u-1} \times t$ binary matrix. Suppose that

$$D(u, s) = \begin{pmatrix} y_0 & y_1 & y_2 & \cdots & y_{t-1} \\ y_t & y_{t+1} & y_{t+2} & \cdots & y_{2t-1} \\ & & \cdots & & \\ y_{t^{l-u}-t} & y_{t^{l-u}-t+1} & \cdots & \cdots & y_{t^{l-u}-1} \end{pmatrix} .$$

We now construct k child databases, $D(u+1, s0)$, $D(u+1, s1)$, $\ldots$, $D(u+1, s(k-1))$.

Recall that $Q(u)$ consists of t group elements $g_{(u,0)}, g_{(u,1)}, \cdots, g_{(u,t-1)}$ in G (defined in (3.2)). We define a group element g_v for each row $v = 0, 1, 2, \ldots, t^{l-u-1} - 1$ as follows. We set

$$f_{(v,w)} = \begin{cases} g_{(u,w)} & \text{if} \quad D(u, s)(v, w) = 1 \\ 1 & \text{if} \quad D(u, s)(v, w) = 0 \end{cases} , \tag{3.3}$$

where $D(u, s)(v, w)$ denotes the (v, w) entry of $D(u, s)$. Then we set

$$f_{D(u,s),v} = \prod_{w=0,1,2,\ldots,t-1} f_{(v,w)} \tag{3.4}$$

for each row $v = 0, 1, 2, \ldots, t^{l-u-1} - 1$. We note that the group element $f_{D(u,s),v}$ $(0 \leq v \leq t^{l-u-1} - 1)$ is of size k, and that $f_{D(u,s),v} \in H$ if and only if $D(u,s)(v, \beta_u) = 0$ by Proposition 1. The rth child database $D(u+1, sr)$ $(0 \leq r \leq k-1)$ is defined to be the sequence consisting of $g_0(r), g_1(r), \ldots, g_{t^{l-u-1}-1}(r)$, where $g_v(r)$ denotes the rth bit of the representation of $f_{D(u,s),v}$. Hence, we have the following matrix equation:

$$\begin{pmatrix} f_{D(u,s),0} \\ f_{D(u,s),1} \\ \cdots \\ f_{D(u,s),t^{l-u-1}-1} \end{pmatrix} = \big(D(u+1, s0) \cdots\ D(u+1, s(k-1))\big) \tag{3.5}$$

where each $f_{D(u,s),v}$ is a row vector and each $D(u+1, sr)$ is a column vector. Thus, $D(u+1, sr)$ is a binary sequence of length t^{l-u-1}. We regard it as a $t^{l-u-2} \times t$ binary matrix. Then the target bit for it (denoted by Target$(D(u+1, sr)))$ is defined to be the $(\alpha_{u+1}, \beta_{u+1})$ entry of $D(u+1, sr)$ for every $r \in \{0, 1, \ldots, k-1\}$ (α_{u+1} and β_{u+1} are obtained in (3.1)).

Step 3 In the last stage of constructing child databases, $\mathcal{DB}$ obtains k^{t-1} databases $D(l-1, s)$ $(s \in \{1, 2, \ldots, k\}^{t-1})$. Note that each $D(l-1, s)$ contains t bits. We regard $D(l-1, s)$ as a $1 \times t$ matrix. For each $D(l-1, s)$, we define a group element $A(s)$ as follows. First, we define

$$f_{(0,w)} = \begin{cases} g_{(u,w)} & \text{if} \quad D(l-1, s)(0, w) = 1 \\ 1 & \text{if} \quad D(l-1, s)(0, w) = 0 \ . \end{cases}$$

Then, we set $f_{D(l-1,s),0} = \prod_{w=0,1,2,\ldots,t-1} f_{(0,w)} = A(s)$. The group element $A(s)$ is of size k for every $s \in \{0, 1, 2, \ldots, k-1\}^{t-1}$. Then the group elements $A(s)$ $(s \in \{0, 1, \ldots, k-1\}^{t-1})$ form the answer Answer(Query(i)) to the query Query(i), and $\mathcal{DB}$ sends Answer(Query(i)) to $\mathcal{U}$.

Step 4 $\mathcal{U}$ receives Answer(Query(i)) consisting of $A(s)$, where $s \in \{o, 1, \ldots, k-1\}^{t-1}$. $\mathcal{U}$ can retrieve the target bit $x_i = \text{Target}(D_{(0,\lambda)})$ in polynomial time in k, n. In fact, the following holds in general.

Theorem 2. *For every database $D_{(u,s)}$, where $0 \leq u \leq l-2$ and $s \in \{1, 2, \ldots, k\}^u$, $\mathcal{U}$ can compute* Target$(D_{(u,s)})$ *in polynomial time in n, k if* Target$(D_{(u+1,s0)})$, Target$(D_{(u+1,s1)})$, $\ldots$, Target$(D_{(u+1,s(k-1))})$ *are given.*

Proof. Suppose that we have the information

$$\text{Target}(D_{(u+1,s0)}),\ \text{Target}(D_{(u+1,s1)}),\ \ldots,\ \text{Target}(D_{(u+1,s(k-1))}) \ .$$

Recall that $\mathcal{U}$ knows the trapdoor for the subgroup membership problem of the subgroup H and the secret information that $g_{(u,\beta_u)} \in S = G \setminus H$ and $g_{(u,0)}, g_{(u,1)}, \ldots, g_{(u,\beta_u-1)}, g_{(u,\beta_u+1)}, \ldots, g_{(u,t-1)} \in H$, where $Q(u) = (g_{(u,0)}, g_{(u,1)}, \cdots, g_{(u,\beta_u-1)}, g_{(u,\beta_u)}, g_{(u,\beta_u+1)}, \cdots, g_{(u,t-1)})$. Note that the number β_u is a private information for $\mathcal{U}$. Recall that Target$(D_{(u,s)})$ is the (α_u, β_u) entry of the database $D_{(u,s)}$. By the computation of $\mathcal{DB}$ in (3.4), we have $f_{D(u,s),\beta_u} = \prod_{w=0,1,2,\ldots,t-1} f_{(\beta_u,w)}$. By Proposition 1 and (3.3), $f_{D(u,s),\beta_u} \in H$ if and only if (α_u, β_u) entry is 0. Moreover, $f_{D(u,s),\alpha_u}$ is the α_uth row of the matrix

$$\big(D(u+1, s0)\ D(u+1, s1)\ D(u+1, s2) \cdots\ D(u+1, s(k-1))\big)$$

by (3.5). Note that α_uth bit in the database $D(u+1, sr)$ is the $(\alpha_{u+1}, \beta_{u+1})$ entry of the matrix $D(u+1, sr)$ for every $r = 0, 1, \ldots, k-1$. On the other hand, the $(\alpha_{u+1}, \beta_{u+1})$ entry of $D(u+1, sr)$ is Target$(D_{(u+1,sr)})$. Since $\mathcal{U}$ knows Target$(D_{(u+1,s0)})$, Target$(D_{(u+1,s1)})$, ..., Target$(D_{(u+1,s(k-1))})$, he can retrieve $f_{D(u,s),\alpha_u}$. After retrieving $f_{D(u,s),\alpha_u}$, $\mathcal{U}$ checks whether or not $f_{D(u,s),\alpha_u}$ is in H. Therefore, $\mathcal{U}$ can retrieve Target$(D_{(u,s)})$ in polynomial time. □

3.3 Privacy

In the proposed scheme, the query Query(i) consists of $Q(1)$, $Q(2)$, ..., $Q(l)$, and each $Q(u)$ consists of

$$\left(g_{(u,0)},\ g_{(u,1)},\ \cdots,\ g_{(u,\beta_u-1)},\ g_{(u,\beta_u)},\ g_{(u,\beta_u+1)},\ \cdots,\ g_{(u,t-1)}\right) ,$$

where one of the components is chosen uniformly and randomly from $S = G \setminus H$ and the others are chosen uniformly and randomly from H. The privacy is assured by the inequality

$$|\mathbf{Prob}(C_k(\text{Query}(i)) = 1) - \mathbf{Prob}(C_k(\text{Query}(j)) = 1)| < \sigma ,$$

where $\sigma = \frac{1}{(\text{Max}(k,n))^c}$, given in (1.1). This is exactly the general pattern indistinguishability assumption in (2.4) if n is bounded by a polynomial in k. Hence, the privacy of the proposed scheme is guaranteed by the subgroup membership assumption by Theorem 1.

3.4 Communication Complexity

In the first step, $\mathcal{U}$ sends Query$(i) = (Q(1), Q(2), \ldots, Q(l))$. Each $Q(u)$ consists of t group elements in G. Since every element in G is represented by a binary sequence of length k, the total bits sent in this stage is $l \times t \times k$. In the second step, $\mathcal{DB}$ sends Answer(Query(i)) consisting of k^{l-1} group elements in G. Therefore, the total bits sent in this stage is $k^{l-1} \times k = k^l$. Consequently, the communication complexity is $ltk + k^l = ln^{\frac{1}{l}}k + k^l$. Suppose that $k = n^c$ and $l = O(\frac{\log n}{\log k})$. Then we have $l = \sqrt{\frac{\log n}{\log k}}$, and $k^l = (2^{\log k})^l = 2^{l \log k} = 2^{\sqrt{\log n \log k}} = 2^{\sqrt{\log nc \log n}} = n^{\sqrt{c}}$. On the other hand, we have $ltk + k^l = k^l(lk+1) < k^l k^l = (k^l)^2$. Hence, we have $ltk + k^l = (n^{\sqrt{c}})^2$. It follows that the communication complexity is $O(n^c)$.

3.5 Small Example

For good understanding of the scheme, we illustrate with a small example. Suppose that the database is given by $X = x_0x_1x_2x_3x_4x_5x_6x_7x_8 = 110010101$. The size of the database is $9 = 3^2$ in this example. Let $t = 3$. The X is identified with the $t \times t$ matrix $D(0,\lambda) = \begin{pmatrix} 1\,1\,0 \\ 0\,1\,0 \\ 1\,0\,1 \end{pmatrix}$. Suppose that the user $\mathcal{U}$ wants to read x_7. He computes 3-adic expansion of 7 as in (3.1). Then we have $7 = 2 \times 3 + 1$, $2 = 0 \times 3 + 2$. Hence, we have $\alpha_0 = 7$, $\alpha_1 = 2$, $\alpha_2 = 0$, $\beta_1 = 1$, $\beta_2 = 2$. Then $\mathcal{U}$ chooses uniformly and randomly 3 group elements $g_{(0,0)}, g_{(0,1)}, g_{(0,2)}$, where $g_{(0,0)}$ and $g_{(0,2)}$ belong to H and $g_{(0,1)}$ belongs to $S = G \setminus H$ since $\beta_1 = 1$. Next, $\mathcal{U}$ chooses uniformly and randomly 3 group elements $g_{(1,0)}, g_{(1,1)}, g_{(1,2)}$, where $g_{(1,0)}$ and $g_{(1,1)}$ belong to H and $g_{(1,2)}$ belongs to $S = G \setminus H$ since $\beta_2 = 2$. The query Query(7) consists of $Q(1) = (g_{(0,0)}, g_{(0,1)}, g_{(0,2)})$ and $Q(2) = (g_{(1,0)}, g_{(1,1)}, g_{(1,2)})$. It is sent to $\mathcal{DB}$ by $\mathcal{U}$. Let us assume that every element of G is represented by a binary sequence of length 4. $\mathcal{DB}$ receives Query(7) and then performs the following computation. Using (3.3), he sets $f_{(0,0)} = g_{(0,0)}$, $f_{(0,1)} = g_{(0,1)}$, $f_{(0,2)} = 1$, $f_{(1,0)} = 1$, $f_{(1,1)} = g_{(2,1)}$, $f_{(1,2)} = 1$, $f_{(2,0)} = g_{(2,0)}$, $f_{(2,1)} = 1$, $f_{(2,2)} = g_{(2,2)}$ corresponding to the database. Then, using (3.4), he computes $f_{D(0,\lambda),0} = f_{(0,0)}f_{(0,1)}f_{(0,2)} = g_{(0,0)}g_{(0,1)}$, $f_{D(0,\lambda),1} = f_{(1,0)}f_{(1,1)}f_{(1,2)} = g_{(0,1)}$, $f_{D(0,\lambda),2} = f_{(2,0)}f_{(2,1)}f_{(2,2)} = g_{(0,0)}g_{(0,2)}$. Suppose that $f_{D(0,\lambda),0}, f_{D(0,\lambda),1}, f_{D(0,\lambda),2}$ are represented by $0110, 1010, 1101$, respectively. It is helpful to see it in the matrix form as follows:

$$\begin{pmatrix} f_{D(0,\lambda),0} \\ f_{D(0,\lambda),1} \\ f_{D(0,\lambda),2} \end{pmatrix} = \begin{pmatrix} 0\,1\,1\,0 \\ 1\,0\,1\,0 \\ 1\,1\,0\,1 \end{pmatrix}.$$

$\mathcal{DB}$ constructs four child databases $D_{1,0}, D_{1,1}, D_{1,2}, D_{1,3}$, where $D(1,0) = (011)^T, D(1,1) = (101)^T, D(1,2) = (110)^T, D(1,3) = (001)^T$. Note that we have $\begin{pmatrix} f_{D(0,\lambda),0} \\ f_{D(0,\lambda),1} \\ f_{D(0,\lambda),2} \end{pmatrix} = \big(D(1,0)\ D(1,1)\ D(1,2) \cdots D(1,3)\big)$. For each database, using $Q(2) = (g_{(1,0)}, g_{(1,1)}, g_{(1,2)})$, $\mathcal{DB}$ compute a group element. For $D(1,0) = (011)^T$, he computes $A(0) = g_{(1,1)}g_{(1,2)}$. For $D(1,1) = (101)^T$, he computes $A(1) = g_{(1,0)}g_{(1,2)}$. For $D(1,2) = (110)^T$, he computes $A(2) = g_{(1,0)}g_{(1,1)}$. For $D(1,3) = (001)^T$, he computes $A(3) = g_{(1,2)}$. He sends $(A(0), A(1), A(2), A(3))$ to $\mathcal{U}$ as Answer(Query(7)) to $\mathcal{U}$. Receiving Answer(Query(7)), $\mathcal{U}$ checks the memberships of $A(0)$, $A(1)$, $A(2)$ and $A(3)$ in H. Since $\mathcal{U}$ keeps the trapdoor for the subgroup membership problem for H, he can check the memberships of these elements in polynomial time. He finds that $A(0), A(1), A(3) \in H$ and $A(2) \in S$ and concludes that $f_{D(0,\lambda),2} = 1101$. Checking the membership of $f_{D(0,\lambda),2}$ in H, he finds that $x_7 = 0$.

References

1. Ambainis,A.: Upper Bound on the Communication Complexity of Private Information Retrieval, Automata, Languages and Programming. Lecture Notes in Computer Science, Vol. 1256. Springer-Verlag, (1997) 401–407
2. Cachin,C., Micali,S., Stadler,M.: Computationally Private Information Retrieval with Polylogarithmic Communication, Advances in Cryptology. Lecture Notes in Computer Science, Vol. 1592. Springer-Verlag, (1999) 402–414
3. Chor,B., Goldreich,O., Kushilevitz,E., Sudan,M.: Private Information Retrieval, IEEE Symposium on Foundations of Computer Science. (1995) 41–50
4. Chor,B., Gilboa,N.: Computationally Private Information Retrieval ACM Symposium on Theory of Computing. (1997) 304–313
5. Goldwasser,S., Micali,S.: Probabilistic Encryption, J. Computer and System Science **28** (1984) 270–299
6. Kushilevitz,E., Ostrovsky,R.: Replication Is not Needed: Single Database, Computationally-private Information Retrieval, IEEE Symposium on Foundations of Computer Science. (1997) 364–373
7. Kurosawa,K, Tsujii,S.: A General Method to Construct Public Key Residue Cryptosystems, Transactions of the IEICE **E-73**, (1990) 1068–1072
8. Naccache,D., Stern,J.: A New Public-key Cryptosystem, Advances in Cryptology. Lecture Notes in Computer Science, Vol. 1233. Springer-Verlag, (1997) 27–36
9. Ostrofsky,R., Shoup,V.: Private Information Storage, ACM Symposium on Theory of Computing. (1997) 294–303
10. Okamoto,T., Uchiyama,S.: A New Public-key Cryptosystem as Secure as Factoring, Advances in Cryptology. Lecture Notes in Computer Science, Vol. 1403. Springer-Verlag, (1998) 308–318
11. Paillier,P.: Public-key Cryptosystems Based on Composite Degree Residuosity Classes, Advances in Cryptology. Lecture Notes in Computer Science, Vol. 1592. Springer-Verlag, (1999) 223–238
12. Saito,T., Koshiba,T., Yamamura,A.: The Decision Diffie-Hellman assumption and the Quadratic Residuosity Assumption, IEICE Transactions on Fundamentals of Electronics (1) **E84-A**, (2001) 165–171

A Practical English Auction with One-Time Registration

Kazumasa Omote and Atsuko Miyaji

School of Information Science
Japan Advanced Institute of Science and Technology,
Asahidai 1-1, Tatsunokuchi, Nomi, Ishikawa, 923-1292 JAPAN
{omote, miyaji}@jaist.ac.jp

Abstract. An English auction is the most familiar type of auctions. Generally, an electronic auction has mainly two entities, the registration manager(RM) who treats the registration of bidders, and the auction manager(AM) who holds auctions. Before starting an auction, a bidder who wants to participate in English auction is registered to RM with her/his information. An electronic English auction protocol should satisfy the following nine properties, (a)Anonymity, (b)Traceability, (c)No framing, (d)Unforgeability, (e)Fairness, (f)Verifiability, (g)Unlikability among different auctions, (h)Linkability in an auction, and (i)Efficiency of bidding. Furthermore from the practical point of view we add two properties (j)One-time registration and (k)Easy revocation. A group signature is adapted to an English auction in order to satisfy (a), (b), and (f)[18]. However such a direct adoption suffers from the most critical drawbacks of efficiency in group signatures. In this paper we propose more realistic electronic English auction scheme, which satisfies all of these properties. Four notable features of our scheme are:
(1) both of bidding and verification of bids are done quite efficiently by introducing a bulletin board,
(2) anonymity for RM, AM and any participant can be realized to plural auctions by only one-time registration,
(3) RM can easily revoke a bidder, and
(4) nobody can impersonate any bidder.

keywords: anonymity, signature of knowledge, bulletin board, easy revocation

1 Introduction

1.1 Background

An English auction is the most familiar type of auctions. In an English auction, each bidder offers the higher price one by one, and finally a bidder who offers the highest price gets a good. An English auction is used on the Internet as well as the real world. In an English auction through the Internet, it is important to spoil the collusion of bidders, because Internet makes the formation of ring members much easier[15]. Therefore anonymity plays an important role in spoiling the collusion of bidders. In an English auction, all bid information is published. Therefore the competition principle well works and any bidder easily knows her/his market price position. This is why an English auction is the most familiar style of auctions. In this paper, we investigate an electronic English auction.

Generally, an electronic auction has mainly two entities, the registration manager(RM) who treats the registration of bidders, and the auction manager(AM) who holds auctions. Before starting an auction, a bidder who wants to participate in English auction is registered to RM with her/his information. As for studies about an electronic auction, a sealed-bid auction has been often investigated[19, 11, 21, 22, 24, 14, 10, 3, 17, 13]. A sealed-bid auction is that each bidder secretly submits a bid to AM only once, and a bidder who offers the highest price gets the goods. A sealed-bid auction has two problems, (1)the competition principle does not work well; (2)a winning bid may be much higher price than market one.

In the case of sealed-bid auction, any canceled bid does not affect the valid bidders. However, in the case of English auction, any bid does not allow to be canceled. If a bid can be canceled in an English auction, the highest bid may be insignificant. Therefore, in an electronic English auction, it is the most important to satisfy the following two properties, (a)Anonymity and (b)Traceablitiy. Although any bidder can participate anonymously, it is necessary to identify a winner after a bidding. This means that every bid placed in an English auction must be verified maintaining the bid anonymity. Addition to the above two properties, an electronic English auction should satisfy the following nine properties:

(a) **Anonymity:** nobody can identify a bidder from her/his signature on a bid.
(b) **Traceability:** A winner cannot deny that she/he submitted the winning bid after the winner decision procedure.
(c) **No framing:** nobody can impersonate a certain bidder.
(d) **Unforgeability:** nobody can forge a bid with a valid signature.
(e) **Fairness:** all bids should be fairly dealt with.
(f) **Verifiability:** anybody can verify a signature on a bid and can confirm whether the bidder is valid or not.
(g) **Unlinkablity among different actions:** nobody can link the same bidder's bids among plural auctions.
(h) **Linkability in an auction:** anybody can link which bids are placed by the same bidder and knows how many times a bidder places a bid in an auction.
(i) **Efficiency of bidding:** the computation and communication amount in both bidding and verifying a bid is practical.

1.2 Related Works

Only a few studies on English auction[18, 23, 15, 16] have been reported as long as we know. On the other hand, many studies on a sealed-bid auction[19, 11, 21, 22, 24, 14, 10, 3, 17, 13] have been proposed because it can realize fairness more easily than English auction of public auction. These studies[15, 16] do not concern with the security aspect of public auctions but describe those different methods. [23] also proposed an electronic English auction using reverse hash chains[20] as a bid, which is similar to multiple sealed-bid biddings in order to satisfy fairness. When a bidder participates in an auction, it has two advantages that a valid bidder can place a bid many times by using only one-time signature and that bidder fairness is satisfied for a non-trusted center. However, in this protocol, the following two problems exist:

1. Anonymity for AM is not satisfied after each bidding since AM knows the bidder's identity.
2. The bidding points are set up discretely. For n bidding points, it is necessary for a bidder to compute hash functions n times. Apparently each bidder cannot place a bid as she/he likes.

[18] proposed an electronic English auction, which keeps a bidder privacy using a slightly modified group signature scheme[7, 5, 6]. So this protocol suffers from the following drawbacks of group signature schemes. In their scheme, a group manager (GM) works as AM and a group member corresponds to a bidder.

The first problem, which is the most serious, is rather complicated signature generation and verification procedure. In [7, 5, 6, 1], a membership certificate is used to reduce the data size of public group key[4]: only a group member has the certificate issued by GM. When each member generates a signature on this certificate and a bid, she/he is required the proof of the knowledge. However the proof of the knowledge needs enormous modular multiplication. In an English auction, signature generation or verification corresponds to bidding or verification of bids respectively, both of which are required in each bidding. In an electronic auction, reducing the computation amount of both signature generation and verification are much concerned compared with reducing the group public key size. Therefore we realize an electronic English auction with both fairly simple bidding and verifying procedures by introducing a bulletin board, which is usually used in putting each bid. The important feature of a bulletin board is that anybody can check the correctness of the board easily. In our protocol, the computation amount for both bidding and verifying a bid can be reduced by using a feature of bulletin board.

The second problem is anonymity. The group signature does not satisfy anonymity for GM at all since GM has a special authority. However, in an electronic auction, any bidder surely desires that nobody knows how much she/he wants to buy goods. Therefore, we need a technique of Escrow scheme[12], in which introduces Identity Escrow Agency(EA) in order to enhance anonymity for GM. This scheme realizes the perfect separability between GM and EA: only EA can identify a user by himself. This means that, in a sense, anonymity for EA is not satisfied at all. In an electronic auction, it is required that neither AM(GM) nor RM(EA) can identify the bidder from a signature on a bid, but cooperation of both parties can certainly recover the identity. In our protocol, neither only AM nor RM identify any bidder but RM can open the signature on a bid with the help of AM and can identify the bidder. Even if a winner is identified in an auction, the winner bidder can participate in the next auction maintaining enough anonymity for both RM and AM satisfied.

The third problem is that it is rather difficult to revoke a bidder since a membership certificate is distributed to each bidder indicated in [2]. Revocation of bidder is necessary when a bidder wants to withdraw from an auction or RM wants to revoke a certain bidder. Therefore RM should be able to revoke a bidder easily. In our protocol, a revocation of bidder is done easily by using a bulletin board: just remove her/him on it.

1.3 Our Result

We propose a practical anonymous electronic English auction protocol satisfying the above eleven properties, (a)Anonymity (b)Traceability, (c)No framing, (d)Unforgeability, (e)Fairness, (f)Verifiability, (g)Unlikability among different auctions, (h)Linkability in an auction, (i)Efficiency of bidding, (j)One-time registration, and (k)Easy revocation. Our protocol satisfies both (a) and (b) simultaneously by using a combination of both the signature of the knowledge and two kinds of bulletin boards. In particular, the computation amount of both bidding and verifying each bid is fairly reduced by introducing a bulletin board. In our protocol, there are two managers RM and AM. RM manages the correspondence of bidder identity to public key, and can identify a winner or a faulty bidder with the help of AM. When a certain bidder is identified after a winner decision procedure or later disputes, AM has only to request RM to identify the bidder.

Notable features of our scheme are as follows:

– both of bidding and verification of bids are done quite efficiently by introducing a bulletin board.
– Any bidder can participate in plural auctions by only one-time registration. Even if a bidder is identified as a winner, she/he can participate in the next auction without repeating registration, maintaining anonymity for RM, AM, and any bidder.
– RM can easily revoke a bidder.
– Even if both RM and AM collude, they cannot impersonate any bidder.

The remaining of this paper is organized as follows. Section 2 summarizes a basic scheme[18] using group signature. Section 3 describes our protocol in detail. Section 4 considers fairness. Section 5 investigates the properties of our scheme.

2 Related Work

Here we summarize a previous English auction scheme[18] which uses an idea of group signature.

2.1 Group Signature

The concept of group signature was introduced by Chaum and van Heyst[8]. Group signature allows any member to sign on behalf of a group and keeps the member identity secret. The work[7] is the first efficient group signature schemes in that the size of both group's public key and of signatures are independent of the number of group members and that a group's public key remains unchanged if a new member is added to a group. Later, group signature schemes with improved performance and better flexibility are proposed in [5, 12, 6, 1]. [18] is based on these group signatures [7, 12, 5, 6].

In an English auction, GM works as AM and a group member corresponds to a bidder. When a bidder places a bid, she/he generates a group signature on a bid. The validity of signature can be verified easily by any participant using a group public key, but any participant does not know who places the bid.

2.2 Previous Scheme

Setup: AM computes an RSA modulus n, where n is the product of two primes, an RSA key pair (e, d), a cyclic group $G = \langle g \rangle$ of order n over the finite field $\mathbf{Z}_p$ for a prime p, an element $a \in \mathbf{Z}_n^*$ that is of the order $\phi(n)/4$, and an upper bound λ on the length of the secret keys: a revocation manager chooses $h \in G$ with order n, computes ElGamal-encryption key pair $(\rho,\ Y_R(= h^\rho)) \in \mathbf{Z}_n \times G$, and sets a constant $b \neq 1$. The group public key is $\mathcal{Y} = (n, e, G, g, a, \lambda, h, Y_R)$. AM's secret key is d and a revocation manager's secret key is ρ.

Registration: Alice randomly generates a secret key $x \in \{0, \cdots, 2^\lambda - 1\}$ and sends the value $y = a^x \pmod n$ and $z = g^y$ to AM; AM returns $v = (y + b)^d \pmod n$. Note that AM cannot see the value of x.

Bidding Phase: In order to put a bid m with her signature, she computes the following values $(d_1, d_2, V_1, V_2, V_3)$:
- $\tilde{g} = g^r$ and $\tilde{z} = \tilde{g}^y$ for $r \in_R \mathbf{Z}_n$;
- $d_1 = Y_R^u g^y$ and $d_2 = h^u$ for $u \in_R \mathbf{Z}_n$;
- $V_1 = SK[(\gamma, \delta) : \tilde{z} = \tilde{g}^\gamma \wedge d_2 = h^\delta \wedge d_1 = Y_R^\delta g^\gamma](m)$;
- $V_2 = SK[(\beta) : \tilde{z} = \tilde{g}^{a^\beta}](V_1)$;
- $V_3 := SK[(\alpha) : \tilde{z}\tilde{g}^b = \tilde{g}^{\alpha^e}](V_2)$

The notation of a signature of knowledge $(x_1, \cdots, x_k)$ on a message m is as follows:

$$SK[(x_1, \cdots, x_k) : z_1 = f_1(x_1, \cdots, x_k) \wedge \cdots \wedge z_\ell = f_\ell(x_1, \cdots, x_k)](m).$$

The secrets $x_1, \cdots, x_k$ satisfy all ℓ statements: $z_1 = f_1(x_1, \cdots, x_k), \cdots, z_\ell = f_\ell(x_1, \cdots, x_k)$. Assume that computing the discrete logarithm, the double discrete logarithms and the e-th root of the discrete logarithm is infeasible. The concrete algorithm for these signatures is referred to [7]. Alice's group signature consists of a set of $(d_1, d_2, V_1, V_2, V_3)$. If the signature (V_1, V_2, V_3) is valid, anyone confirms that (d_1, d_2) is an encryption of z by using ElGamal encryption function with a revocation manager's public key Y_R, and that Alice knows her secret key x and her membership certificate v.

Winner Decision Phase: A revocation manager decrypts (d_1, d_2) using his secret key ρ and identifies a member Alice from z since he knows the correspondence of z to member's identity.

In this scheme, the signature V_3 is slightly modified using a verifiable group signature sharing scheme in order to satisfy anonymity of bidder.

2.3 Undesirable Properties of the Scheme

In this scheme, there exist some problems as follows.

Efficiency: In applying a group signature to an electronic auction, it is necessary to generate or verify a signature on each bid. A signature generation or verification corresponds to bidding or verification of bids respectively, both of which are required in each bidding. However the computation amount for both signature generation and verification is rather large. Therefore it is not realistic to apply directly a group signature to an electronic auction, which requires a real-time operation.

Revocation of Bidder: In an Electronic auction, a revocation of bidder is frequently conducted when a bidder wants to withdraw from an auction or AM wants to revoke a certain bidder. So revocation-procedure should not be complicated. However, in the previous scheme, it is rather difficult to revoke a bidder since a membership certificate has been distributed to each bidder indicated in [2]. Of course, a bidder does not want to publish her/his secret key in revocation procedure. A revocation manager has to keep her/his z in a black list to revoke a certain bidder. Therefore a revocation manager can discover the unacceptable signature generated by a revoked bidder.

3 Our Protocol

In this section, we propose a practical electronic English auction.

3.1 Entities

The entities of our scheme consist of the registration manager(RM), the auction manager(AM) and a bidder($\mathcal{B}$), where each role of AM or RM is slightly different from that of previous scheme. The role of each entity is as follows:

- **RM:**
  - guarantees the correspondence of a bidder to bidder's registration key.
  - works like Identity Escrow Agency and identifies a certain bidder when AM requests.
- **AM:**
  - sponsors several auctions.
  - controls the number of a bidder's bidding in an auction.
- **Bidder($\mathcal{B}$):**
  - participates in an auction that AM holds.

3.2 Notations

Notations are defined as follows:

p, q	: two large primes ($q \mid p-1$)
g	: an element $g \in \mathbf{Z}_p$ with order q
I	: the number of bidders
i	: the index of bidders ($i = 1, \cdots, I$)
$\mathcal{B}_i$	: bidder i
x_i	: a secret key of $\mathcal{B}_i$ ($x_i \in_R \mathbf{Z}_q$)
y_i	: a public key of $\mathcal{B}_i$ ($y_i = g^{x_i}$) (Note that a public key is used as a registration key, and does not reveal bidder's identity.)
r_i	: AM's random number for $\mathcal{B}_i$ ($r_i \in_R \mathbf{Z}_q$)
t_i	: a random number of $\mathcal{B}_i$ ($t_i \in_R \mathbf{Z}_q$)
T_i	: an auction key for $\mathcal{B}_i$
k	: the index of auctions ($k \geq 1$)
Y_{AM}	: AM's public key ($Y_{AM} = g^{\rho},\ \rho \in_R \mathbf{Z}_q$)
Enc	: $Enc(key, data)$ is a secret key encryption function by using a secret key, key, (Note that a cipher text is uniquely determined.)
Enc^j	: $Enc^j(key, data)$ is j-times encryption by using the same key, $Enc(key, Enc(key, \cdots))$.

3.3 Procedure

Initialization: RM publishes p, q and g. AM computes a pair of public key and secret key, ($Y_{AM},\ \rho$) using g and publishes Y_{AM}.

Bidder Registration: A bidder Alice ($\mathcal{B}_j$) registers her registration key in the following steps:

1. Alice chooses her secret key x_j and computes her registration key $y_j = g^{x_j} \pmod p$;
2. She chooses a random number t_j, named *ticket*. She uses her ticket in order to find her auction key T_j on AM's bulletin board. Note that she can also find her auction key T_j without using her ticket by checking that $y_j^{r_j} \overset{?}{=} (g^{r_j})^{x_j}$;
3. She sends $\{y_j, t_j\}$ to RM as her registration key, registers her identity and proves that she knows the discrete logarithm x_j of y_j to the base g by showing V_1,
$$V_1 = SK[(\alpha) : y_j = g^{\alpha}](m_R),$$
where m_R is a message published by RM;
4. When RM accepts that Alice knows the discrete logarithm, he publishes her registration key $\{y_j, t_j\}$ on his bulletin board, while RM keeps her name secretly(Figure 1).

Although Alice's name is not published at RM's bulletin board, she can easily confirm whether there exists her registration key on that board or not. Here a registration key works also as a pseudonym. We assume that RM cannot make up a secret key of a certain bidder.

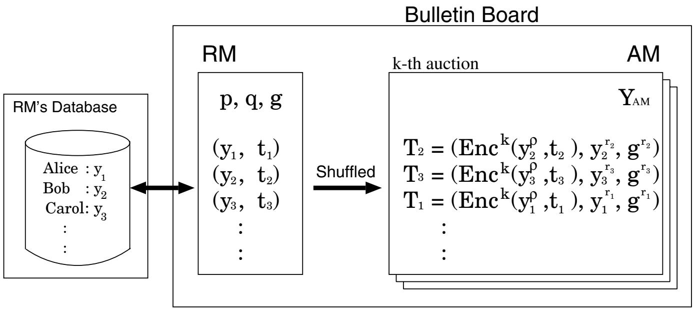


Fig. 1. Bulletin Board

AM's Setup: When a vendor requests AM to hold an auction, AM conducts the following procedure. For simplicity, here a bidder $\mathcal{B}_i$ participates in the k-th auction.

1. AM computes a shared secret key y_i^ρ with each bidder $\mathcal{B}_i$ ($y_i^\rho = Y_{AM}^{x_i}$) by using Diffie-Hellman key-distribution[9].
2. AM generates the random numbers $\{r_1, \cdots, r_I\} \in_R \mathbf{Z}_q$ for each bidder published on RM's bulletin board and keeps the numbers $\{r_1, \cdots, r_I\}$ secret.
3. AM encrypts t_i to $Enc^k(y_i^\rho, t_i) = Enc(y_i^\rho, Enc^{k-1}(y_i^\rho, t_i))$ in the k-time Enc by using a shared key y_i^ρ.
4. AM computes the following auction key T_i for $\mathcal{B}_i$ using $\mathcal{B}_i$'s public key y_i on RM's bulletin board:

$$T_i = (Enc^k(y_i^\rho, t_i),\ {y_i}^{r_i},\ g^{r_i}).$$

5. AM publishes the shuffled auction key T_i of all bidders on his bulletin board.

AM's setup has the following properties:

(A) Nobody except for AM can know the correspondence of y_i to T_i since y_i is concealed to $y_i^{r_i}$ in T_i and shuffled by AM;
(B) AM cannot identify a bidder since he does not know the correspondence of B_i's identity to y_i.

Bidding: Alice who wants to participate in the k-th auction can easily find her bidding key T_j in $\{T_1, \cdots, T_I\}$ published by AM because she knows the value $Enc^k(y_j^\rho, t_j)$ in advance by using $y_j^\rho = Y_{AM}^{x_j}$. Alice generates the signature of knowledge V_2 using both ${y_j}^{r_j}$ and g^{r_j} in T_j.

When she places a bid, she sends the following bid information $(m_j, y_j^{r_j}, g^{r_j}, V_2)$ to AM.

- a bid m_j $\quad (m_j = \textit{auction ID}||\text{bid value})$
- $y_j^{r_j}$ and g^{r_j} (published by AM)
- $V_2 = SK[\alpha : y_j^{r_j} = (g^{r_j})^{\alpha})](m_j)$

Here V_2 implies that $\mathcal{B}_j$ knows the value of $\alpha = x_j$ if V_2 is valid signature. Furthermore both $y_j^{r_j}$ and g^{r_j} also work as a kind of certificate.

Verifiability: We assume that AM checks the validity of the signature V_2 on each bid. Of course, anybody can check the validity. If the signature V_2 is invalid signature, AM removes the bid with V_2

Checking the validity of the signature of knowledge V_2, anybody can confirm that a bidder knows surely her/his secret key. Furthermore anybody can accept that the signer is one of the bidders if the values ${y_j}^{r_j}$ and g^{r_j} in V_2 are published on AM's bulletin board.

Winner Decision: Let Alice's bid m_j be a winning bid. AM proves to RM that the public information ${y_j}^{r_j}$ added to a winning bid m_j corresponds to the registration key y_j by sending RM the value r_j^{-1}. Note that only RM can identify Alice as a winner for the first time, and that AM cannot identify a winner Alice in this winner decision.

Winner Announcement: Only the entity RM knows the winner's identity after the winner decision procedure. This means that all participants including AM cannot identify a winner but can confirm the validity of a winner. If RM informs a vendor of winner's identity after the winner decision procedure, nobody except for RM can identify a winner. Therefore anonymity of a winner is satisfied without changing her/his registration key managed by RM.

Generally, there is a problem of bidder collusion to form a ring. However, in our protocol, even if a winner Alice offers her values of bid, any bidder cannot identify her at the next auction, because AM changes r_j at every auction. Unlikability among different auctions holds in our protocol.

4 Fairness of Bidder

Fairness of bidder in an electronic auction means that any bid is fairly accepted by AM. Generally, in an electronic English auction, fairness of bidder depends on AM. There are two unfairness acts by AM:

1. AM repudiates any higher bids than a certain value.
2. AM repudiates any bidding by a certain bidder.

In order to satisfy the fairness of above 1, a bidder has to conceal a bid value for AM. As for the above 2, a bidder has to place a bid anonymously. Our protocol keeps the fairness of case 2 since bidding is done anonymously but is vulnerable to the case 1 since any value on bids is revealed. In order to avoid the case 2, we may use *non-repudiation protocol* [25, 26].

4.1 Outline of Non-repudiation Protocol

The non-repudiation protocol is that Alice sends a message to Bob and then Bob cannot repudiate a receipt of the message from Alice. We summarize the basic procedure.

1. Alice encrypts a message m into C and sends it to Bob.
2. He sends his signature $S_{Bob}(C)$ back to her after receiving C.
3. She sends the decryption key K of C to him after receiving $S_{Bob}(C)$.

Note that if Bob repudiates K after the deadline, she deposits K in TTP (Alice cannot know whether Bob repudiates K or the network between Alice and Bob is broken down). TTP publishes K using public directory service as soon as TTP receives it. Bob cannot deny receiving a message m if the network between Bob and TTP is not permanently broken down.

4.2 Bidding Procedure with Non-repudiation

Fairness of bidder is realized by introducing an idea of non-repudiation protocol as above. Non-repudiation protocol is added to a bidding procedure of our protocol. Alice and Bob correspond to a bidder $\mathcal{B}_i$ and AM, respectively. RM also plays a role of TTP. In our protocol, both RM and AM use a public bulletin board. A bid m is placed as follows:

1. AM cannot know each bid value since the bid information is encrypted by a bidder.
2. AM publishes $\mathcal{B}_i$'s signature $S_{\mathcal{B}_i}(C)$ in AM's bulletin board instead of returning it since AM does not know who is $\mathcal{B}_i$.
3. Even if AM repudiates a receipt of decryption key K from a bidder, he cannot deny getting bid information since RM publishes K in his bulletin board.

5 Consideration

5.1 Features

We discuss the following eleven properties in our protocol.

(a) **Anonymity:** nobody including either RM or AM can identify a bidder from her/his signature on a bid. Furthermore AM cannot identify a bidder though RM can identify a bidder with the help of AM. More importantly any bidder can anonymously participate in another auction by using the same registration key even if she/he has been identified once.
(b) **Traceability:** RM can open a signature on a bid with the help of AM and can identify the bidder. So a winner cannot deny that she/he has submitted the winning bid after the winner decision procedure.
(c) **No framing:** this will be discussed in chapter 5.2.

(d) **Unforgeability:** nobody can forge a bid with a signature since anybody cannot generate a valid signature using the registration key in AM's bulletin board.
(e) **Fairness:** our scheme has fairness of bidder if it applies non-repudiation protocol to bidding. Otherwise AM may decide on which bids to accept. However AM's misbehavior turn out by a bulletin board. A bidder can point out that AM does not accept her/his bid. Furthermore AM cannot identify a bidder from bids. Therefore such a dishonest act may not have an influence on electronic auction.
(f) **Verifiability:** anybody can verify the signature V_2 on a bid. Furthermore anybody can confirm whether a bidder is valid or not by checking her/his registration key in AM's bulletin board.
(g) **Unlikability among different auctions:** each bidding key generated by AM is different among each auction since AM's secret information r_i, which is different in every auction, is embedded in $y_i^{r_i}$ and g^{r_i} with a bid. So nobody except for AM can link two signatures among different auctions. Although AM can link all bids of $\mathcal{B}_j$ in all auctions, AM cannot get an identity of $\mathcal{B}_j$ except for collusion with RM.
(h) **Linkability in an auction:** a real auction has a linkability in an auction. An auction becomes active by a certain aggressive bidder who always places a higher bid. Anybody knows how many times a bidder places bids in an auction from the signature since a bidder uses both $y_i^{r_i}$ and g^{r_i} as a part of bidding information in an auction.
(i) **Efficiency of bidding:** this will be discussed in chapter 5.3.
(j) **One-time registration:** any bidder can take part in plural auctions as a valid bidder in one-time registration of registration key, maintaining anonymity for RM, AM, and any bidder.
(k) **Easy revocation:** this will be discussed in chapter 5.4.

5.2 No Framing

Here we discuss the security against framing attacks such that an entity impersonates another valid bidder.

Security against Collusion of RM and AM: Even if both RM and AM are colluded, they cannot impersonate a bidder in the following reason. In our protocol, in order to impersonate a bidder, RM and AM must show that they know the bidder's secret key x_i, which is the discrete logarithm of a part of the bidding key in AM's bulletin board. However only a bidder $\mathcal{B}_i$ knows x_i, so they cannot impersonate a bidder.

Security against RM, AM, Other Bidders, and Outsiders: In the same reason as the above, RM, AM, other bidders and outsiders cannot also impersonate another valid bidder.

Table 1. Performance for a bidder

	#Modular multiplications (1024-bit)			Communication amount (kbit)	
	Registration	Bidding	Verification	Registration	Bidding
[18]	1,500	218,600	206,700	1.3	7.6
Our scheme	480	240 (560)[1]	320 (560)	1.3	2.4

5.3 Performance

In this section, we compare our scheme with the previous scheme[18] in section 2 from the viewpoints of computation and communication amount for a bidder, which are shown in Table 2. For simplicity we estimate the computation amount by the number of 1024-bit modular multiplication and let the system parameters be $e = 3, |n| = |p| = 1024, |q| = \lambda = 160, |\mathcal{H}| = 160$ and a security parameter $\ell = 64$[7]. From table 2, we see that the computation amount for a bidder is much reduced compared with the previous scheme. In particular, it is the most important to reduce the modular multiplication amount of bidding and verification, because both are conducted many times in an auction. The computation amount in our scheme is dramatically reduced by introducing two kinds of bulletin boards and an auction key. AM has only to check whether the signature V_2 is valid or not and whether there exists an auction key is in his bulletin board or not when a bidder places a bid. In this way the computation amount of both bidding and verification are reduced. Therefore our scheme can practically realize an electronic auction.

5.4 Easy Revocation

In an Electronic auction, a revocation of bidder can be frequently conducted when a bidder wants to withdraw from an auction or RM wants to revoke a certain bidder. Therefore it should be simple and easy. Furthermore the bidding history is kept secret if a bidder is revoked. In the previous scheme, it is rather difficult to revoke a bidder since a membership certificate is distributed to each bidder. In our protocol, it is easy to revoke a bidder: RM has only to delete a bidder from RM's bulletin board. Note that AM requests RM to revoke a certain bidder informing her/his information(e.g. the value r_i) or that a bidder requests RM to revoke herself/himself.

6 Conclusion

We have proposed a practical electronic auction which satisfies (a)Anonymity, (b)Traceability, (c)No framing, (d)Unforgeability, (e)Fairness, (f)Verifiability, (g)Unlikability among different auctions, (h)Linkability in an auction, (i)Efficiency of bidding, (j)One-time registration, and (k)Easy revocation. Five notable features are:

[1] This value in brackets shows the case that fairness of bidder is realized.

(1) both of bidding and verification of bids are done quite efficiently by introducing a bulletin board,
(2) anonymity for RM, AM and any participant can be realized to plural auctions by only one-time registration,
(3) RM can easily revoke a bidder,
(4) nobody can impersonate any bidder, and
(5) Fairness of bidder can be realized.

References

[1] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A Practical and Provably Secure Coalition-Registant Group Signature Scheme. In *Advances in Cryptology – CRYPTO2000*, pages 255–270, 2000.

[2] G. Ateniese and G. Tsudik. Some Open Issues and New Directions in Group Signatures. In *Proceedings of Financial Cryptography'99*, pages 196–211, 1999.

[3] C. Cachin. Efficient Private Bidding and Auctions with an Oblivious Third Party. In *Proceedings of 6th ACM Conference on Computer and Communications Security*, pages 120–127, 1999.

[4] J. Camenisch. Efficient and Generalized group signatures. In *Advances in Cryptology – EUROCRYPT'97*, pages 465–479, 1997.

[5] J. Camenisch and M. Michels. A Group Signature Scheme with Improved Efficiency. In *Advances in Cryptology – ASIACRYPT'98*, pages 160–174, 1998.

[6] J. Camenisch and M. Michels. Separability and Efficiency for Generic Group Signature Schemes. In *Advances in Cryptology – CRYPTO'99*, pages 106–121, 1999.

[7] J. Camenisch and M. Stadler. Efficient Group Signature Schemes for Large Groups. In *Advances in Cryptology – CRYPTO'97*, pages 410–424, 1997.

[8] D. Chaum and E. van Heyst. Group signatures. In *Advances in Cryptology – EUROCRYPT'91*, pages 257–265, 1991.

[9] W. Diffie and M. Hellman. New direction in cryptography. *IEEE Transactions on Information Theory*, pages 644–654, November 1976.

[10] M. Franklin and M. Reiter. The design and implementation of a secure auction service. *IEEE Transactions on Software Engineering*, 5:302–312, 1996.

[11] H. Kikuchi, M. Harkavy, and D. Tyger. Multi-round anonymous auction protocols. In *Proceedings of the First IEEE Workshop on Dependable and Real-Time E-Commerce Systems*, pages 62–69, 1998.

[12] J. Kilian and E. Petrank. Identity Escrow. In *Advances in Cryptology – CRYPTO'98*, pages 169–185, 1998.

[13] K. Kobayashi, H. Morita, K. Suzuki, and M. Hakuta. Efficient Sealed-bid Auction by Using One-way Functions. *IEICE Trans. Fundamentals*, E84-A(1):289–294, 2001.

[14] M. Kudo. Secure electronic sealed-bid auction protocol with public key cryptography. *IEICE Trans. Fundamentals*, E81-A(1):20–27, 1998.

[15] M. Kumar and S.Feldman. Internet Auctions. In *Proceedings of the Third USENIX Workshop on Electronic Commerce*, pages 49–60, 1998.

[16] T. Mullen and M.Wellman. The auction manager: Market middleware for large-scale electronic commerce. In *Proceedings of the Third USENIX Workshop on Electronic Commerce*, pages 49–60, 1998.

[17] M. Naor, B. Pinkas, and R. Sumner. Privacy Preserving Auctions and Mechanism Design. In *Proceedings of ACM Workshop on Electronic Commerce*, pages 120–127, 1999.

[18] K. Nguyen and J. Traoré. An Online Public Auction Protocol Protecting Bidder Privacy. In *Information Security and Privacy (ACISP2000)*, pages 427–442, 2000.

[19] K. Omote and A. Miyaji. An anonymous auction protocol with a single non-trusted center using binary trees. In *Proceedings of ISW2000*, pages 108–120, 2000.

[20] R.L.Rivest and A.Shamir. Payword and micromint: Two simple micropayment schemes. In *Proceedings of Security Protocols*, pages 69–87, 1996.

[21] K. Sako. An Auction Protocol Which Hides Bids of Losers. In *Proceedings of PKC2000*, pages 422–432, 2000.

[22] K. Sakurai and S. Miyazaki. An anonymous electronic bidding protocol based on a new convertible group signature scheme. In *Proceedings of ACISP2000*, pages 385–399, 2000.

[23] Stuart G. Stubblebine and Paul F. Syverson. Fair On-line Auctions Without Special Trusted Parties. In *Proceedings of Financial Cryptography'99*, pages 230–240, 1999.

[24] K. Suzuki, K. Kobayashi, and H. Morita. Efficient sealed-bid auction using hash chain. In *Proceedings of ICISC 2000*, pages 189–197, 2000.

[25] J. Zhou and D. Gollmann. A fair non-repudiation protocol. In *Proceedings of 1996 IEEE Symposium on Security and Privacy*, pages 55–61, 1996.

[26] J. Zhou and D. Gollmann. An efficient non-repudiation protocol. In *Proceedings of the 10th Computer Security Foundations Workshop (PCSFW). IEEE Computer Society Press*, 1997.

A User Authentication Scheme with Identity and Location Privacy

Shouichi Hirose and Susumu Yoshida

Graduate School of Informatics, Kyoto University, Kyoto, 606–8501 JAPAN
hirose@i.kyoto-u.ac.jp

Abstract. The rapid growth of wireless systems provides us with mobility. In mobile environments, authentication of a user and confidentiality of his identity and location are two major security issues, which seem incompatible with each other. In this manuscript, we propose a user authentication scheme with identity and location privacy. This scheme is an interactive protocol based on public key cryptosystems. In the proposed scheme, to prove his authenticity, a user utilizes a digital signature scheme based on a problem with a random self-reducible relation such as the square root modulo a composite number problem and the discrete logarithm problem. We also define the security requirements for user authentication with identity and location privacy, impersonation-freeness and anonymity, against active attacks, and prove that the proposed scheme satisfies them assuming the security of the cryptographic schemes used in the scheme. Furthermore, we show that we can construct authenticated key agreement schemes by applying the proposed scheme to some existing authenticated key agreement schemes.

1 Introduction

The rapid growth of wireless systems provides us with mobility. In mobile environments, the service area of a service provider is divided into domains, each of which is covered by a network operator. Each user moves around the domains and gets some services through the network operator of the visiting domain. In such a situation, user authentication is necessary for accounting. It is also necessary to ensure the identity and location privacy, that is, users' identity and location information should not be disclosed to unauthorized entities. However, these two requirements seem incompatible with each other.

In this manuscript, we present a solution to this problem, user authentication with identity and location privacy (ILP), in a public key setting. We assume that user authentication is achieved with a challenge-and-response scheme based on a digital signature scheme.

Suppose that a user uses the same public key and proves his authenticity to different network operators. In this case, the location privacy is not provided. These network operators are able to track the user by colluding with each other, even if the user uses some pseudonym and hides his real ID. To avoid the tracking, the user has to generate and use his temporary public key whenever he proves his authenticity. We present an efficient scheme which achieves this goal. As is in the typical situation, it is assumed that the service provider keeps the public keys of its users. Each of the corresponding secret keys is known only to the user. We call these public/secret keys original public/secret keys of the user.

To prove his authenticity to the network operator he is visiting, each user utilizes a digital signature scheme based on a problem with a random self-reducible relation such as the square root modulo a composite number problem or the discrete logarithm problem. Thus, the proposed scheme can be constructed with practical digital signature schemes such as the ElGamal scheme [4], the Fiat-Shamir scheme [5], the Schnorr scheme [12], the Pointcheval-Stern scheme [10] and so on.

When a user proves his authenticity, he computes a pair of temporal public and secret keys from his original key pair and a random seed by utilizing the random self-reducibility of the problem the digital signature scheme is based on. The user signs for the challenge from the network operator he is visiting and the temporal public key by using the temporal secret key. The user's ID and the random seed are sent to his service provider through the network operator after being encrypted so as to be recovered only by his service provider. The validity of the temporal public key is guaranteed by his service provider that is able to compute it from the user's original public key it keeps and the random seed.

In this manuscript, we also initiate the study of provable security of user authentication schemes with ILP. We first define two security requirements for user authentication with ILP, impersonation-freeness and anonymity, both of which are against active attacks. Then we prove that the proposed scheme satisfies these requirements assuming the security of the cryptographic schemes, public key encryption schemes and digital signature schemes, used in the scheme.

Furthermore, we show that we can construct authenticated key agreement schemes by applying the proposed scheme to existing authenticated key agreement schemes such as those in [2, 7]. As an example, we show an authenticated key agreement scheme constructed by applying the proposed scheme to the station-to-station protocol [2].

1.1 Related Works

User authentication schemes have been already incorporated in the specifications of cellular phone systems such as GSM [11] and CDPD [1, 11]. These schemes, however, do not provide anonymity of users.

Molva, Samfat and Tsudik [8] presented an efficient user authentication scheme with anonymity based on KryptoKnight [9]. Their scheme is constructed with private key cryptosystems. Thus, their approach is quite different from ours. In addition, they focused on user authentication and did not fully discuss the anonymity. Their security analysis of the anonymity was quite informal.

Herzberg, Krawczyk and Tsudik [6] discussed the anonymity problem in mobile environments. They reviewed the existing approaches and proposed several potential solutions based on private key cryptosystems or public key cryptosystems. As a scheme based on private key cryptosystems, they presented the same scheme as the one in [8]. For public key based schemes, they focused on providing the framework rather than proposing some concrete schemes. Furthermore, their security analysis is also quite informal.

In [3], for third generation mobile telecommunications systems, several user authentication schemes with ILP are proposed. One of the protocols is based on public key encryption schemes and uses the service provider on-line. Furthermore, it is mentioned that the scheme needs a temporary user public key encryption transformation. However, no solution is provided for this problem.

1.2 Organization of This Manuscript

This manuscript is organized as follows. In Section 2, random self-reducibility, which is the basis of the proposed scheme, is reviewed. The proposed user authentication scheme with ILP is presented in Section 3. As an example, the proposed scheme constructed with the Schnorr scheme is presented in the same section. In Section 4, two security requirements for user authentication with ILP, impersonation-freeness and anonymity, are defined and it is proved that the proposed scheme satisfies these requirements. Efficiency of the proposed scheme is also discussed in this section. In Section 5, it is shown that authenticated key agreement protocols can be constructed by applying the proposed scheme to existing authenticated key agreement schemes.

2 Random Self-Reduciblity

In this section, random self-reducibility, on which the proposed scheme is based, is reviewed [13].

Let $\mathcal{N}$ be a countable infinite set. For any $N \in \mathcal{N}$, let A_N, B_N be finite sets and $R_N \subseteq A_N \times B_N$ be a relation.

$\mathrm{dom}(R_N) \overset{\mathrm{def}}{=} \{a \in A_N \mid (a,b) \in R_N \text{ for some } b \in B_N\}$ is called the domain of R_N, and $R_N(a) \overset{\mathrm{def}}{=} \{b \mid (a,b) \in R_N\}$ is called the image of $a \in A_N$.

Let R be the relation $\{((N,a),b) \mid N \in \mathcal{N} \text{ and } (a,b) \in R_N\}$. R is called a random self-reducible relation, if there exists an algorithm M_1 for R, which has the following properties:

M_1 is an algorithm which outputs $a' \in \mathrm{dom}(R_N)$ which satisfies the following conditions on input $N \in \mathcal{N}$, $a \in A_N$, $r \in \{0,1\}^*$ in $|N|^{O(1)}$ steps.

- If each bit of r is selected randomly, uniformly and independently, then a' is uniformly distributed over $\mathrm{dom}(R_N)$.
- There exists an algorithm $\bar{\mathsf{M}}_1$ which outputs some $b \in R_N(a)$ on input $N \in \mathcal{N}$, $a \in A_N$, $\bar{r}$, $b' \in R_N(a')$ in $|N|^{O(1)}$ steps, where $\bar{r}$ is a finite prefix of r used by $\mathsf{M}_1(N,a,r)$.
- There exists an algorithm M_2 which outputs some $b' \in R_N(a')$ on input N, a, r, $b \in R_N(a)$ in $|N|^{O(1)}$ steps. Furthermore, if each bit of r is selected randomly, uniformly and independently, then b' is uniformly distributed over $R_N(a')$.

For example, the square root modulo a composite number problem, the discete logarithm problem and the graph isomorphism problem are the problems with random self-reducible relations. A random self-reducible relation of the discrete logarithm problem is as follows.

For a positive integer k, let $\mathbf{Z}_k = \{0, 1, 2, \ldots, k-1\}$ and $\mathbf{Z}_k^* = \{x \mid x \in \mathbf{Z}_k, \gcd(x, k) = 1\}$. Let $\mathcal{N} = \{(p, g) \mid p$ is prime, $g \in \mathbf{Z}_p^*$ and g is a primitive element$\}$, and $R_{(p,g)} = \{(a, b) \mid (a, b) \in \mathbf{Z}_p^* \times \mathbf{Z}_{p-1}, a = g^b \bmod p\}$. Since g is a primitive element, $\mathrm{dom}(R_{(p,g)}) = \mathbf{Z}_p^*$. For any $a \in \mathrm{dom}(R_{(p,g)})$ and $r \in \mathbf{Z}_{p-1}$, let $a' = ag^r \bmod p$. Then, if r is randomly selected, then a' is uniformly distributed over $\mathrm{dom}(R_{(p,g)})$. Furthermore, since $b' = b + r \bmod p-1$, b or b' is easily obtained from r and b' or r and b, respectively.

In this manuscript, we present a user authentication scheme which utilizes a digital signature scheme based on a problem with a random self-reducible relation. For this kind of digital signature scheme with a random self-reducible relation $R = \{((N, a), b) \mid N \in \mathcal{N}$ and $(a, b) \in R_N\}$, N is a public key shared by all users, a is a public key of a user and b is a secret key corresponding to a.

3 The Proposed Scheme

3.1 Overview

In this section, we present an overview of the proposed user authentication scheme with ILP, which is based on public key cryptosystems.

We assume that there exists a service provider and that its service area is divided into domains. We also assume that each domain is covered by a network operator. A network operator checks the authenticity of a user who makes a request in the domain it covers.

Each user has his own pair of a public key and a secret key, which is used for proving his authenticity. We call the public/secret keys of a user the original public/secret keys of the user. The service provider maintains the original public key of each user. Notice that the original secret key of a user is known only to the user.

The proposed scheme enables a user to prove his authenticity to the network operator he is visiting without disclosing his ID and location with the aid of the service provider. The user proves his authenticity with a challenge-and-response protocol based on a digital signature scheme.

When a user proves his authenticity, he computes a pair of temporal public and secret keys from his original key pair and a random seed by utilizing the random self-reducibility of the problem the digital signature scheme is based on. The user signs for the challenge from the network operator he is visiting and the temporal public key by using the temporal secret key. The user's ID and the random seed are sent to the service provider through the network operator after being encrypted so as to be recovered only by the service provider. The validity of the temporal public key is guaranteed by the service provider who is able to compute it from the user's original public key it keeps and the random seed.

3.2 Description of the Scheme

Before describing the proposed scheme in detail, we introduce some notations.

Let U be a user, SP be the service provider and NO be a network operator U is visiting. Let $I_{\mathrm{U}}, I_{\mathrm{SP}}, I_{\mathrm{NO}}$ be the ID's of U, SP and NO, respectively. ID's are assumed to be binary strings.

Let Sig be a signing algorithm used by the users to prove their authenticity. The subscript of Sig is the key used when signing. The digital signature scheme with this signing algorithm is assumed to be based on some problem with random self-reducibility. For this scheme, let N be a public key shared by all users and let a, b be the original public key and the original secret key of U, respectively. For N, a, b, let M_1 be an algorithm which outputs a temporal public key on input N, a and a random seed r, and M_2 be an algorithm which outputs a temporal secret key on input N, a, b and a random seed r. These algorithms are publicly available.

Let E_{SP} be an encryption algorithm of some public key encryption scheme. The decryption key corresponding to E_{SP} is kept secret only by SP. Let $S_{\mathrm{SP}}, S_{\mathrm{NO}}$ be signing algorithms of SP and NO, respectively. It is not necessary for E_{SP}, S_{SP}, S_{NO} to be based on the problems with random self-reducibility.

The proposed user authentication scheme with ILP:

1. U randomly selects r and α, and sends $c, I_{\mathrm{SP}}, \alpha$ to NO, where $c = E_{\mathrm{SP}}(I_{\mathrm{U}}, r)$. He also computes a temporal public key $a_{\mathrm{tmp}} = \mathsf{M}_1(N, a, r)$ and a temporal secret key $b_{\mathrm{tmp}} = \mathsf{M}_2(N, a, b, r)$.
2. After receiving $c, I_{\mathrm{SP}}, \alpha$ from U, NO sends c, I_{NO} to SP. He also randomly selects β, computes $S_{\mathrm{NO}}(\alpha, \beta)$, and sends $\beta, S_{\mathrm{NO}}(\alpha, \beta)$ to U.
3. After receiving c, I_{NO} from NO, SP recovers I_{U}, r from c. If the plaintext obtained from c is invalid, then SP terminates the execution. Otherwise, he computes the temporal public key of U, $a_{\mathrm{tmp}} = \mathsf{M}_1(N, a, r)$ and sends $S_{\mathrm{SP}}(c, a_{\mathrm{tmp}})$, a_{tmp} to NO.
   After receiving $\beta, S_{\mathrm{NO}}(\alpha, \beta)$ from NO, U verifies the validity of $S_{\mathrm{NO}}(\alpha, \beta)$. If it is invalid, then he terminates the execution. Otherwise, he computes a signature of β, a_{tmp} with the temporal secret key b_{tmp} and sends $Sig_{b_{\mathrm{tmp}}}(\beta, a_{\mathrm{tmp}})$ to NO.
4. NO receives $S_{\mathrm{SP}}(c, a_{\mathrm{tmp}}), a_{\mathrm{tmp}}$ from SP and verifies the validity of $S_{\mathrm{SP}}(c, a_{\mathrm{tmp}})$. If it is invalid, then NO terminates the execution. Otherwise, after receiving $Sig_{b_{\mathrm{tmp}}}(\beta, a_{\mathrm{tmp}})$ from U, NO verifies its validity with the temporal public key a_{tmp}. NO accepts U if and only if it is valid.

The above scheme is also shown in Fig. 1.

3.3 Example

The proposed scheme can be constructed with digital signature schemes based on problems with random self-reducibility: the ElGamal scheme [4], the Fiat-Shamir scheme [5], the Schnorr scheme [12], the Pointcheval-Stern scheme [10] and so on.

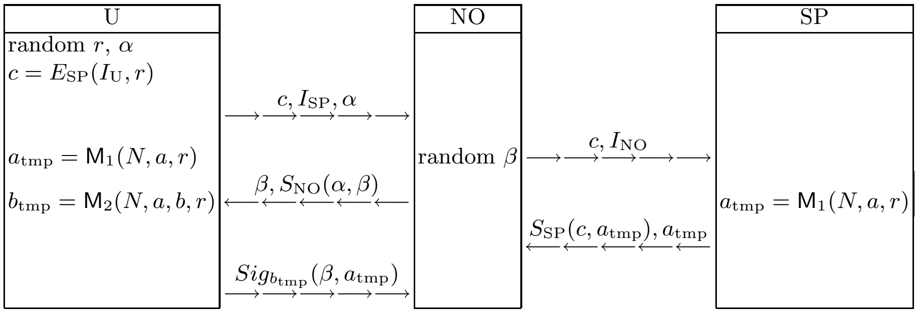

U
random r, α
$c = E_{\mathrm{SP}}(I_{\mathrm{U}}, r)$
$c, I_{\mathrm{SP}}, \alpha$
$a_{\mathrm{tmp}} = \mathsf{M}_1(N, a, r)$
$b_{\mathrm{tmp}} = \mathsf{M}_2(N, a, b, r)$
$\beta, S_{\mathrm{NO}}(\alpha, \beta)$
$Sig_{b_{\mathrm{tmp}}}(\beta, a_{\mathrm{tmp}})$
NO
random β
c, I_{NO}
$S_{\mathrm{SP}}(c, a_{\mathrm{tmp}}), a_{\mathrm{tmp}}$
SP
$a_{\mathrm{tmp}} = \mathsf{M}_1(N, a, r)$

Fig. 1. The proposed scheme. U is a user and NO is a network operator U is visiting. SP is the service provider.

In the following, we present an example with the Schnorr scheme. Let p and q be primes and q be a divisor of $p-1$. Let g be an element of $\mathbf{Z}_p^*$, whose order is q. Let p, q, g be the public keys of the Schnorr scheme shared by all users. Let $b \in \mathbf{Z}_q$ be the original secret key of the user U and $a = g^b \bmod p$ be the original public key of U. Let h be a collision-free hash function. Furthermore, α and β are assumed to be binary strings in $\{0,1\}^\kappa$, where κ is appropriately determined.

An example of the proposed scheme constructed with the Schnorr scheme:

1. U randomly selects $r \in \mathbf{Z}_q$ and $\alpha \in \{0,1\}^\kappa$, and sends $c, I_{\mathrm{SP}}, \alpha$ to NO, where $c = E_{\mathrm{SP}}(I_{\mathrm{U}}, r)$. He also computes a temporal public key $a_{\mathrm{tmp}} = a\, g^r \bmod p$ and a temporal secret key $b_{\mathrm{tmp}} = b + r \bmod q$.
2. After receiving $c, I_{\mathrm{SP}}, \alpha$ from U, NO sends c, I_{NO} to SP. He also randomly selects $\beta \in \{0,1\}^\kappa$, computes $S_{\mathrm{NO}}(\alpha, \beta)$, and sends $\beta, S_{\mathrm{NO}}(\alpha, \beta)$ to U.
3. After receiving c, I_{NO} from NO, SP recovers I_{U}, r from c. If the plaintext obtained from c is invalid, then SP terminates the execution. Otherwise, he computes U's temporal public key $a_{\mathrm{tmp}} = a\, g^r \bmod p$ and sends $S_{\mathrm{SP}}(c, a_{\mathrm{tmp}})$, a_{tmp} to NO.

   After receiving $\beta, S_{\mathrm{NO}}(\alpha, \beta)$ from NO, U verifies the validity of $S_{\mathrm{NO}}(\alpha, \beta)$. If it is invalid, then he terminates the execution. Otherwise, he computes a signature of β, a_{tmp} with the temporal secret key b_{tmp}. That is, he randomly selects $x \in \mathbf{Z}_q$ and computes $y = g^x \bmod p$, $e = h(y, \beta, a_{\mathrm{tmp}})$, and $w = x - e\, b_{\mathrm{tmp}} \bmod q$. Then U sends e, w to NO.
4. NO receives $S_{\mathrm{SP}}(c, a_{\mathrm{tmp}}), a_{\mathrm{tmp}}$ from SP and verifies the validity of $S_{\mathrm{SP}}(c, a_{\mathrm{tmp}})$. If it is invalid, then he terminates the execution. Otherwise, after receiving e, w, he verifies its validity. That is, he computes $z = g^w a_{\mathrm{tmp}}^e \bmod p$ and checks whether $e = h(z, \beta, a_{\mathrm{tmp}})$ or not. NO accepts U if and only if it is valid.

4 Discussions

4.1 Security

In this section, we discuss the security of the proposed user authentication scheme with ILP. We first define two security requirements, impersonation-freeness and anonymity, and then prove that the proposed scheme satisfies these requirements.

In the proposed scheme, the service provider should be trusted because it guarantees the validity of temporal public keys of the users. A dishonest service provider is able to impersonate any user by randomly selecting a temporal secret key, generating the corresponding temporal public key and stating that the temporal public key is a valid key for the user. Thus, the service provider is assumed honest.

Let ℓ be the security parameter. Let $\mathcal{U}, \mathcal{O}$ be the set of the users and the set of the network operators, respectively. Both of $|\mathcal{U}|$ and $|\mathcal{O}|$ are assumed to be bounded by some polynomial of ℓ.

We consider the security against active attacks. A malicious adversary A is a probabilistic algorithm and is assumed to operate in two successive phases: the observation phase and the trial phase. In the observation phase, A can fully control the network. A can arbitrarily select U's in $\mathcal{U}$ and NO's in $\mathcal{O}$, and make them execute the protocol with SP. A can modify, replay or not deliver the messages exchanged during the executions of the protocol. Furthermore, A can corrupt U's and NO's. A can obtain the secret keys of corrupted U's and NO's and also control them arbitrarily. On the other hand, A cannot corrupt SP. A cannot obtain the secret keys of SP: the decryption key for E_{SP} nor the signing key for S_{SP}. A cannot control SP, neither.

We define the security requirements, impersonation-freeness and anonymity, against the active adversary A.

impersonation-freeness. A user authentication scheme with ILP is impersonation-free if no polynomially bounded adversary A succeeds with non-negligible probability in impersonating an uncorrupted user in the observation phase.

A's behavior in the observation phase is described as above. At the end of the observation phase, A selects a network operator NO.

In the trial phase, A executes the protocol with NO and SP, and tries to impersonate some user. Notice that A may not determine the user he tries to impersonate.

anonymity. To define anonymity, we consider the most advantageous scenario for an adversary. A's behavior in the observation phase is described as above. At the end of the observation phase, the adversary A selects two users $\mathrm{U}_0, \mathrm{U}_1$ and a network operator NO. $\mathrm{U}_0, \mathrm{U}_1$ and NO may be corrupted in the observation phase.

In the trial phase, the user $\mathrm{U}_i \in \{\mathrm{U}_0, \mathrm{U}_1\}$, who is randomly selected, executes the protocol with NO and SP. We assume that A does not know which one of

$\mathrm{U}_0, \mathrm{U}_1$ is selected. We also assume that $\mathrm{U}_0, \mathrm{U}_1$ are not corrupted and that NO is corrupted. After the execution of this protocol, A outputs the value of i that he guesses.

An user authentication scheme with ILP satisfies anonymity if $|\Pr[A\text{'s guess is correct}] - 1/2|$ is negligible.

In the definition of impersonation-freeness, impersonation of a network operator NO is not mentioned. If the signature scheme of NO, S_{NO} is existentially unforgeable against the adaptive chosen-message attack, then it is impossible to impersonate NO.

The definition of anonymity implies the anonymity against passive eavesdroppers though it describes the anonymity against network operators.

Impersonation-Freeness. First, we make an assumption on the unforgeability of the digital signature scheme of the users.

Assumption 1. For the digital signature scheme of the users, there exists a probabilistic polynomial time algorithm which computes the secret key with non-negligible probability by using an algorithm which can forge a signature with the adaptive chosen-message attack as an oracle.

This assumption holds for the digital signature schemes such as the Schnorr scheme [12] and the Pointcheval-Stern scheme [10].

Theorem 1. Suppose that the digital signature scheme of the service provider SP is existentially unforgeable against the adaptive chosen-message attack. If there exists an adversary A who succeeds in impersonation with non-negligible probability, then there exists a probabilistic polynomial time algorithm which is able to compute the original secret key of some user with non-negligible probability in cooperation with SP by using A as an oracle.

Proof. In the observation phase, an adversary A can make the adaptive chosen-message attack on the digital signature scheme of the users and that of SP. At the end of the observation phase, suppose that A selected a network operator NO.

In the trial phase, A executes the protocol with NO and SP, and tries impersonation.

Since the digital signature scheme of SP is assumed to be existentially unforgeable against the adaptive chosen message attack, we do not consider the attack such that A generates a temporal key pair $(a_{\mathrm{tmp}}, b_{\mathrm{tmp}})$ in some way, forges $S_{\mathrm{SP}}(c, a_{\mathrm{tmp}})$, and succeeds in impersonation with b_{tmp}.

During the execution of the protocol with NO and SP, A first sends $c, I_{\mathrm{SP}}, \alpha$ to NO.

Notice that A does not necessarily know the plaintext corresponding to c nor may there exist no valid plaintext corresponding to c. NO then sends c, I_{NO} to SP.

If SP recovers a valid plaintext (I_{U}, r) for some $\mathrm{U} \in \mathcal{U}$, then he computes the temporal public key of U, $a_{\mathrm{tmp}} = \mathsf{M}_1(N, a, r)$ and sends $S_{\mathrm{SP}}(c, a_{\mathrm{tmp}})$, a_{tmp} to NO.

If A succeeds in impersonation with non-negligible probability, then he succeeds in forging signatures for random challenges from NO with non-negligible probability. ¿From Assumption 1, there exists a probabilistic polynomial time algorithm Alg which can compute the temporal secret key b_{tmp} corresponding to a_{tmp} by using A as an oracle. Thus, Alg can compute the original secret key of U in cooperation with SP from $N, a, r, b_{\mathrm{tmp}}$. □

This theorem implies that, for example, the proposed scheme constructed with the Schnorr scheme is impersonation-free if users' public keys are selected so that it is intractable to compute the corresponding secret keys.

Anonymity. To prove anonymity of the proposed scheme, we only consider a random self-reducible relation $R = \{((N, a), b) \mid N \in \mathcal{N} \text{ and } (a, b) \in R_N\}$ which satisfies the following conditions:

- For any $N \in \mathcal{N}$, r is selected from a finite set Q_N, and the random sampling from Q_N is feasible.
- There exists an algorithm which outputs $r \in Q_N$ on input $N \in \mathcal{N}$, $a \in A_N$, $b \in R_N(a)$, $a' \in A_N$, $b' = M_2(N, a, b, r) \in R_N(a')$ in $|N|^{O(1)}$ steps.

These conditions are not restrictive. For example, the random self-reducible relation of the discrete logarithm problem presented in Section 2 satisfies them. It is easy to show that the digital signature schemes such as the ElGamal scheme, the Fiat-Shamir scheme, the Schnorr scheme and the Pointcheval-Stern scheme can be constructed based on the problems with random self-reducibility satisfying the conditions.

We further assume that an adversary A is able to obtain the temporal secret key b_{tmp} that the user uses during the execution in the trial phase.

Theorem 2. If the public key encryption scheme of the service provider SP satisfies the indistinguishability of encryptions against the adaptive chosen-ciphertext attack, then the proposed scheme satisfies the anonymity.

Proof. In the observation phase, since an adversary A can fully control the network, he can apply the adaptive chosen-ciphertext attack on the public key encryption scheme of SP.

Suppose that an adversary A has chosen two users $\mathrm{U}_0, \mathrm{U}_1$ at the end of the observation phase, whose original secret keys, b_0, b_1, are known to A. Furthermore, in the trial phase, suppose that A obtained b_{tmp}, which is used by the user to compute $Sig_{b_{\mathrm{tmp}}}(\beta, a_{\mathrm{tmp}})$ during the execution of the protocol. Then, from the conditions on the random self-reducibility described above, A can compute r_i from $N, a_i, b_i, a_{\mathrm{tmp}}, b_{\mathrm{tmp}}$ for $i = 0, 1$, and the plaintext of the ciphertext c is (I_{U_0}, r_0) or (I_{U_1}, r_1). Since E_{SP} satisfies the indistinguishability of encryptions against the adaptive chosen-ciphertext attack, A's advantage of guessing i over random selection is negligible. □

4.2 Efficiency

In [3], for third generation mobile telecommunications systems, an authentication scheme with ILP based on public key cryptography is shown which uses the service provider on-line. In this section, the proposed scheme is compared with the above scheme in terms of the efficiency, and the advantage of the proposed scheme is made clear. In the following the scheme in [3] is called the 3GS3 scheme.

The communication overhead of the proposed scheme is lower than that of the 3GS3 scheme, while the number of the passes of the proposed scheme is equal to that of the 3GS3 scheme. In the proposed scheme, the network operator NO communicates on-line with the service provider SP in order to receive a temporal public key of U and its certificate. However, NO can send a challenge β to U without waiting for the response from SP. On the other hand, in the 3GS3 scheme, NO can send a challenge to U only after receiving the response from SP.

As is mentioned in Introduction, it is observed in [3] that the 3GS3 scheme needs a temporary user public key encryption transformation. For this problem, the proposed scheme provides a solution based on the random self-reducibility, which can also be applied to the 3GS3 scheme. The solution is more advantageous than the naive solution in that a user need not prove to the service provider that he really generates the temporal public key a_{tmp} that he sends. The naive solution possibly requires the user to prove to the service provider that he really knows the corresponding temporal secret key. On the other hand, the proposed solution guarantees that the one who knows the secret key of $a_{\mathrm{tmp}} = \mathsf{M}_1(N, a, r)$ is the one who knows both the random seed r and the secret key of the public key a.

5 Extension

In this section, we show that we can construct an authenticated key agreement protocol by applying the proposed scheme to an existing authenticated key agreement scheme. The constructed key agreement protocol also provides the ILP. With this key agreement protocol, a user can share a common secret session-key with the network operator without the aid of the service provider, that is, the shared session-key is known only to the user and the network operator he is visiting, not known to the service provider.

To construct an authenticated key agreement protocol, we can apply the proposed scheme to some of the existing protocols, such as those in [2, 7]. As an example, we show the application of the proposed scheme to the station-to-station (STS) protcol [2]. In the following description, let $\langle\cdot\rangle_K$ be a ciphertext obtained by encrypting the plaintext in $\langle\,\rangle$ with the secret key K and some symmetric key encryption function.

An authenticated key agreement scheme constructed by applying the proposed scheme to the STS protocol:

1. U randomly selects $k_{\mathrm{U}} \in \mathbf{Z}_q$, computes $u_{\mathrm{U}} = g^{k_{\mathrm{U}}} \bmod p$ and sends $c, u_{\mathrm{U}}, I_{\mathrm{SP}}$ to NO, where $c = E_{\mathrm{SP}}(I_{\mathrm{U}}, r)$. He also computes the temporal public key $a_{\mathrm{tmp}} = \mathsf{M}_1(N, a, r)$ and the corresponding temporal secret key $b_{\mathrm{tmp}} = \mathsf{M}_2(N, a, b, r)$.
2. NO receives $c, u_{\mathrm{U}}, I_{\mathrm{SP}}$ from U and sends c, I_{NO} to SP. He also randomly selects $k_{\mathrm{NO}} \in \mathbf{Z}_q$, computes $u_{\mathrm{NO}} = g^{k_{\mathrm{NO}}} \bmod p$ and the session-key $K = {u_{\mathrm{U}}}^{k_{\mathrm{NO}}} \bmod p$, and sends $u_{\mathrm{NO}}, S_{\mathrm{NO}}(\langle u_{\mathrm{U}}, u_{\mathrm{NO}}\rangle_K)$ to U.
3. After receiving c, I_{NO} from NO, SP recovers I_{U}, r from c. If the plaintext obtained from c is invalid, then he terminates the execution. Otherwise, he computes the temporal public key of U, $a_{\mathrm{tmp}} = \mathsf{M}_1(N, a, r)$, and sends $S_{\mathrm{SP}}(c, a_{\mathrm{tmp}})$, a_{tmp} to NO.
   After receiving $u_{\mathrm{NO}}, S_{\mathrm{NO}}(\langle u_{\mathrm{U}}, u_{\mathrm{NO}}\rangle_K)$ from NO, U computes the session-key $K = {u_{\mathrm{NO}}}^{k_{\mathrm{U}}} \bmod p$ and verifies the validity of the signature $S_{\mathrm{NO}}(\langle u_{\mathrm{U}}, u_{\mathrm{NO}}\rangle_K)$. If it is invalid, then he terminates the execution. Otherwise, he signs for $\langle u_{\mathrm{NO}}, u_{\mathrm{U}}\rangle_K, a_{\mathrm{tmp}}$ with his temporal secret key b_{tmp} and sends $Sig_{b_{\mathrm{tmp}}}(\langle u_{\mathrm{NO}}, u_{\mathrm{U}}\rangle_K, a_{\mathrm{tmp}})$ to NO.
4. NO receives $S_{\mathrm{SP}}(c, a_{\mathrm{tmp}}), a_{\mathrm{tmp}}$ from SP and verifies the validity of the signature. If it is invalid, then he terminates the execution. Otherwise, after receiving $Sig_{b_{\mathrm{tmp}}}(\langle u_{\mathrm{NO}}, u_{\mathrm{U}}\rangle_K, a_{\mathrm{tmp}})$ from U, he verifies the validity of the signature with the temporal public key a_{tmp}. NO accepts U if and only if it is valid.

In the above protocol, $\langle u_{\mathrm{NO}}, u_{\mathrm{U}}\rangle_K$ is used as a challenge from NO to U. The above protocol is also shown in Fig. 2. In this figure, the interactions between NO and SP are omitted because they are same as those of the proposed scheme.

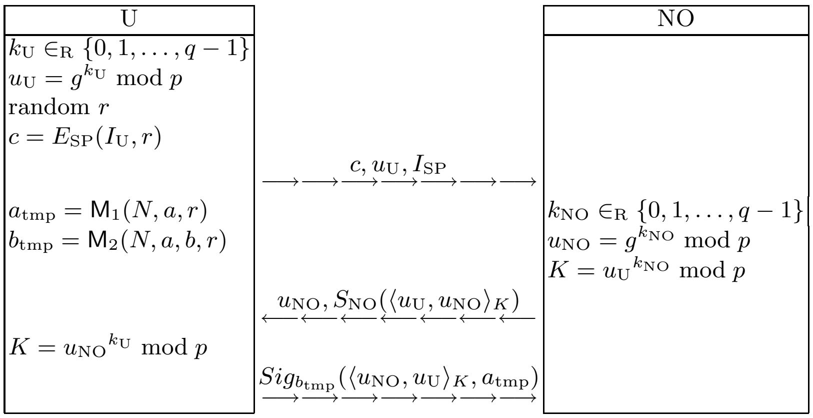

U
$k_{\mathrm{U}} \in_{\mathrm{R}} \{0, 1, \ldots, q-1\}$
$u_{\mathrm{U}} = g^{k_{\mathrm{U}}} \bmod p$
random r
$c = E_{\mathrm{SP}}(I_{\mathrm{U}}, r)$
$c, u_{\mathrm{U}}, I_{\mathrm{SP}}$
$a_{\mathrm{tmp}} = \mathsf{M}_1(N, a, r)$
$b_{\mathrm{tmp}} = \mathsf{M}_2(N, a, b, r)$
NO
$k_{\mathrm{NO}} \in_{\mathrm{R}} \{0, 1, \ldots, q-1\}$
$u_{\mathrm{NO}} = g^{k_{\mathrm{NO}}} \bmod p$
$K = {u_{\mathrm{U}}}^{k_{\mathrm{NO}}} \bmod p$
$u_{\mathrm{NO}}, S_{\mathrm{NO}}(\langle u_{\mathrm{U}}, u_{\mathrm{NO}}\rangle_K)$
$K = {u_{\mathrm{NO}}}^{k_{\mathrm{U}}} \bmod p$
$Sig_{b_{\mathrm{tmp}}}(\langle u_{\mathrm{NO}}, u_{\mathrm{U}}\rangle_K, a_{\mathrm{tmp}})$

Fig. 2. An authenticated key agreement scheme constructed by applying the proposed scheme to the STS protocol. The interactions between NO and SP is omitted, which are same as those of the proposed protocol.

6 Conclusion

In this manuscript, we have proposed a user authentication scheme with ILP. We have also discussed the provable security and the efficiency of the proposed scheme. Furthermore, we have shown that we can construct authenticated key agreement schemes by applying the proposed scheme to some of the existing authenticated key agreement schemes.

Acknowledgement

The authors would like to thank anonymous refrees for their valuable comments.

References

[1] *Cellular Digital Packet Data (CDPD) System Specification*, release 1.0 edition, July 1993.

[2] W. Diffie, P. C. van Oorschot, and M. J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 2(2):107–125, 1992.

[3] DTI/EPSRC LINK Personal Communications Programme. *Third Generation Mobile Telecommunications Systems Security Studies Technical Report 2: Security Mechanisms for Third Generation Systems*, May 1996.

[4] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-31(4):469–472, 1985.

[5] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO'86*, pages 186–194, 1987. Lecture Notes in Computer Science 263.

[6] A. Herzberg, H. Krawczyk, and G. Tsudik. On travelling Incognito. In *Proceedings of IEEE Workshop on Mobile Computing Systems and Applications*, 1994.

[7] S. Hirose and S. Yoshida. An authenticated Diffie-Hellman key agreement protocol secure against active attacks. In *PKC'98*, pages 135–148, 1998. Lecture Notes in Computer Science 1431.

[8] R. Molva, D. Samfat, and G. Tsudik. Authentication of mobile users. *IEEE Network*, 8(2):26–35, 1994.

[9] R. Molva, G. Tsudik, E. V. Herreweghen, and S. Zatti. KryptoKnight: Authentication and key distribution system. In *Proceedings on 1992 European Symposium on Research in Computer Security*, pages 155–174, 1992.

[10] D. Pointcheval and J. Stern. Security proofs for signature schemes. In *EUROCRYPT'96*, pages 387–398, 1996. Lecture Notes in Computer Science 1070.

[11] M. Rahnema. Overview of the GSM system and protocol architecture. *IEEE Communications Magazine*, 31:92–100, 1993.

[12] C. P. Schnorr. Efficient identification and signatures for smart cards. In *CRYPTO'89*, pages 239–252, 1990. Lecture Notes in Computer Science 435.

[13] M. Tompa and H. Woll. Random self-reducibility and zero knowledge interactive proofs of possession of information. In *1987 IEEE Symposium on Foundations of Computer Science*, pages 472–482, 1987.

An End-to-End Authentication Protocol in Wireless Application Protocol

Jong-Phil Yang[1], Weon Shin[1], and Kyung-Hyune Rhee[2]

[1] Department of Computer Science, Pukyong Nat'l Univ.,
599-1, Daeyeon3-Dong, Nam-Gu, Pusan 608-737, Republic of Korea
{bogus, redcomet}@unicorn.pknu.ac.kr
[2] Division of Electronic, Computer and Telecommunication Engineering,
Pukyong Nat'l Univ.,
599-1, Daeyeon3-Dong, Nam-Gu, Pusan 608-737, Republic of Korea
khrhee@pknu.ac.kr

Abstract. Mobile commerce is becoming more and more commonplace, but security is still a major concern. To provide security, the WAP (Wireless Application Protocol) forum suggests the WAP security architecture. However, it needs the WAP gateway for intermediate process between the WTLS (Wireless Transport Layer Security) and the SSL (Secure Socket Layer) protocol, and it does not guarantee end-to-end security between the mobile devices and the WAP servers. In this paper, we propose a new authentication protocol to solve this problem. Our solution is based on the design of a new network component that is called CRL-agent. Furthermore, we also analyze and evaluate the security strength of the proposed protocol.

1 Introduction

Recently, the new customers and services have been developing due to the rapid growth of the wireless Internet market. Operators and manufacturers established the WAP forum, which defines a set of protocols in transport, security, transaction, session and application layers to meet the challenges of the advanced, distinguished, fast and flexible services. The WAP forum has developed the WTLS layer for secure communication in the WAP environment. The primary goal of the WTLS is to provide privacy, data integrity and authentication between communicating applications. The WTLS provides functions similar to SSL 3.0 and incorporates new features such as datagram service, optimized handshake and dynamic key refreshing. The WTLS is optimized for low-bandwidth bearer networks with a relatively long latency[1],[2].

Presently, the serious security problem in the WAP is caused by not the WTLS in itself but importing WAP gateway. Since the WTLS is not compatible to the SSL protocol used on the web, the WAP gateway must decrypt the received data from the WAP client and re-encrypt the data to transfer to the WAP server. Existence of the WAP gateway at the network has the advantage that a content provider does not need a new software to overcome differences between wire and wireless networks. However, the drawback is that we have no real end-to-end security between the WAP client and the WAP server.

In this paper, we design a new authentication protocol that can provide several security services such as a WAP client's user authentication, session key establishment, dynamic key refresh and end-to-end security between a WAP client and a WAP server in the WAP environment.

The rest of this paper is organized as follows. In Section 2, we show the architecture of WAP and shortcomings of it. Moreover, we present necessity of designing a new protocol. In Section 3, we introduce the whole operating rule of the proposed protocol and a new network component that is called CRL-agent. In Section 4, we introduce notations and the proposed protocol. In Section 5, we evaluate the proposed protocol from the security point. Finally, we present some tips that can be used to embody the proposed protocol and make conclusions in Section 6.

2 The Security Architecture for WAP and Its Shortcomings

2.1 The Security Architecture for WAP

The WTLS that is provided by the WAP forum is composed of four subprotocols. The Handshake Protocol, Alert Protocol and Change Cipher Spec Protocol are used for managing the operation of the WTLS, and the Record Protocol provides actual security services. For connecting a secure session between the WAP client and the WAP server through the WTLS, the Handshake Protocol is processed in advance to allow peers to agree upon security parameters such as a session key, a peer certificate, compression method, master secret and a key refresh to be performed. The negotiated security parameters are used to provide security services in the Record Protocol[1],[2].

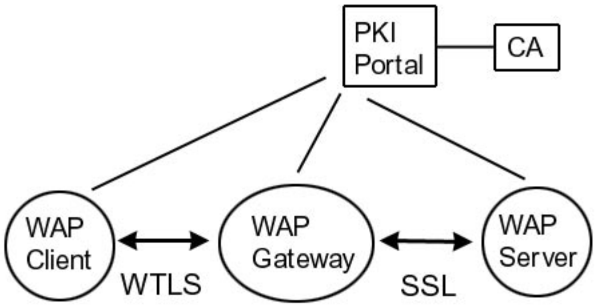


Fig. 1. The security architecture in WAP environment

The Security architecture for WAP is shown in Fig. 1. The WTLS is used for secure services in wireless environment between the WAP client and the WAP gateway, and SSL is used for secure services in wire environment between the WAP gateway and the WAP server.

2.2 The Shortcomings of Security Architecture for WAP

The security architecture for WAP based on WTLS/SSL has the following several shortcomings.

– It does not provide end-to-end security between the WAP client and the WAP server. That is, the WAP gateway decrypts the data that is received from the WAP client and re-encrypts it for wire Internet. Finally, plaintext is exposed in the WAP gateway.
– For a secure session, several public certificates based on X.509 v3 are used. Furthermore, the verification of a received certificate needs to query and verify the CRLs (Certificate Revocation Lists). This traffic leads to extravagant resources of the WAP client that has only low-processing power and memory. Even if the WAP forum recommends short-lived certificate to the verification of certificate, it is not a clear solution[3].
– The security vulnerability of the WTLS has been publishing[4],[5],[6],[7].
– The authentication based on public certificate only supports the legality of the communicating entity. Therefore, it does not sufficiently present the authorization information of the communicating entity[8].

In this paper, we propose a security protocol that can be applicable in the WAE (Wireless Application Environment) layer of the WAP protocol stack. Since it operates in the different layer with the WTLS/SSL, it can be applicable together with them. The proposed protocol in this paper which is called E2ESP(End-to-End shared Security Protocol) resolves the shortcomings of the original WAP security architecture and supports basic access control based on the identity of the WAP client's user.

In this paper, we assume that the WAP gateway is located at the subnetwork of the network operator. At present, the input equipment of a mobile phone is very inferior and network operators do their best to support more useful and interesting WAP portal that has power to collect subscribers. Therefore, most users search the linked sites of the WAP portal that is provided by the WAP gateway and it is few for users to search the unlinked-sites by typing directly the URL[9]. Hence, E2ESP focuses on the usual wireless Internet services based on the WAP portal. However, E2ESP can be operated in a situation where a special company has its own WAP gateway and needs to do access control for the internal users that connect from the outside.

3 CRL-Agent & Assumptions

When a user starts the service of a wireless Internet, the home page of the WAP browser for the WAP client is established as the WAP portal. If the WAP servers that are linked from the WAP portal are the sites that need the security such as on-line banking, stock services and m-commerce, these sites have already agreed with network operator for the support of E2ESP. Fig. 2 shows the architecture of E2ESP.

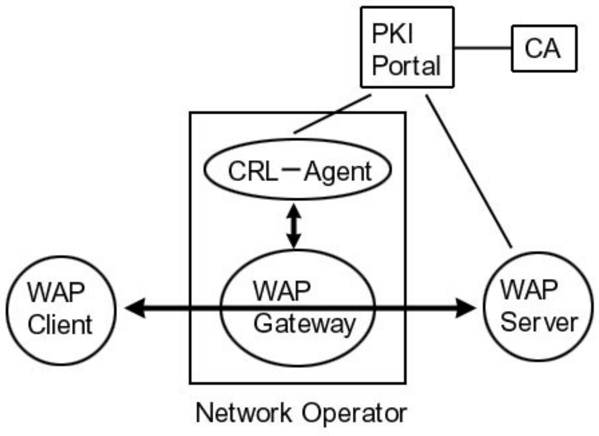


Fig. 2. The architecture of E2ESP

When a user does not want to use E2ESP for communication with a linked WAP server or tries to communicate with an un-linked WAP server, a session between the WAP client and the WAP server can be made securely by the WTLS/SSL. However, when a WAP client uses E2ESP for a secure session with a WAP server, it is not surely necessary to use the WTLS/SSL for a secure session.

We assume that the CRL-agent is located in the subnetwork of network operator and it is a secure system. The main function of the CRL-agent is to investigate the present state of public certificates for public-keys that are used in E2ESP. The present state of the public certificate means that it is one of the following three states; "applicable", "revoked" or "updated". We consider the traffic overhead of the CRL-agent and restrict that only several specific negotiated CAs (Certificate Authorities) with network operator can issue public certificates for E2ESP. The CRL-agent periodically investigates CRLs to ensure the state of the WAP servers and the public keys of the WAP servers. When a WAP client makes a request to the WAP portal page to the WAP gateway, the WAP gateway makes a request to the public key state information that is related to the WAP client. Then, the CRL-agent gives a response of the public key state information to the WAP gateway. The WAP gateway that received the response sends the WAP portal page and the public key state information to WAP client.

When the CRL-agent sends the public key state information to the WAP gateway, it does not perform any translation for privacy. Moreover, the public key of the WAP client for communicating with the CRL-agent has been already known to the CRL-agent and the CRL-agent can periodically investigate the state of the public certificate of the WAP client for E2ESP. If the public key of the WAP client is revoked, CRL-agent can make a request for a new certificate URL to the WAP client. More detail procedures are introduced in [3].

In this paper, we assume that the WAP client trusts network operator as the following point of view.

– The WAP client trusts that the CRL-agent is a secure system. That is, the WAP client trusts that the public key of the CRL-agent is safe and already knows the public key of the CRL-agent from the initial establishment. However, it can query the CRLs to verify the public certificate for the CRL-agent.
– The WAP client can verify the received information that is signed by the CRL-agent.

4 End-to-End Authentication Protocol in WAP

4.1 Notations

In this section, we introduce some notations as follows.

– C_i : The identity of *i-th* WAP client, for $1 \leq i \leq n$.
– S_j : The identity of *j-th* WAP server, for $1 \leq j \leq m$. It may be linked to WAP portal pages.
– $CRLA$: The identity of the CRL-agent.
– U_{ij} : The identity of *i-th* WAP client's user at *j-th* WAP server. For example, the user's account name in the WAP server.
– PU_X : A public key of communication entity X.
– PR_X : A private key of communication entity X.
– $Hash(m)$: A one-way hash value of message *m*.
– $Ppass_j$: A public password that is a hash value for the public key of *j-th* WAP server.

$$Ppass_j = Hash(PU_{S_j})$$

– $Upass_{ij}$: A user defined weak password that corresponds to user's account U_{ij}.
– r_X : A random challenge of communication entity X. The main function of this is both response and timeness. In initialization step for E2ESP, it can be replaced with time-stamp value.
– $refresh$: It defines how often session keys are updated. A new session key is calculated at every

$$n = 2^{refresh}$$

message, i.e. the sequence number of new session key is $0, n, 2n, 3n$ etc respectively.

- $Krls$: It is a list that includes all kind of refresh periods which an WAP server can support and is called *key refresh list*.
- $Svls$: It is a list that includes the identities of the WAP servers whose certificates are revoked and is called *server list*.
- $Info$: It is a notification message that the identity of an WAP server is deleted from certificate query lists for a specific WAP client.
- *seqnum* : The sequence number of transmitted message.
- ZZ_{ij} : A shared secret value that is derived from key agreement between *i-th* WAP client and *j-th* WAP server through E2ESP.
- SK_{ij} : A session key that is used for a secure session between *i-th* WAP client and *j-th* WAP server and is derived from ZZ_{ij} and the other parameters.
- $E_A(m)$: Message m is encrypted with key A.
- $D_A(m)$: Message m is decrypted with key A.
- $SIG_A(m)$: Message m is signed with key A.

4.2 Initialization for E2ESP

After establishing a session between the WAP client and the WAP gateway (optionally through WTLS), the process that the WAP client receives the WAP portal page is shown in Fig. 3. The detailed procedures are given as follows;

Message 1 : [WAP client ⇒ WAP gateway]
WAP portal request, $C_i, SIG_{PR_{C_i}}(C_i, r_{C_i})$
Message 2 : [WAP gateway ⇒ CRL-agent]
$C_i, SIG_{PR_{C_i}}(C_i, r_{C_i})$
Message 3 : [CRL-agent ⇒ WAP gateway]
$CRLA, SIG_{PR_{CRLA}}(Svls, CRLA, C_i, r_{C_i})$
Message 4 : [WAP gateway ⇒ WAP client]
WAP portal pages,
$CRLA, SIG_{PR_{CRLA}}(Svls, CRLA, C_i, r_{C_i})$

Fig. 3. Initialization for E2ESP

Message 1 *WAP portal request*, $C_i, SIG_{PR_{C_i}}(C_i, r_{C_i})$
i-th WAP client C_i requests the WAP portal page to the WAP gateway.
Message 2 $C_i, SIG_{PR_{C_i}}(C_i, r_{C_i})$
The WAP gateway requests the public key state information of WAP servers for C_i to the CRL-agent before sending the WAP portal pages to C_i.
Message 3 $CRLA, SIG_{PR_{CRLA}}(Svls, CRLA, C_i, r_{C_i})$
The CRL-agent has a database that is called *certificate query lists*. This database defines which WAP servers' certificates are necessary to be verified for each WAP client.

The CRL-agent receives message 2, and detects the required public key state information for the WAP servers which are included in the certificate query lists for C_i and makes *server list*. Then, the CRL-agent sends message 3 to the WAP gateway.

Message 4 *WAP portal pages*, $CRLA, SIG_{PR_{CRLA}}(Svls, CRLA, C_i, r_{C_i})$
After receiving message 3, the WAP gateway sends message 4 to C_i. Then, C_i verifies the signed part of message 4 by the public key of the CRL-agent. Finally, C_i displays the WAP portal pages in the WAP browser and recognizes whether each WAP server's public key is revoked or not.

If the public certificate of *j-th* WAP server S_j to be accessed by *i-th* WAP client C_i is revoked or updated, C_i must update the public password for S_j or delete the identity of S_j from certificate query lists. We will refer to the public password at section 4.3 in more detail. The detailed procedures are shown in Fig. 4 and given as follows;

Message 1 : [WAP client ⇒ WAP gateway]
$C_i, SIG_{PR_{C_i}}(S_j, C_i, r_{C_i})$
Message 2 : [WAP gateway ⇒ CRL-agent]
$C_i, SIG_{PR_{C_i}}(S_j, C_i, r_{C_i})$
Message 3-1 : [CRL-agent ⇒ WAP gateway]
$CRLA, SIG_{PR_{CRLA}}(CRLA, S_j, C_j, PU_{S_j}, r_{C_i})$
Message 3-2 : [CRL-agent ⇒ WAP gateway]
$CRLA, SIG_{PR_{CRLA}}(CRLA, S_j, C_j, Info, r_{C_i})$
Message 4-1 : [WAP gateway ⇒ WAP client]
$CRLA, SIG_{PR_{CRLA}}(CRLA, S_j, C_j, PU_{S_j}, r_{C_i})$
Message 4-2 : [WAP gateway ⇒ WAP client]
$CRLA, SIG_{PR_{CRLA}}(CRLA, S_j, C_j, Info, r_{C_i})$

Fig. 4. Updating public password for WAP server

Message 1 $C_i, SIG_{PR_{C_i}}(S_j, C_i, r_{C_i})$
Message 1 has two meanings according to different circumstances. If the identity of S_j is not included in the certificate query lists inside the CRL-agent for requesting C_i, it means that C_i wants to add S_j into the certificate query lists. If the identity of S_j is included in the *server list* of message 4 of the former initialization step, it means that C_i wants to receive the updated public key of S_j or delete the S_j from the certificate query lists.

Message 2 $C_i, SIG_{PR_{C_i}}(S_j, C_i, r_{C_i})$
The WAP gateway only forwards the received message from C_i to the CRL-agent.

Message 3-1 $CRLA, SIG_{PR_{CRLA}}(CRLA, S_j, C_j, PU_{S_j}, r_{C_i})$

Message 3-2 $CRLA, SIG_{PR_{CRLA}}(CRLA, S_j, C_j, Info, r_{C_i})$

After receiving message 2, the CRL-agent performs the following steps.

– If the received S_j is included in the certificate query lists for C_i,
  - In the case of update of S_j's public certificate for E2ESP, the CRL-agent sends the message 3-1 that includes a new public key for S_j to the WAP gateway.
  - In the case of revocation of S_j's public certificate for E2ESP, the CRL-agent deletes S_j from the certificate query lists for C_i and makes a notification message *Info* that is added into message 3-2. The CRL-agent sends message 3-2 to the WAP gateway.

– If the currently received S_j has linked site with the WAP portal pages and is not included in the certificate query lists for C_i, the CRL-agent adds the received S_j to certificate query lists for C_i and sends message 3-1 that includes a new public key for S_j to the WAP gateway.

Message 4-1 $CRLA, SIG_{PR_{CRLA}}(CRLA, S_j, C_j, PU_{S_j}, r_{C_i})$

Message 4-2 $CRLA, SIG_{PR_{CRLA}}(CRLA, S_j, C_j, Info, r_{C_i})$

The WAP gateway only forwards message 3-1 or message 3-2 to C_i . C_i received message 4-1 or 4-2 performs one of the following two steps according to circumstances.

– If C_i receives message 4-1, C_i hashes the received public key of S_j and allows the user to use it as a public password.

– If C_i receives message 4-2, E2ESP for communicating with S_j will be inactive. However, it is independent of the existing WTLS/SSL protocol.

4.3 E2ESP

When the WAP client's user receives the WAP portal pages and public key state information from WAP gateway, the user can start the login process and key agreement for E2ESP based on the user-defined password. E2ESP adopts the EKE (Encrypted Key Exchange) method based on Diffie-Hellman key agreement[10],[11],[12]. The WAP client's user and the WAP server agree on a large prime p and g, such that g is primitive *mod p*. These two integers do not have to be secret; the WAP client and WAP server can agree to them over some insecure channel. The user has a hash value for the public key of the WAP server as a public password. The user does not even need to type such a digest, but just recognize it when it is displayed. The WAP server has a user-defined password as

$$v_{ij} = g^{Hash(Upass_{ij},\, U_{ij},\, S_j)}, for\ 1 \le i \le n, \quad 1 \le j \le m.$$

The value v_{ij} will allow the WAP server later to get or store the password itself; this way we can limit the damage if the WAP server is corrupted or the database is leaked. v_{ij} is stored together with the WAP client's identity in the WAP server's user database.

The procedure of the WAP client's user who wants to login to the WAP server is given in Fig. 5. In this paper, we omit the *mod p* calculation for convenience.

Message 1 : [WAP client ⇒ WAP server]
User login request
Message 2 : [WAP server ⇒ WAP client]
$r_{S_j}, PU_{S_j}, g^x, g^{x'}, Krls$
Message 3 : [WAP client ⇒ WAP server]
$E_{PU_{S_j}}(U_{ij}, S_j, g^y, refresh, E_{SK_{ij}}(r_{S_j})), r_{C_i}$

Fig. 5. A secure session establishment over E2ESP

Message 1 *User login request*

Message 1 may be sent by *i-th* WAP client's user to *j-th* WAP server at any time. This message just means that U_{ij} can negotiate several security parameters such as ZZ_{ij}, SK_{ij} and *refresh* and U_{ij} wants to login to S_j. In message 1, it does not include user's identity U_{ij} but the identity of *i-th* WAP client C_i.

Message 2 $r_{S_j}, PU_{S_j}, g^x, g^{x'}, Krls$

S_j that has received message 1 chooses two random big integers x and x', and computes g^x and $g^{x'}$ for key agreement with C_i. Of course, x and x' are secure information for S_j. Then, S_j generates r_{S_j} as a random challenge and makes a *key refresh list.* Finally, S_j configures message 2 and sends it to C_i.

Message 3 $E_{PU_{S_j}}(U_{ij}, S_j, g^y, refresh, E_{SK_{ij}}(r_{S_j})), r_{C_i}$

C_i that has received message 2 compares the public password $Ppass_j$ with the hashed value of PU_{S_j}. The implementation technique of this comparison will be handled at Section 6 in more detail. If two values do not agree, C_i may request the CRL-agent to give the public key state information for S_j. If the two values agree, C_i requests the user to input his identity U_{ij} and password $Upass_{ij}$, chooses an appropriate *refresh* from the received *key refresh list.* Then, C_i chooses a random big integer y and generates g^y which is secure information for C_i.

C_i generates a shared secret ZZ_{ij} and a session key SK_{ij} for a secure session with S_j in the following way.

$$ZZ_{ij} = Hash((g^x)^y, (g^{x'})^{Hash(Upass_{ij}, U_{ij}, S_j)})$$

$$SK_{ij} = Hash(ZZ_{ij}, r_{S_j}, r_{C_i}, seqnum)$$

Finally, C_i generates message 3 and sends it to S_j.

Now, S_j receives message 3, and decrypts it by its own private key PR_{S_j} and becomes to know the user's identity U_{ij} and g^x. S_j generates a shared secret and a session key in the following way.

$$ZZ_{ij} = Hash((g^x)^y, (v_{ij})^{x'})$$

$$SK_{ij} = Hash(ZZ_{ij}, r_{S_j}, r_{C_i}, seqnum)$$

As a result, S_j can decrypt an encrypted part of message 3 by the SK_{ij} and compares r_{S_j} with the received one. If two values agree, S_j knows that the generated session key and the U_{ij} are correct.
The first *seqnum* value is 0 and this value is updated every message exchange between C_i and S_j. While updating SK_{ij} according to *refresh*, the above procedures for generation of session key SK_{ij} must be recomputed where only *seqnum* is changed.

5 Security Evaluation of E2ESP

In this section, we evaluate the security of E2ESP through some attack scenarios.

– When the attacker compromises v_{ij} and masquerades *j-th* WAP server S_j.
  - The attacker generates disguised $r^\star_{S_j}, x^\star, x'^\star$ and sends

    $$r^\star_{S_j}, PU_{S_j}, g^{x^\star}, g^{x'^\star}, Krls$$

    to *i-th* WAP client C_i. Then, C_i receives the above messages, and computes a shared secret

    $$ZZ^\star_{ij} = Hash((g^{x^\star})^y, (g^{x'^\star})^{Hash(Upass_{ij}, U_{ij}, S_j)})$$

    and a session key $SK^\star_{ij}$ correspondent to $ZZ^\star_{ij}$. Finally, C_i sends

    $$E_{PU_{S_j}}(U_{ij}, S_j, g^y, refresh, E_{SK^\star_{ij}}(r^\star_{S_j})), r_{C_i}$$

    to the attacker. In this situation, the attacker must know PR_{S_j} to compromise U_{ij} or SK_{ij}. That is, attacker must decrypt message that is encrypted by PU_{S_j}.
  - If the attacker uses its own public key to make a response for the user login request of C_i, he may send

    $$r^\star_{S_j}, PU^\star_{S_j}, g^{x^\star}, g^{x'^\star}, Krls$$

    as response to C_i . Since the verification process of the public password is performed, this attack would fail. However, when g^y is exposed by accident, the attacker could make a session key successfully.

– When the attacker eavesdrops on the inner traffic of the WAP gateway and masquerades U_{ij}.
  This active attack is possible when an attacker knows $Upass_{ij}$ and U_{ij}. If the attacker who masquerades U_{ij} randomly generates $r^\star_{C_i}, g^{y^\star}, U^\star_{ij}$ and $Upass^\star_{ij}$ and computes a shared secret

  $$ZZ^\star_{ij} = Hash((g^x)^{y^\star}, (g^{x'})^{Hash(Upass^\star_{ij}, U^\star_{ij}, S_j)})$$

  and a session key.

  $$SK^\star_{ij} = Hash(ZZ^\star_{ij}, r_{S_j}, r^\star_{C_i}, seqnum)$$

He chooses a masqueraded $refresh^{\star}$ and sends

$$E_{PU_{S_j}}(U^{\star}_{ij}, S_j, g^{y^{\star}}, refresh^{\star}, E_{SK^{\star}_{ij}}(r_{S_j})), r^{\star}_{C_i}$$

to S_j , then S_j tries to decrypt the received message and finds $U^{\star}_{ij}$ and its related information $Upass^{\star}_{ij}$ in the system. If S_j cannot find the correct $U^{\star}_{ij}$ and $Upass^{\star}_{ij}$, it terminates the present session. However, if S_j can find a correct $U^{\star}_{ij}$ and $Upass^{\star}_{ij}$ by accident, the secure session is established between the attacker and S_j, where the attacker has come to understand the identity of user and user-defined password.

The second possible attack can be protected by the following two methods.

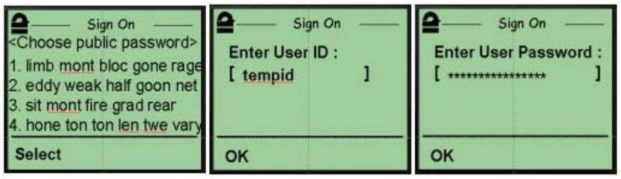
- S_j which has already sent message 2 in Fig.5 closes the session after waiting for a defined time interval. Since the transmitted $r_{S_j}, g^x, g^{x'}$ values are changed every time, it is impossible for an attacker inside the WAP gateway to perform an off-line password guessing attack. Moreover, S_j can give a time delay for establishing a session to a user who had already failed to login S_j several times, hence, it can be protected on-line password guessing attack.
- Since the input mechanism of the WAP client is so simple, it is a basically weak situation against a dictionary attack. However, since the identity of user and user-defined password are not sent in plaintext over the insecure channel, it is difficult for an attacker to perform on-line and off-line password guessing attacks, and to know the identity of the user and user-defined password. Therefore, these attacks can be efficiently protected in E2ESP.

6 Implementation Techniques of E2ESP & Conclusion

For example, the following situation can be considered. " When does the WAP client request the CRL-agent to manage the updated or revoked WAP servers' public certificate after receiving the public key state information from WAP gateway ?" There are two possible ways to implement this. The first one is that the WAP client requests updating process of public password to the CRL-agent right after receiving the *WAP portal pages* and *server list*. The second one is that the WAP client requests updating process of public password to the CRL-agent right before trying to connect with the WAP server that has revoked the public certificate.

For a user to be able to read, recognize, and type the public password, it is advisable to have a user-readable format for these passwords. A representation which maps arbitrary binary strings into easy-to-read words was introduced in the context of one-time passwords in detail[13]. When the WAP client receives the WAP server's public key (message 2 in E2ESP), it displays a verification screen of the public key to user, as given in Fig. 6.

In Fig. 6, one of number 1, 2, 3, 4 is a correct password. The user must choose the number corresponding to the correct public password. That is, if the public password is *"limb mont bloc gone rage pit"*, the user may choose "1". The others



Sign On
<Choose public password>
1. limb mont bloc gone rage
2. eddy weak half goon net
3. sit mont fire grad rear
4. hone ton ton len twe vary
Select

Sign On
Enter User ID :
[tempid]
OK

Sign On
Enter User Password :
[***************]
OK

Fig. 6. Example of user interface to input public password, user ID and user-defined password

from 2 to 4 are randomly selected words in the word dictionary of the S/Key system by the WAP client. The next step of verifying the public password is that the user must input his identity and user-defined password. Fig. 6. shows this process. After ending all of the above steps, the WAP client sends message 3 of Fig. 6 to the WAP server. The above mentioned implementation is a typical example, so the various alternatives are possible if we consider user's convenience.

In this paper, we propose a new security protocol that serves as an end-to-end security between the WAP client and the WAP server in WAP environment. Since E2ESP adopts the EKE method for a user to login to the WAP server, the user's identity is not exposed over the insecure channel. Moreover, E2ESP supports that the WAP server can successfully do basic access control based on the user's identity and key agreement. E2ESP can operate securely alone and more securely together with the WTLS/SSL.

Acknowledgements

This work was supported by **MSRC**(Mobile Network Security Technology Research Center).

References

1. WAP Forum : Wireless Application Protocol Architecture Specification, 1998.
2. WAP Forum : Wireless Application Protocol Wireless Transport Layer Security Specification, 1999.
3. WAP Forum : Wireless Application Protocol Public Key Infrastructure Definition, 2000.
4. David Wagner, Bruce Schneier : Analysis Of The SSL 3.0 Protocol. Proceedings of 2nd USENIX Workshop on Electronic Commerce 2104 USENIX Press, November 1997, pp.29-40.
5. Markku-Juhani Saarinen : Attack Against The WAP WTLS Protocol. Communications and Multimedia Security Joint working conference IFIP TC6 and TC11 Katholieke Universiteit Leuven, 1999, Belgium.
6. Sami Jormalainen, Jouni Laine : Security In The WTLS. http://www.hut.fi/ jt-laine2/wtls/, 1999.
7. Steven M. Bellovin : Problem Areas For The IP Security Protocols. Proceedings of the Sixth USENIX Security Symposium, 1996, pp. 205-214.

8. Rolf Oppliger : Security Technologies For The World Wide Web. ARTECH HOUSE. INC, 2000.
9. Charles Arehart, Nirmal Chidambaram etc : Professional WAP. Wrox Press Ltd, 2000, pp.10-41.
10. Peter Buhler, Thomas Eirich, Michael Stenier, Michael Waidner : Secure Password-Based Cipher Suite For TLS. In Symposium on Network and Distributed Systems Security (NDSS '00), pages 129-142, San Diego, CA, Internet Society, 2000.
11. S. Halevi and H. Krawczyk : Public-Key Cryptography And Password Protocols. In 5th ACM Conference on Computer and Communication Security", San Francisco, California. ACM Press, 1998.
12. Steven M. Bellovin : Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks. Proceedings of the IEEE Symposium on research in Security and Privacy, Oakland, May 1992.
13. N. Haller : The S/KEY One-Time Password System. RFC 1760, Feb 1995.

Error Detection and Authentication in Quantum Key Distribution

Akihiro Yamamura[1] and Hirokazu Ishizuka[2]

[1] Communications Research Laboratory,
4-2-1, Nukui-Kitamachi, Koganei, Tokyo 184-8795, Japan
aki@crl.go.jp
[2] Mitsubishi Electric Corporation,
5-1-1, Ofuna, Kamakura, Kanagawa 247-8501, Japan
ishizuka@isl.melco.co.jp

Abstract. Detecting errors in a raw key and authenticating a private key are crucial for quantum key distribution schemes. Our aim is to propose practical methods for error detection and authentication in quantum key distribution schemes. We introduce several concepts about neighborhood collision free properties of Boolean functions, which are closely related to hash functions, and propose methods based on neighborhood collision free functions and error correcting codes such as Reed-Solomon code. We also examine whether or not widely used cryptographic hash functions SHA-1 and MD5 satisfy the neighborhood collision free property by computation experiments.

1 Introduction

Quantum key distribution schemes have been introduced and studied in detail up to date (e.g. [1], [2], [8]). Under an ideal circumstance like an experiment in a laboratory without any physical interferences, quantum key distribution schemes enjoy the unconditional security. Since an eavesdropper Eve's unlawful access to the quantum channel causes disturbance of bit patterns of photons sent by Alice due to the Heisenberg uncertainty principle, Alice and Bob can detect Eve's intervention by estimating error rate after the data transmission through the quantum channel. Error estimation can be carried out by discussion through the classical channel. Physical errors inevitably occur in data transmission through the quantum channel under realistic circumstances. Eve may want to obtain only small amount of information concerning the private key shared by Alice and Bob. Then Eve's best strategy is to wiretap the quantum channel only small fraction of the total data transmission, and deceive Alice and Bob as if the resulting disturbance is caused by the physical defects of the quantum channel and other peripherals. By the attack, Eve may be able to obtain partial information on the private key shared by Alice and Bob. Under such a scenario, bits, where errors may have happened, are more suspicious of Eve's intervention than the other bits and should be dumped to prevent Eve from gaining any partial information. The following are essential to attain the virtually unconditional security. The first is to lower the error rate in the data transmission through the quantum channel. This depends on improvements of physical devices such as optical fibers, single photon source generators, avalanche-photo-diode detectors and so on. The error rate depends on the distance of the quantum data transmission: the longer the channel gets, the higher the error rate rises. See [15], [16] for recent experimental results. The second is to efficiently detect (and correct) errors in the raw keys, remove the leaked information and confirm the integrity of the private key agreed by Alice and Bob. Our aim in this paper is to propose practical methods forward the second goal.

We briefly explain the general scheme of a quantum key distribution (see Chapter 2 of [6] for more detail). First, Alice generates a (sufficiently long) random bit string and sends photon pulses according to the random bit string through the quantum channel, where the basis and the polarization are randomly determined. Bob also generates a random bit string and measures the photon pulses with the basis determined according to his random bit string. Then Alice and Bob obtain bit strings, called *raw keys*, respectively. We should note that Bob's raw key is totally different from Alice' raw key because Bob does not know Alice's choice of bases and cannot get to know the bits in Alice's raw key unless he chooses the same basis. Checking their choice of bases through the classical channel, they estimate errors existing in Bob's raw key and then obtain *sifted keys* (this process is called *sifting*). The error rate is supposed to be kept under a previously fixed value, which is determined by the quality of the physical devices, unless Eve intervened. If Eve wiretapped substantial amount of data transmission from Alice to Bob through the quantum channel, Eve's intervention can be detected in this stage because Alice and Bob will find the error rate is larger than the previously fixed value. Eve's best strategy to eavesdrop is to wiretap only small fraction of the total data transmission through the quantum channel. It follows that the leaked information to Eve is at most the physical error rate.

Second, errors must be removed or corrected. After the error correction process, Alice and Bob possess an identical key called *reconciled key.* Note that Eve might have partial information on the reconciled key because Eve could eavesdrop the communication through the quantum and the classical channel even though the potentially leaked information is almost negligible.

Third, Eve's information is reduced substantially using *privacy amplification* that is the method to lower Eve's information exponentially by sacrificing bits in the reconciled key linearly ([3], [4], [10]). Privacy amplification can be carried out using *t-resilient functions* [4] (also known as *(N,J,K) functions* [7]). The resulting key is called a *private key.*

Lastly, Alice and Bob confirm the integrity of their private key and obtain an *authenticated private key.* We illustrate a typical process of key distribution in Fig.1 in a quantum key distribution scheme.

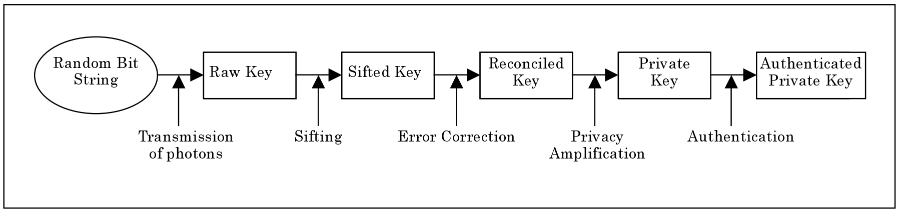


Fig. 1. Data Processing in Quantum Key Distribution

We introduce a concept of a *(globally, locally) neighborhood collision free function* and show that SHA-1 [9] and MD5 [12] enjoy the neighborhood collision free property by experiments with computers. We present methods to detect errors in the raw keys and to authenticate the private key in a quantum key distribution scheme using a neighborhood collision free function. Our methods realize the error detection (correction) and authentication procedures in Fig.1.

2 Several Error Correction Methods

We briefly explain the error correction methods in [4] and [5] in this section. Suppose Alice and Bob possess their sifted keys after the sifting process in a quantum key distribution scheme. If Alice has a sifted key r, then Bob has a sifted key $r \oplus e$, where $\oplus$ denotes the bitwise exclusive or, and e represents the errors occurred. The Hamming weight of e depends on the physical error rate of data transmission through the quantum channel, and the recent physical experiments show relatively low error rate for short distance transmission. See, for example, [15] and [16]. The physical error rate is the fraction of occurrence of errors in the total data transmission through the quantum channel. Under the most ideal assumption, we have $e = 0$, and hence, Alice and Bob share the identical key, on which Eve has no chance to get any information on it. Although physical errors unavoidably occur at some rate under the realistic situation, they are very rare. Therefore, the Hamming weight of e is in proportion to the error rate and so slightly greater than 0. We may assume that most of bits in e are 0. To share the identical private key, Alice and Bob need to get rid of the error bits. Especially, if they intend to use the key as the secret key for a symmetric cipher, it is crucial to share an identical authenticated private key.

First, we explain the error correction method by Bennett, Bessette, Brassard, Salvail and Smolin [5]. Alice divides her sifted key into blocks. Bob also divides his sifted key in the same way as Alice does: if Alice has the sifted key r and r is divided as $r = r_1 r_2 \cdots r_n$, then Bob has the sifted key $r \oplus e$ and it is divided as $r \oplus e = (r_1 \oplus e_1)(r_2 \oplus e_2) \cdots (r_n \oplus e_n)$, where $e = e_1 e_2 \cdots e_n$ represents the error bits. Then Alice computes the parity of each block r_i and sends them all to Bob through the classical channel. Eve can wiretap the classical channel and is able to obtain the parities of the blocks. The parity of each block is considered as one bit information, and so, Alice and Bob take it for granted that one bit information is leaked for each block. Bob computes the parities of the corresponding blocks of his sifted key and compares them with the parities sent by Alice. If all of them coincide, then Alice and Bob probably possess the identical key. Otherwise, some of Alice's block and Bob's block must be different at least one position. In such a case, Alice and Bob divide the block whose parities are different into shorter blocks and continue the process until they do not find any different parity. In any stage, Alice and Bob delete one bit from each block at the same position in order to make the leaked information to Eve meaningless. Repeating the process several times, Alice and Bob eventually establish an identical key with a high probability. Demerits of this method are following: Alice and Bob are not guaranteed to share the identical reconciled key. It wastes numerous bits and requires considerable computation. In the process of generating raw keys, Alice and Bob cannot theoretically predict the number of necessary bits to establish the reconciled key, that is, it is quite hard to theoretically estimate the efficiency of the error correction.

Second, we explain one of the methods in Bennett, Brassard and Robert [4]. They proposed that Alice sends the hash value of her sifted key through the classical channel. Bob computes the hash value of his sifted key as well. Bob compares these two hash values. If they are identical, they share the identical reconciled key. Otherwise, Bob turns around a few bits in his sifted key, computes the hash value of the altered key then and checks whether or not it coincides with the hash value of Alice's sifted key. Bob continues this process until he finds the one whose hash value coincides with the hash value of Alice's sifted key. Bob basically carries out the exhaustive search to find positions in his bit string, where the errors happen, until he detect the errors. The method is called a *bit twiddling*. The defect of the method is that Bob is required to carry out substantial computation, and the hash value transmitted through the classical channel gives substantial information to Eve as well. Only under the very restricted assumption that the error rate is very low and the bit string is short, the exhaustive search can be carried out. Otherwise, the task is impossible. It is also proposed in [4] that Alice encodes her sifted key by an error correcting code and sends only the redundancy part of the encoded sifted key. The defect of this method is again that the redundancy part of encoded sifted key gives substantial information to Eve. This method has several demerits, nevertheless, these can be remedied as we will see in Section 4.

3 Neighborhood Collision Free Functions

Let H be a Boolean function of $\mathbb{Z}_2^l$ to $\mathbb{Z}_2^k$. Intuitively, H is *neighborhood collision free* if H maps any two bit strings with a small Hamming distance to bit strings with a large Hamming distance. Recall that the *Hamming distance* of bit strings x_1 and x_2 is the number of positions where the entry of x_1 is different from that of x_2. The *Hamming weight* of a bit string x is the Hamming distance between

x and the zero (that is, the string consisting of only 0). This property should be satisfied by all (symmetric and asymmetric) encryption functions, although it is not sufficient for secure communication. Recall that a Boolean (hash) function H is *(strongly) collision free* if it is hard to find bit strings r_1 and r_2 with $r_1 \neq r_2$ and $\mathrm{H}(r_1) = \mathrm{H}(r_2)$. In other words, H is (strongly) collision free if it is hard to find bit strings r_1 and r_2 such that $r_1 \neq r_2$ and the Hamming distance between $\mathrm{H}(r_1)$ and $\mathrm{H}(r_2)$ is 0. This concept is generalized as follows. Let us denote the Hamming distance between r and s by $d(r, s)$, where $r, s \in \mathbb{Z}_2^l$. For $t \in \mathbb{Z}_2^l$, the set $\{s \in \mathbb{Z}_2^l \mid d(s,t) \leq i\}$ is called the *neighborhood* around t of radius i and denoted by $\mathrm{N}(t, i)$. We define several neighborhood collision free properties. Let H be a Boolean function of $\mathbb{Z}_2^l$ to $\mathbb{Z}_2^k$.

- H is a *globally j-neighborhood collision free function* if it is hard to find $s, t \in \mathbb{Z}_2^l$ such that $\mathrm{H}(s) \in N(\mathrm{H}(t), j)$, equivalently $\mathrm{H}(t) \in N(\mathrm{H}(s), j)$ (or $N(\mathrm{H}(s), \frac{j}{2}) \cap N(\mathrm{H}(t), \frac{j}{2})$ is not empty).
- H is a *locally j-neighborhood collision free function in i-neighborhood* if for every $u \in \mathbb{Z}_2^l$ it is hard to find $s, t \in N(u, i)$ such that $\mathrm{H}(s) \in N(\mathrm{H}(t), j)$, equivalently $\mathrm{H}(t) \in N(\mathrm{H}(s), j)$ (or $N(\mathrm{H}(s), \frac{j}{2}) \cap N(\mathrm{H}(t), \frac{j}{2})$ is not empty).
- H is a *globally collision free function* if it is hard to find $s, t \in \mathbb{Z}_2^l$ such that $\mathrm{H}(s) = \mathrm{H}(t)$.
- H is a *locally collision free function in i-neighborhood* if for every $u \in \mathbb{Z}_2^l$ it is hard to find $s, t \in N(u, i)$ such that $\mathrm{H}(s) = \mathrm{H}(t)$.

These concepts play a vital role in construction of our error detection and authentication scheme. The concept of the *hardness* depends on the context, and it may be information theoretic or computational. A globally collision free property coincides with a *(strongly) collision free property* for cryptographic hash functions. It is easy to see that a globally j-neighborhood collision free function is a locally j-neighborhood collision free function in i-neighborhood, a globally j-neighborhood collision free function is a globally collision free function, a globally collision free function is a locally collision free function in j-neighborhood and a locally j-neighborhood collision free function in i-neighborhood is a locally collision free function in i-neighborhood. The converses are not necessarily true. See Fig. 2 for the relationships among the concepts.

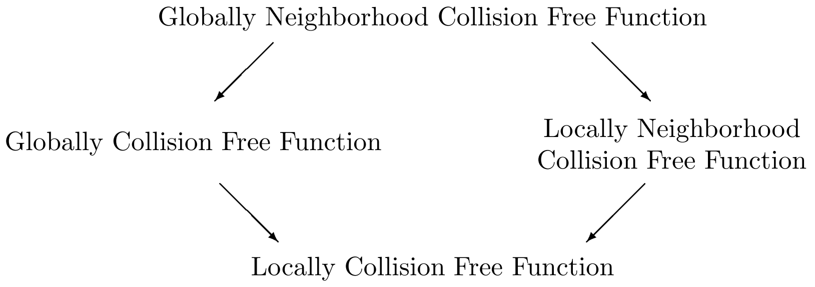

$$\begin{tikzcd}
& \text{Globally Neighborhood Collision Free Function} \arrow[dl] \arrow[dr] & \\
\text{Globally Collision Free Function} \arrow[dr] & & \text{Locally Neighborhood Collision Free Function} \arrow[dl] \\
& \text{Locally Collision Free Function} &
\end{tikzcd}$$

Fig. 2. Hierarchy of Collision Free Functions

For example, good block ciphers show the strong avalanche effect, and hence, they satisfy the globally neighborhood collision free property even under a low round. The globally neighborhood collision free property can be considered as a generalization of the avalanche effect. We shall show, in Section 5, that SHA-1 and MD5 satisfy the globally neighborhood collision free property by experiments by computers. Our experiments show that SHA-1 has the 43-neighborhood collision free property, and MD5 has the 34-neighborhood collision free property, however, it is difficult to prove theoretically and rigorously that they really do.

4 Error Detection Using Locally Neighborhood Collision Free Functions

The methods explained in Section 2 waste numerous bits and require considerable computation such as iterations of random permutations to detect and correct errors. Moreover, it is difficult for us to predict the number of necessary bits, that is, the length of raw keys, to succeed in establishing an authenticated private key in the final stage. It is desired to invent a simple efficient method so that we can predict easily and theoretically the number of necessary bits in advance. We employ a locally neighborhood collision free function to detect errors in the sifted keys.

Suppose that the physical error rate of the quantum data transmission is $\epsilon > 0$. We note that Alice and Bob should operate a random permutation to their sifted keys after the sifting process. If they have done so, we can suppose the errors are random, that is, the errors are uniformly distributed in Bob's sifted key. If Eve eavesdrops the bits located at specific positions in the private key (according to his eavesdropping strategy) and Alice and Bob do not operate a random permutation, then the errors are burst, that is, they are distributed non-uniformly in Bob's sifted key. After the error estimation process, Alice and Bob have their sifted keys, r and s, where $r, s \in \mathbb{Z}_2^l$ for some integer l, respectively. Then $r \oplus s$ shows the error bit pattern and its Hamming weight is approximately $\epsilon \times l$. Suppose $0 < \epsilon < 1$ and $0 < \alpha < 1$ are constants such that α is sufficiently larger than ϵ. Let H be a locally neighborhood collision free function of $\mathbb{Z}_2^l$ to $\mathbb{Z}_2^k$ with $\theta(H, \epsilon, \alpha)$ that is the probability of the event $d(\mathrm{H}(r_1), \mathrm{H}(r_2)) \leq \alpha \times k$ when we choose randomly and uniformly a pair (r_1, r_2) of distinct bit strings from $\mathbb{Z}_2^l$ such that the Hamming distance between r_1 and r_2 is less than or equal to $\epsilon \times l$. A Boolean function H is considered locally neighborhood collision free if $\theta(H, \epsilon, \alpha)$ is negligible for some constants ϵ and α such that $0 < \epsilon \ll \alpha < 1$.

We now explain the basic idea of an error detection method. Suppose that H is a locally neighborhood collision free function and $\theta = \theta(H, \epsilon, \alpha)$ is small. This implies that the probability that $d(\mathrm{H}(r), \mathrm{H}(s)) < \alpha \times k$ for $r \neq s \in \mathbb{Z}_2^l$ with $d(r, s) < \epsilon \times l$ is negligible. We assume the Hamming weight of $r \oplus s$ is less than $\epsilon \times l$. Hence, if $r \neq s$, then $\mathrm{H}(s)$ is not in $N(\mathrm{H}(r), \alpha \times k)$, equivalently the Hamming weight of $\mathrm{H}(r) \oplus \mathrm{H}(s)$ is bigger than $\alpha \times k$, by the locally neighborhood collision free property of H. If $r = s$, $\mathrm{H}(r) = \mathrm{H}(s)$ and so the Hamming weight of $\mathrm{H}(r) \oplus \mathrm{H}(s)$ is 0.

We now suppose Alice and Bob possess t and $t \oplus e$ as parts of their sifted keys, respectively, where $t, e \in \mathbb{Z}_2^k$ and the Hamming weight of e is approximately $\epsilon \times k$. Then the Hamming distance between $\mathrm{H}(r) \oplus t$ and $\mathrm{H}(s) \oplus (t \oplus e)$ is given by $(\mathrm{H}(r) \oplus t) \oplus (\mathrm{H}(s) \oplus (t \oplus e)) = (\mathrm{H}(r) \oplus \mathrm{H}(s)) \oplus e$. Hence, the Hamming distance is approximately $\epsilon \times k$ if $r = s$, otherwise, it is more than $\alpha \times k$. So if we set $\frac{\epsilon+\alpha}{2}k$ as a threshold, Bob can determine whether or not $r = s$ by checking whether the Hamming distance between $\mathrm{H}(r) \oplus t$ and $\mathrm{H}(s) \oplus (t \oplus e)$ is smaller or bigger than $\frac{\epsilon+\alpha}{2}k$.

We combine this criterion to find the existence of errors and several methods to find the exact bit positions where the errors occurred. We discuss several methods in the following subsections. The difference among the first three methods lies in the consumption of resources (computation, quantum data transmission and classical data transmission). This difference indicates the existence of a trade-off relation among computation, quantum communication and classical communication.

4.1 Method 1

Suppose that l is the intended size of a reconciled key. Let H be a locally neighborhood collision free function of $\mathbb{Z}_2^l$ to $\mathbb{Z}_2^k$ such that the probability $\theta(H, \epsilon, \alpha)$ is negligible and $\epsilon \ll \alpha$. We assume Alice and Bob can make use of H. Note that H is not necessarily kept secret, and hence, Eve can also make use of it. Alice and Bob first establish $2l + k$ bit sifted keys in the sifting process. Alice and Bob have $2l + k$ bit binary strings r and $r \oplus e$ as their sifted keys, respectively. Here, e represents the errors. The Hamming weight of e is approximately $\epsilon \times |e| = \epsilon \times l$. The basic idea is that Alice and Bob sacrifice $l + k$ bits of their sifted keys and detect error bits in e without leaking any information to Eve. Then they share r and agree that r is their reconciled key.

Suppose Alice has r as her sifted key and $r = r_1 r_2 r_3$, where $r_1, r_2 \in \mathbb{Z}_2^l$ and $r_3 \in \mathbb{Z}_2^k$. Alice computes the hash value $\mathrm{H}(r_1)$, then sends $r_1 \oplus r_2$ and $\mathrm{H}(r_1) \oplus r_3$ to Bob through the classical channel. Eve can wiretap the classical channel. Bob has $r \oplus e$ as his sifted key and $r \oplus e = (r_1 \oplus e_1)(r_2 \oplus e_2)(r_3 \oplus e_3)$, where $e = e_1 e_2 e_3$ and $e_1, e_2 \in \mathbb{Z}_2^l$ and $e_3 \in \mathbb{Z}_2^k$. Bob, receives $r_1 \oplus r_2$ and $\mathrm{H}(r_1) \oplus r_3$. Thus, Bob possesses $r_1 \oplus e_1, r_2 \oplus e_2, r_3 \oplus e_3, r_1 \oplus r_2, \mathrm{H}(r_1) \oplus r_3$. He computes the hash value $\mathrm{H}(r_1 \oplus e_1)$. Next he computes $(r_1 \oplus r_2) \oplus (r_2 \oplus e_2) = r_1 \oplus e_2$ and $(r_1 \oplus e_2) \oplus (r_1 \oplus e_1) = e_1 \oplus e_2$. The bit string $e_1 \oplus e_2$ contains considerable information on the bit string $e_1 e_2$. Bob now computes $(\mathrm{H}(r_1) \oplus r_3) \oplus (r_3 \oplus e_3) = \mathrm{H}(r_1) \oplus e_3$ and $(\mathrm{H}(r_1) \oplus e_3) \oplus \mathrm{H}(r_1 \oplus e_1) = \mathrm{H}(r_1) \oplus \mathrm{H}(r_1 \oplus e_1) \oplus e_3$. If e_1 contains no 1, that is, $r_1 = r_1 \oplus e_1$, then we have $\mathrm{H}(r_1) = \mathrm{H}(r_1 \oplus e_1)$. In this case, $\mathrm{H}(r_1) \oplus \mathrm{H}(r_1 \oplus e_1) \oplus e_3 = e_3$. Hence, the Hamming weight of $\mathrm{H}(r_1) \oplus \mathrm{H}(r_1 \oplus e_1) \oplus e_3$ is smaller than $\frac{\alpha+\epsilon}{2}k$ with a high probability. On the other hand, if e_1 contains 1, then $\mathrm{H}(r_1) \oplus \mathrm{H}(r_1 \oplus e_1) \oplus e_3$ is larger than $\frac{\alpha+\epsilon}{2}k$ with a high probability. So we can decide whether or not $e_1 = 0$ by the threshold criterion that Hamming weight of $\mathrm{H}(r_1) \oplus \mathrm{H}(r_1 \oplus e_1) \oplus e_3$ is bigger than or smaller than $\frac{\alpha+\epsilon}{2}k$.

If $e = 0$, then Alice and Bob established the identical key r_1 of size l. If $\mathrm{H}(r_1) \neq \mathrm{H}(r_1 \oplus e_1)$, then Bob guesses e_1 from the information $e_1 \oplus e_2$ (bit twiddling). Then he computes the hash values of the bit string twiddled from $r_1 \oplus e_1$ according to the information $e_1 \oplus e_2$ and compares them with $\mathrm{H}(r_1) \oplus e_3$. Bob can eventually finds e' such that $\mathrm{H}(r_1) = \mathrm{H}(r_1 \oplus e_1 \oplus e')$ (strictly speaking, e' such that the Hamming distance between $\mathrm{H}(r_1) \oplus e_3$ and $\mathrm{H}(r_1 \oplus e_1 \oplus e')$ is smaller than $\frac{\alpha+\epsilon}{2}k$. Since H is locally neighborhood collision free, it is implausible that he finds $e' \neq e_1$ and $\mathrm{H}(r_1) = \mathrm{H}(r_1 \oplus e_1 \oplus e')$. Hence, $e' = e_1$ holds with a high probability and Bob can detect all errors occurred in quantum data transmission. Alice and Bob can delete or correct these error bits $e_1 = e'$ and establish a reconciled key r'_1 of the length slightly shorter than l (when the errors are deleted). We should note that if Alice and Bob correct (not to delete) and reuse the error bits, then they share the reconciled key r_1 of exactly size l. Amplifying privacy, they can reduce enemy's information at their own will. The method is schematically illustrated in Fig. 3.

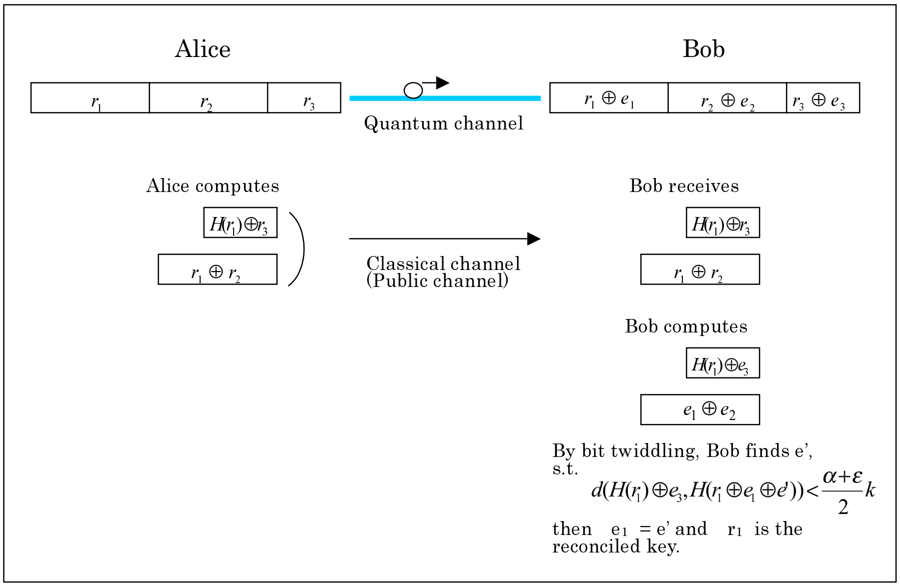

Alice
Bob
r_1 | r_2 | r_3
Quantum channel
$r_1 \oplus e_1$ | $r_2 \oplus e_2$ | $r_3 \oplus e_3$
Alice computes
$H(r_1) \oplus r_3$
$r_1 \oplus r_2$
Classical channel (Public channel)
Bob receives
$H(r_1) \oplus r_3$
$r_1 \oplus r_2$
Bob computes
$H(r_1) \oplus e_3$
$e_1 \oplus e_2$
By bit twiddling, Bob finds e', s.t.
$d(H(r_1) \oplus e_3, H(r_1 \oplus e_1 \oplus e')) < \frac{\alpha+\varepsilon}{2}k$
then e_1 = e' and r_1 is the reconciled key.

Fig. 3. Method 1

We briefly discuss the security of the method. Eve can only obtain information out of communication through the classical channel under the assumption that the process of establishing the shifted key is sound. Thus, Eve can obtain only $r_1 \oplus r_2$ and $\mathrm{H}(r_1) \oplus r_3$. By the mechanism of quantum key distribution scheme, r_1, r_2, r_3 are mutually independent random bit strings. We can consider r_1 and $\mathrm{H}(r_1)$ are encrypted by the one-time pad, also known as the Vernam encryption [14], sacrificing r_2 and r_3, respectively. This implies that Eve can obtain virtually no information as the one-time pad enjoys the perfect secrecy [13].

However, physical implementation problem leaves room for Eve to obtain small amount of information. In the case that Eve wiretapped only small fraction of the total data transmission, succeeded in her attack and obtained partial information of the reconciled key r_1, the information is estimated at most $2\epsilon \times l$ bits. This leaked information can be removed by the privacy amplification process.

4.2 Method 2

We suppose Bob has strong computation power and then discuss a method to reduce the amount of quantum data transmission by demanding Bob substantial computation as a trade-off. Data transmission through the quantum channel costs much more than data transmission through the classical channel and computation, and hence, it is reasonable to require Bob to perform substantial computation if he has abundant computation resource. As before, H is a locally neighborhood collision free function of $\mathbb{Z}_2^l$ to $\mathbb{Z}_2^k$ such that the probability $\theta(H, \epsilon, \alpha)$ is negligible and $\epsilon \ll \alpha$.

Suppose that Alice has $r_1 r_2$ as her sifted key, where $r_1 \in \mathbb{Z}_2^l$ and $r_2 \in \mathbb{Z}_2^k$, whereas Bob has $(r_1 \oplus e_1)(r_2 \oplus e_2)$ as his sifted key, where e_1 and e_2 represent the errors. Alice computes the hash value $\mathrm{H}(r_1)$ and sends $\mathrm{H}(r_1) \oplus r_2$ to Bob through the classical channel. The communication can be considered encrypted by the one-time pad. Note that the amount of bits transmitted is the constant k. Bob computes $\mathrm{H}(r_1 \oplus e_1)$ and $(\mathrm{H}(r_1) \oplus r_2) \oplus (r_2 \oplus e_2) = \mathrm{H}(r_1) \oplus e_2$. If $\mathrm{H}(r_1) = \mathrm{H}(r_1 \oplus e_1)$, then $\mathrm{H}(r_1 \oplus e_1) \oplus \mathrm{H}(r_1) \oplus e_2 = e_2$ and its Hamming weight is approximately $k \times \epsilon$. If $\mathrm{H}(r_1) \neq \mathrm{H}(r_1 \oplus e_1)$, then the Hamming weight of $\mathrm{H}(r_1 \oplus e_1) \oplus \mathrm{H}(r_1) \oplus e_3$ is approximately $k \times \alpha$ since H is locally neighborhood collision free. Since α is sufficiently larger than ϵ, we can conclude with a high probability that $\mathrm{H}(r_1) = \mathrm{H}(r_1 \oplus e_1)$ if the Hamming weight of $\mathrm{H}(r_1) \oplus e_2$ is smaller than $\frac{\alpha+\epsilon}{2}k$, and $\mathrm{H}(r_1) \neq \mathrm{H}(r_1 \oplus e_1)$ otherwise. If $\mathrm{H}(r_1) \neq \mathrm{H}(r_1 \oplus e_1)$, then Bob twiddles randomly up to $\epsilon \times l$ bits of $r_1 \oplus e_1$, computes the hash values of them and then compares with $\mathrm{H}(r_1) \oplus r_2$. Bob can eventually find e_1 by the exhaustive search, however, e_1 has approximately $\epsilon \times l$ bits of 1 and so Bob twiddles only up to about $\epsilon \times l$ bits of $r_1 \oplus e_1$. Clearly Bob's computation task depends on the length of r_1 and the error rate ϵ.

Let us discuss the amount of data transmission through the quantum and classical channels. In Method 1, Alice and Bob have to generate sifted keys of size $2l + k$ to generate a reconciled key of length l bits. The amount of the quantum data transmission is proportion to $2l + k$. The amount of the classical data transmission is $l + k$. In Method 2, on the other hand, the amount of the quantum data transmission is proportion to $l + k$ and the amount of the classical data transmission is k.

Another merit in Method 2 is that information potentially leaked to Eve is reduced compared with Method 1. The reason is that the total (quantum and classical) communication is less than in Method 1. In Method 1, it is estimated that Eve may have stolen at most $\epsilon \times (2k + l)$, whereas in Method 2, at most $\epsilon \times (k + l)$.

A defect of Method 2 is to require Bob considerable amount of computation. If ϵ is small and the length of the established key is small, then Bob's computation can be carried out by a desktop computer. However, if ϵ is large and the key length is long, then the computation becomes an impossible task.

4.3 Method 3

We give an intermediate between Method 1 and Method 2. Suppose H is a locally neighborhood collision free function of $\mathbb{Z}_2^l$ to $\mathbb{Z}_2^k$ such that the probability $\theta(H, \epsilon, \alpha)$ is negligible and $\epsilon \ll \alpha$. Alice has $r_1r_2r_3r_4$ as her sifted key and $r_1, r_2, r_3 \in \mathbb{Z}_2^{(\frac{l}{2})}$ and $r_4 \in \mathbb{Z}_2^k$. Similarly Bob has $(r_1 \oplus e_1)(r_2 \oplus e_2)(r_3 \oplus e_3)(r_4 \oplus e_4)$ as his sifted key, where $e_1, e_2, e_3 \in \mathbb{Z}_2^{(\frac{l}{2})}$ and $e_4 \in \mathbb{Z}_2^k$. The string $e_1e_2e_3e_4$ represents the errors. Alice and Bob intend to establish a reconciled key r_1r_2. The bit string e_1e_2 contains approximately $\epsilon \times l$ bits of 1. Alice computes $r_1 \oplus r_2 \oplus r_3$ and $\mathrm{H}(r_1r_2) \oplus r_4$ and sends it to Bob through the classical channel. Bob computes $(r_1 \oplus r_2 \oplus r_3) \oplus (r_3 \oplus e_3) = r_1 \oplus r_2 \oplus e_3$ and $(\mathrm{H}(r_1r_2) \oplus r_4) \oplus (r_4 \oplus e_4) = \mathrm{H}(r_1r_2) \oplus e_4$. He computes $(r_1 \oplus e_1) \oplus (r_2 \oplus e_2) = r_1 \oplus r_2 \oplus (e_1 \oplus e_2)$, and then $(r_1 \oplus r_2 \oplus e_3) \oplus (r_1 \oplus r_2 \oplus (e_1 \oplus e_2)) = e_1 \oplus e_2 \oplus e_3$. If r_1r_2 is equal to $(r_1 \oplus e_1)(r_2 \oplus e_2) = (r_1r_2) \oplus (e_1e_2)$, then the Hamming distance between $\mathrm{H}(r_1r_2) \oplus e_4$ and $\mathrm{H}((r_1 \oplus e_1)(r_2 \oplus e_2))$ is approximately $\epsilon \times k$. On the other hand, if r_1r_2 is not equal to $(r_1 \oplus e_1)(r_2 \oplus e_2)$, then the Hamming distance between $\mathrm{H}(r_1r_2) \oplus e_4$ and $\mathrm{H}((r_1 \oplus e_1)(r_2 \oplus e_2))$ is more than $\alpha \times k$. Since α is sufficiently larger than ϵ, Bob can decide whether or not $e_1e_2 = 0$ by the threshold criterion that the Hamming distance between $\mathrm{H}(r_1r_2) \oplus e_4$ and $\mathrm{H}((r_1 \oplus e_1)(r_2 \oplus e_2))$ is bigger or smaller than $\frac{\epsilon+\alpha}{2}k$. If $\mathrm{H}(r_1r_2) = \mathrm{H}((r_1 \oplus e_1)(r_2 \oplus e_2))$, then Alice and Bob agree the reconciled key r_1r_2. If $\mathrm{H}(r_1r_2) \neq \mathrm{H}((r_1 \oplus e_1)(r_2 \oplus e_2))$, then Bob guesses e_1e_2 using the information $e_1 \oplus e_2 \oplus e_3$ (bit twiddling). Clearly it is much easier to find e_1e_2 than Method 2, but more difficult than Method 1.

For Alice and Bob to establish a reconciled key of length l, r_1r_2 must be of length l. Note that $|r_1| = |r_2| = |r_3| = \frac{l}{2}$ and $|r_4| = k$. Hence, Alice and Bob have to generate a sifted key of length $\frac{3l}{2} + k$. If we ignore k, they need to generate a bit string of length almost $\frac{3l}{2}$ of the reconciled key length whereas sifted keys of size $2l$ and l are required in Method 1 and Method 2, respectively.

4.4 Method Using Error Correcting Codes

We briefly discuss a method using error correcting codes. Suppose H is a locally neighborhood collision free function of $\mathbb{Z}_2^l$ to $\mathbb{Z}_2^k$ such that the probability $\theta(H, \epsilon, \alpha)$ is negligible and $\epsilon \ll \alpha$. To correct the errors in sifted keys of Alice and Bob, Alice may want to encode her sifted key by a classical error correcting code and transmit only the redundancy part of the encoded sifted key. However, the redundancy part gives substantial information of Alice's sifted key, and hence, the redundancy part must be encrypted to prevent Eve from obtaining any information. We propose to encrypt the redundancy part by the one-time pad. Suppose Alice has r_1r_3 as her sifted key, where $r_1 \in \mathbb{Z}_2^l$ and $r_3 \in \mathbb{Z}_2^k$, and Bob has $(r_1 \oplus e_1)(r_3 \oplus e_3)$ as his sifted key, where $e_1 \in \mathbb{Z}_2^l$ and $e_3 \in \mathbb{Z}_2^k$. Alice computes the redundancy (denoted by $\mathrm{C}(r_1)$) of the encoded word of r_1 by the error correcting code C. Bob can detect and correct the error bit string e_1 if he has most correct bits of $\mathrm{C}(r_1)$ with his sifted key $r_1 \oplus e_1$. Alice sends $\mathrm{C}(r_1) \oplus r_3$, and hence, $\mathrm{C}(r_1)$ is encrypted by the one-time pad and so it gives virtually no information to Eve even if she can eavesdrop it. Bob can compute $(\mathrm{C}(r_1) \oplus r_3) \oplus (r_3 \oplus e_3) = \mathrm{C}(r_1) \oplus e_3$. Hence, if the error rate is small enough, then Bob can correct the error bits due to the error-correcting ability of C. For instance, we can use the Reed-Solomon code [11] for our purpose because of its capability of correcting random errors. Note that we may assume that errors distribute uniformly all over the sifted keys because Alice and Bob operated a random permutation to their sifted keys after the sifting process.

4.5 Authentication

After generating a reconciled key, Alice and Bob carry out privacy amplification and obtain their private key. Next they confirm the integrity of their private key. We can employ the same idea to authenticate a private key. We should note that the existing methods basically require the previously shared authenticated private key, while ours do not. Suppose that after the privacy amplification process, Alice has her private key r_1 and Bob has his private key r_1', where $r_1, r_1' \in \mathbb{Z}_2^l$. When making their raw keys, Alice and Bob generate extra sifted keys r_3 and $r_3 \oplus e_3$, respectively, where $r_3 \in \mathbb{Z}_2^k$ and e_3 represents the errors. Alice sends $\mathrm{H}(r_1) \oplus r_3$ to Bob. This transmission is considered as encrypted by the one-time pad, and hence, Eve obtains virtually no information. Bob checks whether or not the Hamming distance between $\mathrm{H}(r_1) \oplus r_3$ and $\mathrm{H}(r_1 \oplus e_1)$ is smaller than the threshold $\frac{\epsilon+\alpha}{2}k$. If so, $r_1 = r_1 \oplus e_1$ and $e_1 = 0$, otherwise, $r_1 \neq r_1 \oplus e_1$. This authentication method can be applied after the error correction process. We also note that the method can be employed after any error correction and privacy amplification method.

5 Experimental Results

To implement our error detection method, we need a concrete locally neighborhood collision free function. We show by experiment with computers that SHA-1 [9] and MD5 [12] satisfy the locally neighborhood collision free property. If a function H satisfies the locally neighborhood collision free property, then the Hamming distance of $\mathrm{H}(x_1)$ and $\mathrm{H}(x_2)$ is expected to be relatively large with a high probability for any bit strings x_1, x_2 having a small Hamming distance. In our experiments, we choose randomly $N = 100,000,000$ pairs (x_1, x_2) of bit strings having Hamming distance 1 (10, 20, respectively). Then we count the frequency of the Hamming distance of the pair $(\mathrm{H}(x_1), \mathrm{H}(x_2))$. If H is a cryptographic hash function, we easily imagine that H exhibits a normal distribution. If the standard deviation is relatively small, that is, most samples yields a Hamming distance close to the mean value, then we can conclude that it is a good neighborhood collision free function.

We consider SHA-1 as a function of $\mathbb{Z}_2^{512}$ to $\mathbb{Z}_2^{160}$, that is, we restrict its domain to $\mathbb{Z}_2^{512}$ in our experiments. We expect the mean value to be 80, and Hamming distance $d(\mathrm{H}(x_1), \mathrm{H}(x_2))$ is close to 80 for most pairs (x_1, x_2). Actually, our experiments for SHA-1 with $10,000,000$ samples of Hamming distance $1(10, 20)$ show that the mean value is about 80, the standard deviation is 6.3, the minimum of $d(\mathrm{H}(x_1), \mathrm{H}(x_2))$ is 44, and the maximum of $d(\mathrm{H}(x_1), \mathrm{H}(x_2))$ is 115. See Table 1 for the statistic and Fig. 4 and Fig. 5 for the histograms in Appendix. Our experiments show that the deviation is small enough. Hence, SHA-1 has the good neighborhood collision free property, and hence, most pairs of bit strings with Hamming distance 1 are mapped to the strings with Hamming distance close to 80. For example, we may set $\alpha = \frac{1}{4}$. Then the probability $\theta(\mathrm{H}, \alpha, \epsilon)$ is negligible for any error rate $0 < \epsilon < \alpha$. In this case, the threshold value is around $\frac{\epsilon + \frac{1}{4}}{2} \times 180$.

We consider MD5 as a function of $\mathbb{Z}_2^{512}$ to $\mathbb{Z}_2^{128}$. Hence, we expect the mean value to be 64, and Hamming distance $d(\mathrm{H}(x_1), \mathrm{H}(x_2))$ is close to 64 for most pairs (x_1, x_2). Our experiments for MD5 with $10,000,000$ samples of Hamming distance $1(10, 20)$ show that the mean value is about 64, the standard deviation is 5.6, the minimum of $d(\mathrm{H}(x_1), \mathrm{H}(x_2))$ is 34, and the maximum of $d(\mathrm{H}(x_1), \mathrm{H}(x_2))$ is 95. See Table 1 for the statistic and Fig. 4 and Fig. 5 for the histograms in Appendix.

Our experiments show that the deviation is small enough. Hence, MD5 has the good neighborhood collision free property, and hence, most pairs of bit strings with Hamming distance 1 are mapped to the strings with Hamming distance close to 64. For example, we may set $\alpha = \frac{1}{4}$. Then the probability $\theta(\mathrm{H}, \alpha, \epsilon)$ is negligible for any error rate $0 < \epsilon < \alpha$. In this case, the threshold value is around $\frac{\epsilon + \frac{1}{4}}{2} \times 128$.

Aknowledgement
The authors thank Yasuyuki Sakai for valuable suggestions.

References

1. Bennett,C.H., Brassard,G.: Quantum cryptography: Public-key distribution and coin tossing. Proc. Int. Conf. on Computers, Systems and Signal Processing, Bangalore, India (1984) 175–179
2. Bennett,C.H.: Quantum Cryptography Using Any Two Nonorthogonal States. Phys. Rev. Lett. **68** (1992) 3121–3124
3. Bennett,C.H., Brassard,G., Crépeau,C., Maurer,U.M.: Generalized Privacy Amplification. IEEE Trans. Information Theory **41** (1995) 1915–1923
4. Bennett,C.H., Brassard,G., Robert,J.M.: Privacy Amplification by Public Discussion. SIAM J Comput. **17** (1988) 210–229
5. Bennett,C.H., Bessette,F., Brassard,G., Salvail,L., Smolin,J.: Experimental Quantum Cryptography. J.Cryptology **5** (1992) 3–28
6. Bouwmeester,D., Ekert,A., Zeilinger,A.: The Physics of Quantum Information. Springer-Verlag, Berlin Heidelberg New York (2000)

7. Chor,B., Goldreich,O., Hastad,J., Freidmann,J., Rudich,S., Smolensky,R.: The Bit Extraction Problem or t-resilient Functions. 26th IEEE Symp. Foundations of Computer Science, (1985) 396–407
8. Ekert,A.K.: Quantum Cryptography Based on Bell's Theorem. Phys. Rev. Lett. **67**, No.6 (1991) 661–663
9. FIPS 180-1: Secure Hash Standard, Federal Information Processing Standard (FIPS), Publication 180-1, National Institute of Standards and Technology, US Department of Commerce, Washington D.C., April (1995)
10. Maurer,U.M.: Secret Key Agreement by Public Discussion from Common Information. IEEE Trans. Information Theory **39** (1993) 733–742
11. Reed, I.S., Solomon, G.: Polynomial Codes over Certain Finite Fields, J. Soc. Indust. Appl. Math. **8** (1960) 300–304
12. Rivest,R.L.: The MD5 Message-digest algorithm, Request for Comments (RFC) 1321, Internet Activities Board, Internet Task Force, April (1992)
13. Shannon,C.E.: Communication Theory of Secrecy Systems. Bell Syst.Tech.J. **28** (1948) 656–715
14. Vernam,G.S.: Cipher Printing Telegraph Systems for Secret Wire and Radio Telegraphic Communications. J. Amer. Inst. Elect. Eng. **55** (1926) 109–115
15. Zbinden,H., Bechmann-Pasquinucci,H., Gisin,N., Ribordy,G.: Quantum Cryptography. Applied Physics B **67** (1998) 743–748
16. "http://www.mitsubishielectric.com/pressreleases/2000/mel0502.html"

Appendix: Statistic and Histogram

Table 1. Statistic of Experiments on SHA-1 and MD5

Algorithm	ID	#data	MEAN	S.D.	max.h.d	min.h.d
SHA-1	1	10^8	80.000029	6.327076	115	44
	10	10^8	80.004204	6.334314	109	49
	20	10^8	79.994482	6.321717	111	47
MD5	1	10^8	63.999359	5.656389	95	34
	10	10^8	63.998326	5.658194	93	38
	20	10^8	63.995178	5.655455	92	37

In Fig. 4 and Fig. 5, the graph labeled by Dis:1, Dis:10 and Dis:20 shows the histogram of Hamming distance of 1, 10 and 20, respectively.

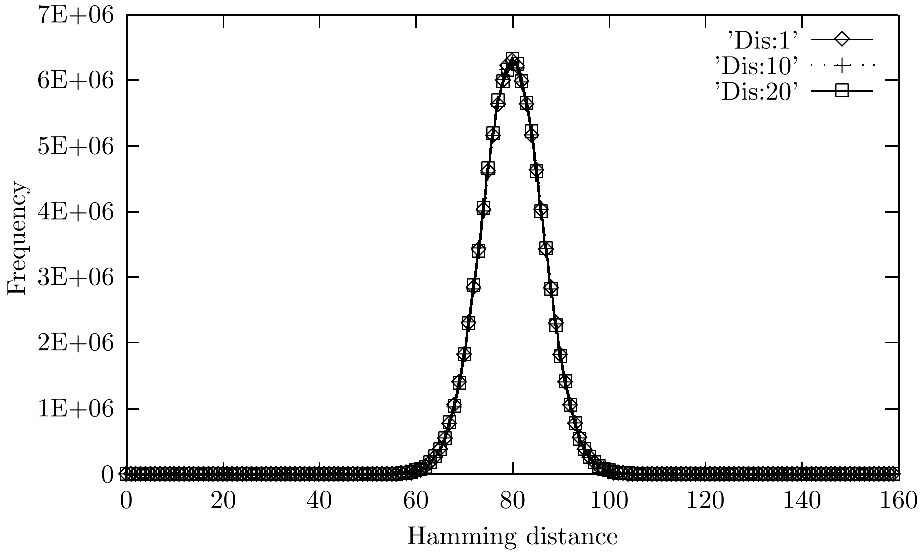


Fig. 4. Hamming Distance Histogram of SHA-1

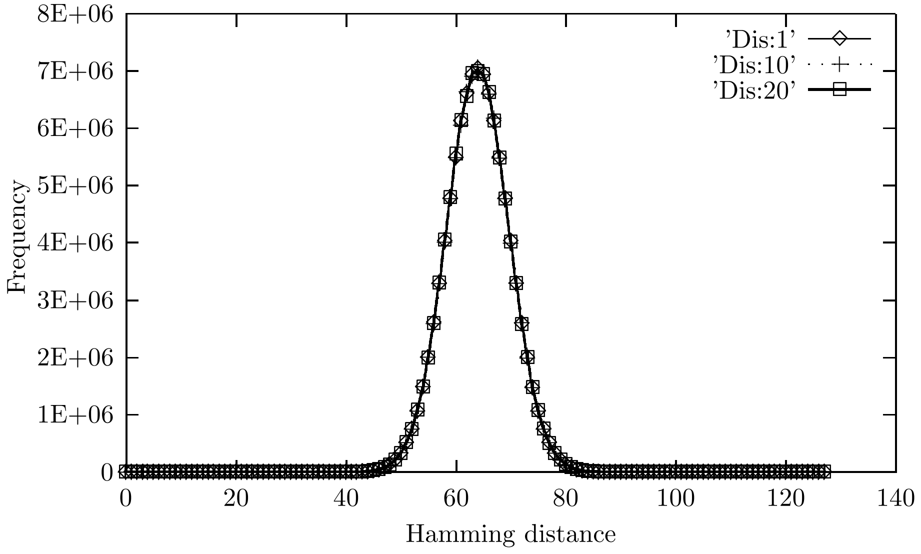


Fig. 5. Hamming Distance Histogram of MD5

An Axiomatic Basis for Reasoning about Trust in PKIs

Chuchang Liu, Maris Ozols, and Tony Cant

Information Technology Division
Defence Science and Technology Organisation
PO Box 1500, Salisbury
SA 5108, Australia
{Chuchang.Liu,Maris.Ozols,Tony.Cant}@dsto.defence.gov.au

Abstract. Trust is essential to a communication channel. The trust relationships, which play an important role in Public Key Infrastructures (PKIs), need to be formalized for providing a reliable modelling methodology to support secure digital communications. In this paper, we present a typed modal logic used for specifying and reasoning about trust in PKIs. In order to study trust relationships within PKIs, we define **TA** (a set of trust axioms), **TB** (a trust base) and **TC** (a set of trusted certificates). In our method, the trust relation in a given PKI is formalized by trust axioms. Based on trust axioms, an agent can have its own trust base that contains all agents whom the agent trusts, and can derive and extend its trusted certificates set. The trust theory for a given PKI, which consists of our modal logic and a set of trust axioms proposed for the PKI, is the basis of the certificate verification function.

Keywords: certificate, CA (Certificate Authority), PKI (Public Key Infrastructure), trust, trust theory, certificate verification, information security.

1 Introduction

Public key technology within Public Key Infrastructure (PKI) has widely been recognised as a fundamental technology for supporting secure digital communication (for example: electronic commerce and secure messaging). A PKI can be viewed as a system consisting of the entire, generally heterogeneous, set of components, which are involved in issuing, rekeying, revoking and managing public key certificates. It has two essential relations, the *certification* relation and the *trust* relation.

The certification relation is usually defined based on the roles that agents (or participants) play in the PKI. For instance, RFC 1422 [7] defines a rigid hierarchical structure for the Internet Privacy Enhanced Mail (PEM) [4]. There are three types of PEM CAs: Internet Policy Registration Authority (IPRA) acts as the root of the PEM certification hierarchy at level 0, and issues certificates only for the next level of authorities, called PCAs (Policy Certification Authorities); PCAs, at level 1 of the PEM certification hierarchy, take the responsibility for establishing and publishing the certification policy with respect to certifying users or subordinate certification authorities; and CAs, which are at level 2 of the hierarchy and can also be at lower levels (those at level 2 are certified by PCAs). We will adopt the concept of PKI certification topology, proposed by Liu *et. al.* [10], to describe the certification path architecture of a PKI.

The trust relation is somewhat different from the certification relation. It captures agents' beliefs and can be modelled by belief logics similar to the BAN logic [2]. Trust depends on the observer (agent), and there is no absolute trust. Two different agents may not equally trust any received information. A message may carry some information, and different people (agents) may act differently depending whether they believe this information or not.

Linguistically, "trust" is closely related to "true" and "faithful", with a usual dictionary meaning of "assured reliance on the character, the integrity, justice, etc., of a person, or something in which one places confidence". So, in common English usage "trust" is what one places his confidence in, or, expects to be truthful. In a PKI, one of the main concerns for an agent is whether a certificate is trustworthy or not. In managing public key certificates, a PKI provides mechanisms allowing an agent to determine whether a needed certificate can be trusted (or, in the agent's view, that the certificate is valid).

PKIs simplify key management but create trust management problem [6]. Blaze *et. al.* [1] have identified such trust management problems as a distinct and important component of security in network services. Recently, several trust models with PKIs have also been proposed, which involve the development of effective formalisms used to define and express trust relations between entities involved in a PKI [14], and the investigation of techniques for dealing with trust management [1,3,9] and the uncertainty in a trust model [5,8,11]. In a PKI, what makes a public key certificate trustworthy? How can one specify and reason about trust? We have to deal with these sorts of questions. However, trust models and management techniques in present implementations are very limited. More rich trust models and new techniques for specifying and reasoning about trust for PKIs are therefore highly desirable.

Trust is essential to a communication channel. The trust relationships, which play an important role in Public Key Infrastructures (PKIs), need to be formalized for providing a reliable modelling methodology to support secure digital communications. This paper presents an axiomatic approach to the description of trust in PKIs. It proposes a typed modal logic for specifying and reasoning about trust in a PKI, which is an extension of first-order logic with typed variables and modal operators representing agents' beliefs. In order to study the trust relationship withing a PKI, we define **TA** (a set of trust axioms), **TB** (a trust base) and **TC** (a set of trusted certificates). In our model, the trust relation in a PKI is formalized by **TA**. Based on **TA**, the set of trust axioms, an agent can have its own **TB**, the trust base, that contains all "persons" whom the agent trusts. The agent can also derive and extend its **TC**, i.e., the set of certificates trusted by itself. The axiomatic approach proposed in the paper allows us to build a trust theory that consists of the logic and a set of trust axioms for a given PKI. The trust theory is a basis for a main client function, the certificate verification function.

This paper is structured as follows. Section 2 discusses the format of PKI certificates. Section 3 is a brief introduction to the state-based model for PKIs, and Section 4 talks about trust relationship involved in PKIs. Section 5 presents a logc for trust transferring in PKIs. Section 6 discusses trust ABC: trust axioms (TA), trust bases (TB) and trust certificates (TC) for a PKI. Section 7 discusses the application of our method in the certificate path validation. The last section concludes the paper with a short discussion about possible future work.

2 PKI Certificates

The PKI entities, which we call *agents* in this paper, are classified into two classes: *Certification Authorities* (CAs)[1] and *Users*. CAs can have their own certificates, and they also issue certificates for others within the PKI. Users, also called *End Entities* (EEs), are people or devices that may hold certificates issued by some CAs, but cannot issue valid certificates themselves.

Without loss of generality, we assume that PKI certificates have a "standard" public-key certificate format, which contains the basic information that most kinds of public key certificates should provide as follows: the name of the certificate issuer, the start and expiry dates, the subject (i.e., the name of the holder of the private key for which the corresponding public key is being certified), the value of the public key, the extension field, and the signature of the issuer. Formally, we define a PKI certificate to have the following form:

$$\mathtt{Cert}\ (\!|\mathtt{I}, \mathtt{DS}, \mathtt{DE}, \mathtt{S}, \mathtt{PK}, \mathtt{E}, \mathtt{Sig}|\!)$$

where $\mathtt{I}$ is the issuer, $\mathtt{DS}$ and $\mathtt{DE}$ are the start date and expiry date respectively, $\mathtt{S}$ is the subject of the certificate, $\mathtt{PK}$ is the value of the public key for $\mathtt{S}$, $\mathtt{E}$ is the value of the extension field, and $\mathtt{Sig}$ holds the signature of the issuer $\mathtt{I}$.

Given a certificate

$$C = \mathtt{Cert}\ (\!|\mathtt{I}, \mathtt{DS}, \mathtt{DE}, \mathtt{S}, \mathtt{PK}, \mathtt{E}, \mathtt{Sig}|\!)$$

the following projection functions can be used to obtain the value of each component contained in C:

$$\begin{array}{lll} \overline{\mathtt{I}}(C) = \mathtt{I} & \overline{\mathtt{DS}}(C) = \mathtt{DS} & \overline{\mathtt{DE}}(C) = \mathtt{DE} \\ \overline{\mathtt{S}}(C) = \mathtt{S} & \overline{\mathtt{PK}}(C) = \mathtt{PK} & \overline{\mathtt{E}}(C) = \mathtt{E} \\ \overline{\mathtt{Sig}}(C) = \mathtt{Sig} & & \end{array}$$

The public key $\overline{\mathtt{PK}}(C)$ is bound to the entity $\overline{\mathtt{S}}(C)$, the subject of the certificate. The private key corresponding to the public key $\overline{\mathtt{PK}}(C)$ is denoted by $SK(\overline{\mathtt{PK}}(C))$. Therefore, the key pair possessed by the subject is $(\overline{\mathtt{PK}}(C), SK(\overline{\mathtt{PK}}(C)))$.

[1] We do not consider *Registration Authorities* (RAs) as separate entities. RAs carry out parts of the CA function, and are logically part of the CA, but are implemented elsewhere for performance, cost and usability reasons.

The extension field of a certificate may include

– an extension named `authorityKeyIdentifier` for providing a means to identify the particular private key used to sign the certificate, and
– an extension named `subjectKeyIdentifier` for differentiating the keys held by the subject.

There are no requirements for PKI implementations to process these extensions. However, for our purposes, we assume that in certificate process, for a given certificate C, the identifier of the certificate authority's key and the identifier of the certificate subject's key can be identified by the extensions `authorityKeyIdentifier` and `subjectKeyIdentifier`, respectively.

In practice, a PKI may use a certificate format different from the standard format given above. However, any PKI certificate format should consist of two parts: the data part and the signature part of the certificate issuer. In the standard public key certificate format, $\overline{\mathtt{Sig}}(C)$ is the signature part of the certificate C, while the data part is a combination of the values of all other components to be signed. We write `tbs` representing "to be signed", and define:

$$\overline{\mathtt{tbs}}(C) = (\mathtt{I}, \mathtt{DS}, \mathtt{DE}, \mathtt{S}, \mathtt{PK}, \mathtt{E}),$$

then $\overline{\mathtt{tbs}}(C)$ is just the data part of the certificate, i.e. the argument to signature function carried out by the certificate issuer.

3 A State-Based Model for PKIs

We now give a brief introduction to the state-based model for PKIs, which is based on the model proposed by Liu *et. al.* [10] for CMSs (Certificate Management Systems).

In our view of a PKI, all the agents of the PKI are organized based on a certification relation over the set of these agents. That is, for any pair of agents, say A and B, if B is within the domain of agents which A may potentially certify (for example, A is an organisation and B is an employee), we write $A \downharpoonleft B$, and call $\downharpoonleft$ the certification relation of the PKI.

We define the *total certificate set* of a given PKI, denoted as $\mathcal{C}$, to be the set of all certificates issued by CAs in the PKI. This definition indicates that any certificate issued by some CA should belong to the total certificate set $\mathcal{C}$, because it contain "all" certificates.

At any moment in time, an agent in the PKI should hold zero or more certificates. Also, for a CA, it is at times necessary to revoke certificates, for example when the certificate holder leaves the issuing organization or when the private key is compromised. A mechanism defined in X.509 for revoking certificates is the *Certificate Revocation List* (CRL). A CRL is a list, signed by a CA, of unexpired, revoked certificates. In our model, we assume that any agent is associated with a CRL issued by itself periodically. However, if the agent is an end-entity (EE), the CRL should be empty, because we assume no EE will issue certificates to others and cannot therefore revoke any certificates. Thus, we define PKI states as follows:

We call $\langle \Omega, \downarrow \rangle$ the *topology* of a given PKI where Ω is the set of all agents in the PKI and $\downarrow$ the certification relation. Let $\mathcal{C}$ be the total certificate set of the PKI. Then a *state* s of the PKI is a relation from Ω to $2^{\mathcal{C}} \times 2^{\mathcal{C}}$ satisfying the following conditions: For any $A \in \Omega$,

(1) there exists an unique set $\zeta(\subset \mathcal{C})$ associated with A such that $C \in \zeta$ if and only if A is the subject of C, and
(2) there exists an unique set $\eta(\subset \mathcal{C})$ associated with A such that $C \in \eta$ if and only if A is the issuer of C and it has revoked the certificate C.

where $2^{\mathcal{C}}$ is the power set of $\mathcal{C}$. Under a state s, we call $s(A, \zeta, \eta)$ a triple, where $\zeta(\subset \mathcal{C})$ is a set of certificates issued to A and $\eta(\subset \mathcal{C})$ is a set of certificates issued by A.

Let s be a PKI state. If we have $s(A, \zeta, \eta)$, then ζ is called the *possessed certificate set* of A, which lists all certificates possessed by A at the state s, and η is called the *revoked certificate set* of A, which represents the CRL issued by A at the state s. In the following, we will often use $\mathbf{PCS}_A$ and $\mathbf{CRL}_A$ to denote the possessed certificate set and the revoked certificate set of an agent A, respectively, at a given PKI state.

The state of a PKI can be changed by application of some PKI functions, such as certificate issuing, certificate rekeying and certificate revocation. These actions could be viewed as transitions which change one PKI state into another. Thus, a PKI can be described as a state machine. In the following, we focus on discussing the role of trust in the certificate verification based on a given PKI state, so we do not attempt to consider the formalization of the state changes here, which will be covered in future work.

4 The Trust Relation

In a PKI, the operations CAs may execute include: issuing, revoking and rekeying certificates. We make the following assumptions concerning trust between the agents in it:

(1) All agents (CAs and users) trust all CAs to faithfully execute their CA operations; and
(2) All agents trust that it is not viable to tamper with PKI certificates.

These assumptions can be well founded and supported by PKI practices. Firstly, assurance is provided for (1) through the use of accreditation of CAs, Certificate Practice Statements published by CAs and the implementation of appropriate policy[2]. Assurance is provided for (2) through the use of digital signatures, and good control of private keys.

[2] Note that policy for CAs can be listed and checked in much the same way as in which certificates are checked and can even be included as an extension in certificates.

In the following, we focus on discussing a trust relation that is tightly related to the certificate verification. Let $\langle \Omega, \downarrow \rangle$ be the topology of a given PKI. Then the trust relation of the PKI is a binary relation over Ω, i.e., a subset of $\Omega \times \Omega$. For any $A, B \in \Omega$, if (A, B) belongs to the subset, we say that A trusts B, denoted as $A \Uparrow B$.

Note that in general the trust relation may not have the following properties: transitivity, and symmetry, i.e., we cannot obtain the conclusion '$A_1 \Uparrow A_3$' from '$A_1 \Uparrow A_2$' and '$A_2 \Uparrow A_3$', and cannot derive the formula '$A_1 \Uparrow A_2$' from '$A_2 \Uparrow A_1$'. These are consistent with the model of trust in the real world: A man may not trust his friend's friend although he may trust his friend and his friend may also trust the friend of herself; and the fact that Alice trusts Bob does not necessarily mean that Bob should trust Alice. However, the trust relation can be reflexive: an agent may trust himself. This property will be expressed as a trust axiom in our model (see Section 6).

In our model, both the certification and trust relations are a binary relation over the set of agents. The difference between the two relations is that the certification relation is static in our model whilst the trust relation may dynamically change from time to time because agents may change their beliefs.

Yahalom *et. al.* [14] identied and described various types of trust, and used the term *trusts* subscripted with types to represent that an agent trusts another in some aspect. For exmaple, $A \Uparrow_{kg} B$ can be intepreted to mean that an agent A trusts an agent B with resepct to quality random key generation. In general, the expression $A \Uparrow_x B$ means that an agent A trusts an agent B with resepct to x, where x is a variable ranging over trust types. In our model, we do not use subscripts attached to the trust relation, and leave a freedom for the PKI designer or someone who concerns reasoning about trust to explain the trust relation $\Uparrow$. For more explanation, see Section 6.

5 A Logic for Trust Transferring in PKIs

A *theory* is based on a *logic*. Briefly, a logic of any sort consists of a *language*, a set of *axioms* and a set of *rules of inference*. The language defines the set of *well-formed formulas* (WFFs) in the logic. An axiom is a WFF and a rule of inference is a transformation from one WFF to another. A theory consists of a logic and a set of WFFs called *proper axioms*. A *proof* starts out from the axioms, repeatedly uses rules of inference and arrives at a WFF. A WFF that is the result of a proof is called a *theorem* of the theory.

The logic we adopt in this paper is a typed modal logic, which is an extenstion of first-order logic with variables typed and modal operators expressing beliefs of a rational agent. A *trust theory* for a given PKI consists of the logic and trust axioms which we discuss late.

5.1 The Syntax

In this logic, all variables as well as functions are typed. Examples of simple types are numerical numbers and boolean values. For our purpose, we introduce the following primitive types: Ω (a set of agents), $\mathcal{C}$ (a set of certificates), $\mathcal{K}$ (a set of keys), and $\mathcal{N}$ (the set of natural numbers). Other types may be introduced at any time as the need arises.

In particular, we use

- $A, B, A_1, A_2, \ldots$ agent variables ranging over the type Ω,
- $C, C_1, C_2, \ldots$ certificate variables ranging over the type $\mathcal{C}$,
- $PK, PK_1, PK_2, \ldots$ public key variables ranging over the type $\mathcal{K}$,
- $SK, SK_1, SK_2, \ldots$ private key variables ranging over the type $\mathcal{K}$, and
- $T, T_1, T_2, \ldots$ time variables ranging over the type $\mathcal{N}$.

We may also use agent constants *alice*, *bob*, ...; certificate constants $c, c_1, c_2, \ldots$; public key constants pk, $pk_1, pk_2, \ldots$; private key constants sk, $sk_1, sk_2, \ldots$; and time constants $t, t_1, t_2, \ldots$, and a special time constant *today* representing the current time.

An n-ary function symbol represents functions of n variables, for finite integer n. The types of all functions are defined. The main functions include all functions given in Section 2, such as $\overline{\mathtt{I}}(C)$, $\overline{\mathtt{S}}(C)$, $\overline{\mathtt{DS}}(C)$, $\overline{\mathtt{Sig}}(C)$, $\overline{\mathtt{tbs}}(C)$, $\overline{\mathtt{PK}}(C)$ and so on. We also have some variable and constant symbols for representing certificate sets, such as $\mathbf{PCS}_A$ and $\mathbf{CRL}_A$ that we introduce for representing the possessed certificates set and certificate revoked list of an agent A.

We write PK and SK for the public and private keys associated with a public key pair K, that is, $K \equiv (PK, SK)$ means that the public key of the key pair K is PK and the private key corrensponding to the public key is SK. Note that, as we said before, no one can calculate the private key from the public key although the corresponding relation has been represented by the formula.

Let X be a public key or a private key. Thus, we have the following notations to define encryptions and decryptions: $\{M\}_X$ represents M encrypted under the key X, and $\langle M \rangle_X$ represents M decrypted under the key X.

In our logic, we distinguish two different concepts, *messages* (in the first-order logic, called terms) and *formulae*. In our logic, messages can be names of agents, certificates, public keys, private keys, dates, strings having particular meanings, or other things. They can also be a combination (or sequence) of other messages. Messages are not formulae although formulas are built from messages. Only formulae can be true or false or have agent's beliefs attributed to them. Formally, messages can inductively be defined as follows:

- X is a message if X a variable or a constant representing an agent, a certificate, a public key, a private key, a time, or a string such as an extension field value of a certificate.
- $F(X_1, \ldots, X_n)$ is a message if $X_1, \ldots, X_n$ are messages and F is any function.

In first order logic, with a given n-ary predicate symbol such as p, in the formula $p(e_1, \ldots, e_n)$ all $e_1, \ldots, e_n$ are defined on the same domain in a given interpretation. Our logic is a typed modal logic, so in the predicate $p(X_1, \ldots, X_n)$,

$X_1, \ldots, X_n$ may be defined with different types. Therefore, in this logic, an interpretation of a formula should be based on the corresponding types of variables appearing in the formula.

In the vocabulary of our logic, apart from variables, function and predicate symbols, we have the primitive propositional connectives, $\neg$ and $\wedge$, universal quantifier "$\forall$" and modal operators: $\mathbb{B}_A$, for all $A \in \Omega$. The formulae of the logic are therefore inductively defined as follows:

- $p(X_1, \ldots, X_n)$ is a formula if p is a n-ary predicate symbol and $X_1, \ldots, X_n$ are the terms (messages) with corresponding types to p. In particular, we have:
  (1) $A \downarrow B$ and $A \Uparrow B$ are formulae when A and B are agents.
  (2) $X \in S$ is a formula if X is a certificate and S is a set of certificates or X is an agent and S is a set of agents.
  (3) $X = Y$ is a formula if X and Y are messages.
  (4) $\mathtt{Valid}(X)$ if X is a certificate, the signature of a certificate, a key or a key pair.
- $\neg\varphi$ and $\varphi \wedge \psi$ are formulae if φ and ψ are formulae.
- $\forall X \varphi(X)$ is a formula if X is a free variable in the formula $\varphi(X)$.
- $\mathbb{B}_A\varphi$ is a formula if φ is a formula, for all agent $_A$.

Here, most of the expressions just given either are standard notation or have been defined before. We only need to give a brief description for the following: '$\mathtt{Valid}(X)$' means that X is valid where X may be a certificate or a key or something else, and, for '$\mathbb{B}_A\varphi$', $\mathbb{B}_A$ is read as "agent A believes", so it means that the agent A believes φ. In the language, other connectives, $\vee, \rightarrow$ and $\leftrightarrow$, and $\exists$ can be defined in the usual manner.

5.2 The Proof System

The proof system consists of a set of axioms and a set of rules of inference. Our logic has the following rules of inference:

Modus Ponens:	From φ and $\varphi \rightarrow \psi$ infer ψ.
Instantiation:	From $\forall X \varphi(X)$ infer $\varphi(Y)$.
Generalisation:	From $\varphi(X)$ infer $\forall X \varphi(X)$.
Necessitation:	From $\vdash \varphi$ infer $\vdash \mathbb{B}_A\varphi$.

where X is a free variable and A is any agent. $\vdash$ is a metalinguistic symbol. '$\Gamma \vdash \varphi$' means that φ is derivable from the set of formulae Γ (and the axioms). '$\vdash \varphi$' means that φ is a theorem, i.e., derivable from axioms alone.

Apart from all instances of tautologies of classical first logic, our logic also has the following axiom schemata:

(A1) $\mathbb{B}_A(\varphi \to \psi) \wedge \mathbb{B}_A\varphi \to \mathbb{B}_A\psi$, for any formulae φ and ψ.

(A2) $\mathbb{B}_A\varphi \to \mathbb{B}_A(\mathbb{B}_A\varphi)$, for any formula φ.

(A3) $\forall C(\texttt{Valid}(C) \to \texttt{Valid}(\overline{\texttt{PK}}(C)))$.

(A4) $\forall K(K = (\overline{\texttt{PK}}(C), SK(\overline{\texttt{PK}}(C))) \wedge \texttt{Valid}(\overline{\texttt{PK}}(C)) \to \texttt{Valid}(K))$.

(A5) $\forall K \forall M(K = (PK, SK) \wedge \texttt{Valid}(K) \to (\langle\{M\}_{SK}\rangle_{PK} = M))$.

(A6) $\forall K \forall M(K = (PK, SK) \wedge \texttt{Valid}(K) \to (\langle\{M\}_{PK}\rangle_{SK} = M))$.

(A7) $\forall C(\exists C'(\texttt{Valid}(C') \wedge (\overline{\texttt{I}}(C) = \overline{\texttt{S}}(C'))\wedge$
$(\overline{\texttt{tbs}}(C) = \langle\overline{\texttt{Sig}}(C)\rangle_{\overline{\texttt{PK}}(C')})) \to \texttt{Valid}(\overline{\texttt{Sig}}(C)))$.

(A8) $\forall C(\texttt{Valid}(\overline{\texttt{Sig}}(C)) \wedge today \geq \overline{\texttt{DS}}(C)\wedge$
$today < \overline{\texttt{DE}}(C) \wedge \neg(C \in \mathbf{CRL}_{\overline{\texttt{I}}(C)}) \to \texttt{Valid}(C))$.

Axiom (A1) says that every agent believes everything that can logically be derived from his beliefs. Axiom (A2) says in effect that an agent knows and is able to tell what he believes. Axiom (A3) says that, if a certificate is valid, then the public key contained in the certificate is valid. Axiom (A4) says that, if the public key bound to the subject of a certificate is valid, then the key pair consisting of the public key and the private key corresponding to it is valid. Axiom (A5) says that, for any message M, we have $\langle\{M\}_{SK}\rangle_{PK} = M$ if the key pair (PK, SK) is valid. The meaning of Axiom (A6) is similar to (A5). Axioms (A7) and (A8) allow agents to verify the signature of a certificate as well as the certificate itself based on another certificate whose validity has been proved.

Note that digital signature algorithms usually involve use of a hash function. However, to simplify our discussion, we do not consider this. So, in axiom (A7), to verify the signature of the certificate C, we only check whether $\overline{\texttt{tbs}}(C) = \langle\overline{\texttt{Sig}}(C)\rangle_{\overline{\texttt{PK}}(C')}$ holds when C is signed by $SK(\overline{\texttt{PK}}(C'))$ and we believe that the certificate C' is valid.

5.3 Transfer of Trust

Suppose that a certificate C_2 is signed by the subject of the certificate C_1 with the private key corresponding to the public key of C_1. We also assume that an agent A trust the certificate C_1, i.e., it believes that C_1 is valid. If the agent does not trust the certificate C_2 but wishes to use it, then the agent must verify this certificate based on its own beliefs.

Using our logic, the verification process can be outlined as follows:

(1) $\mathbb{B}_A\texttt{Valid}(C_1)$. (assumption)
(2) $\overline{\texttt{I}}(C_2) = \overline{\texttt{S}}(C_1)$. (assumption)
(3) $\mathbb{B}_A(\overline{\texttt{I}}(C_2) = \overline{\texttt{S}}(C_1))$. (by the rule of necessitation)

(4) $\overline{\texttt{tbs}}(C_2) = \langle \overline{\texttt{Sig}}(C_2) \rangle_{\overline{\texttt{PK}}(C_1)}$. (be checked and assumed to be true)
(5) $\mathbb{B}_A(\overline{\texttt{tbs}}(C_2) = \langle \overline{\texttt{Sig}}(C_2) \rangle_{\overline{\texttt{PK}}(C_1)})$. (by the rule of necessitation)
(6) $\mathbb{B}_A(\texttt{Valid}(C_1) \wedge (\overline{\texttt{I}}(C_2) = \overline{\texttt{S}}(C_1)) \wedge (\overline{\texttt{tbs}}(C_2) = \langle \overline{\texttt{Sig}}(C_2) \rangle_{\overline{\texttt{PK}}(C_1)}))$. (from (1), (3) & (5))
(7) $\texttt{Valid}(C_1) \wedge (\overline{\texttt{I}}(C_2) = \overline{\texttt{S}}(C_1)) \wedge (\overline{\texttt{tbs}}(C_2) = \langle \overline{\texttt{Sig}}(C_2) \rangle_{\overline{\texttt{PK}}(C_1)})$
$\rightarrow \texttt{Valid}(\overline{\texttt{Sig}}(C_2))$. (by axiom (A7) & the rule of instantiation)
(8) $\mathbb{B}_A(\texttt{Valid}(C_1) \wedge (\overline{\texttt{I}}(C_2) = \overline{\texttt{S}}(C_1)) \wedge (\overline{\texttt{tbs}}(C_2) = \langle \overline{\texttt{Sig}}(C_2) \rangle_{\overline{\texttt{PK}}(C_1)})$
$\rightarrow \texttt{Valid}(\overline{\texttt{Sig}}(C_2)))$. (by the rule of necessitation)
(9) $\mathbb{B}_A\texttt{Valid}(\overline{\texttt{Sig}}(C_2))$. (from (6) & (8), and by (A1) and Modus Ponens)

Furthermore, if the following formulas:

(10) $\textit{today} \geq \overline{\texttt{DS}}(C_2)$,
(11) $\textit{today} < \overline{\texttt{DE}}(C_2)$ and
(12) $\neg(C \in \mathbf{CRL}_{\overline{\texttt{I}}(C_2)})$

are all checked and hold, then, form (10) – (12) and by the rule of necessitation, we can have

(13) $\mathbb{B}_A(\textit{today} \geq \overline{\texttt{DS}}(C_2) \wedge \textit{today} < \overline{\texttt{DE}}(C_2) \wedge \neg(C \in \mathbf{CRL}_{\overline{\texttt{I}}(C_2)}))$.

Thus, from (9) and (13), we have

(14) $\mathbb{B}_A(\texttt{Valid}(\overline{\texttt{Sig}}(C_2)) \wedge \textit{today} \geq \overline{\texttt{DS}}(C_2) \wedge$
$\textit{today} < \overline{\texttt{DE}}(C_2) \wedge \neg(C \in \mathbf{CRL}_{\overline{\texttt{I}}(C_2)}))$.

According to Axiom (A8) and by the rule of instantiation, we have

(15) $\texttt{Valid}(\overline{\texttt{Sig}}(C_2)) \wedge \textit{today} \geq \overline{\texttt{DS}}(C_2) \wedge \textit{today} < \overline{\texttt{DE}}(C_2) \wedge$
$\neg(C \in \mathbf{CRL}_{\overline{\texttt{I}}(C_2)}) \rightarrow \texttt{Valid}(C_2)$.

Then, by the rule of necessitation, we have

(16) $\mathbb{B}_A(\texttt{Valid}(\overline{\texttt{Sig}}(C_2)) \wedge \textit{today} \geq \overline{\texttt{DS}}(C_2) \wedge \textit{today} < \overline{\texttt{DE}}(C_2) \wedge$
$\neg(C \in \mathbf{CRL}_{\overline{\texttt{I}}(C_2)}) \rightarrow \texttt{Valid}(C_2))$.

Thus, from (14) and (16), we obtain

(17) $\mathbb{B}_A\texttt{Valid}(C_2)$.

Having completed the proof, we can therefore have

(*) $\mathbb{B}_A\texttt{Valid}(C_1) \vdash \mathbb{B}_A\texttt{Valid}(C_2)$.

This expression (*) can formally be read as “the fact that agent A believes that the certificate C_2 is valid is derived from the fact that agent A believes that certificate C_1 is valid”. Intutively, it represents a trust transfer: Agent A’s trust in the certificate C_2 is transferred from its trust in C_1. In general, an expression ‘$\mathbb{B}_A\varphi \vdash \mathbb{B}_A\psi$’ represents that an agent’s trust in ψ is tranferred from its trust in φ (or its belief in ψ is tranferred from its belief in φ).

This indicates that PKIs provide a mechanism for agents to transfer their trust from where it exists to where it is needed, while our logic allows agents to check the correctness of trust transferring. However, we have to note that PKIs do not create trust [6]. Any PKI is only able to propagate it: agents must initially trust something. Ususlly, initial trust is established off-line. In our approach, initial trust will be formalized as proper axioms in the trust theory of the PKI. Once the set of trust axioms for a given PKI is given, agents can obtain their trust bases as well as the initial trusted certificate set. These will be discussed in the next section.

The reader may note that we did not directly use axioms (A4) – (A6) in the above proof process. However, we have to point out that checking if $\overline{\mathtt{tbs}}(C_2) = \langle \overline{\mathtt{Sig}}(C_2) \rangle_{\overline{\mathtt{PK}}(C_1)}$ holds lies in the validity of the key $K = (\overline{\mathtt{PK}}(C_1), SK(\overline{\mathtt{PK}}(C_1)))$, and the fact that the agent believes that

if K is valid, then $\langle \{M\}_{SK(\overline{\mathtt{PK}}(C_1))} \rangle_{\overline{\mathtt{PK}}(C_1)} = M$ for any message M.

Therefore, these axioms are also needed.

6 Trust Framework for a PKI

This section discusses the trust framework for a PKI, i.e. trust axioms (TA), trust bases (TB) and trust certificates (TC), which are formed as the basis of specifying and reasoning about trust in the PKI.

6.1 Trust Axioms

In our approach, the trust relation in a PKI is formalized as a set of *trust axioms*. That is, we will use a set of proper axioms to define the trust relation. Obviously, different kinds of PKIs may have different axioms to define the trust relation.

In the hierarchical PKI, all CAs and users would trust the certificate of the *paa*, the top CA, because any certificate may be verified by verifying the certification path starting from the "root" certificate held by the *paa*. Therefore, any agent may trust the *paa*. Also, in the PKI, every agent may trust itself. Thus, the PKI may have the following axioms to define the trust relation:

(T1) $\forall A(A \Uparrow paa)$.
(T2) $\forall A(A \Uparrow A)$.

The **TA** = {(T1),(T2)} is the set of trust axioms, which specifies trust in the PKI.

In our approach, a trust theory for a PKI consists of a logic and a set of trust axioms. Now, for the hierarchical PKI, we have a trust theory, for which the proper axioms specifying some trust are contained in **TA**.

What are the effects when adding a new axiom to the existing theory? If the new axiom can be proved as a theorem in the theory, there is no need to add it; if the new trust axiom is not a theorem, adding it gives rise to a new theory.

Let us continue to consider the example given above. We denoted the trust theory for the hierarchical PKI as $\mathcal{T}$. Obviously, adding a trust that "*alice* trusts the *paa*" to the trust theory $\mathcal{T}$ is not necessary, because it cannot make the theory to contain more trusts. In fact, the trust "*alice* trusts the *paa*" can formally be expressed as

$$alice \Uparrow paa,$$

which can directly derived from Axiom (T1). However, if we add a trust stated that all EEs (i.e., users) trust their parents, i.e., those who can certify them, we will obtain a new theory $\mathcal{T}_1$. The trust can be expressed as:

(T3) $\quad \forall A \forall B(\texttt{Is\_EE}(A) \wedge (B \downarrow A) \rightarrow A \Uparrow B),$

which says that for any agents A and B, if A is an EE and can be certified by B, then A trusts B.

(T3) cannot be derived from the theory $\mathcal{T}$, it should therefore be viewed as a new axiom. Thus, the new theory $\mathcal{T}_1$ has an extended set of trust axioms: $\mathbf{TA}_1 = \{$(T1),(T2),(T3)$\}$.

Furthermore, if we add the formula

(T4) $\quad \forall A \forall B(B \downarrow A \rightarrow A \Uparrow B)$

as an axiom into the theory $\mathcal{T}_1$, a new trust theory $\mathcal{T}_2$ is obtained. Thus, the set of trust axioms for $\mathcal{T}_2$ contains 4 axioms, i.e., (T1) – (T4). However, if (T3) is moved out from the set of trust axiom, we still have the same theory. That is, the set of trust axioms for the theory $\mathcal{T}_2$ can be $\mathbf{TA}_2 = \{$(T1),(T2),(T4)$\}$.

We do not attempt to discuss all the issues about how to construct a set of trust axioms in a trust theory for a PKI. We only need to note that we have to consider the consistency of a trust theory when adding a new trust into it. That is, we must maintain the soundness of our theory when extending the theory by adding new axioms.

6.2 Trust Bases

Given the topology $\langle \Omega, \downarrow \rangle$ of a PKI. We define that the *trust base* of any agent $A (\in \Omega)$, denoted as $\mathbf{TB}_A$, is a set consisting of all agents whom the agent trusts. We argue that, for any agent in a PKI, there should be a trust basis that the agent places his trust in and from where it can therefore obtain the information it expects.

Our model is conservative: in particular, an agent A is assumed not to trust another agent B unless there is an explicit expression $A \Uparrow B$ that can be derived from the theory of trust. That is

- If $\mathcal{T}$ is the trust theory of a PKI, then, for any agents A and B, $B \in \mathbf{TB}_A$ if and only if $A \Uparrow B$ is a theorem of $\mathcal{T}$.

For example, assume that $alice \in \Omega$, and the PKI is hierarchical and has the trust theory $\mathcal{T}$ given above. Then, it is not difficult to show that $\mathbf{TB}_{alice} = \{paa, alice\}$.

As we will see, based on the trust base, an agent can build its own trusted certificate set, which is needed for the certificate verification.

6.3 Trusted Certificates

When an agent wants to verify a required certificate, it must construct a certification path starting with a certificate trusted by itself and ending with the required certificate. If it has no trusted certificates, it cannot accept any certificate as valid. We now analyse why an agent needs a set of trusted certificates and how to derive it based on our logic.

A PKI provides mechanisms for agents to retrieve information, such that agents are able to verify the validity of every certificate that a security application uses. Assume that an agent Alice wants to retrieve Bob's certificate together with evidence used for checking if the certificate is valid. The certificate held by Bob carries the message that allows Bob to say, for example, "I am *bob* and have the certificate to which the public key *pk* (and the corresponding private key *sk*) is bound. The certificate is issued by ca_1 and valid from 12th October 2000 to 31st October 2001". If Alice trusts Bob, she may believe that Bob's certificate is valid, i.e., she may accept Bob's certificate as valid. In particular, Alice believes that Bob's public key is really Bob's, so that she can use Bob's public key *pk* to decrypt a message signed with Bob's private key.

However, if Alice does not trust Bob, she must verify Bob's certificate before she uses it. To do this, Alice may employ the proof procedure presented in the last section to transfer some of her trust which has existed to Bob's certificate, and to determine whether to trust it. The procedure can be outlined as follows:

- verifying the identity of the certificate issuer and owner (checking if *bob* is the owner and, and checking if *bob* belongs to the certification domain of the issuer);
- verifying the validity dates of the certificate;
- verifying the certificate against the issuer's latest CRL list to make sure it has not been revoked;
- verifying extension fields (such as certificatePolicies) if necessary; and
- verifying the signature on the certificate.

In order to verify the signature on Bob's certificate, Alice needs to check if the issuer holds a valid certificate, or, more precisely, Alice must verify the certificate which is held by the issuer and used to sign Bob's certificate in the same way if she does not trust the issuer's certificate. Therefore, the verification process Alice uses is iterative. She cannot accept Bob's certificate as valid unless she reaches a certificate she trusts in the verification procedure.

This indicates that, if Alice has no trusted certificates, either she cannot prove that Bob's certificate is valid or her proof process can never terminate. That is, an agent who is involved in certificate verification should have a non-empty set of certificates which it trusts.

Let $\langle \Omega, \downarrow \rangle$ be the topology of a PKI. Given a state, for all agent $A(\in \Omega)$, we denote the set of certificates trusted by the agent A at the given state by $\mathbf{TC}_A$. For the agent, it has a basic belief that, if a certificate belongs to its trusted certificate set, then the certificate can be accepted as valid. Formally, we have:

(A*) $\mathbb{B}_A(C \in \mathbf{TC}_A \rightarrow \mathtt{Valid}(C))$, for all $C \in \mathcal{C}$.

(A*) is an auxiliary axiom of the proof system, which is related to constructing initial trust.

Suppose that at the current state the trusted certificate set of an agent A is TC, and the agent has also verified that a certificate C is valid and put C into its trusted certificate set, then its new trusted certificate set would be $TC \cup \{C\}$. This indicates that, if there has existed a trusted certificate set for an agent, then there is no problem to extend the set by adding new certificate that the agent trusts. The question is: how does the agent derive the initial trusted certificate set for itself? A simple rule the agent A may adopt is that, if a certificate is owned by someone whom A trusts, then the certificate can be trusted by A itself and should belong to the trusted certificate set $\mathbf{TC}_A$. This rule can be expressed as follows:

Belief Rule: If $\overline{\mathtt{S}}(C) = B$ and $B \in \mathbf{TB}_{\mathtt{A}}$, then $C \in \mathbf{TC}_A$.

As we said before, we allow freedom to define the meaning of "trust". In our model, as shown in the above rule, a simple explanation of trust could be that the fact "one agent trusts someone" means that the agent trusts the person's certificate, i.e., it believes that the certificate is valid. The above rule is flexible, and may be modified by giving different explanations for the meaning of "$A \Uparrow B$" (depending on the designer and/or security requirements). For instance, instead of the above rule, we may adopt a rule as follows:

If $\overline{\mathtt{S}}(C) = B$ and $A \Uparrow B$, then $\mathbb{B}_A(B\ \mathit{controls}\ SK(\overline{\mathtt{PK}}(C)))$,

which means that if an agent A trusts an agent B and B is the subject of a certificate C, then the agent A believes that B controls the private key corresponding to the public key of C. (Note that, if this rule is adopted, we may need to make slight changes to the axioms of our proof system.)

Trust assessment must be based on some initial trust combined with trust propagating [6]. For certificate verification, one must obtain an initial trusted certificate set. Initial trust is usually established off-line. Our model allows an agent to gain trust by the proof system of our logic.

7 Certificate Verification

Certificate verification is a client function, which is responsible for verifying the validity of every certificate that a security application uses. This section discusses certificate verification in a given state for a PKI.

7.1 The Concept of Certificate Verification

The certificate verification is always based on a given state, in which any agent has a certain set of certificates possessed by itself and a certain revoked certificate set. For verifying a required certificate, the agent must also have its own trusted certificate set at the given state.

Verifying the validity of a required certificate involves obtaining and verifying the certificates from a trusted certificate to the target certificate. Obtaining the certificates is referred to as *certificate path development* and checking the validity of the certification path is referred to as *certification path validation.* A *certification path* is usually defined to be a non-empty sequence of certificates $\langle C_0, \ldots, C_n \rangle$, where C_0 is the *target* certificate, C_n is a *trusted* certificate, and for all i $(0 \leq i < n)$ the subject of C_{i+1} is the issuer of C_i.

The path development module discovers certification paths and sends them to the path validation module for processing; the path validation module takes a given certification path and determines whether the target certificate is valid or invalid.

For certificate verification, we have an essential principle stated as follows:

– *In a certificate verification process, when the verifier (an agent) has found a certification path constructed for verifying a required certificate in which he believes that all certificates are valid it may accept this certificate as valid, and in all other cases the certificate is regarded as invalid.*

Note that, according to the certificate verification principle, it can happen that a certificate may actually be valid but the verifier did not find a corresponding certificate path in which all certificates are valid. In such a case, the verifier cannot accept this certificate as valid. This is the correct choice on security grounds.

7.2 Path Development and Validation

How the certification path is obtained is dependent on the structure of the PKI. However, in any PKI, starting with the target (required) certificate and building a certificate chain back towards a certificate trusted by the verifier is usually an efficient means for developing a certificate path.

A sequence of certificates starting with the target certificate constructed in developing a certification path may eventually be a part of some certification path; however, in some cases, it may be discarded as not being a part of any certification path if the verifier could not reach any trusted certificate along this sequence. In either case, the verifier may need to check whether there is any possibility to reach a trusted certificate along such a sequence of certificates, such that a certification path can be constructed.

We assume that, at a given state, C_0 is the target certificate required by the agent A, and $\mathbf{TC}_A$ is the trusted certificate set of the agent A. Recalling the notations for possessed and revoked certificate sets, we have: for any agent

B, $\mathbf{PCS}_B$ represents B's possessed certificate set, and $\mathbf{CRL}_B$ represents B's revoked certificate set.

Thus, for a hierarchical PKI, the agent A, as a verifier of the certificate C_0, may adopt the following algorithm in developing a certification path by constructing candidate paths step by step.

1. Set $i = 0$, $P = \langle C_0 \rangle$ and $\zeta_0 = \mathbf{PCS}_{\bar{\mathtt{I}}(C_0)}$;
2. If $C_i \in \mathbf{TC}_A$, return $P = \langle C_0, \ldots, C_i \rangle$ as the certification path, then stop; otherwise
3. If ζ_i is not empty, choose C_{i+1} from ζ_i, set
$\zeta_i = \zeta_i - \{C_{i+1}\}$, $P = \langle C_0, \ldots, C_i, C_{i+1} \rangle$, and $\zeta_{i+1} = \mathbf{PCS}_{\bar{\mathtt{I}}(C_{i+1})}$,
then reset $i = i + 1$ and go to Step 2; otherwise
4. If ζ_i is empty, and $i > 0$, delete the last element in P, i.e., reset $P = \langle C_0, \ldots, C_{i-1} \rangle$, then set $i = i - 1$, go to Step 3; otherwise
5. If ζ_i is empty and $i = 0$, return fail (which means that no certification path is found), then stop.

By this method, the verifier A may construct all possible certification paths starting with the target certificate and ending with a trusted certificate. All these certification paths can be used for verifying the target certificate. However, it may find that there are no such certification paths, in which case, it cannot accept the certificate as valid.

In particular, if A only trusts the certificate held by the *paa* in a hierarchical PKI, then any certification path constructed by itself is always a certification path starting with the certificate held by the *paa*.

Since the certificate set of an agent may contain multiple certificates, without the use of the key-identifier information, certification path development become increasingly complex as the number of paths that need to developed may grow exponentially. To avoid this, in Step 3 "choose C_{i+1} from ζ_i", we may use the key-identifier information to reduce the number of choices.

Suppose a certification path $\langle C_0, \ldots, C_n \rangle$ bas been developed for agent A to verify cerificate C_0, and sent to the path validation module, where C_n is a certificate belonging to A's trusted certificate set. ¿From the fact that A trusts the certificate C_n, i.e., $C_n \in \mathbf{TC}_A$, and Axiom (A*), we can have

$$\mathbb{B}_A\mathtt{Valid}(C_n).$$

The path validation module needs to check whether A's trust in C_n can be transferred to its trust in C_0, i.e., it needs to prove all the following trust transferring:

$$\begin{aligned}
&\mathbb{B}_A\mathtt{Valid}(C_n) \vdash \mathbb{B}_A\mathtt{Valid}(C_{n-1}),\\
&\mathbb{B}_A\mathtt{Valid}(C_{n-1}) \vdash \mathbb{B}_A\mathtt{Valid}(C_{n-2}),\\
&\qquad\qquad\cdots\cdots\\
&\mathbb{B}_A\mathtt{Valid}(C_1) \vdash \mathbb{B}_A\mathtt{Valid}(C_0).
\end{aligned}$$

Unless all proofs for these trust transferrings are successfully completed, the agent A cannot accept C_0 as valid by this path.

A framework for path processing has been proposed based on a natural separation of the entire certification path validation problem into distinct types of checking requirements. For the details about the framework and a mechanism dealing with various different checks, we refer the reader to Ozols *et. al.* [12].

8 Conclusion

There are two major relations involved in a PKI, one is the certification relation and the other the trust relation. The certification relation has been formalized by Liu *et. al.* [10,12]. The trust relation is the basis of certificate verification. This paper, focusing on the trust relation, has present a typed modal logic for specifying and reasoning about trust for a PKI. We have proposed a trust theory for formalizing the trust relation in a PKI. In particular, we have discussed trust axioms, trust bases and trusted certificates. In our model, the trust relation in a PKI is formalized by **TA**, a set of trust axioms. Based on the trust axioms, an agent has its own trust base that contains all agents whom the agent would trust. The trusted certificate sets are essential for certificate verification. Without a trusted certificate set, an agent cannot prove that a certificate is valid or he should not use any PKI function for itself.

Our axiomatic approach is flexible: it is easily modified or extended for a specific purpose. The logic is sound. Because of space limitation, no proof of soundness is given in this paper. Also, we did not give a formal semantics for this logic. All these will be addressed in future work. We also plan to mechanise the theory in a general theorem prover, Isabelle [13]. Once a reasoning system for PKIs has been developed, certificate verification and the proof of security properties of a PKI could be automatically performed.

Future work also includes investigating the different distributions of trust points within a PKI. Comparisons of solutions and suggestions as to how distribution of trust points could be implemented in these extended PKI structures need to be considered. Joining PKI's, with so called cross-certificates or bridging CAs, is another important issue. This paper is based on first order logic. The PKI functions, include certificate verification, could also be described based on a temporal reasoning framework.

References

1. M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *Proceedings of the 1993 IEEE Computer Society Symposium on research in Security and Privacy*, pages 164–173, 1996.
2. M. Burrows, M. Abadi, and R. M. Needham. A logic of authentication. In *Proceedings of the Royal Society of London* **426**, pages 233–271, 1989.
3. A. Herzberg, Y. Mass, and J. Mihaeli. Access control meets public key infrastructure, or: Assigning roles to strangers. In *Proceedings of the 2000 IEEE Computer Society Symposium on research in Security and Privacy*, pages 2–14, 2000.

4. R. Housley, W. Ford, W. Polk, and D. Solo. RCF 2459, *Internet X.509 Public Key Infrastructure - Part I: Certificate and CRL Profile*. Internet Request for Comments 2459, January 1999.
5. A. Jøsang and S.J. Knapskog. A metric for trusted systems. In *Proceedings of the 21st National Security Conference*, NSA 1998.
6. A. Jøsang, I. G. Pedersen, and D. Povey. PKI seeks a trusting relationship. In *Proceedings of the 5th Australasian Conference on Information Security and Privacy (ACISP 2000)*, volume 1841 of *Lecture Notes in Computer Science*, pages 191–205. Springer, 2000.
7. S. Kent. Privacy Enhancement for Internet Electronic Mail, Part II: Certificate-Based Key Management, Request for Comments 1422. Network Working Group, 1993.
8. R. Kohlas and U. Maurer. Confidence valuation in a public-key infrastrucutre based on uncertain evidence. In *Proceedings of the 3rd International Workshop on Practice and Theory in Public Key Cryptosystems (PKC 2000)*, volume 1751 of *Lecture Notes in Computer Science*, pages 93–113. Springer, 2000.
9. N. Li, B. Grosof, and J. Feigenbaum. A practically implementable and tractable delegation logic. In *Proceedings of the 2000 IEEE Computer Society Symposium on research in Security and Privacy*, pages 27–42, 2000.
10. C. Liu, M. A. Ozols, M. Henderson, and T. Cant. A state-based model for ceritificate management systems. In *Proceedings of the 3rd International Workshop on Practice and Theory in Public Key Cryptosystems (PKC 2000)*, volume 1751 of *Lecture Notes in Computer Science*, pages 75–92. Springer, 2000.
11. Ueli Maurer. Modeling a public-key infrastructure. In E. Bertino, H. Knurth, G. Martella, and E. Montolivo, editors, *Computer Security – ESORICS'96 (LNCS 1146)*. Springer-Verlag, 1996.
12. M. A. Ozols, M. Henderson, C. Liu, and T. Cant. The PKI specification dilemma: A formal solution. In *Proceedings of the 5th Australasian Conference on Information Security and Privacy (ACISP 2000)*, volume 1841 of *Lecture Notes in Computer Science*, pages 206–219. Springer, 2000.
13. L. C. Paulson. *ML for Working Programmer*. Cambridge University Press, 1991.
14. R. Yahalom, B. Klein, and Th. Beth. Trust relationships in security systems - A distributed authentication prespective. In *Proceedings of the 1993 IEEE Computer Society Symposium on research in Security and Privacy*, pages 151–164, 1993.

A Knowledge-Based Approach to Internet Authorizations

Along Lin

Hewlett-Packard Laboratories
Filton Road, Stoke Gifford
Bristol BS34 8QZ, U.K.
alin@hplb.hpl.hp.com

Abstract. This paper proposes a knowledge-based approach to Internet authorizations using Public-Key Infrastructure (PKI) based digital certificates and Role-Based Access Control (RBAC). First, we introduce several existing access control models. Second, a logic-based policy specification language is given. Third, a policy-driven RBAC is presented. Fourth, a method of automatically assigning roles to users using digital certificates is discussed. Then, the architecture for Internet authorizations is described. Finally, a solution to remote policy enforcement is proposed. We also give the syntax of a role definition language and illustrate it in appendices A and B, respectively.

1 Introduction

The Internet provides an excellent infrastructure for supporting information sharing and the collaboration between business partners. One of the most important challenges is to control Internet users accesses to resources, without asking the users to pre-register with the resource providers. In the following, we will use the terms resource and service exchangeably. First, we introduce some of the existing access control models.

1.1 Discretionary Access Control (DAC)

DAC enforces the rules specified by an access matrix, which describes the operations each subject is authorized to perform on various objects, and is typically implemented by either Access Control Lists (ACLs) associated with resources or users capability lists. Each time a user is added into or removed from a system, security managers have to administer the relevant ACLs for those resources affected. Similarly, this applies to users capability lists, when a resource is added into or removed from the system. The extensions to the conventional ACL include the addition of an optional field to each ACL entry for specifying restrictions on access rights [8]. In [17], a generalized ACL supporting authorizations delegation is given. However, they do not scale well as the number of subjects or objects in the system increases. If an authorization policy changes, the ACL has to be modified dramatically. Therefore, DAC is not the best solution to the access control in Internet environments from a user-resource management perspective.

1.2 Mandatory Access Control (MAC)

MAC attaches security levels to objects based on their information sensitivity, and to subjects, reflecting the degree to which they are trusted to not disclose sensitive information. Security levels are partially ordered in a lattice-structured hierarchy, with each level dominating itself and the ones below it [4, 19, 23]. Some MAC systems also introduce categories and assign them to subjects and objects. Thus, a node in the hierarchy consists of a security level and a set of categories. MAC enforces a specific security policy so that it prevents information flow from high-level objects to low-level subjects. However, when a user is unknown to a resource provider, the prior art MAC model still cannot solve the general problem of Internet authorizations.

1.3 Role-Based Access Control (RBAC)

In RBAC models [2,6,7,9,13,14,16,24], a role is represented by a set of permissions that allow its role members to perform operations on objects. Security officers and system administrators create and assign roles to users based on their responsibilities and obligations in their organizations. An RBAC system determines a user s permissions according to the role(s) the user plays at the time of requesting a resource. A user can be easily reassigned from one role to another. A role may be granted new permissions as new resources are provided, and permissions can be revoked from roles as needed. Therefore, security management is significantly simplified. However, given an unknown Internet user s request, assigning roles to the user dynamically still needs to be solved.

1.4 Certificate-Based Access Control

Attribute and authorization certificates may contain access rights [11,15]. Attribute Certificates (ACs) bind attributes to users Distinguished Names (DNs), and can be used with identity certificates to achieve the mapping: attributes → DN → public key. Having a delegation tag within it, a SPKI attribute certificate allows its recipients to delegate privileges to other people, achieving the distributed authorizations [1,3,5]. However, having authorizations in an AC has several issues. First, AC authorities must issue ACs to users before the users accesses to the controlled resources. Second, for those ACs that have a long period of validity, revoking them can be much of a burden to the issuing AC authorities, especially when the accessing rights within them have to be updated due to the resource provider s security policy changes. Third, it is too expensive to issue ACs to potential Internet users.

In this paper, it is assumed that an AC contains only users non-volatile attributes without access rights. A user may have several ACs issued by different trusted authorities that have intimate knowledge of the user. E.g., an accredited university issues its graduates normal ACs containing such information as degrees, qualifications, majors, graduation dates, and the like. A user s ACs will then be used to make access control decisions by resource providers, based on their security policy and the user s requests [25,26,27,28].

In the following, a logic-based policy specification language is introduced first. Second, a policy-driven RBAC is presented. Third, we discuss a method of automatically assigning roles to Internet users using digital certificates. Fourth, the architecture for knowledge-based Internet authorizations is described. Finally, a solution to remote policy enforcement is proposed. We also give the syntax of a policy-based role definition language and illustrate how it can be used to specify security policies in appendices A and B, respectively.

2 A Logic-Based Policy Specification Language

Our policy specification language is defined as a subset of Horn clauses. A clause, also known as a rule, takes the following form: *H* if *B*, where *H*, *B* are called the head and body of the rule, respectively. *H* takes the form of *pred*(t_1,   , t_n), where *pred* is an n-ary predicate symbol and each t_i is a term. *B* is a conjunction of literals. A term is either a variable or a constant and each rule has a bounded number of variables. When B is empty, the if part of the rule can be omitted. A policy base is thus just a finite set of rules. Given a policy base *PB*, a request *r* is to be granted if and only if *PB* ! *r*.

The logic-based policy specification language has the following advantages [18]:

- The separation of domain-specific policy base from its implementation mechanism, increasing the flexibility of an authorization system. Based on the same implementation mechanism, users can deploy various authorization policies.
- The ability of expressing constraints and security policies as declarative rules. In most security systems, policies are hard-coded into programs, which is inflexible for configuration changes.
- The capability of expressing information implicitly. E.g., a role membership can be specified by a predicate, rather than enumerating its members explicitly.
- Policy conflict detection can be done by checking if the corresponding rule set is consistent or not, based on the model-theoretic semantics of first order logic.

3 Policy-Driven RBAC

The known prior art RBAC is policy-neutral. Some extensions to it include the introduction of role hierarchies, and constraints that apply to user-role and role-permission assignment, and so on [2,6,7,9,12,13,16]. In our policy-driven RBAC model, user-role assignment policy and the application-specific business logic on role permissions are provided as a configurable knowledge base. Any changes to it will dynamically and automatically drive the changes in an organization s security policies on its business processes, thus alleviating the security management. In our approach, each role is extended by a predicate describing its membership policy, and role permissions are generic. For example, there are several permissions defined for the withdrawal of money in a traditional RBAC system:

- A normal customer can withdraw money from its own bank account, but overdrawing is disallowed.
- However, a silver customer, whose annual salary is over £20000, can overdraw up to £1000 from a bank if the customer has been with the bank for more than 5 years; otherwise, the limit for overdrawing is reduced to £300.

The business logic on roles permissions can be captured in the knowledge base so that when the bank s business policy on a customer s withdrawal of money changes, all that needs to be done is to modify the affected policy rule rather than adding a new permission or modifying an existing permission. A simplified definition of role silver_customer is described below, based on the role syntax given at appendix A.

```
Name: silver_customer.
Role-Assigning Policy: salary_based_silver_customer(Certificates, Request).
Authorizations:
          Request = [withdraw, Account, Amount|_],
                    overdraw_policy(Certificates, Limit),
                    withdraw_test(Limit, Account, Amount),
                    withdraw_money(Account, Amount). // It is a method.
          // For simplicity, other permissions are omitted here.
          .
salary_based_silver_customer(Certificates, _) :-
          get_salary (Certificates, Salary), // Its definition is omitted here.
          Salary > 20000.

overdraw_policy(Certificates, 1000) :-
          time_with_the_bank_over_months(Certificates, 60).
          // For simplicity, its definition is omitted here.
overdraw_policy(_, 300).

withdraw_test(Limit, Account, Amount) :-
          balance_of_the_account(Account, Balance),
          Amount <= Balance + Limit.
```

Besides the application-specific business logic on role permissions, the user-role assignment policy can also be captured in the knowledge base. If it is empty, roles must be assigned to users manually with business logic hard-coded into the system; otherwise, roles may be assigned to users automatically and dynamically, depending on whether the role assignment policies evaluate true. For a role that can be assigned to any user, the role assignment policy attribute is true in its definition. If a role assignment policy evaluates false for a given user, the role cannot be assigned to the user. For other roles that must be manually assigned to users, their user-role assignment policy attributes are null by default. Furthermore, because a resource provider may trust only those digital certificates issued by particular Certificate

Authorities (CAs), the trust relationships between resource providers and CAs may also be captured in the knowledge base.

There are several significant differences between our approach to Internet authorizations and others like PolicyMaker, KeyNote, and SPKI. First, a resource provider does not need to issue digital certificates to an Internet user before the user can access the resources controlled by the resource provider. The user may need to have a set of digital certificates issued by trusted third parties. Second, the user s presented digital certificates do not contain any access rights. A user s access rights are implied by the permissions of the roles assigned to the user based on the user s presented certificates. They will be used for a wide range of general purposes rather than specific requests for particular resources. Third, for security reasons, access control policies do not appear on the certificates in the form of credential conditions or whatsoever. Fourth, when either the access control policies or access rights change, our approach will not be affected. The separation of access rights and authorization policy from a user s digital certificates makes our approach very flexible.

4 Automated Role-Assignment Using Digital Certificates

Most RBAC systems assign roles to users manually and involve a lot of user-role administration. An approach has been proposed to dynamically map a user to predefined business roles, using the user s presented digital certificates issued by Certificate Authorities (CAs) and role-assigning policies pre-set by resource providers [11]. Because the user does not know about the role-assignment policy and multiple roles may have the required permission for a given user s request, various digital certificates will be requested to present to the resource provider [18], based on the evaluation of security policies and business logic associated with the required permission.

There are two major differences between our automated role-assignment approach and that described in [11]. First, we adopt a non-deterministic approach due to the fact that the resource provider does not know which role should be dynamically assigned to a user for its current request. The non-deterministic role assignment is achieved by using a powerful backtracking mechanism. Second, our approach is more efficient by enforcing server side security policies on a client s machine remotely, which will be discussed later.

The principle of *separation of duties* can be dynamically accomplished by not satisfying the policies on the assignment of conflicting roles to the same user simultaneously [20]. Alternatively, we can store the roles already assigned to a user in a database persistently, such that the candidate roles being in conflict with them will not be assigned to the user (see Fig. 1). If such a conflict does occur, the security manager will be informed and will take necessary measures as required. In this paper, we adopt the first approach due to its simplicity for implementation.

Although a user may be assigned multiple roles, only some of them will authorize the user s current request. For each role assignable to a user, if either it does not have a matching privilege for the user s current request or the business logic associated with the privilege evaluates false, the current role assignment will be backtracked automatically for alternative roles; otherwise, the user s current request is authorized. If all roles have been tried and none of them authorize the given request, the user will be denied. In our approach, a role-filtering sub-system is used to pre-compute the candidate roles assignable to a user for efficiency.

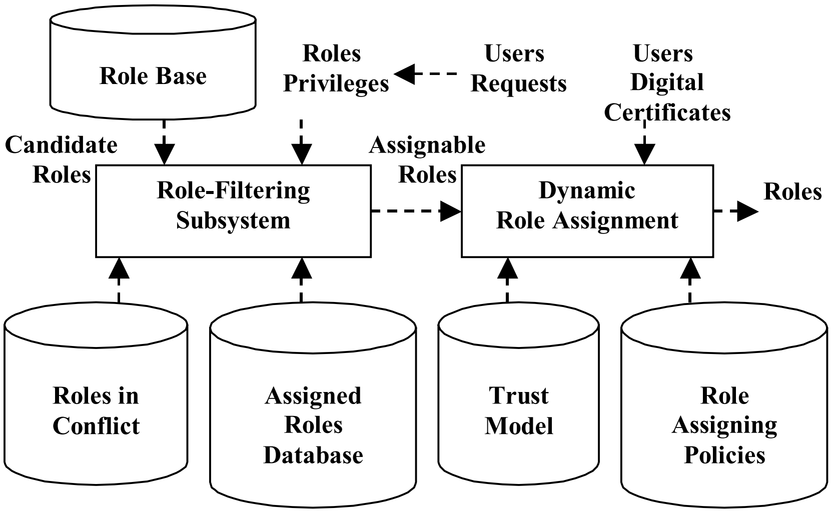


Figure 1. Automated Role Assignment Model

5 A Knowledge-Based Approach to Internet Authorizations

The architecture for our knowledge-based solution to Internet authorizations is described in Fig. 2. Given a user s digital certificates and the knowledge base on the server side, it does not make sense to ask the user to submit all of its digital certificates to the service provider for accessing the requested service, most of which are irrelevant to the current request. The user may not be willing to send them either.

During the access control decision-making, if any of the following conditions holds, the Server Security Agent (SSA) will automatically redo the current user-role assignment and re-evaluate the business logic on a role s associated privileges:

- Some of the user s presented digital certificates have already been revoked by their issuing CAs, using either Certificate Revocation Lists (CRLs) or the Online Certificate Status Protocol (OCSP) for their validations.
- Some certificates presented either have expired or are unacceptable to the SSA, based on the trust models described in the server-side knowledge base.

- The user refuses to send the requested digital certificates to the SSA.
- Given a user s current request, the user s presented digital certificates do not satisfy the resource provider s role-assignment policy.
- The business logic on the assigned role s matching privilege evaluates false, based on the server-side knowledge base.

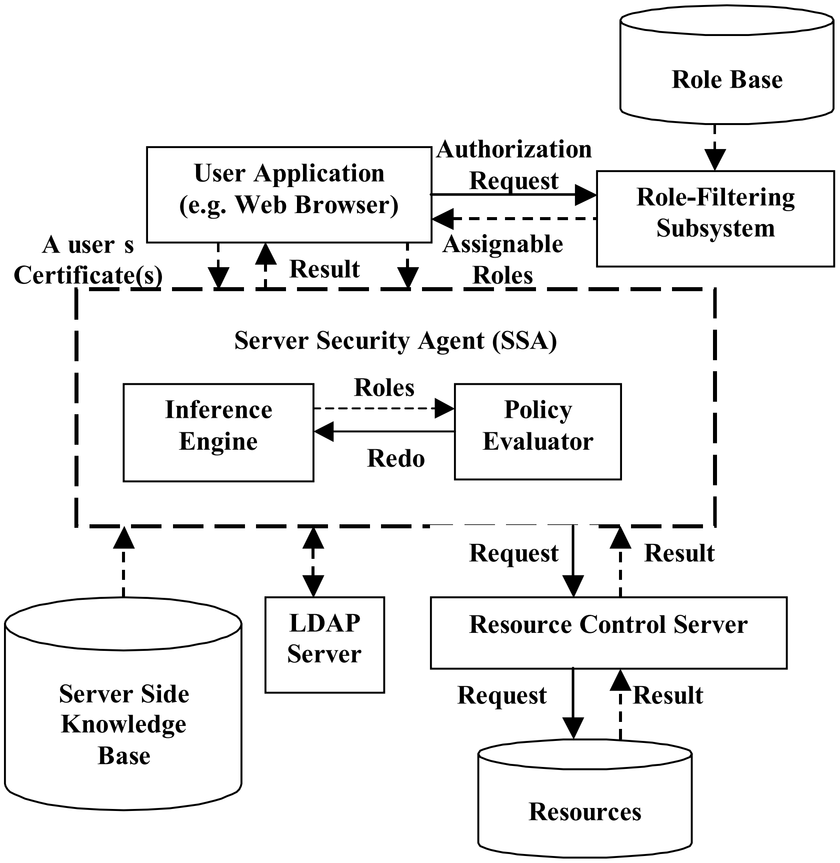


Fig. 2. Architecture for Knowledge-Based Internet Authorization Using PKI

Given a user s request, there are several steps in sending the user s digital certificates to the SSA after the role-filtering sub-system identifies the candidate roles assignable to the user. First, the SSA finds a set of digital certificates required for assigning each of the candidate roles to the user, based on the certificate acceptance policies and trust models specified in the server-side knowledge base. Second, if some of those certificates have already been cached in a local directory and are still valid, the SSA only requests for other unavailable certificates from the user, who then sends back the requested certificates to the SSA according to the user s certificate-sending policy. Finally, the SSA caches the received certificates in its directory using

Lightweight Directory Access Protocol (LDAP). If the user refuses to send a requested certificate to the SSA, an alternative candidate role will have to be tried for its assignment to the user.

Users certificates can be sent to the SSA by an application, a trusted applet digitally signed by the resource provider, or a plug-in from a trusted third party for the user s Web browser. The user may configure its certificate-sending policy so that some of its certificates can be automatically sent to the SSA. The default configuration will inform the user of the SSA s request for them whenever it happens. If the user s certificate-sending policy evaluates false, the user will not send the requested certificates; otherwise, they will be sent to the SSA. The client side may need a knowledge base that describes the client s certificate-sending policy and trust models, as shown in Fig. 3.

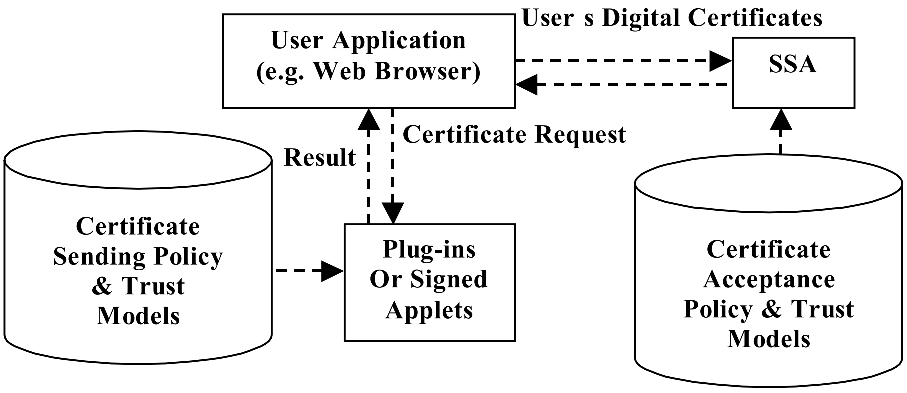

User s Digital Certificates
User Application (e.g. Web Browser)
SSA
Result
Certificate Request
Certificate Sending Policy & Trust Models
Plug-ins Or Signed Applets
Certificate Acceptance Policy & Trust Models

Figure 3. Knowledge-Based Certificate Sending

Our knowledge-based approach to Internet authorizations using PKI-based digital certificates and RBAC has several unique advantages over the traditional RBAC and other access control models. First, it enables users to specify security policies, trust models, and business logic in a configurable knowledge base separately rather than hard-coding them into the system or the role privileges. Second, based on digital certificates, it allows unknown Internet users to access resources more conveniently without having to register with the resource provider in advance. Third, the approach supports automated, non-deterministic role-assignment. Because the user-role administration in conventional RBAC systems has been automated, the security manager s work is reduced to a minimum, and thus can be focused on the declarative specification of security policies, trust relationships between the service provider and various CAs. Finally, our approach provides a flexibility that any changes to the knowledge base will automatically drive the changes in an organization s security policies on its business processes, thus alleviating the security management.

6 Remote Policy Enforcement in Internet Authorizations

The solution to reducing the traffic caused by runtime digital certificate exchange between a client and a server is minimizing the computation-intensive non-deterministic policy evaluation over the Internet. This can be achieved by downloading the server side knowledge base, which includes certificate acceptance policy and role-assigning policy, onto a client s machine whenever the client requests a service.

However, the server side security polices will not be enforced unless, on the client side, the inference engine is trusted and the knowledge base used is the same as that from the service provider. One potential solution is generating the set of a client s digital certificates required for its current request and sending them to the server side for checking [25]. Because much of the backtracking has already been done on the client side during the finding of the set of the client s certificates that satisfy both the client s certificate sending policy and the server side certificate acceptance policy, the server side checking no longer needs to ask the client to send various digital certificates to it, and therefore the client s privacy is protected to some extent.

To further reduce the traffic between a client and a server, we only need to download onto the client s machine a much smaller server-side knowledge base, which is relevant to the client s current request and can be pre-computed. Before discussing how to enforce security policy remotely and efficiently, we give the following notations first. Let ***R*** be the set of role, and ***KB*** the set of rule in the knowledge base on the server side, respectively.

1. For a given role ***r***, its privileges and role assignment policies are denoted by $\boldsymbol{P}(\boldsymbol{r})$, and $\boldsymbol{RA}(\boldsymbol{r})$, respectively.
2. For a given privilege ***p*** in role ***r***, its business logic is denoted by $\boldsymbol{A}(\boldsymbol{p},\boldsymbol{r})$, and the partial ***KB*** relevant to it is defined as $\boldsymbol{K}(\boldsymbol{p},\boldsymbol{r}) = \boldsymbol{RA}(\boldsymbol{r}) \text{ " } \boldsymbol{A}(\boldsymbol{p},\boldsymbol{r})$.
3. For a given privilege ***p***, the set of candidate role assignable to a user is defined as $\boldsymbol{C}(\boldsymbol{p}) = \{ \boldsymbol{r} \mid \boldsymbol{r} \# \boldsymbol{R} \$ \boldsymbol{q} \# \boldsymbol{P}(\boldsymbol{r}) \$ \%\boldsymbol{u} (\boldsymbol{p}\circ\boldsymbol{u} = \boldsymbol{q}\circ\boldsymbol{u}) \}$, where ***u*** is a most general unifier for ***p*** and ***q***.
4. The privileges of all of the roles in ***R***, are defined as $\boldsymbol{Priv}(\boldsymbol{R}) = \underset{\& r \# \boldsymbol{R}}{\text{"}} \boldsymbol{P}(\boldsymbol{r})$.
5. Let ***T*** be the trust model for the validation and acceptance of the client s digital certificates by the service provider.
6. For a given privilege ***p***, the partial ***KB*** relevant to it, is defined as follows:

$$knowledge\_base(\boldsymbol{p}) = \Big(\underset{\& r \# \boldsymbol{C}(\boldsymbol{p})}{\text{"}} \boldsymbol{K}(\boldsymbol{p},\boldsymbol{r}) \Big) \text{ " } \boldsymbol{T}.$$

During the compilation of ***KB***, for every role ***r*** and each privilege ***p*** in ***r***, $\boldsymbol{C}(\boldsymbol{p})$ and $\boldsymbol{K}(\boldsymbol{p},\boldsymbol{r})$ are computed first, and finally *knowledge_base*(***p***) is computed. We store $\boldsymbol{C}(\boldsymbol{p})$ and *knowledge_base*(***p***) in a file whose name is denoted by $\boldsymbol{F}(\boldsymbol{p})$.

The client s service request now can be processed as follows. First, the SSA translates it into a corresponding privilege ***p*** in ***Priv*(*R*)**. Then, the SSA sends the file ***F*(*p*)** to the client. After the client application receives the partial server-side knowledge base, it retrieves ***C*(*p*)** and *knowledge_base*(***p***) immediately, and tries to enforce server-side security policies based on the client s current request and available digital certificates. Finally, if the client is authorized for the current service request, it will return a set of the client s digital certificates that satisfy both the client s certificate-sending policy and the server-side security policies to the SSA; otherwise, an empty set will be returned.

7 Conclusions

In this paper, a knowledge-based approach to Internet authorizations is proposed by using PKI-based digital certificates and RBAC. After introducing a logic-based policy specification language, we present a policy-driven RBAC. Security policies are expressed as the rules in an application-specific, configurable knowledge base. An inference engine is utilized to evaluate policies, automatically assign roles to Internet users based on their digital certificates, and redo role assignment as required. The approach is capable of dealing with unknown Internet users and automatically managing user-role assignment by using digital certificates, which makes the administration of unknown Internet users access to services less of a burden to security managers. Finally, we discuss an efficient method of remote security policy enforcement by downloading a partial knowledge base that is relevant to a client s current request.

As pointed out in [11,12,14,18,25], the expressiveness of a logic-based specification language allows security policies to be succinctly and uniformly specified. PROLOG is based on Horn-clauses, which is a subset of first order logic, and has a solid theoretical foundation for reasoning. Both policy evaluation and policy conflict detection can be easily done within the logic framework. Therefore, PROLOG is adopted as the generic policy representation language in this paper. However, a policy-authoring tool is strongly recommended for alleviating the policy-writing task of security managers. The tool should be capable of mapping a user s access request to a role s privilege intelligently and refining a high-level security policy into an executable rule, so that security managers can focus on the specification of authorization policies, trust models, and role-assigning policies in the knowledge base.

Acknowledgement

We would like to thank the following people for their reviews: Pete Bramhall, Brian Monahan, Keith Harrison, Martin Sadler, Yolanta Beres at Hewlett-Packard Laboratories, United Kingdom. The constructive discussions with them are very useful. Especially, we would like to thank Pete Bramhall for his strong support. We are also grateful to the anonymous referees for their valuable comments and suggestions, which have significantly improved the paper.

References

1. Attribute certificates a new initiative in PKI technology, http://www.baltimore.com/library/whitepapers/acswp-hm.html.
2. Dandhu, R., Bhamidipati, V., Munawer, Q.: The ARBAC97 Model for Role-Based Administration of Roles, ACM Transactions on Information and System Security, 2(1), (1999) 105-135.
3. Ellison C.: SPKI Certificate Documentation, http://www.pobox.com/~cme/html/spki.html.
4. Edwards, N., Rees, O.: High Security Web Servers and Gateways, In Proc. of 6th International WWW Conference, (1997) 927-938.
5. Ellison, C., Frantz, B., Lampson, B., Rivest, R., Thomas, B., Ylonen, T.: SPKI Certificate Theory Request for Comments: 2693, ftp://ftp.isi.edu/in-notes/rfc2693.txt.
6. Ferraiolo, D. F., Barkley, J. F., Kuhn, D. R.: A Role-Based Access Control Model and Reference Implementation Within a Corporate Intranet, ACM Transactions on Information and System Security, 2(1), (1999) 34-64.
7. Ferraiolo, D. F., Cugini, J. A., Kuhn, D. R.: Role-Based Access Control (RBAC): Features and Motivations, In Proc. of 11th Annual Computer Security Applications Conf., (1995) 241-248, http://hissa.ncsl.nist.gov/rbac/newpaper/rbac.html.
8. Gheorghiu, G., Ryutov, T., Neuman, B. C.: Authorization for Metacomputing Applications, In Proc. of the IEEE 7th International Symposium on High Performance Distributed Computing, (1998) 132-139.
9. Giuri, L., Iglio, P.: A formal model for role based access control with constraints, In Proc. of the Computer Security Foundations Workshop, (1996) 136-145.
10. Hashii, B., Malabarba, S., Pandey, R., Bishop, M.: Supporting Reconfigurable security Policies for Mobile Programs, Computer Networks, 33 (2000) 77-93.
11. Herzberg, A., Mass, Y., Mihaeli, J., Naor, D., Ravid, Y.: Access Control Meets Public Key Infrastructure, Or: Assigning Roles to Strangers, http://www.hrl.il.ibm.com/TrustEstablishment/paper.htm.
12. Hitchens, M., Varadharajan, V.: Issues in the Design of a Language for Role-Based Access Control, In Proc. of the 2nd International Conference on Information and Communications Security (ICICS 99), (1999) 22-38.
13. Hitchens, M., Varadharajan, V.: Elements of A Language for Role-Based Access Control, In Proc. of IFIP TC11 Sixteenth Annual Working Conference on Information Security, (2000) 371-380.
14. Jajodia, S., Samarati, P.: A Unified Framework for Enforcing Multiple Access Control Policies, SIGMOD RECORD, 26(2), (1997) 474-485.
15. Johnston, W., Mudumbai, S., Thompson, M.: Authorization and Attribute Certificates for Widely Distributed Access Control, In Proc. of the IEEE 7th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE-98), (1998) 340-345.
16. Kuhn, D. R.: Mutual Exclusion of Roles as a Means of Implementing separation of Duty in Role-Based Access Control Systems, In Proc. of the 2nd ACM Workshop on Role-Based Access Control, 1997.
17. Li, N., Feigenbaum, J., Grosof, B.: A Logic-based Knowledge Representation for Authorization with Delegation, In Proc. of the 12th IEEE Computer Security Foundations Workshop, (1999) 162-174.
18. Lin, A., Brown, R.: The Application of Security Policy to Role-Based Access Control and the Common Data Security Architecture, Computer Communications, Vol. 23, No. 17, (2000) 1584-1593.
19. Lin, P., Lin, L.: Security in Enterprise Networking: A Quick Tour, IEEE Communications Magazine, (1996) 56-61.
20. Liu, Q., Shi, J., You, J.: Separation of Duty in Role-Based Access Control Model, In Proc. of IFIP/SEC2000 (part of the16th IFIP World Computer Congress), (2000) 240-243.

21. Massacci, F.: Reasoning about Security: a Logic and a Decision Method for Role-Based Access Control, In Proc. of the International Joint Conference on Qualitative and Quantitative Practical Reasoning (ECSQARU/FAPR-97), Lecture Notes in Artificial Intelligence, Vol. 1244 (1997) 421-435.
22. Na, S., Cheon, S.: Dynamic Role Assignment in Role-Based Access Control Systems, In Proc. of IFIP TC11 Sixteenth Annual Working Conference on Information Security, (2000) 248-252.
23. Sandhu, R. S.: Lattice-Based Access Control Models, IEEE Computer, 26(11), (1993) 9-19.
24. Sandhu, R. S., Coyne, E., Feinstein, H., Youman, C.: Role-Based Access Control Models, IEEE Computer, 29(2), (1996) 38-47.
25. Seamons, K. E., Winsborough, W., Winslett, M.: Internet Credential Acceptance Policies, In Proc. of the Workshop on Logic Programming for Internet Applications, July 1997, http://www.transarc.com/~winsboro/papers/CAP.html.
26. Thomson, M., Johnston, W., Mudumbai, S., Hoo, G., Jackson, K., Essiari, A.: Certificate-Based Access Control for Widely Distributed Resources, In Proc. of 8th Security Symposium, (1999) 215-227.
27. Winslett, M., Ching, N., Jones, V., Slepchin, I.: Using Digital Credentials on the World-Wide Web, Journal of Computer Security 5(3) (1997) 255-267, http://drl.cs.uiuc.edu/pubs/jcs97.ps.
28. Woo, T. Y. C., Lam, S. S.: Designing a Distributed Authorization Service, In Proc. of IEEE 17th International Conference on Computer Communications (INFOCOM 98), Vol. 2 (1998) 419-429.

Appendix A The Role Syntax

```
 <Role>        ::=  Name:                    <Role Name> .
                    Role-Assigning Policy:   {<Predicate> | null | true } .
                    Authorizations:          <Privileges>
                    .
<Role Name> ::= <Symbolic Atom>
<Privileges>  ::=  <Privilege>*
<Privilege>   ::=  <Pattern><Policy>+ { , <Method>}* .
```

Where, <Pattern> is a user s request pattern expressed by a logical term, <Policy> is a predicate defined by a set of PROLOG clauses, and <Method> is defined as an external function in models.

Appendix B A Bank Example

We use a simple example to demonstrate the ideas contained in the paper. It is assumed that a full-time student has already been issued an identity certificate by an accredited university, and has got a digital driving license issued by the Driving License Agency (DLA) after having passed both the theory and practical tests. The student wants to get the service of car rental or hotel reservation on the Internet, which requires the student to open an account in a recognized bank first to get digital credit certificates. The bank provides a set of E-services, including new account creation, normal bank account transactions. In the following, various kinds of policies are modeled by PROLOG rules, and predicates are only simply defined due to space limitation.

```
Name: account_owners.
Role-Assigning Policy: bank_account_ owners (Accounts, Certificates, Request ).
Authorizations:
            Request = [balance|_], true, get_balance( Account, Balance ).
            // get bank account balance.
            Request = [deposit|_], true, deposit_money( Account, Amount ).
            Request = [withdraw, Account, Amount|_],
                        overdraw_policy( Certificates, Limit),
                        withdraw_test(Limit, Account, Amount),
                        withdraw_money( Account, Amount ).
            .
// Request attribute certificates issued by the bank to the user, and collect bank
account    // numbers into Account_numbers. If the user doesn t have valid and
acceptable
// attribute certificate issued by the bank, it returns false.
bank_account_owners( Accounts, Certificates, Request ) :-
            request_certificates( Certs ),  // Its definition is omitted
            valid_and_accepted_certificates( Certs, Accepted_Certs ),
             //collect valid and accepted certificates from Certs into Accepted_Certs
            get_account_numbers( Accepted_Certs, Accounts),
            // get account numbers of valid and accepted bank account attribute
certificates
            bank_account_no( Request, Account ),
member( Account, Accounts ).
```

Applications of Trusted Review to Information Security

John Yesberg[1] and Marie Henderson[2]*

[1] Information Technology Division
Defence Science and Technology Organisation
PO Box 1500, Salisbury, SA 5108, Australia
john.yesberg@defence.gov.au
[2] School of Computer Science and Electrical Engineering
University of Queensland, Qld 4072, Australia
marie@csee.uq.edu.au

Abstract. The review process is an important part of many everyday activities. We introduce the concept of trusted review for electronic data. The review process is performed using an insertable security device called a Trusted Reviewer. The Trusted Reviewer can be designed to satisfy high assurance evaluation requirements. We show how the Trusted Reviewer can offer increased security in messaging, certification authorities, funds transfer, witnessing, and information downgrade.

1 Introduction

Computer systems and networks are an increasingly integral part of our everyday operations. As a result, we are becoming increasingly dependent on these systems. Another trend, relevant to this article, is that humans are becoming separated from computer processes as use of automation increases and computers begin to undertake interactions between themselves on our behalf. As our traditional paper medium is replaced by electronic methods and computers become more pervasive, we are becoming further dis-associated from tasks and actions once under our control. With the many advantages of computerisation it is evident that these changes will be permanent with the "brave new world of ubiquitous computing" [21]. At the same time computer security incidents are widely acknowledged to be on the rise. There are many reports of attacks, for example, [2, Section 1] describes a number of successful attacks against well known organisations ranging from the Pentagon to NATO. Seemingly, we face a brave new world ridden with risks.

There are several approaches to mitigate these problems. For example, security evaluation and accreditation schemes are in place to provide a measure of confidence where required. Legislation exists for the provision of due care

* This work was performed in part while with DSTO, and in part while with the ISRC at QUT.

with the handling of certain information [10]. Another approach is to use *insertable* security products like firewalls, intrusion detection systems, etc. Indeed, insertable security products can focus on securing particular applications or components rather than the entire system. An example along these lines is to use tamper resistant devices, such as smartcards, to provide key storage and perform cryptographic operations thus protecting cryptographic keys from exposure on a Personal Computer (PC).

In this article, we consider a process familiar to humans that can be transferred to the electronic world to improve security, namely, the process of review. This is achieved using an insertable security device which we call the *Trusted Reviewer*. This device may be used by a human to review data in a high assurance setting. Prototype devices are currently under development at the Defence Science and Technology Organisation.

This article describes both the Trusted Reviewer (TR) itself and a number of its applications. In Section 2 we briefly outline a number of existing information security problems relevant to our discussion. In Section 3 we describe the Trusted Reviewer and outline possible applications in Section 4. Conclusions are given in Section 5.

2 Hurdles: A Brief Survey of Existing Problems

High development costs are making proprietary or custom hardware and software unattractive. As a consequence, commercial off the shelf (COTS) products are being incorporated into sensitive and critical processes. Even defence organisations, which require the most dependable systems, have acknowledged their increased reliance on the use of COTS products [21,23]. In [8], attention is drawn to the contrast between the acceptance of software failures and normal consumer expectations of dependability with other products. Wide utilisation of COTS products and the on-going interconnection of systems has increased exposure to computer attacks in the following ways.

– Decreases in diversity of available products have led to wider susceptibility.
– Inter-connection has enabled remote attacks and increased the scope of attacks as infected computers and systems can be used to spread attacks.
– Economic pressures for rapid development and system evolution has decreased consideration of security issues.
– The increased complexity of systems has made security harder to achieve.

This list is by no means complete but highlights some of the issues now affecting computer security.

The concept of trusted computing has largely remained within the military and research domains. The increased reliance on computer systems to support critical processes will drive this concept into the commercial sector. The first steps have already been taken in this direction, [26], [32][1]. Without trusted platforms it may be necessary to employ external trusted devices to provide the

[1] the intentions of this group may not be as altruistic as they first appear [30]

required process security and assurance. Separate trusted devices can run simplified programs for specific applications allowing users to keep their untrusted platforms for running comprehensive, full featured applications. The idea of using insertable security devices in systems has already generated some high assurance devices, such as [3,24,34]. Some, recognising the vulnerabilities in general purpose computers, have proposed trusted devices for certain applications, such as voting [37] or smartcard PIN entry [31].

Baker [8] argues that the *trusted system* concept is still relevant in modern computer systems. She states that "Any system component is only as trustworthy as the components upon which it depends." This point has been widely overlooked, even with security products. For example, cryptography can be used to secure transmissions between two end points but it cannot secure the ends: indeed it imposes additional security requirements. End point attacks still have to be considered, but are all too frequently ignored.

The following quote from Baker extends the idea of trusted components:

> "Applications derive their functionality and assurance from the strength of the underlying infrastructure; unfortunately dependability of that infrastructure has largely been ignored."

This comment may seem to contradict the concept of insertable security. However, we will show that it does not.

We support efforts to improve the security of application and operating system software. However, with current technology, it would not be cost effective to develop and evaluate them all to the highest standards. Instead, we should try to isolate the security enforcement into small components, so that if we can trust only those modules, the entire system will be secure. A physical world analogy is that by securing the *external* doors, walls, and windows of a house, we reduce the need to have the same level of security on inner doors and walls.

3 Trusted Review: Concepts

Two established principles for the design of a secure system [28] are:

Economy of mechanism: keep the design as simple and small as possible.

Complete mediation: every access to an object must be validated. There must be no path to the object that bypasses validation.

The traditional result of applying these rules is the Reference Monitor, which requires that a secure kernel at the heart of an operating system mediates access to every file. Although many systems contain a reference monitor, it has not proven to provide a very high level of security.

If we assume that there is limited opportunity to improve commodity computing systems (including operating systems), then we need to look for a different way to apply these principles. The trusted reviewer described in this paper mediates access to a special private key. Access is based on decisions an authenticated user makes in response to the displayed document.

In a simple scenario, illustrated in Figure 1, a user Alice has a trusted reviewer (TR) device on her desk, next to her PC. At the beginning of a session, she authenticates to the TR. When it is necessary for her to confirm some critical information, the PC transmits it to the TR[2]. The TR displays the information to Alice, who has an opportunity to read it carefully. If she decides to confirm the information, she presses the "Accept" button on the TR to indicate this. The TR then creates a pair of digital signatures for the information to indicate that it has been confirmed within the trusted environment. The first signature is calculated using Alice's private key, and the second uses a private key belonging to the device itself. The signed information is then sent back to the PC.

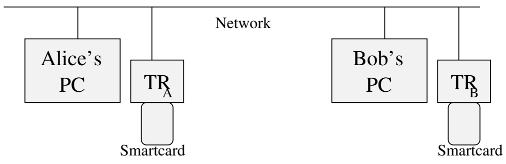


Fig. 1. Network topology for simple Trusted Reviewer scenario

Alice can now email the (TR-signed) information to Bob. Bob's email program can check the signature(s), and it informs him that it is valid. However, Bob is aware (possibly through bitter experience) that COTS computers cannot be trusted to any great degree. Before he relies on, or trusts, this signature he sends it to his own trusted reviewer. The TR verifies the signature, and displays the details of the signature and the document to Bob. Because the TR is *not* susceptible to threats like his PC, Bob is able to trust the TR's display.

This scenario illustrates the essential security functions of the TR:

– Authentication of the user
– Display of information to the user[3]
– Receiving an indication of the user's confirmation
– Generating a digital signature
– Verifying a digital signature

Figure 2 is a logical view of the TR's internal functions and its external interfaces.

[2] Any convenient medium such as USB, Ethernet, or SCSI could be used.
[3] Trusted display and receipt of user's confirmation constitute a "Trusted Path" in the terminology of [25].

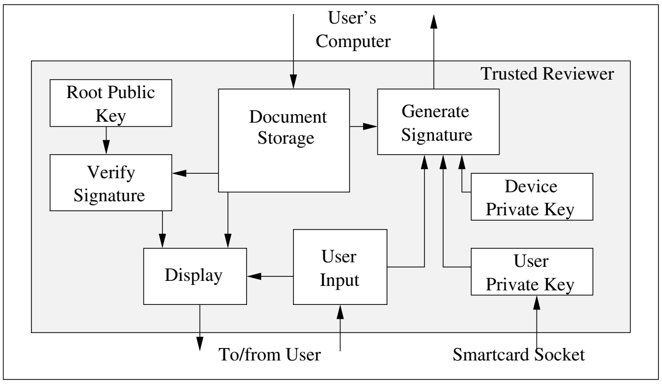


Fig. 2. Functional view of the Trusted Reviewer

3.1 Authentication Mechanism

It would be possible for the TR to use a variety of different authentication mechanisms. Here, we propose the use of a smartcard or similar token [11, for example] that requires entry of a PIN for activation. The smartcard is inserted into the TR (not the computer), and the PIN is entered directly into the TR. This means that a PIN need never be entered into the computer[4] (where untrustworthy software may have access to it). We suggest using public key cryptography to authenticate the user to the TR [1] as this technology can then be used to apply digital signatures to reviewed data. With public key cryptography each entity has a public key and a private key. The public key is made available to all entities. The private key is protected from disclosure and use by anyone other than the rightful owner. We propose that the user's smartcard contain their private signing key.

The TR may present a challenge for the smartcard to sign or use other mechanisms to obtain some signed data from the smartcard. The TR must also have access to the associated public key. We do not outline a preference for the mechanism to achieve this in this article. However, we note that this can be done using TR loaded public keys or by employing Public Key Infrastructure (PKI) methods [1].

Whenever a user is able to insert a smartcard and enter the appropriate PIN, that user is said to be authenticated.

3.2 Display

We have explained that most commodity computers are vulnerable to malicious software. Thus, we cannot rely on these computers to display the actual contents of a document or file to us reliably.

[4] Some commercially available smartcard readers [31] have PIN entry systems which do not require the computer to process the PIN.

We solve this problem by arranging for the information to be forwarded to the TR for display. The TR may have its own built-in display mechanism, or by suitable arrangement of video cables and some switches, it is possible to use the computer's monitor, as shown in Figure 3.

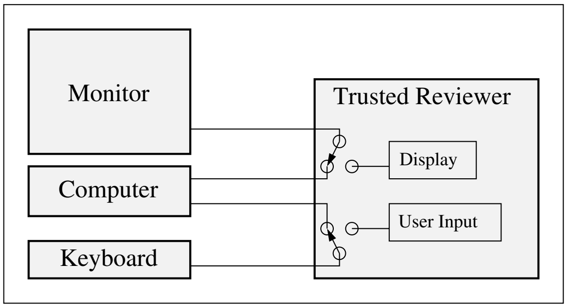


Fig. 3. Video and keyboard switching for the Trusted Reviewer

It is important for the user to be confident that the display is an accurate representation of the information. For example, if the user is about to certify that the document contains no classified information, it may be important that it is not a Microsoft Word file containing some "Track Changes" information, as this would allow a recipient to undo some recent deletions. Similarly, if it were a contract, it would be important to make sure that the user can see every page, and all information on those pages, before they sign. For this reason, it may be appropriate for the TR to use a specialised markup format to indicate how the document is to be displayed.

Providing the user with a reliable, complete, and easy-to-interpret display of the document's contents is a difficult task. Initial implementations of the Trusted Reviewer have been limited to simpler document formats, such as plain text, RTF, and HTML. For perfect security, it would be necessary for the Trusted Reviewer to expose (or filter out) any steganographic [6] content. This is, by definition, challenging, and the subject of continuing DSTO research.

We can see that there will also be a need for a user to scroll or page through a long document. One possibility is to have dedicated keys on the TR to do this. An alternative would be to switch the PC's keyboard, as shown in Figure 3, so that it could be used to operate the trusted reviewer.

3.3 Signature Generation

One of the attractive security features of some smartcards is that the user's private signature key can never leave the card. This means that there is a high degree of certainty that the user's signature can only be created by that card (assuming that the keys and algorithms are not weak, broken, or compromised).

However, this does not guarantee that the user has approved or seen the information being signed, or is even aware of signing events. There are three components involved in creating a digital signature: the private key, the algorithm, and the data to be signed. The overall security of the signature creation process is limited by the security of each of these components. A smartcard can be used to secure the key and algorithmic processing, but it is not able to ensure that only appropriate information is signed. No matter what data is sent to the card, the card will securely create a signature for that data.

We solve this problem by programming the TR so that it can *only* sign information after it has been reviewed and confirmed by an authenticated user. Now it may be possible to use similar software on a PC. However, because the TR is a rigorously evaluated embedded device, it will not be possible for any malicious software to bypass this mechanism in the TR. The same cannot be said for the PC environment.

The user's smartcard may sometimes also be used in an untrustworthy process, such as on the PC. It is necessary, therefore, to ensure that any signature created by a TR is clearly identifiable. We propose to solve this problem by having the TR append its own additional signature to that of the user.

This implies that the TR needs to be an end-entity in the public key world, having its own identity, public-private key pair, and certificate. Also, it means that the private key needs to be stored securely inside the TR, or otherwise communicated securely to it from some central store. We propose that the TR would have its own smartcard locked (via some physically secure means) into place.

3.4 Signature Validation

The verification of digital signatures is subject to the attacks that we outlined for digital signatures themselves. Signatures can be validated on a PC, and the results displayed to the user. However, as indicated earlier, when a PC is vulnerable to potentially malicious code, we cannot trust what the PC says. Therefore, if we need to rely on a signature, it is important that the validation of that signature, and the display of the result, are both carried out in a trustworthy way. This motivates use of the TR for signature validation.

To support validation it may be necessary for the TR to contain a root public key (a key from which all trust is referenced) in secure storage. There are a number of complex issues associated with public key infrastructures upon which the Trusted Reviewer will rely, such as the discovery of valid certificate paths. In this paper, we do not attempt to solve these problems. However, we do note that the TR does not need to be able to perform complex searches of directories; such tasks, as well as preliminary validation of certificates, can be delegated to the PC, with the results being forwarded to the TR for final verification.

3.5 Assurance

In regards to security, we need to address the question of why we can rely on the trusted reviewer when we cannot rely on the user's computer. There are really two questions here that need answering.

Question 1: why is the TR implicitly more trustworthy?

The TR is a far simpler device than a PC. It has a specific purpose and performs a small number of tasks. The TR would be an embedded device with all software in ROM and consist of a simplified operating system, with no disk drives or any other built in non-volatile storage. It would be limited to a single, well defined function. Hence it is feasible for it to be evaluated very carefully. In fact, we propose that EAL-7 (Evaluation Assurance Level 7 of the Common Criteria [9]) would be an achievable level. This level is the highest supported by the Common Criteria. This judgement is based on our experience with the evaluation of other devices to similar levels, [35,36,34,12]. In comparison, a PC is composed of complicated hardware and software, none of which can be considered to be in a stable or static state. Evaluation is difficult at best, with only lower EAL levels attainable.

Question 2: why can we trust a system composed of untrusted PCs and trusted reviewers?

The function of the TR is to display the content to the user no matter what this content contains. Once the information is passed to the TR from the PC then the information can not be altered and can be considered to be in a static state. The content is displayed under the control of the TR to the user. The user is wholly responsible for judging whether this content is acceptable. If the user decides to accept the content then the TR will cover the information with its own digital signature before passing the information and the signature back to the PC. Similarly, a TR can be used to check a TR's signature and the content it is associated with. Therefore the untrusted PCs do not affect the TR operation[5].

3.6 Threat Model

In this section, we explain and summarise how the TR can offer improved security. The essential security-breach conditions that we aim to prevent are:

1. Information being signed with a user's (Alice's) signature, without that user's knowledge of what is being signed, and/or without their deliberate decision to make and be bound by that signature; or
2. A user (Bob) being convinced that another user (Alice) deliberately chose to sign and be bound by a document, when this is not the case.

[5] Of course, the PCs can cause a denial-of-service attack, but this is true whether or not the TR is involved.

The TR offers the following features:

- *Allows PCs to be virus-prone.* Assume firstly that Alice's computer contains malicious software.
    - If this software modifies the document before the signature is applied, Alice will detect this when she reads it in the trusted display.
    - If this software modifies the document after the signature is applied, the signature will not be valid.
    - If this software refuses to send the information to the TR, no signature will be created. (This may be a denial of service, but it doesn't fit within our definition of a security breach.)
    - This software cannot obtain Alice's or the TR's private keys, nor cause any information to be signed with them, until it has been reviewed and physically accepted by Alice.

  Now, assume that Bob's computer contains malicious software.
    - Bob's computer may display information to Bob claiming that it has been signed by Alice. But Bob will not believe this unless the TR is in control of the screen. (The TR may use a LED or other device to indicate that the screen is presently trustworthy.)
    - Bob's computer may not pass information to the TR when it should. This is another denial of service, but not a security breach as we have defined it.
    - Bob's computer may modify the information before it reaches the TR. But then the signature(s) will not be valid, and the TR will indicate that the document has not been validly signed.
    - Bob's computer may modify the information after the TR has displayed it. But Bob will already have seen the real information and that it was validly signed.
- *Allows communications between PC and TR to be open and unauthenticated.* The TR does not trust the source of any communications. The user is required to visually verify information before it is signed.
- *Allows user's smartcard to be used in untrustworthy devices.* If the user's smartcard is used in a PC containing malicious software, a document could be signed with the user's signature, without the user's knowledge. However, such a document could not be signed with the TR's private key, because this key is never made available outside the TR. Therefore, the recipient's TR will not indicate that the document has been validly signed.
- *Protects the PINs of the users' smartcards.* Unless the users choose to use their cards in untrusted environments, we can guarantee that the PIN is never made available outside the TR.

The TR does not solve all security problems. The security that it offers rests on a number of pillars.

Public Key Cryptography

The TR uses public key cryptography to create signatures. If the keys or algorithm can be broken, either by brute-force or other attacks, then the system will not be secure.

Public Key Infrastructure
The TR relies on the security of its private keys, and of the root public key. If these can be compromised, the overall system will not be secure.

User
If the authorised user behaves irresponsibly, then the system will not be secure. Examples of irresponsible behaviour include habitually pressing the “Accept” button, without carefully reading the information displayed by the TR; and believing the reported validity of a signature if the TR is not in trusted display mode. Another problem, is that the information displayed may be ambiguous[6]. However, this problem is not specific to the TR.

4 Trusted Review: Applications

4.1 High Assurance Signatures

In the previous section, we described a high assurance signature scenario for the trusted reviewer. Such a system provides a recipient with a very high level of confidence that it not only came from the particular signer, but also that this signer deliberately confirmed that she would accept and be legally bound by the information that was displayed. This *high assurance* signature can be contrasted with existing systems which rely on operating system services to provide:

– authentication of the user;
– access and reuse control for the PIN or private key;
– display of the information to the user; and
– confirmation that the information should be signed.

When an operating system vendor builds in features that allow automatic exploitation of one or more of these services, the signature may not be very meaningful. Even the lay community is aware of the Melissa [7] and I-Love-You [19] viruses: malicious software which transmit email from your account without you knowing about it. It would not be difficult to construct a virus which would create and append a valid digital signature to such emails.

4.2 Certification Authority

We have described some problems with current mechanisms for creating digital signatures. One important area where signatures are used is in the creation of digital certificates. Trusted authorities, known as Certificate Authorities (CAs), create and digitally sign certificates. The CAs signature attests to the correctness of the details within a certificate (usually linking an identity or authorisation to a public key value).

[6] As an example of undesirable ambiguity, the verb *cleave* can mean “stick together” or “break apart”.

It is normal for CA software to be loaded onto an existing (and relatively untrustworthy) operating system. Even if the CA software itself has been stringently evaluated, so that it only creates signatures for appropriate certificates, there is nothing to stop the operating system from accessing the CA's private key[7], or from providing inappropriate input to the CA software.

For critical applications, it would be appropriate to use a TR as a CA workstation. This would have the advantage that no untrustworthy software could ever access the CA's private key, or cause a signature to be created without the operator's knowledge.

4.3 Witnessing

Many legal aspects of our civil society require the witnessing of documents and signatures [13]. For example, a person's signature on their legal will needs to be witnessed by two other people. The major application is to prevent forgery and fraud. The witness may provide evidence regarding the signing event if it is later called into question or denied. With paper based witnessing of a signature, this usually means checking that a document doesn't have blank spaces or alterations, and then watching a person make a mark on the page, before the witness applies their mark. The witness's mark attests that they have observed this event. It does not require authentication of the signatory or careful review of the meaning of the document, as the witness is not bound by its contents. Currently, no method for electronic witnessing exists, although [22] presents some requirements.

The TR may be used to support a witnessing application in the following way. The signatory uses a TR to apply their digital signature to the reviewed information (as described in Section 4.1) in the presence of the witness. The witness can then use a TR to review the information and verify the associated signature. The TR can display the information to the witness and also the presence of a valid digital signature. If the witness is satisfied by the results of this process then they digitally sign over the entire document and signature (or some fingerprint of this information).

With paper-based witnessing it is sometimes recommended that everyone uses the same pen (this strengthens the evidence that the signing was performed with everyone present). This could be achieved electronically by using a single TR device which applied its signature to each signature in turn. By including a trusted real-time clock within the TR, signatures could also be reliably timestamped. This would provide additional benefits in witnessing applications, and possibly auditing and other non-repudiation services.

4.4 Electronic Funds Transfer

Banks and commercial organisations use dedicated Automatic Teller Machines and special purpose EFTPOS devices. Yet many banking operations can today

[7] Even if the key is stored on a smartcard, the OS could cause the smartcard to sign an inappropriate certificate, without the operator's knowledge.

be performed over the Internet. Although there are many risks [27,5], banks (and their customers) appear to have decided that it is commercially worthwhile to offer (use) these services. If TR devices became prevalent, it might be appropriate for banks to require their use for fund transfer requests. Manninger [20] has proposed a similar system for Internet banking.

4.5 Multilevel Security Information Downgrade

The final application we describe in this paper is actually the one for which the TR was originally invented. Consider an organisation with two networks, one classified, say "Secret", and the other one Unclassified. In an ideal situation, there would be no need to transfer information between these two networks. However, in reality, these information flows are required.

From a confidentiality view-point, there are no problems with allowing information to pass from the Unclassified to the Secret network (integrity and availability issues are outside the scope of this paper). Transferring information from a Secret network to an Unclassified network is, however, potentially dangerous. We assume that each network is comprised of COTS (untrusted) hardware and software[8]. Therefore, we need to cater for the possibility that there may be malicious software on the Secret network. Such software may insert Secret information inside what would otherwise be Unclassified files. If an operator transfers the file onto a magnetic medium, and then to the Unclassified network, we would have a security breach.

Some "guard" systems [15,17,16,33] use labels to indicate which files are suitable for passing from a Secret system, through a filter, to the Unclassified system. However, these labels are typically generated on untrusted platforms. The security therefore relies on the correct operation of an untrustworthy system. This is hardly a desirable situation.

With these problems in mind, we propose that a TR could be used to attach a signed label to a file. This label could not be forged, nor created without a human user's deliberate confirmation. A Guard (a sibling in the TR product range) would verify the signature of the label, and check that the classification within the label was appropriate before transferring it to the unclassified network.

In Figure 4, we see Charlie's PC connected to the Secret network. He wants to send an Unclassified email to Dora, whose PC is connected to the Unclassified network. Charlie first creates the email, and then his PC sends it to his TR. The TR generates a "trusted display" of the contents of the message (including the label "Unclassified"), which Charlie is able to review. When he presses his "Accept" button, the TR signs the message with Charlie's signature and its own signature, and returns it to the PC.

[8] The security of the Secret network is provided by physical access control means, rather than any logical mechanism: e.g. only people with Secret clearances are allowed to be in the same room as the Secret network.

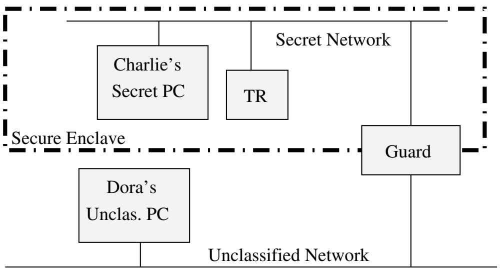


Fig. 4. Multilevel Security Downward Transfer Scenario

The PC now forwards the message to the Guard via the email system. The Guard checks that the message:

– has an "unclassified" label;
– has been signed by an authorised user; and
– has been signed by an authorised TR.

After these checks are passed, the Guard is able to forward the message to Dora's computer, again via the email system.

Space limitations preclude a detailed discussion of the frameworks for managing authorisation. Readers will recognise that a supporting PKI could be designed with ease. A separate option might require more than one person on the Secret system to approve the release of information.

5 Conclusions

The concept of review is an important part of many of our paper processes. Until now there has been no secure electronic analogue. Unfortunately, software technology does not currently allow vendors to deliver highly dependable products. The risks associated with the performance of many security-critical operations using commodity applications and operating system software are therefore unacceptably high. We have shown that a new device, called the Trusted Reviewer, allows a number of the risks to be reduced to much more acceptable levels.

The Trusted Reviewer provides authentication, review, and digital signature creation and validation functions. Because it is so simple in both function and design, it will be feasible to develop, evaluate, and certify the device to a high level, such as EAL-7 in the common criteria. As we have pointed out in this article, a device like the Trusted Reviewer has applications in many areas, including Certification Authorities (CAs), organisational messaging, Electronic Funds Transfer (EFT), witnessing, and inter-security-domain information transfer. The Trusted Reviewer can achieve this without limiting users' access to commodity PCs running diverse applications on the shared network.

The Defence Science and Technology Organisation is developing prototype Trusted Reviewers to explore and demonstrate a number of these issues.

6 Acknowledgements

The authors wish to thank and acknowledge the contributions made by colleagues at the Defence Science and Technology Organisation, particularly Peter Drewer, Maris Ozols, Peter Duffett, and Mark Anderson, who have played crucial roles in the development of the Trusted Reviewer. Further thanks is offered to the anonymous referees who gave constructive reviews.

References

1. C. Adams and S. Lloyd, *Understanding Public-key infrastructure: concepts, standards, and deployment considerations*, Macmillan Technical Publishing, 1999.
2. J. Allen, A. Christie, W. Fithen, J. McHugh, J. Pickel and E. Stoner, *State of the Practice of Intrusion Detection Technologies*, Technical Report
3. M. Anderson, C. North, J. Griffin, R. Milner, J. Yesberg, and K. Yin, *Starlight: Interactive Link*, Proc. 12th. Annual Computer Science Security Applications Conference, IEEE Computer Society Press, 55–63, 1996.
4. M. Anderson, J. Yesberg, D. Marriott, L. Nayda, K. Hayman, M. Stevens, and B. Beahan *Communications Security and Trusted Path Method and Means* Australian Patent 706073, 1991.
5. R. Anderson, *Liability and Computer Security - Nine Principles*, ESORICS 94, `http://www.ftp.cl.cam.ac.uk/ftp/users/rja14/liability.pdf`).
6. R. Anderson and F. Petitcolas, *Information Hiding: An Annotated Bibliography*, 1999. `http://www.cl.cam.ac.uk/~fapp2/steganography/bibliography/Annotated_Bibliography.pdf`
7. Australian Media Pty Ltd, *Melissa.Net Portal*, 2001, `http://melissa.net/melissavirus.htm`
8. D. Baker, *Fortresses Built Upon Sand*, Proc. of the New Security Paradigms Workshop, pp. 148–153, 1996.
9. Common Criteria, `http://www.commoncriteria.org/`
10. Commonwealth of Australia, *Privacy Act, 1988*, Australian Government Printing Service. `http://www.austlii.edu.au/au/legis/cth/consol_act/pa1988108/index.html`
11. Dallas Semiconductor, *iButton* 2001. `http://www.ibutton.com`
12. Defence Signals Directorate, *Evaluated Products List*, April 2000. Available at `http://www.dsd.gov.au/infosec/aisep/EPL.html`
13. Department of Justice and Attorney-General, Queensland, *Administrative Duties of Commissioners for Declarations and Justices of the Peace*, 2000.
14. C. Ellison and B. Schneier *Ten Risks of PKI: What You're Not Being Told About Public Key Infrastructure*, Computer Security Journal, **16**(1), 1–7, 2000. `http://www.counterpane.com/pki-risks.html`
15. T. Fiorino, P. Casey, M. Easley, and R. Jordan. *Lessons learned during the life cycle of an MLS guard deployed at multiple sites* Proc. IEEE 11th Annual Computer Security Applications Conference, 1995. pp. 99–107.
16. L. Gagnon, *An overview of the USAFE Guard System*, Proc. 13th National Computer Security Conference, 1990. pp. 218-227
17. P. Greve, J. Hoffman, and R.E. Smith, *Using type enforcement to assure a configurable guard*, Proc. IEEE 13th Annual Computer Security Applications Conference, 1997. pp. 146–154.

18. M. Henderson, M. Burmester, E. Dawson and E. Okamoto, *The Dark Side of Digital Signatures*, Business Briefing — Global Information Security, to appear.
19. McAfee.com Associates, *Virus Profile: VBS/Loveletter.a* 2001, `http://vil.mcafee.com/dispVirus.asp?virus_k=98617&`
20. M. Manninger and R. Schischka, *Adapting an electronic purse for Internet payments*, ACISP'98, Springer-Verlag LNCS 1438, pp 205–214, 1998.
21. J. McLean and C. Meadows, *The Future of Information Security*, Themes and Highlights of the New Security Paradigms Workshop. `http://chacs.nrl.navy.mil/publications/CHACS/1999/index1999.html`
22. A. McCullagh, W. Caelli, and P. Little, *Electronic Signatures: Understand the Past to Develop the Future*, University of New South Wales Law Journal, 1998. `http://www.law.unsw.edu.au/unswlj/ecommerce/mccullagh.html`
23. C. Meadows and J. McLean, *Security and Dependability: Then and Now*, Computer Security, Dependability, and Assurance: From Needs to Solutions, IEEE Society Press, 166–170, 1999. `http://chacs.nrl.navy.mil/publications/CHACS/1999/index1999.html`
24. A. Moore, *Network Pump (NP) Security Target*, Naval Research Laboratory, Memorandum Report 5540–00-8459, May 2000.
25. National Computer Security Center, *Department of Defense Trusted Computer Security Evaluation Criteria*, Report DOD5200.28-STD 1985.
26. National Security Agency, *Security-Enhanced Linux* `http://www.nsa.gov/selinux/`
27. T. Redhead and D. Povey, *The Problems With Secure On-line Banking.* In Proceedings of the XVIIth annual South East Asia Regional Conference (SEARCC'98). July, 1998, `http://security.dstc.edu.au/papers/searcc98-bank/`
28. J. Saltzer and M. Schroeder *The protection of information in computer systems* Proc. IEEE **63**(9) 1278–1308, 1975.
29. B. Schneier *Why Digital Signatures are Not Signatures*, Cryptogram, November 15 2000. `http://www.counterpane.com/crypto-gram-0011.html#1`
30. E. Smith, *Trusted Computing: Trusted by Whom?* `http://www.brouhaha.com/~eric/editorials/trusted_computing.html`
31. Spyrus, *Rosetta Personal Access Reader*, 2001. `http://www.spyrus.com/content/products/rosetta/PAR2.asp`
32. Trusted Computing Platform Alliance, (for information `http://www.trustedpc.org/home/home.htm`)
33. R. Vick, *An overview of the AMC WWMCCS CAT Guard*, Proc. IEEE Eighth Annual Computer Security Applications Conference, 1992. pp. 46–54.
34. Vision Abell Pty Ltd (now Tenix Defence Systems), *Data Diode v1.2 Security Target*, 1998.
35. Vision Abell Pty Ltd (now Tenix Defence Systems), *Interactive Link v3.0 Security Target*, 1999. `http://www.systems.tenix.com/PRODUCTS/c3i/INTERLINK.HTML`
36. Vision Abell Pty Ltd (now Tenix Defence Systems), *Interactive Link Multiple Computer Switch v2.0 Security Target*, 1999.
37. P. Winkler, *Electronic Trusted Party*, US Patent 5117358, 1992. `http://www.delphion.com/details?pn=US05117358__`

Network Security Modeling and Cyber Attack Simulation Methodology

Sung-Do Chi[1], Jong Sou Park[1], Ki-Chan Jung[1] and Jang-Se Lee[1]

[1] Department of Computer Engineering
Hangkong University, Seoul, KOREA
{sdchi, jspark, prayccc, jslee2}@mail.hangkong.ac.kr

Abstract. The major objective of this paper is to develop the network security modeling and cyber attack simulation that is able to classify threats, specify attack mechanisms, verify protection mechanisms, and evaluate consequences. To do this, we have employed the advanced modeling and simulation concepts such as System Entity Structure / Model Base framework, DEVS (Discrete Event System Specification) formalism, and experimental frame concept underlying the object-oriented S/W environment. Our approach is to show the difference from others in that (i) it supports a hierarchical and modular modeling environment, (ii) it generates the command-level behavior of cyber attack scenario, (iii) it provides an efficient model building environment based on the experimental frame concept, and (iv) it supports the vulnerability analysis of given node on the network. Simulation test performed on sample network system will illustrate our techniques.

1 Introduction

For all practical purposes, international boundaries have been eliminated in cyberspace. The growth of information technology and almost universal access to computers has enabled hackers and would be terrorists to attack information systems and critical infrastructures worldwide [1]. A cyber attack is an attack on a computer and network system, consisting of computer actions (e.g., remote or local connection, computer file access, program execution, etc.) to compromise the secure operation of the computer and network system. As we increasingly rely on information infrastructures to support critical operations in defense, banking, telecommunication, transportation, electric power and many other systems, cyber attacks have become a significant threat to our society with potentially severe consequences.

A computer and network system must be protected to assure security goals such as availability, confidentiality and integrity. That is, the deep understanding of system operation and attack mechanisms is the foundation of designing and integrating information protection activities [2]. Therefore, the advanced modeling and simulation methodology is essential for classifying threats, specifying attack mechanisms, verifying protective mechanisms, and evaluating their consequences. That means, we need to establish the advanced simulation methodology for analyzing the vulnerability, survivability, etc. of given infrastructure as well as the expected consequences of successful attacks and the effect of the defense policy. Such a methodology may be able to support to find unknown attack behavior if more refined models are allowed. Actually, many fields use modeling and simulation technique to support the analysis and insight into building better systems, but the field of information protection has not produced significant research results to date. Perhaps this is due to the extreme complexity of the cyber attack and defense problem, the enormous size of the search space, the lack of good data on attacks and defenses, the inability to derive consequences in a systematic way, or the lack of a coherent view of information protection [3].

In order to overcome these limitations, we have proposed the network security modeling and cyber attack simulation by employing the advanced modeling and simulation concepts such as System Entity Structure / Model Base framework, DEVS formalism, and experimental frame concept [4] underlying the object-oriented S/W environment. Our approach is to show the difference from others in that (i) it supports a hierarchical and modular modeling environment, (ii) it generates the command-level behavior of cyber attack scenario, (iii) it provides an efficient model building environment based on the experimental frame concept, and (iv) it supports the vulnerability analysis of given node on the network.

The remainder of this paper is organized as follows. First, it briefly reviews a background on conventional information security modeling approaches. Then, it proposes a model-based approach for designing the network security modeling and cyber attack simulation system. This is followed by the case study.

2 Background on Network Security Modeling and Simulation

Many fields use modeling and simulation to provide analysis and insight into building better systems, but the field of network security has not produced significant research results to date. Since we are modeling very complex phenomena involving mixes of human behavior and interactions of complex interdependent systems with time bases ranging from nanoseconds to years. There is no widely accepted information physics that would allow us to make an accurate model, and the sizes of the things we are modeling are so large and complex that we cannot describe them with any reasonable degree of accuracy. Also there are no consensus on how to describe a network security system, and no set of commonly accepted metrics upon which to base a set of measurements to be used for simulation. Despite of these difficulties of network security modeling and simulation, it actually provides much of the best justification for actively pursuing it. The high cost of running real-world attacks, the limited extent to which they exercise the space of actual attacks, and the high potential for harm from a successful attack conspire to make some other means of analysis an imperative [3].

Cohen [3], who was a pioneer in the field of network security modeling and simulation, interestingly suggested a simple network security model which is composed of network model represented by node and link, cause-effect model, characteristic functions, and pseudo-random number generator. However, cyber attack and defense representation which is based on cause-effect model [3] is so simple that practical difficulty in application comes about. Amoroso suggested that the intrusion model [6] should be represented by sequence of actions, however, the computer simulation approach was not considered clearly. Wadlow [7] suggested an intrusion model with four classified states such as COOL, WARM, HOT, and COOLDOWN, but it failed to go beyond the conceptual modeling level. Finally, Nong Ye [2] noticeably proposed a layer-based approach to complex security system, but failed to provide a practical modeling and simulation techniques of the relevant layers. Nong Ye s approach is that the high layers among four such as objectives, conceptual, functional, and physical level can be represented in a simple model, and have rapid simulation run with minimal number of parameters, however, it is too simple to be meaningful. In the low layers, it is represented in complex model with accuracy, but it requires the massive amount of data and enormous amount of simulation time so that it is hard to model. To deal with these problems, this paper attempts to provide an appropriate modeling approach through functional level (command-level) access to cyber attack, and provide a network security model and its simulation by applying a discrete event simulation technique.

3 Proposed Approach

The overall design methodology for the network security simulation systems can be better understood by organizing them within a set of layers that characterizes its design structure as shown in Fig. 1. Layer I provides a hierarchical and modular modeling and simulation S/W environment. Layer II supports the command-level dynamic model construction based on the experimental frame concept. Finally, the network security simulation system can be accomplished in the Layer III.

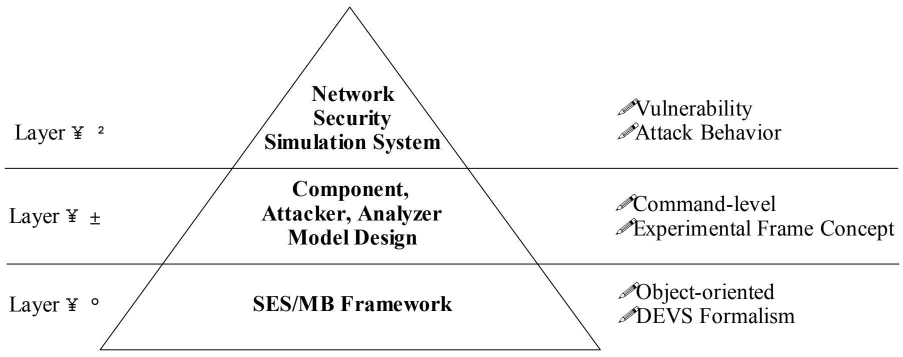


Fig. 1. Layered approach for network security simulation systems

3.1 Layer I: SES/MB Framework

This layer relies on the object-oriented programming environment to provide the ability to specify models that populate the model base that it organizes. The properties of this lowest layer make it possible to realize similar properties at the higher layers. The SES/MB framework [4] as a step toward interfacing the dynamic-based formalism of simulation with the symbolic formalism of AI can be suitably adopted for this layer. It basically consists of a system entity structure (SES) and model base (MB). The SES represents the knowledge of decompositions, taxonomies, coupling specification and constraints. Hierarchical and modular simulation models may be constructed by applying the transformation operation to the SES. The model base contains models that are procedural in character, expressed in discrete event system specification (DEVS) formalism, a theoretically well-grounded means of expressing modular discrete event simulation models. A DEVS is a structure [4,5]:

$$M = < X , S , Y , \delta_{int} , \delta_{ext} , \lambda , ta >$$

where *X* is the set of input event types, *S* is the sequential state set, *Y* is the set of external event types generated as output, δ_{int} (δ_{ext}) is the internal (external) transition function dictating state transitions due to internal (external input) events, λ is the output function generating external events as the output, and *ta* is the time advanced function.

Fig. 2 shows SES of the network security model for information infrastructure. NETWORK, which is the root entity, can be classified into ATOMIC with single network and COMPOSITE with multiple networks. ATOMIC is divided into COMPONENT for security factors consideration and SECURITY-AGENTS for multiple entity. COMPONENT is again divided into O/S and several SERVICES. Besides, it can be divided into more detailed subsystems such as BORDER, INFRASTRUCTURE, BUSINESS, DESKTOP. In parallel, SECURITY-AGENTS can be divided into ATTACKER which generates attack scenario and ANALYZER which analyzes the simulation results. COMPOSITE is divided into more detailed levels such as NETWORKS, a multiple entity, which can link multiple network groups and LINK which links them all. Fig. 3 shows a pruned entity structure (PES) obtained by applying the pruning operation into the SES. A final simulation model in each entity of the PES can be established by attaching the dynamics models discussed next.

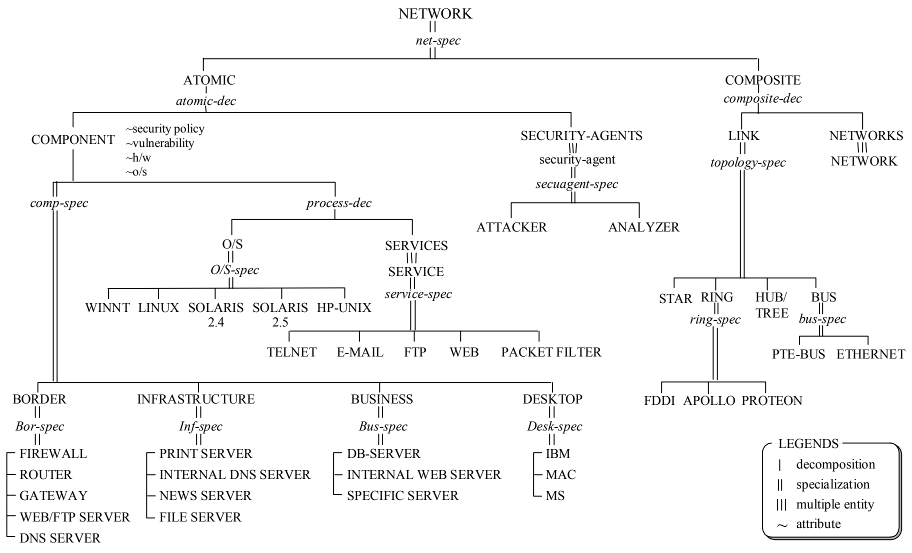


Fig. 2. SES representation of network systems

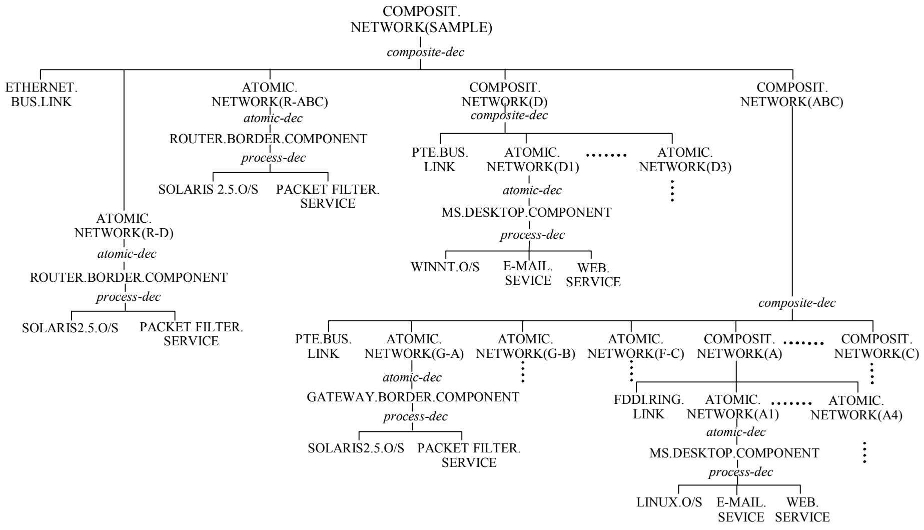


Fig. 3. A PES example

3.2 Layer II: Component, Attacker, and Analyzer Model Design

In this layer, the command-level component modeling can be constructed on the basis of the experimental frame concept. Although a network model can be tested in a stand-alone fashion, it really does not come to life until it is coupled with modules capable of providing it input and observing its output. Thus, the experimental frame concept [4] may be suitably utilized to couple with a given model (network model), generates input external events (cyber attack commands), monitor its running (consequences), and process its output (vulnerability). Fig. 4 depicts the modeling approach with the experimental frame module underlying SES/MB framework.

In Fig. 4, Attacker inputs planned commands one by one into Network as well as Analyzer. Simulation proceeds by Network s responding to Attacker as well as Analyzer. If enough data are collected for analysis, Analyzer terminate simulation by sending *stop* command to Network and Attacker. Then it analyzes each component s vulnerability through statistical procedure on the collected commands and attack results. A detailed modeling method can be illustrated as follows:

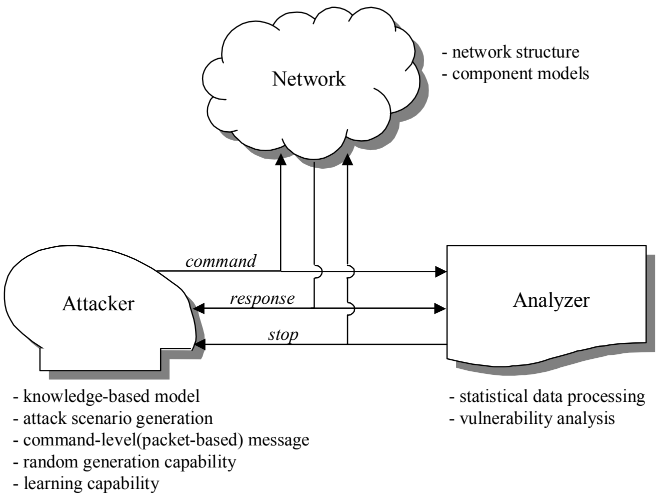


Fig. 4. Network security modeling approach

- Network Component Modeling

As described in the preceding section, network component model comprises various services such as Telnet, E-mail, Ftp, Web, and Packet Filtering. Dynamics of these component models can be represented in various ways according to their respective state variables such as service type, H/W type, and O/S type, etc. Fig. 5(a) is a typical example of DEVS representation of component model. In Fig. 5(a), the external transition function processes the external input through the *in* port by applying command-table represented in pre/post-condition when phase is *passive*. During the procedure, it remains in *busy* state. On the other hand, the internal transition function, when phase is *busy*, is converted to *passive* and the output function delivers processed results in packet through *out* port.

```
Component Model
state variables
    Service-type, H/W-type, O/S-type
    Registered-User-list, Queue-size, etc.

external transition function
    case input-port
        in: case phase
                passive: execute command-table(command)
                         hold-in busy processing-time
                busy :   continue
        else: continue

internal transition function
    case phase
        busy: passive

output function
    case phase
        busy: send packet(result) to port out
```

(a) Network component model

```
Attacker Model
state variable
    scenario-type, target-host

external transition function
    case input-port
        in: case phase
                passive: next command := scenario-table
                         hold-in active attacking-time
                active : continue
        else: continue

internal transition function
    case phase
        active: passive

output function
    case phase
        active: send packet(command) to port out
```

```
Analyzer Model
state variable
    num-attack, num-success-attack, vulnerability

external transition function
    case input-port
        in: store result-table(command, states)
        else: continue

internal transition function
    case phase
        active: passive

output function
    case phase
        active: analyze result
```

(b) Attacker model (c) Analyzer model

Fig. 5. DEVS representation of network security models

Based on this basic behavior model, command-level modeling can be accomplished by grouping and characterizing of commands that are used in various services. Table 1 shows an example of command-level modeling using pre/post-condition representation in Unix. Here pre-condition represents the condition for executing the command, output represents the results by command execution, and post-condition represents the changed properties after command execution. For example, pre-condition for the execution of rmdir command is to confirm the emptiness of the directory for deletion, output should be a directory deletion, and finally as a post-condition, the directory property should be changed.

Table 1. Pre/post-condition representation of Unix commands (partially-shown)

Command	Pre-condition (current states)	Output	Post-condition (next states)
more	-	Brows of page through a text file	-
pwd	-	Return working dire-ctory name	-
rmdir	Check the file exist-ence	Remove directory en-tries	Change directory attr-ibutes
cd	Check the file exist-ence	Change working dire-ctory	Change directory attr-ibutes
vi	Check the file exist-ence	Display or edit file	Change file attributes
mv	Check the file exist-ence	Move files	Change file attributes
rm	Check the file exist-ence	Remove file entries	Change file attributes
chmod	Check the file exist-ence	Change the permiss-ion mode	Change permission attribute

- Attacker Modeling

The attacker model outputs a sequence of attacking commands according to its attacking scenario. The basic mechanism that produces this behavior is the next command := scenario-table and hold-in active attacking-time phrase in the external transition function shown in Fig. 5(b). This phrase returns the model to the same phase, *active* after each external transition and schedules it to undergo a next transition in a time given by attacking-time. Just before the internal transition takes place, the output of next command is proceeded by the pre-defined scenario table.

- Analyzer Modeling

The analyzer model is designed to gather the statistics and analyze the performance index such as the vulnerability of each component on given network. For the simulation convenience, we have defined the component vulnerability as the number of successful attacks divided by the total number of attempted attacks. To do this, the analyzer stores commands that arrive at its *in* port on the result table as shown in Fig. 5(c).

3.3 Layer III: Network Security Simulation System

Fig. 6 shows the overall methodology using the SES/MB. Phase I represents the conceptual specification stage, in which the decomposition, taxonomies, coupling specification and constraints of given information network system can be specified by SES. In Phase II, the network component models as well as the cyber attack, defense, and consequence models can be built and saved into MB. In phase III, the simulation model may be constructed by integrating the dynamic models in MB along with the network structure of the SES so that the cyber attack simulation can be performed. Finally, the simulation result can be analyzed in Phase IV so that the security characteristics and policies of each network component may be evaluated.

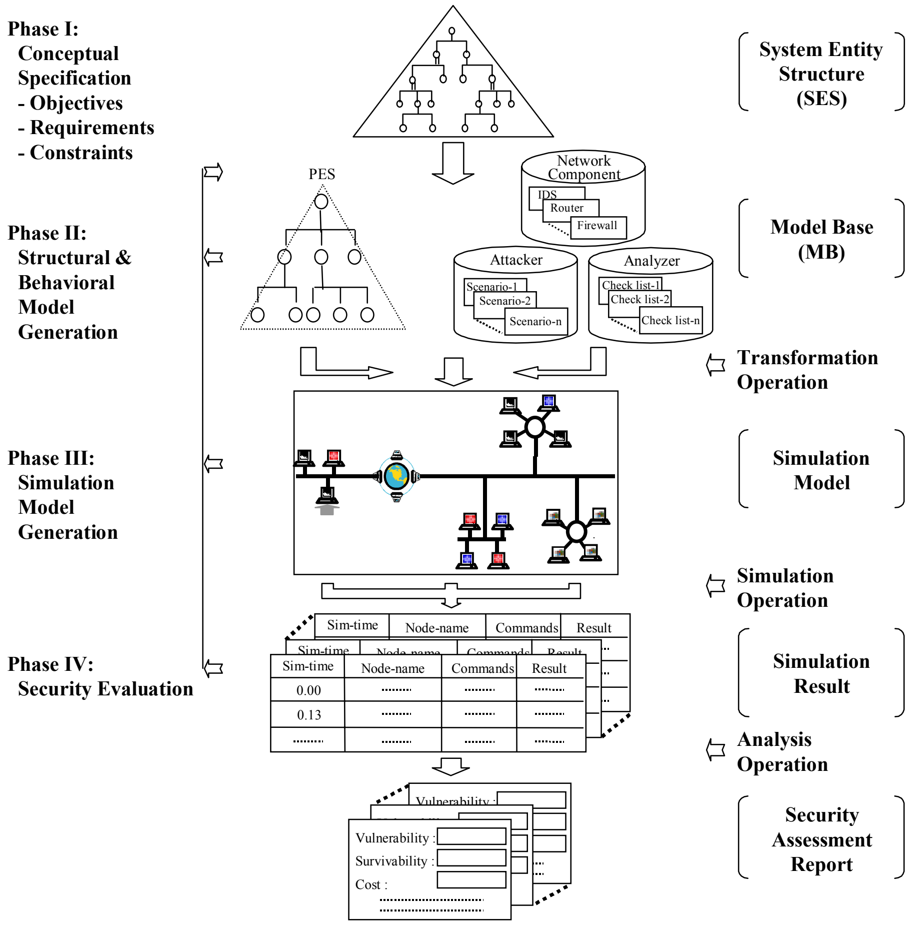


Fig. 6. Overall methodology

4 Case Study

This section examines the feasibility of the proposed methodology through the case study on the sample network. Fig. 7 shows a sample network for simulation test. It includes LAN with client and server computers, topology like ring, bus, etc., and WAN with multiple LAN via router or gateway on Internet. In addition, each node may be connected to attacker model, thus any node can generate packet. These packets can move to destination node through node models, topology models, and/or router models. Node model which receives packet responds in the same way after conducting commands that are received in the packet. Due to the space limitation, this case study will provide only an example of simple scenario as follows: *How to access to a system with no user account by acquiring general user account through SYN flooding, IP spoofing, and old bugs in SUN O/S v1.4.x [8,9]*

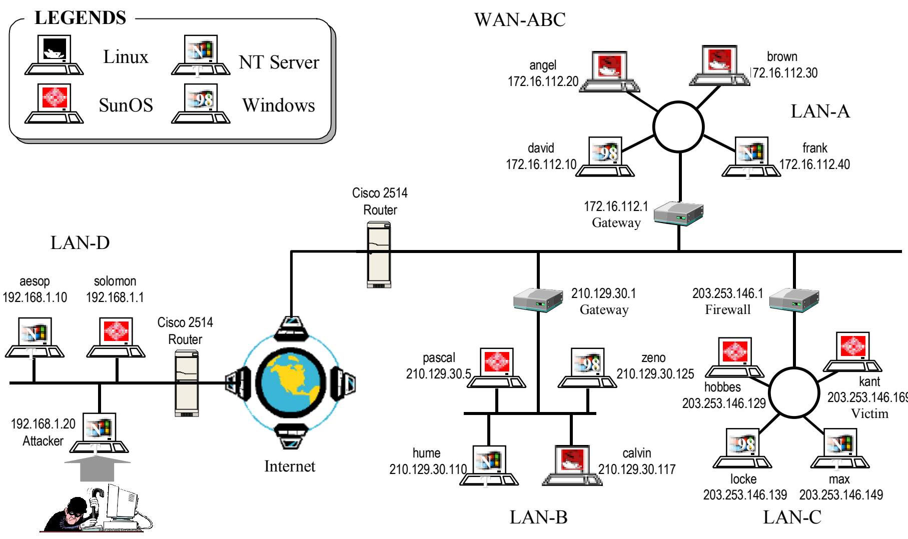


Fig. 7. Simulation model example

Table 2 illustrates the simulation results from cyber attack scenario to acquire general user account. Here, Time means a simulation run time, and Node signifies the name of the nodes showing simulation result and IP address. What signifies the command generated from node or command put in node or processed result. Remarks provides the explanation for What . Let us look at simple examination of cyber attack simulation. First, each of the commands from cyber attack scenario starts from attacker model, moves to destination node with packet via Link, Router, and Gateway model. Destination node again delivers command result to attacker node. In Table 2, Attacker attacks Max node with SYN flooding and the system of Max downs. Attacker disguises its source address with Max s IP address and sends a command to Kant showmount e 203.253.146.169 on packet. Due to Kant s system bug, input command is performed and Kant delivers the result to Attacker s (disguised Max address). By using his disguised address, Attacker performs consequential commands mount 203.253.146.169:/usr/foo , echo prayccc: 1230:10001:1::: >> passwd , echo 192.168.1.20 >> .rhosts and succeeds in acquisition of his own account. Finally, Attacker sends to Kant rlogin 203.253.146.169 command by using his own account successfully.

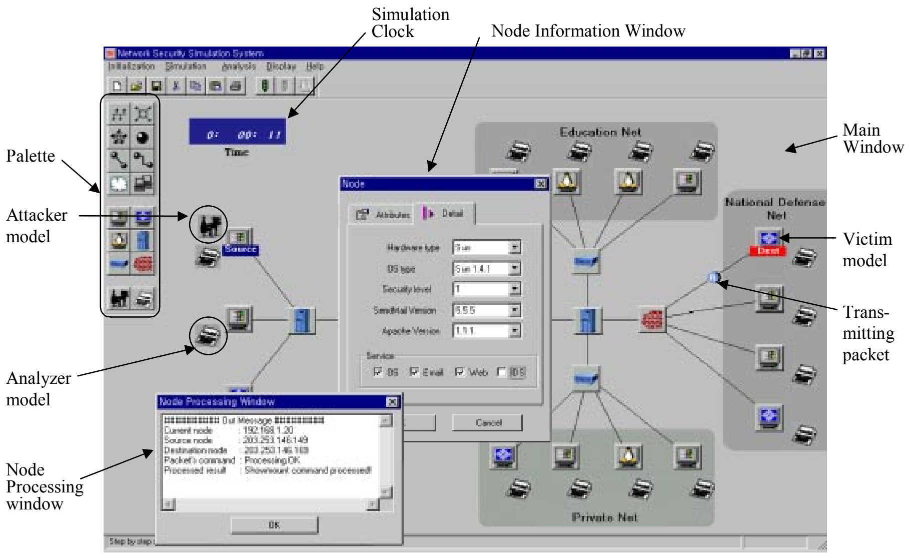
Table 2. Simulation trajectory: Scenario of general user account acquisition

Time	Node	What	Remarks
0 : 0	Attacker (192.168.1.20)	S) 192.168.1.20 SYN flooding 203.253.146.149	SYN flooding attack
0 : 2	Max (203.253.146.149)	SYN flooding 203.253. 146.149	System down
0 : 9	Attacker (192.168.1.20)	S) 203.253.146.149 showmount e 203.253. 146.169	Showmount command
0 : 11	Kant (203.253.146.169)	showmount e 203.253. 146.169 Processing OK!!!	Command processed and reply to 203.253.146.149
0 : 16	Attacker (192.168.1.20)	S) 203.253.146.149 mount 203.253.146.169: /usr/foo	Mount command
0 : 18	Kant (203.253.146.169)	mount 203.253.146.169: /usr/foo Processing OK!!!	Command processed and reply to 203.253.146.149 - Increased Mount Vul-nerability
0 : 23	Attacker (192.168.1.20)	S) 203.253.146.149 cd /foo	Cd command
0 : 25	Kant (203.253.146.169)	cd /foo Processing OK!!!	Change directory to /foo and reply to 203.253.146. 149
0 : 30	Attacker (192.168.1.20)	S) 203.253.146.149 ls alg	Ls command
0 : 32	Kant (203.253.146.169)	ls alg Processing OK!!!	List up ==> -alg Reply to 203.253.146. 149
0 : 37	Attacker (192.168.1.20)	S) 203.253.146.149 echo prayccc:1230:1000 1:1::: >> passwd	Echo command
0 : 39	Kant (203.253.146.169)	echo prayccc:1230:1000 1:1::: >> passwd Processing OK!!!	Increased user number Reply to 203.253.146. 149
0 : 44	Attacker (192.168.1.20)	S) 203.253.146.149 ls alg	Ls command

Table 2. Simulation trajectory: Scenario of general user account acquisition (continued)

Time	Node	What	Remarks
0 : 46	Kant (203.253.146.169)	ls alg Processing OK!!!	List up ==> -alg Reply to 203.253.146. 149
0 : 51	Attacker (192.168.1.20)	S) 203.253.146.149 su prayccc	Su command
0 : 53	Kant (203.253.146.169)	su prayccc Processing OK!!!	Changed user ID to prayccc and reply to 203. 253.146.149
0 : 58	Attacker (192.168.1.20)	S) 203.253.146.149 echo 192.168.1.20 >> .rhosts	Echo command
1 : 00	Kant (203.253.146.169)	Echo 192.168.1.20 >> .rhosts Processing OK!!!	Increased rhost number Reply to 203.253.146. 149
1 : 5	Attacker (192.168.1.20)	S) 192.168.1.20 rlogin 203.253.146.169	Rhost command
1 : 7	Kant (203.253.146.169)	rlogin 203.253.146.169 Processing OK!!!	Command processed and reply to 192.168.1.20
1 : 12	Attacker (192.168.1.20)	S) 192.168.1.20 Bye	Finished
1 : 14	Kant (203.253.146.169)	Attack Succeeded!!!	Attack succeeded and reply to 192.168.1.20

Fig. 8 shows screen copy of SECUSIM system, a network security simulation system under current implementation. SECUSIM is implemented on the basis of MODSIM III [10] and enables a simulation of various attack patterns against various network components. Users can set up initial conditions for simulation by using windows of each node. They can also try to test various cases by attaching attacker and analyzer to any particular node. Procedures of simulation can be checked by the packet-based animation and more detailed procedures can be checked through given windows. The simulation result can be represented in the total number of attacks and successful attacks on each node analyzed in analyzer model, and the vulnerability value.



Hardware type: Sun
OS type: Sun 1.4.1
Security level: 1
SendMail Version: 5.5.5
Apache Version: 1.1.1
Service: ☑ OS ☑ Email ☑ Web ☐ IDS

```
########## Out Message ##########
Current node       : 192.168.1.20
Source node        : 203.253.146.149
Destination node   : 203.253.146.169
Packet's command   : Processing OK
Processed result   : Showmount command processed!
```

Fig. 8. Implemented cyber attack simulation system: SECUSIM

5 Conclusions

This study has discussed so far the network security modeling and cyber attack simulation methodology that can classify threats, specify attack mechanisms, verify protection mechanisms, and evaluate consequences. It has employed the advanced modeling and simulation concepts such as SES/MB framework, DEVS formalism, and experimental frame concept underlying the object-oriented S/W environment. Our approach in this work is different from others in that (i) it supports a hierarchical and modular modeling environment, (ii) it generates the command-level behavior of cyber attack scenario, (iii) it provides an efficient model building environment based on the experimental frame concept, and (iv) it supports the vulnerability analysis of given node on the network. As normal and attack activities are systematically organized, understood, and captured in our model-based approach for the network security system, information protection techniques may be designed more efficiently to cover attacks at various levels and scales of the system for layered, complimentary defense mechanisms. We leave here future further studies for intelligent attacker model, distributed simulation, and vulnerability.

Acknowledgements

The authors thank Hwan-Kuk Kim and Jeong-Rye Jeong for their valuable discussions. This work is supported in part by the Ministry of Information & Communication of Korea (Support Project of University Information Technology Research Center supervised by IITA).

References

1. T. A. Longstaff, Clyde Chittister, Rich Pethia, Yacov Y. Haimes, Are We Forgetting the Risks of Information Technology , *IEEE Computer*, pp 43-51, December, 2000.
2. N. Ye, C. Hosmer, J. Giordano, J. Feldman, Critical Information Infrastructure Protection through Process Modeling and Model-based Information Fusion , *Proceedings of the Information Survivability Workshop*, 1998.
3. F. Cohen Simulating Cyber Attacks, Defenses, and Consequences , *IEEE Symposium on Security and Privacy Special 20th Anniversary Program*, Berkeley, CA, May, 1999.
4. B. P. Zeigler, *Object-oriented Simulation with Hierarchical, Modular Models: Intelligent Agents and Endomorphic systems*, Academic Press, 1990.
5. B.P. Zeigler, *Multifacetted Modeling and Discrete Event Simulation*, Academic Press, 1984.
6. E. Amoroso, *Intrusion Detection,* AT&T Laboratory, Intrusion Net Books, January, 1999.
7. T. A. Wadlow, *The Process of Network Security*, Addison-Wesley, 2000.
8. W. Kaufman, *Running LINUX*, Oreilly, 1999.
9. S. Northcutt, *Network Intrusion Detection An Analyst's Handbook*, New Riders, 1999.
10. *MODSIM Manual*, CACI, 1997.

Cryptographic Salt: A Countermeasure against Denial-of-Service Attacks

DongGook Park [1, 2], JungJoon Kim[1], Colin Boyd [2] and Ed Dawson[2]

[1] Korea Telecom, Access Network Laboratory,
17 WooMyeon-Dong, SeoCho-Gu, 137-792, Seoul, Korea
{dgpark6, jungkim}@kt.co.kr
[2] Queensland University of Technology, Information Security Research Centre,
2 George Street, GPO Box 2434, Brisbane, Queensland 4001, Australia
{c.boyd, e.dawson}@qut.edu.au

Abstract. Denial-of-service (DoS) attack is one of the most malicious Internet-based attacks. Introduction of cryptographic authentication protocols into Internet environment does not help alleviate the impact of denial-of-service attacks, but rather increases the vulnerability to the attack because of the heavy computation associated with cryptographic operation. Nevertheless, many Internet security protocols including SSL/TLS protocol do not consider this aspect. We consider this overlooked issue in authentication protocol design, and propose an effective countermeasure applicable to authentication protocols like SSL/TLS protocol which adopt public-key based encryption to authenticate the server to the client.

1 Introduction

Recently, denial-of-service (DoS) attacks have become a growing concern as Internet services have been used in more aspects of human life. Many things in human life, turned out to have their counterpart in the Internet world: the DoS attack would be one example. In this paper, we focus on the most typical DoS attacks which may be called *connection depletion attacks* or *resource clogging attacks*: an attack in which an attacker seeks to initiate and leave unresolved a large number of connection requests to a Web server, exhausting its resources and rendering it incapable of servicing legitimate connection (or service) requests. SYN flooding attack in TCP/IP networks is the most well known example of this kind [Cert96, Fred99]. This attack exploits a weakness in the TCP connection establishment protocol. Attempting to establish a TCP connection, the client sends the server a SYN message. In response, the server sends a SYN-ACK message, and prepares the connection by allocating buffer space. The client then finishes establishing the connection by responding with an ACK message. After this sequence, both entities can exchange the service-specific data. The attacker, however, does not follow the above sequence of messages. He simply fails on purpose to send the third message, namely ACK to the server, leaving the session half-open. The attacker may initiate large amounts of SYN messages simultaneously, causing the server to be unable to handle the legitimate connection requests. A detailed analysis of this attack and possible remedies are described by Schuba et al. [Schu97].

Using an authentication protocol in Internet environment is orthogonal to prevention of DoS attacks. Authentication protocols themselves do not help prevent denial-of-service attacks but instead may give rise to another environment for denial-of-service attacks. Usually to run an authentication protocol, the involved entity has to assign to it a particular session and some memory to keep relevant data resulting from message exchanges and related computation during the execution of it. Thus, although the notorious SYN flooding attacks can be minimized through careful design and operation of the Internet communication systems, the introduction of authentication protocols just opens up another door to similar denial-of-service attacks.

This problem concerning authentication protocols and DoS attacks is well understood and a lot of previous work is invested to address it; a detailed survey of the related work can be found in [ANL01, LNA00]. The most well studied and promising approach to date seems to be for the server to use *cookies* against a potential attacker. The concept of *cookies* for use in the context of client-server transactions started from Netscape Cookie in 1994 as part of the feature set of Netscape Version 1.1 [Laur98]. Since then, most Web browsers including Microsoft Explorer adopted cookies.

Cookies are pieces of information generated by a Web server and stored in the user s computer, ready for future access [Scho99]. Basically the same concept of cookies started to be used to thwart DoS attacks on cryptographic protocols, the first example of which seems to be Photuris protocol by Karn and Simpson (most recent version 1999 [KaSi99] but originally published 1995). Several Internet security protocols followed this trend, including SKEME [Kraw96], OAKLEY [Orma98]. The basic idea of cookies in these protocols is as follows. When a client attempts to make a connection the server sends back a *cookie* which is a function of a secret known only to the server and other information unique to the particular connection. At this stage the server stores no state for this request. The client needs to return the cookie in the next message and its validity can be checked by the server from the information sent and its secret. The idea is to ensure, before investing significant resources, that the client is making a unique request for connection. This technique addresses the denial of service attacks in which the adversary sends random connection requests.

The benefits of *stateless* connections in the beginning of an authentication protocol were recognized by Janson et al. [JTY97] in the KryptoKnight protocol suite, and this concept was generalized by Aura and Nikander [AuNi97]. Their idea is to make the client store all the state information required by the server and return it to the server as necessary with each message sent. In this way the server need not store any state information. The cookie approach can be considered a special instance of the stateless connection approach in the sense that a cookie generated by the server can contain a session specific information, is stored in the client system, and later delivered back to the server to be verified.

A *cryptographic puzzle* for the client to solve to initiate a connection with the server is another approach to solving DoS attack problems. Dwork and Naor [DwNa98] first presented this concept in the context of electronic junk mailing, and later Juels and Brainard [JuBr99] presented a simpler client puzzle for the server to combat TCP SYN flooding attacks. The same concept was further developed by Aura et al. [ANL01] to address DoS attacks against authentication protocols. In this scenario, the server in an authentication protocol can ask the client to solve a puzzle before the server creates a protocol state or computes expensive public-key related computations. In this way, the puzzle helps improve the DoS-resistance of an authentication protocol.

It can be seen that each of the countermeasures against DoS attacks that we have described carries some cost. If cookies are used as an initial stage in an authentication protocol then additional message exchanges are usually required; this can be a significant overhead in some applications such as the limited signalling channels in mobile communications. Making a protocol stateless may require significant changes to the protocol structure and also increases storage and bandwidth requirements on the client side. Finally the use of cryptographic puzzles imposes a computational burden on both client and server as well as requiring additional message exchanges.

In this paper, we propose a new countermeasure against DoS attacks for client-server security protocols in which *the client authenticates the server by sending a random nonce encrypted under the public encryption key of the server*. Such protocols include SSL/TLS [RFC99], SKEME [Kraw96], and the authentication and key agreement protocol of the PACS (Personal Access Communication System), one of the six PCS standards in North America [Bell94], [JTC94].

Our approach is on the same line of Aura and Nikander s stateless connection concept in that both approaches purport to make the cryptographic protocols themselves more robust against the attacks. Our approach has something in common with the client puzzle concept in that both use some cryptographic mechanisms to combat the DoS attacks, but differs in that our method can apply and be integrated directly into the authentication and key-establishment protocols themselves. This provides a mechanism for the design of more robust protocols. Our scheme requires only a minimal overhead on both the client and the server. The only limitation of the new method is its usage applies only to a specific type of authentication protocols as stated earlier.

Notation

Throughout the paper, symbols *A* and *B* will denote the identities of the client and the server. Symbols like r_X denote a random number or nonce generated by principal *X*. The private and public keys will be written as K_X and K_X^{-1}, respectively. The encryption of some message under key *K* will be denoted by $\{\bullet\}_K$ and the digital signature under key of some message under *X* s private key by $\{\bullet\}_{K_X^{-1}}$. The hash operation of some message will be denoted by $hash(\bullet)$ or $H(\bullet)$.

2 Server Authentication and Random Numbers

To authenticate the server with any cryptographic challenge-response mechanism, the client chooses a random number and sends it to the server. According to the way this random challenge is handled, we may have two different methods of authentication. The first is that the client can send it in the clear and then the server signs over it with its own private key. The corresponding certified public verification key is available publicly and so the client can check whether the signature was generated by and came from the server. The unpredictability and randomness of the random challenge guarantees the required freshness of the signature: i.e., the server has generated the signature for the current session, not for another old session.

The second alternative is to encrypt the random number under the public encryption key of the server before delivery to the server. The authentic server is then the only entity to be able to retrieve the random number from the ciphertext. The server s response to the client with the decrypted random number provides the authenticity of the server s identity.

Each of the above two schemes has its own strengths and weaknesses. As far as denial-of-service attack is concerned, however, the latter method is preferable. This is because in the latter method the random number from the client is not just a random number but an encrypted message thereof, which may be exploited to accommodate a countermeasure against the DoS attack. The basic idea of the countermeasure is to implant a cryptographic salt or a random number chosen by the server in the public-key encryption operation by the client. That is, the client is required to encrypt a random nonce which he received from the server as well as his own fresh nonce. This is quite an unusual usage of random nonce encryption in public-key based authentication protocols. On receipt of the encryption message of random nonces, the server is able to check whether the message has been formed correctly since it leads to the successful retrieval of the server s random nonce after decryption *only when the message has been formed correctly*.

Figure 1 depicts this concept in more detail.

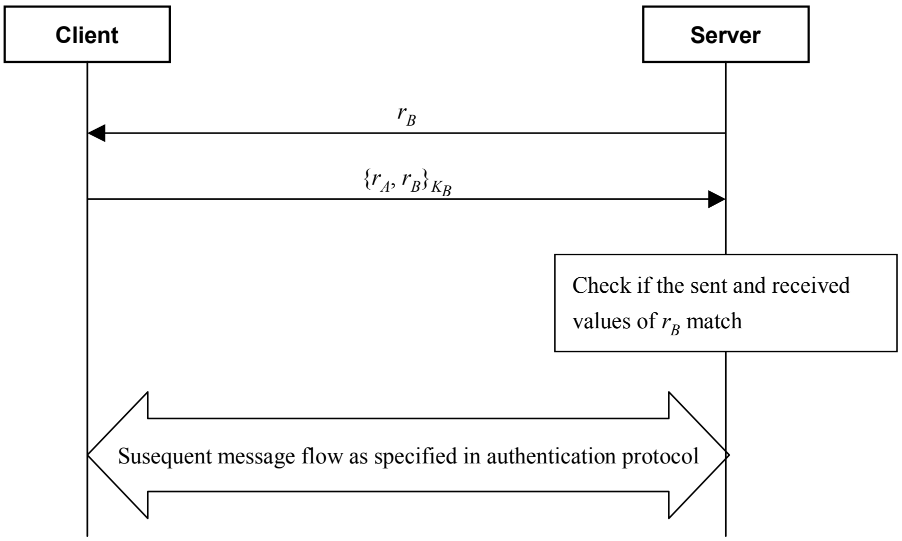

Client
Server
r_B
$\{r_A, r_B\}_{K_B}$
Check if the sent and received values of r_B match
Susequent message flow as specified in authentication protocol

Fig. 1. A random number can be used as a kind of cryptographic salt to combat the DoS attack.

In the above figure, we assume that the client authenticates the server by sending the second message which is encrypted under the server s public encryption key, K_B. Furthermore, it should be noted that the first two messages just comprise a part of an authentication and key establishment protocol, which we want to make more robust against DoS attacks. The steps in this scheme can be outlined as follows.

1. The server B chooses a random number r_B and sends it to the client A.
2. On receipt of r_B, the client chooses its own random number r_A and encrypts it together with r_B using the server s public key K_B; the resulting ciphertext $\{ r_A , r_B \}_{K_B}$ is sent back to the server.
3. On receiving the encryption message, the server decrypts and retrieves r_B and r_A from the received ciphertext. The value of the retrieved r_B and the value of r_B which has been sent to the client should match; otherwise the server concludes that the received message is simply a garbage value sent by a malicious attacker.

Without using this kind of countermeasure, there is no way for the server to check whether the received ciphertext is really the result of a proper cryptographic computation and whether the computation has occurred for the current session. Otherwise even for a garbage or old message attack the server will execute a public key computation for decryption, send the subsequent message to the attacker, and finally will result in a state of the session left open waiting the next message from the attacker, which is simply given up by the attacker.

It can be seen that in a protocol in which the client already sends the challenge r_A encrypted using the server s public key K_B there is only a small change in the protocol messages and minimal additional computational effort required. This is in contrast to other DoS countermeasures which may require additional messages, extra computation, and/or significant alterations to the protocol specification. In practice it is often possible to include the challenge from the server in an existing message, as we will see below.

In the next section, we demonstrate that the above technique can be easily applied to a typical Internet security protocol SSL/TLS where the *ServerHello* message and the *ClientKeyExchange* message correspond to the first and the second messages, respectively.

3 SSL/TLS Protocol

The SSL protocol has become a de facto standard for the Internet security, and its latest version 3.0 is used as the core protocol TLS by the IETF Transport Layer Security working group. The SSL/TLS protocol uses public key cryptography for authentication and key-establishment. Some analyses of cryptographic security of the protocol have been published, such as Paulson s formal inductive analysis [Paul99], and Wagner and Schneier s informal analysis [WaSc96]. Both analyses concluded that the protocol has no weakness with regard to its basic structure. We show below its simplified abstract description which is adopted from Paulson s abstract version of the protocol, where optional messages are boxed in dotted line.

Protocol 1. A simplified description of the TLS handshake protocol

A: Client, *B*: Server

1. $A \rightarrow B$: $A,\ r_A,\ Sid,\ Pa$ `client hello`
2. $A \leftarrow B$: $r_B,\ Sid,\ BCert,\ Pb$ `server hello, server certificate`
3. $A \rightarrow B$: $\boxed{ACert,}\ \{r'_A\}_{K_B},\ \boxed{\{hash(r_B, B, r'_A)\}_{K_A^{-1}},}\ \{finished\}_{K_{AB}^A}$
   `client certificate, client key exchange, certificate verify, client finished`
4. $A \leftarrow B$: $\{finished\}_{K_{AB}^B}$ `server finished`

A, B: $M = hash(r_A, r_B, r'_A)$, $finished = hash(Sid, M, r_A, r_B, Pa, A, Pb, B)$

Here we use slightly different notation from Paulson s description of the TLS protocol; r_A and r_B replaces the original *Na* and *Nb* called client random and server random, respectively. Another random nonce r'_A denotes the pre-master-secret (*PMS*), which serves as a challenge data to the server *B*. The public key certificates of the client and the server are denoted as *ACert* and *BCert*, respectively. *Sid* means the session identifier. The notations $\{\bullet\}_{K_B}$ and $\{\bullet\}_{K_A^{-1}}$ stand for the message encryption under *B* s public encryption key K_B and the signature with the *A* s private signature key. Using r_A, r_B and *M*, the principals *A* and *B* compute the session keys K_{AB}^A and K_{AB}^B to be used for *A*-to-*B* and *B*-to-*A* encryptions, respectively. *Pa* and *Pb* comply with the original notation, which mean the sets of *A* and *B* s preferences for encryption and compression, respectively.

We can see that r_B in the message 2 of the SSL/TLS protocol, and $\{r_A\}_{K_B}$ can serve a good vehicle for the countermeasure described in the previous section. That is, $\{r_A\}_{K_B}$ can be modified to $\{r_A, r_B\}_{K_B}$. The countermeasure is a very reasonable mechanism worthy to be considered for the SSL protocol because there is no additional public-key encryption/decryption required. Furthermore, it should be noted that the concatenation of r_A and r_B is not the only way to implement the idea of the countermeasure. For instance, instead of $\{r_A, r_B\}_{K_B}$, we can adopt, for example, the following alternative:

$$\{r'_A + r_B\}_{K_B},\quad hash(r'_A).$$

In this way, we can keep the length of the encrypted message as the original one. The server decrypts the received ciphertext and subtracts the value of r_B from the decrypted value and takes hash value of it, comparing it with the received value of hash. The benefit of this countermeasure can be made clearer by comparing the significance of DoS attacks for both cases: the original protocol and the modified one, as shown in the table below.

	Original SSL/TLS	Modified SSL/TLS
After a DoS attack the server		
has spent	*one* decryption and *one* or *two* signature verifications,	*one* decryption,
and is left in a state of	*one* half-open session.	*no* half-open session.

Here both decryption and verification are public-key based operations, and the original protocol requires two signature verifications when the client authentication is needed: one for the client certificate verification and another for the client signature verification. The countermeasure cannot prevent DoS attacks completely, but significantly mitigates the damage of the attacks with no additional public key operation or extra message exchange at all.

It is very important to note that the cryptographic salt explained so far is distinct from the idea of cookies. A cookie is a function of session specific information whereas a cryptographic salt is simply a nonce chosen arbitrarily by the server. Both ideas, however, may be combined as shown in the next section.

4 Cookies Combined with the New Countermeasure

The random number r_B in the countermeasure can be generated in a way similar to *cookies* in the Photuris protocol, thus enabling the server to achieve even more robustness against DoS attacks. Usually at the point of the delivery of r_B the server is expected to assign a unique session to the service requesting client. In this situation, a particular value of r_B is also uniquely related to the corresponding session. The value of r_B is stored in a memory within the server system to be compared with the received value of r_B from the client. The problem of this scheme is very similar to that of TCP/IP based client-server model which leads to the notorious SYN flooding attacks. In other words, the server must wait the second message in the above figure after it sends r_B to the client. This problem can be avoided by the server delaying the assignment of a particular session resource to the client until the client proves that he has correctly carried out the encryption of the two random nonces. In other words, the server does not couple a specific value of r_B with a particular client before the client computes and sends the required cryptographic message.

To obviate the need to store the values of r_B, the server prepares a suitable hash function *H*, selects a random master key K_{master} and selects a sufficiently large value as the modulus *M* of the index for r_B. Here, the index runs from 0 to M 1. When a new value of r_B is required, the server runs the hash function with the master key and the current index as the inputs, the hash result of which will be used as the value of r_B (Figure 2).

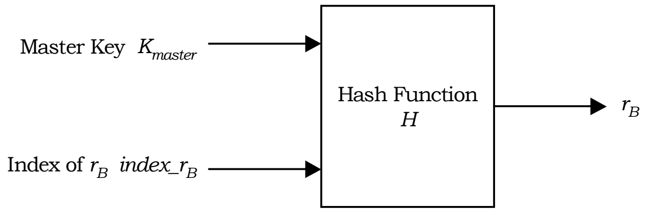


Fig. 2. Generation of random number r_B

The following Figure 3 shows an example using this r_B generation method together with the countermeasure described before. The process is outlined in the following steps.

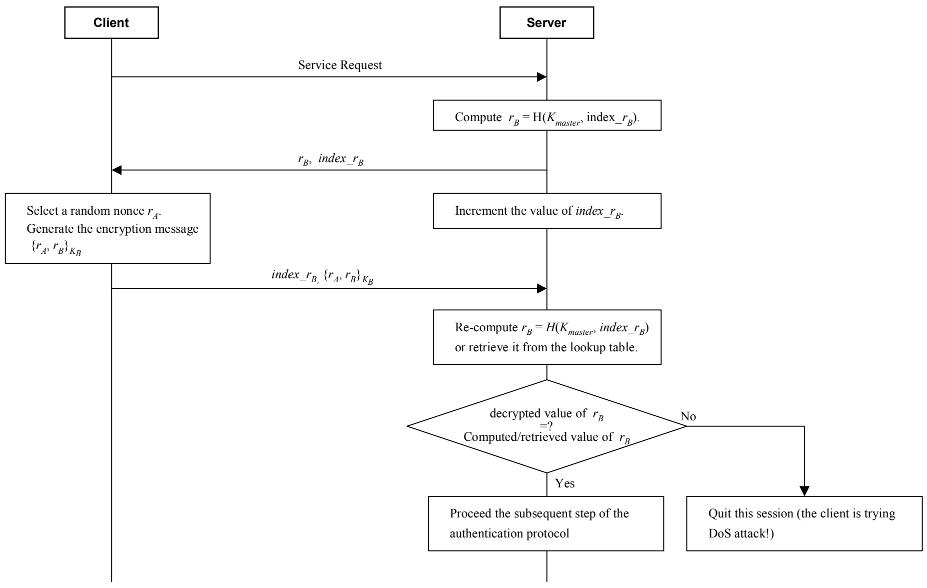

Client
Server
Service Request
Compute $r_B = H(K_{master}, index\_r_B)$.
r_B, $index\_r_B$
Select a random nonce r_A.
Generate the encryption message $\{r_A, r_B\}_{K_B}$
Increment the value of $index\_r_B$.
$index\_r_B$, $\{r_A, r_B\}_{K_B}$
Re-compute $r_B = H(K_{master}, index\_r_B)$ or retrieve it from the lookup table.
decrypted value of r_B =? Computed/retrieved value of r_B
No
Yes
Proceed the subsequent step of the authentication protocol
Quit this session (the client is trying DoS attack!)

Fig. 3. The cryptographic salt r_B as a *cookie*

1. In response to a service request from the client, the server generates a new value of $r_B = H(K_{master}, index\_r_B)$, increments the index parameter $index\_r_B$, and sends the client the values of r_B and $index\_r_B$.
2. On receipt of r_B and $index\_r_B$, the client generates his own random nonce r_A, encrypts r_A and r_B under the public encryption key K_B, and sends the server the plaintext $index\_r_B$ and the ciphertext $\{r_A, r_B\}_{K_B}$.
3. When the server receives the response from the client, using the received value of the parameter $index\_r_B$, it retrieves from a look-up table or, alternatively, re-computes the corresponding value of r_B. The server also decrypts the received ciphertext $\{r_A, r_B\}_{K_B}$, and retrieves the value of r_B, which is compared with the value of r_B which was generated by itself using the given value of $index\_r_B$.
4. If both values match, the server is assured that the client has formed the ciphertext honestly and sent the ciphertext $\{r_A, r_B\}_{K_B}$. This leads the server to the next step specified in the authentication protocol to which the protection scheme is applied.
5. On the other hand, if the match fails, the server may conclude that the client is trying a DoS attack by sending a bogus message which has nothing to do with the correct cryptographic operation to compute the cipher text $\{r_A, r_B\}_{K_B}$.

The usage scenario of r_B as a cookie is just an example, and there may be as many usages as different uses of cookies. It should be noted, however, that the basic idea of the new countermeasure presented in this paper is rather independent from cookies. In other words, the cryptographic salt r_B as described in this paper may or may not be cookies. Rather, the new countermeasure can be more effective when combined with the cookie scheme.

5 Conclusion

We proposed a new concept of protecting a particular form of authentication protocols like the SSL/TLS protocol against the connection depletion attack. The cookie, an existing countermeasure, is useful against the DoS attack, but useless for a determined attacker because the cookie data can be eavesdropped by the attacker. The client puzzle approach solves the problem but requires additional computational overhead in both the client and the server. Our new concept solves all these problems without minimal overhead because it requires no additional public-key operation. In some concrete implementations of the concept, it may require one extra hash computation, which is practically insignificant. Furthermore, this protection method can be combined successfully with the existing cookie mechanism as well, providing more robustness against the DoS attack.

References

[ANL01] T. Aura, P. Nikander, J. Leiwo, DOS-resistant authentication with client puzzles , *Proc. Security Protocols Workshop 2000*, Lecture Notes in Computer Science, Cambridge, UK, April 2000, Springer-Verlag 2001.

[AuNi97] T. Aura and P. Nikander, Stateless connections , *International Conference on Information and Communications Security ICICS'97*, Lecture Notes in Computer Science 1334, Springer-Verlag, 1997, pp. 87-97.

[Bell94] TR-INS-001313, Generic Criteria for Version 0.1 Wireless Access Communications Systems (WACS), Bellcore, Revision 1, June 1994.

[Cert96] CERT, Advisory CA-96.21: TCP SYN Flooding and IP Spoofing Attacks , (Available from http://www.cert.org/advisories/index.html)

[DwNa98] C. Dwork and M. Naor, Pricing via processing or combatting junk mail , In *Advances in Cryptology - Proc. CRYPTO 98*, volume 740 of LNCS, pages 139 147, Santa Barbara, CA USA, August 1992. Springer-Verlag.

[Fred99] Stephen Frede, Attack Scenarios , Security Systems & Technologies, September 1999, pp.4-11.

[JTC94] JTC, Text Modification to JTC(AIR)/94.02.07-119R6, September 15, 1994.

[JuBr99] A. Juels and J. Brainard, Client puzzles: A cryptographic counter-measure against connection depletion attacks , *Proc. 1999 Network and Distributed System Security Symposium (NDSS)*, Internet Society, March 1999, pp. 151-165.

[JTY97] P. Janson, G. Tsudik, and M. Yung, Scalability and flexibility in authentication services: The KryptoKnight approach , *IEEE INFOCOM 97*, Tokyo, April 1997.

[KaSi99] P. Karn and W. A. Simpson. Photuris: Session-key management protocol. RFC 2522, IETF Network Working Group, March 1999.

[Kraw96] H. Krawczyk, SKEME: A Versatile Secure Key Exchange Mechanism for Internet , *Proc. of the Internet Society Symposium on Network and Distributed System Security*, February 1996.

[Laur98] S. St.Laurent, *cookies*, McGraw-Hill, 1998.

[LNA00] J. Leiwo, P. Nikander, T. Aura, Towards network denial of service resistant protocols , *Proc. Sixteenth Annual Working Conference on Information Security (SEC2000)*, IFIP Series, Vol. 175, Beijing, China, August 2000, Kluwer Academic Publishers.

[Orma98] H. Orman. The oakley key determination protocol. *RFC2412*. The Internet Society, November 1998.

[Paul99] L. C. Paulson, Inductive Analysis of the Internet Protocol TLS , ACM Transactions on Computer and System Security **2** 3, 1999, pp.332-351.

[RFC99] T. Dierks and C. Allen, The TLS Protocol, [RFC 22246], January 1999.

[Scho99] V. M.-Schonberger, The Internet and Privacy Legislation: cookies for a Treat? . Available from `http://www.wvjot.wvu.edu/wvjolt/current/issue1/article`

[Schu97] C. L. Schuba et al., Analysis of a denial of service attack on TCP , *Proc. 1997 IEEE Symposium on Security and Privacy*, May 1997, IEEE Computer Society Press. pp. 208 223.

[WaSc96] D. Wagner and B. Schneier, Analysis of the SSL 3.0 Protocol , The Second USENIX Workshop on Electronic Commerce Proceedings, USENIX Press, November 1996, pp.29-40.

Enhanced Modes of Operation for the Encryption in High-Speed Networks and Their Impact on QoS

Oliver Jung, Sven Kuhn, Christoph Ruland, and Kai Wollenweber

University of Siegen, Institute for Data Communications Systems,
Hölderlinstraße 3, 57068 Siegen, Germany
{jung, kuhn, wollenweber}nue.et-inf.uni-siegen.de

Abstract. The internet revolution and modern applications require more bandwidth capacity as a result of the increasing amount of people using e.g. web-based applications with their enhanced quality and performance. Today, modern networks like ATM and SDH/SONET do not only have to fulfill the demand of higher transmission rates but also have to provide and to guarantee data security and especially data confidentiality. Therefore, new or modified cryptographic modes of operation are required. These modes provoke an error propagation which has an impact on the Quality of Service (QoS) parameters of the network. The influences on an ATM network are examined for the CBC, Statistical Counter Mode, a new mode of operation and the ATM Counter Mode, which needs additional bandwidth for synchronization purposes.
For SDH/SONET networks we suggest another mode of operation, called the Statistical Self-Synchronization, combining the advantages of the CFB and OFB mode. In synchronous networks it is the only mode that does not require additional bandwidth and is self-synchronizing with acceptable augmentation of error rates. The impact on the error performance is discussed and guidelines for adjusting selected cryptographic parameters are presented.

1 Introduction

The development of efficient, digital transmission systems is a result of the increased requirement of bandwidth capacity over the last years. The *Synchronous Digital Hierarchy (SDH)*, *Synchronous Optical Network (SONET)* and the *Asynchronous Transfer Mode (ATM)* are the technologies that do not only fulfill this demand, but also offer the possibility of enhanced network management and controllable *Quality of Service (QoS)* for different services.

These networks require adequate security features to protect the processed information and the network management. The ATM Forum has established a framework of specifications that defines objectives for security requirements. The security requirements for ATM networks originate from confidentiality, data integrity and accountability for all ATM network service invocations and management activities. To ensure data confidentiality during the transmission, encryption technology is highly important. Because of the high transmission rates and the consideration of the negotiated QoS-parameters new or modified modes of operation are required.

The ATM Forum recommends the *Cipher Block Chaining (CBC)* and *ATM Counter Mode* for the usage in ATM networks. The ATM Counter Mode has the disadvantage that additional bandwidth for security purposes is necessary and that the occupied bandwidth is no longer available for user data. This results in further cell losses, depending on traffic load and buffer capacities of the security devices. To overcome this disadvantage and new mode of operation, the Statistical Counter Mode is introduced.

In SDH the standardized and fixed frame structure does not allow additional synchronization information of the crypto algorithm. The Statistical Self-Synchronization is the suggested technique which ensures the synchronization even in case of bit-slipping. This mode of operation guarantees that the correct plaintext is computed at the receiver's side after an error propagation has occurred.

The paper is organized as follows: In Sec. 2 the ATM technology is briefly introduced. Section 3 gives a short overview of SDH/SONET. Section 4 concentrates on the modes of operation of block ciphers and two new modes of operation, the Statistical Self-Synchronization and Statistical Counter Mode are presented. Sections 5 and 6 focus on the impact of modes of operation on the error performance in SDH networks and Quality of Service parameters in ATM networks. Finally, the paper is summarized and suggestions for future work are given.

2 ATM

Asynchronous Transfer Mode is based on the definition of the B-ISDN Protocol Reference Model specified in ITU I.321 [5]. It supports integrated voice, data, and video communications for available services as well as for future services not yet defined. ATM has become one of the leading network protocols, because it is highly scalable, fast and efficient, suitable for service integration, and provides Quality of Service.

In ATM the information to be transmitted is divided into short 53 byte fixed-length units called cells, which have a 5 byte header and a 48 byte payload. The reason for such a short cell length is that ATM must deliver real time service at low bit rates and thus it minimizes packetization delay. ATM networks are connection oriented with virtual channels and virtual paths. The virtual channel carries one connection while a virtual path may carry a group of virtual channels. This ensures that cell sequence is maintained throughout the network. The virtual channel is identified by the *Virtual Channel Identifier (VCI)*, and the virtual path is identified by the *Virtual Path Identifier (VPI)*. Both the VCI and VPI are stored in the header of the cell and may change within the network. These values are assigned during call establishment while using a *Switched Virtual Connection (SVC)* or *Permanent Virtual Connection (PVC)*.

The reference-model consists of the ATM Adaption Layer (AAL), the ATM Layer and the Physical Layer:

- **ATM Adaption Layer (AAL)**
  The ATM Adaption Layer adapts the data structure of higher-layer-services to the cell structure of the ATM Layer.
- **ATM Layer**
  The ATM Layer is responsible for the transparent transfer across pre-established connections and is independent of the used service and physical infrastructure.
- **Physical Layer**
  The main task of the Physical Layer is to adapt cells from the ATM Layer to the physical infrastructure.

3 SDH

SDH and SONET are transmission systems with unlimited increasing transmission rates that originally have been developed for the use in Wide Area Networks (WANs). Nowadays they are also used for ATM in Local Area Networks (LANs). The standards for SDH and SONET contain not only the definitions of interfaces, e.g. transmission rates, formats and multiplexing techniques and error performance objectives, but also recommendations for network management. SDH and SONET have similar characteristics. SONET is mainly used in North America and is based on a standard frame, called *Synchronous Transport Signal Level 1 (STS-1)*, with a transmission rate of 51.84 Mbit/s. SDH, which is based on a standard rate of 155.52 Mbit/s is widely spread in Europe. The standard SDH frame is called *Synchronous Transport Module 1 (STM-1)* [6].

SDH has a modular structure, in which the STM-1 builds the basis for all higher transmission rates. Higher transmission rates are gained by bytewise multiplexing $4 \cdot n$ STM-1 frames ($n = 1$,2, etc) to one STM-$4 \cdot n$-frame. In this way the next level of the hierarchy is the STM-4, which offers a capacity of 622 Mbit/s.

In synchronous networks the clocks of the network providers synchronous switching equipment are locked to one common clock. The objective of network synchronization is to minimize the number of byte slips. In case of clock differences between network nodes, up to three bytes can positively or negatively be stuffed into one transmission frame. Therefore, bit- or byte-slipping can only occur if these thresholds are exceeded or in case of a frame buffer overflow or underflow. This error needs to be taken into consideration even if the probability of bit- or byte-slipping is extremely low.

The overhead on the STM-1 frame contains bytes for frame synchronization, signaling of the frame structure, service quality monitoring, path identification, alerts and alert responses. These specific parts of the frame must not be encrypted. Some overhead bytes bypass the encryption and enter the next overhead respectively to be transmitted in plaintext. Section 5 summarizes the error performance objectives which are described in ITU-T G.826 [8] and ITU-T G.829 [10].

4 Encryption

A block cipher encrypts plaintext blocks of a fixed size of n bits, whereby usual values for n are 64 or 128. For messages exceeding n bits, the simplest approach is dividing it into n-bit blocks and encrypt each block separately. This native mode of a block cipher, the *Electronic Codebook Mode (ECB)*, has disadvantages in most applications. Enciphering each block separately results in separate pieces of ciphertext which the adversary can analyze and abuse. A method of enciphering successive blocks is necessary, making the cipher meaningless except in the given sequence.

Two ways of generating such a sequence are common. The first solution is the concatenation of a ciphertext blocks with all preceding blocks due to a chaining operation like in Cipher Block Chaining mode or a feedback operation as in *Cipher Feedback Mode (CFB)* mode. The other one is the generation of a pseudo-random binary sequence that is added modulo-2 with the plaintext binary sequence like it is done in *Output Feedback Mode (OFB).* These ciphers are called binary additive stream ciphers.

The four modes of operation, defined so far in ISO 10116 [2], are quite different in their properties regarding security, synchronization, error propagation, delay and throughput. (Note: ISO 10116 is currently been revised. It will be extended by new modes appropriate for high-speed applications, e.g. the Counter Mode and the Statistical Self Synchronization, at the next release).

4.1 CBC-Mode

A method of using the algorithm in which the ciphertext blocks are concatenated is called the (CBC) mode. Figure 1 demonstrates how the CBC mode is used to encrypt a message. The content of the *Shift Register* at the beginning is called the *Initialization Vector.* The CBC mode requires complete blocks of 64 bits until the final block is encrypted. The length of the feedback register is extended to allow parallel encryption (pipelining) for usage in high-speed networks.

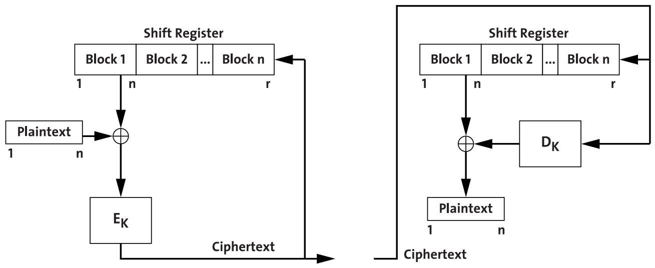


Fig. 1. CBC-Mode

The operation of enciphering each plaintext variable employs the following steps. First the leftmost n-bit of the feedback register have to be selected. The input to the cipher engine is defined to be the XOR of the data and the content of the feedback register. The ciphertext is generated by ciphering the cipher engine input. The bits of the feedback register are shifted left by n places and the ciphertext is inserted in the rightmost n places, to produce the new value of the feedback register. The new n leftmost bits are used as the next input of the encipherment process. These steps are repeated until the final block is encrypted. The final data block of a message or record may contain less than 64 bits when processing in the CBC mode. In this case either the plaintext block must be padded to 64 bits or the terminal block must be enciphered in a way that yields the same number of bits as the input. The steps to encipher the data are repeated in reverse order to decipher the data again.

One or more bit errors within a single cipher block affect the decryption of two blocks (the block in which the error occurs and a succeeding block). If the errors occur, each bit of the corresponding plaintext block has an average error rate of 50%. A succeeding plaintext block has one bit error at the same position. Therefore the CBC mode recovers from bit errors, it also recovers from losses of whole ciphertext blocks, but it does not recover if the block boundaries are lost. The error propagation of the CBC mode in ATM-networks is described in Sec. 6.1.

4.2 The CFB Mode

In CFB mode encryption is achieved by XORing the key stream with the plaintext the output of a key stream generator, where the size of a plaintext character is n bits. The key stream is generated by the block cipher E_K, whereby K is a secret key. The algorithms input data is buffered in an input shift register. The ciphertext is fed back into the input shift register, n bits at a time (Fig. 2).

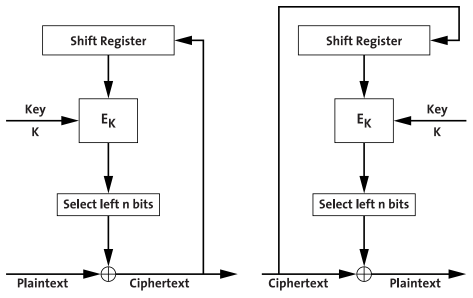


Fig. 2. CFB-Mode

The CFB mode is self-synchronizing. If a synchronization error occurs by erasing or adding a ciphertext unit of n bits, the decrypting side only generates corrupt plaintext as long as defect ciphertext units remain in the input shift register, this requires $\frac{n}{r}$ ciphertext blocks where r is the length of the feedback shift register. The same behavior occurs, if bits have been modified during transmission.

CFB mode is quite inefficient in terms of encryption speed. One block cipher operation is required for enciphering n-bit of plaintext. This applies also, if $n = 1$ is selected to gain a self-synchronization even in the case of bit slipping. The one-bit CFB is not suitable for the encryption of broadband networks. In general, assuming V is the throughput rate of the block cipher implementation, the effective encryption rate, can be calculated by:

$$\nu = V \cdot \frac{n}{64}$$

4.3 The OFB-Mode

The *Output Feedback Mode (OFB)* differs from CFB mode that the output of the encryption block is fed back into the input shift register instead of the ciphertext. Hence, a complete output block of the key stream can be XORed with the plaintext for encryption even if this is achieved only n-bit by n-bit (Fig. 3). The effective encryption rate therefore equals the encryption rate V of the key stream generator. Since the key stream does not depend on the plaintext or ciphertext it may be generated in advance. This type is also called a synchronous stream cipher.

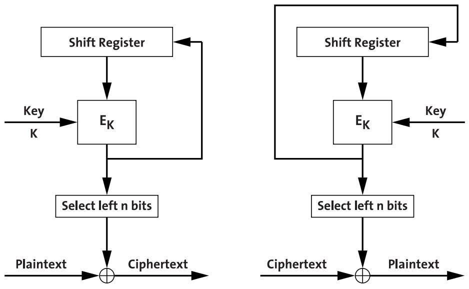


Fig. 3. OFB-Mode

The fact that the transmitted ciphertext is not used for the generation of the key stream means that the cryptographic synchronization is completely lost and cannot be recovered after the occurrence of synchronization errors. The advantage of the OFB mode is that no error propagation occurs if bits have been modified during transmission.

4.4 The Statistical Self-Synchronization

The two stream cipher modes of operation of block ciphers described in section 4.3 and 4.2 show big differences in their properties. The CFB is self-synchronizing, but only offers a low data throughput and error propagation. The OFB in contrast is not self-synchronizing, but has no error propagation and offers a higher encryption rate.

The optimal solution would be the combination of both modes of operation. This is achieved by the *Statistical Self-Synchronization* [12].

The Statistical Self-Synchronization switches from one mode of operation to the other and back, whereby synchronization is reached between encryption and decryption by using the CFB mode. OFB mode is used between the synchronization phases. Loss of synchronization occurs in case of bit- or byte-slipping. In order to re-synchronize both sides need to be switched to CFB mode. The encryption and decryption are kept in CFB mode unless the input shift registers are filled with a complete block of ciphertext. This has to be identical on both sides. The content is used as a new starting value whereby the OFB mode is re-used afterwards (Fig. 4).

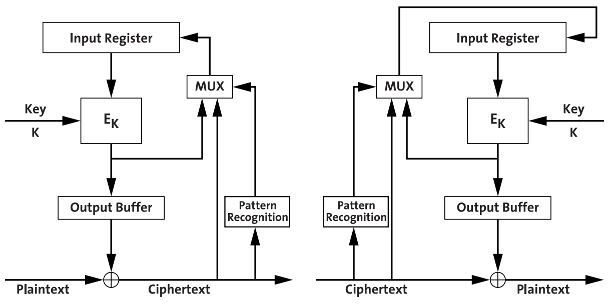


Fig. 4. Statistical Self-Synchronization

The decryption side is not able to recognize a loss of synchronization. Both sides search for a fixed statistically distributed bit pattern in the ciphertext as there is no additional communication capacity between the encryption and decryption entities to signal a switch in modes. Once the pattern is found, both sides switch to CFB mode. The length of the bit pattern defines the probability of the synchronization and needs to be chosen in relation to the probability of bit slipping. The content of the bit pattern can be selected randomly as all bit patterns of a fixed length are equally probable in the ciphertext. A bit-slip causes a loss of synchronization, because the OFB mode is used between the synchronization phases. Encryption and decryption are out of synchronization until the bit pattern occurs again in the ciphertext. Switching to CFB mode is achieved even in the case that no synchronization loss has occurred.

It should be emphasized again, that the bit pattern is generated by the encryption process itself as a result of the encryption of the plaintext. No additional bandwidth is necessary to signal the synchronization start or re-synchronization start, respectively.

A switching to the slower CFB mode implies for the encryption that during the operation in OFB mode, as many key stream blocks as necessary need to be stored in the output buffer to encrypt the plaintext during the next synchronization phase. Therefore, the encryption rate in OFB mode must be higher than the transmission rate.

The bit pattern recognition is switched off to permit another synchronization during the synchronization process.

Assuming that the same key is used for long messages the statistical self-synchronization might be weakend compared to the OFB in case of known-plaintext resp. chosen-plaintext attacks. Changing the keys more often and choosing a block length of more then 64 bit e.g. 128 bit makes the attacks harder to achieve. As encryption works in output feedback mode the generated key stream is cyclic like the data of any other pseudo random generator. The maximum possible cycle length for a 64 bit OFB mode is 2^{64}. Leaving the key unchanged means that the stream generator jumps from one point in the cycle to another every time re-synchronization is performed.

4.5 The ATM Counter Mode

Overview The ATM Counter Mode, specified in the ATM Forum Security Specification [1] is a modified version of the Counter Mode, which will be part of the next revision of the ISO 10116 standard [4],[3]. The difference between the ATM Counter Mode and the previously introduced CBC Mode is the absence of a feedback function. Instead encryptor and decryptor are synchronously producing identical key streams, based on so-called *State Vectors (SV)*. After the determination of the key stream the encryptor XORs the key stream with the plaintext and generates the ciphertext. On the decryption side the XOR of the key stream and the ciphertext recovers the original plaintext (Fig. 5).

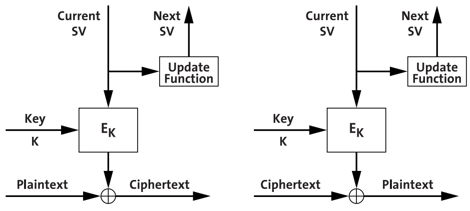


Fig. 5. ATM Counter Mode

Because the Counter Mode allows direct parallelization of the encryption algorithm, it is well suited for high-speed implementations, such as required for modern ATM networks. The advantage arises from the fact that the 48 byte payload can be divided in 6 segments with a length of 64 bit each.

Cryptographic Synchronization Dependent on the used ATM Adaption Layer type, a cell loss or bit error in the ATM network can cause a loss of cryptographic synchronization on the decryption side. The result would be that the encrypted cell can not be correctly decrypted again. To prevent this the decryptor has to use the same SV as the encryptor. To accomplish this the encryptor sends re-synchronization cells (*Session Key Changeover (SKC)*) including the actual State Vector. The SKC cells are sent on a regular basis, dependent on the bandwidth of the used connection and the negotiated Quality of Service parameters. The so-called re-synchronization rate (R) is based on the Cell Loss Ratio (CLR) and the Sustainable Cell Rate (SCR): $R = 10 \cdot CLR \cdot SCR$

State Vector The synchronously produced key stream is based on identical *State Vectors (SV)*. The SV consists of several counters and a Linear Feedback Shift Register to ensure that unique key stream values are generated for each encrypted block (Figure 6):

– *Galois Linear Feedback Shift Register (LFSR)*
  The LFSR is a 21 bit linear non-repeating sequence that is stepped once per cell or per sequence of cells depending on the used AAL type. In case of a re-synchronization the LFSR is preset to its initial value.
– *Initiator/Responder bit*
  The initiator/responder bit is used to prevent the generation of identical key streams for each direction in duplex connections.
– *Sequence number*
  The 4 bit sequence number field is filled with the sequence number extracted from the encrypted/decrypted ATM cell. The length of the sequence number within the ATM cell is dependent on the used AAL type.
– *Segment number*
  The 48 byte payload of the ATM cell is seperated into 64 bit segments for encryption and decryption. The 3 bit segment number defines which segment is being encrypted/decrypted. All other State Vector fields are held constant for the entire cell payload.
– *Jump number*
  The 35 bit jump number is preset to all zeros at call setup and is incremented for each re-synchronization and in case of an AAL-5 connection for each received End-of-Message cell. Because of its length the jump number ensures an always unique SV.

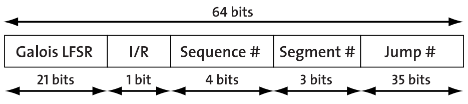


Fig. 6. State Vector (SV)

4.6 Statistical Counter Mode

The ATM Counter Mode has the disadvantage that it allocates bandwidth for synchronizing en- and decryption. This disadvantage can be overcome by using a new mode, the *Statistical Counter Mode*.

On the analogy of the ATM Counter Mode, the Statistical Counter Mode encrypts the plaintext by XOR-ing the enciphered State Vector with the plaintext. Compared to the ATM Counter Mode, a modified SV is used, as the Jump Number is not considered. Instead the LFSR is extended to 56 bits for AAL-1 and AAL-3/4 and to 44 bits for AAL-5. Additionally, for AAL-5 a 16 bit counter for the cells between two *End of Message (EOM)* cells is used (Fig. 7). The purposes of the other fields remain unchanged. The update of the SV is processed in accordance to the update mechanisms of the ATM Counter Mode.

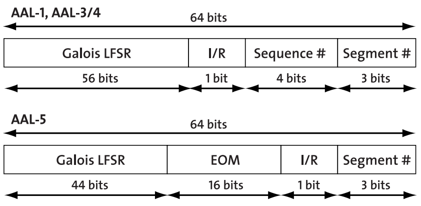


Fig. 7. State Vector of the Statistical Counter Mode

Without allocating bandwidth and without sending re-synchronization cells the synchronization process is maintained by scanning the ciphertext for a predefined *Bit Pattern (BP)* in the 2nd byte of each user cell payload. The cell containing the bit pattern is called *Bit Pattern Cell (BPC)*. Since the first byte of the payload is not random, it cannot be used as it may contain the sequence number. The re-synchronization rate is determined by the length of the bit pattern. If the defined bit pattern occurs in the ciphertext, the following 56 bits are stored and encrypted. They serve as a new value for the LFSR. The value that has been extracted from the ciphertext is random and therefore an already used value can recur. To prevent an attacker from getting knowledge about the used LFSR, the ciphertext is enciphered again before it is used in the new SV, even if the probability of the repeated LFSR content is very low. Furthermore, in contrast to the ATM Counter Mode, the value of the LFSR should be kept secret. Figure 8 shows the Statistical Counter Mode for ATM.

The re-synchronization rate is defined by the length of the bit pattern. The shorter the bit pattern is chosen, the more often re-synchronization is performed. The average re-synchronization rate should be the same as in the ATM Counter Mode. It can be calculated by:

$$R = \frac{1}{2^{BPL}} = 10 \cdot CLR \cdot SCR \tag{1}$$

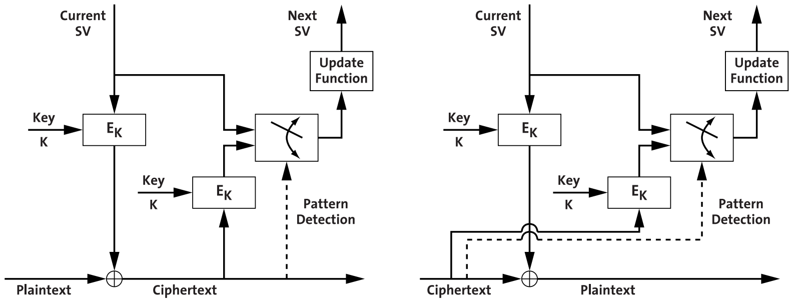


Fig. 8. Statistical Counter Mode

Respectively, the length of the bit pattern is derived by:

$$BPL = \frac{\log(2)}{\log(10 \cdot CLR \cdot SCR)} \tag{2}$$

Since different bit patterns of the same length have the same probability ($P = 1/2^{BPL}$), the distances between the synchronization events are geometrically distributed with mean 2^{BPL}. Thus, the probability of the distances d can be derived by:

$$f(d) = \begin{cases} \frac{1}{2^{BPL}} \cdot (1 - \frac{1}{2^{BPL}})^{d-1} & \text{for } d \,\epsilon\, \{1, 2, ...\} \\ 0 & \text{others} \end{cases} \tag{3}$$

The exact time of re-synchronization cannot be foreseen, as the occurrence of the bit pattern is statistically distributed.

New problems arise due to transmission errors that affect the bit pattern cell. Under the assumption that all bits in an errored cell are unusable, in an errored BPC, the bit pattern itself as well as the new starting value are errored. Therefore, the decryption is not able to detect the bit pattern which finally leads to loss of synchronization. The same problem arises if a cell is errored in that way, that a new bit pattern is created. The decryption would detect a pattern which the encryption did not detect. If the BPC is lost, the decryption produces corrupt plaintext until re-synchronization is performed again.

5 Impact of Security on Error Performance in SDH/SONET

In contrast to ATM, SDH does not offer any possibility to provide additional bandwidth for cryptographic synchronization methodologies. Hence, the mode of operation for an encryption in SDH networks has to be self-synchronizing and to offer an adequate throughput rate. Due to this needs the only appropriate mode of operation is the Statistical Self Synchronization.

Using the Statistical Self-Synchronization provokes a new source of errors. Loosing the cryptographic synchronization means that all bits are errored until the synchronization is re-established. There are three possible reasons for the loss of synchronization:

– A bit- or byte-slip (Event E_1)
– An errored synchronization pattern (Event E_2)
– A new synchronization pattern is generated by an error (Event E_3)

Frame alignment in SDH is found by searching for the framing pattern contained in the SOH of the STM-N signal as specified in ITU-T G.783 [9]. The frame signal is continuously checked with the presumed frame start position for alignment. If no framing pattern is found in the presumed bit position *Out-of-Frame (OOF)* is declared. In the OOF condition the SDH payload may be corrupt. If in the OOF state the maximum frame alignment time shall be 250 μs for an error-free signal with no emulated framing patterns. In case of a bit- or byte-slip the frame alignment is lost as well as the cryptographic synchronization. Frame alignment is gained again 250 μs after the correct framing pattern has been found. The encryption algorithm is not synchronized until the synchronization pattern is found in the ciphertext. For this period of time the decryption generates erroneous plaintext.

An errored synchronization pattern as well as an erroneous generated synchronization pattern have the effect that cryptographic synchronization is lost. In that case only one of the participating parties performs a re-synchronization. The encryption is out of synchronization until the next occurrence of the synchronization pattern.

The probability for both events, the errored synchronization pattern and the erroneous generated synchronization pattern are equal. The length of the synchronization pattern has to be adapted to the BER and the slip rate. The dominating events are the events E_2 and E_3 as slipping occurs rarely. The longer the synchronization pattern is chosen the smaller is the probability for these events and for a disturbed decryption. The occurrence of a longer synchronization pattern has a lower probability which results in a longer period without synchronization.

A demand on the encryption is the transparency regarding the SDH frame structure and management data. Erroneous decryption cannot be detected by the error detection mechanisms of SDH. If e.g. an ATM signal is mapped into the VC-4, errors can only be detected by the ATM network equipment.

Monitoring the BER in terms of G.826, a loss of cryptographic synchronization causes a burst error and thus a SES, but it is not detected by the SDH equipment. This is troublesome for the signal that is carried by the SDH network and recovered from a path terminating node as the data is corrupt. The encryption has to be adjusted for an optimized error performance which is reached by choosing the appropriate length of the synchronization pattern.

6 Impact of Security on QoS in ATM

Providing security in ATM networks is an important aspect to guarantee the confidentiality of the transmitted data. New security devices which offer different encryption algorithms and modes of operation are necessary to fulfill these demands. The CBC and ATM Counter Mode are the recommended modes of operation by the ATM Forum in their Security Specifications. The ATM Forum suggests the DES and FEAL algorithms for the encryption in ATM networks.

All security devices have in common that the performance parameters (Tab. 1) defined in ITU-T I.356 [11] are influenced. This arises because of the transit delays, delay variations and obviously the error propagation of the modes. Hence, the degradation of the QoS parameters depends on the used mode of operation and implementation.

Table 1. ATM Performance Parameters

Acronym	Parameter	Meaning
CER	Cell Error Ratio	ratio of total errored cells to total transferred cells
CLR	Cell Loss Ratio	ratio of total lost cells to total transmitted cells
CMR	Cell Misinsertion Rate	the number of misinserted cells per connection second
CTD	Cell Transfer Delay	the time between the occurrence of two corresponding successful cell transfer events
CDV	Cell Delay Variation	variability in the pattern of cell arrival events at an measuring point
SECBR	Severly Errored Cell Block Ratio	ratio of total severely errored cell blocks to total cell blocks

Especially the time dependent parameters (CTD and CDV) are highly addicted to the hardware or software implementation. Features like *key agility* and *algorithm agility* have a strong impact on the performance because these features lead to additional delay and delay variations, but are out of scope.

If modes of operation that require additional bandwidth (e.g. the ATM Counter Mode) for synchronization purposes are used, it is recommended to assume that the additional bandwidth is available and reserved during the call-establishment. Otherwise it would lead to an increment of the cell loss ratio, because a cell loss may be unavoidable each time a SKC cell is inserted. Evidently, the influence on the QoS parameter not only depend on the mode of operation, but strongly on the implementation and traffic characteristics.

The influence of the used mode of operation on the dependability parameters (CER, CLR, CMR) are described in the following sections. The impact of the implementation and the specific aspects are out of scope.

6.1 Impact of the CBC-Mode

As described in ITU I.432.1 [7], the synchronization mechanism verifies the integrity of every incoming cell and rejects invalid cells. Consequently only valid cells are transmitted to security devices and therefore, the CBC Mode in ATM networks self-synchronizes from bit errors and cell losses.

The CBC Mode does not require any bandwidth for re-synchronization purposes. Thus, this mode does not have any influence on the CLR. Misinserted cells are usually caused by errored routing tables. Naturally, security devices do not have any routing purposes and therefore the CMR does not change.

The modification of the CER depends on the CLR, CMR and CER itself. Supposing the parallel encryption of m blocks, the cell loss of an encrypted cell leads to $A = 1 + \left\lfloor \frac{m-1}{6} \right\rfloor$ errored cells. The same number of cells is errored after each misiserted cell. An errored block in cell i leads to one additional errored cell in cell $i + A$.

6.2 Impact of the ATM Counter Mode

Compared to the CBC-Mode the ATM Counter Mode is a non-self-synchronizing stream cipher which results in the necessity for a re-synchronization protocol. The re-synchronization with SKC cells (Sect. 4.5) implicates a demand for additional bandwidth. Assumed that the additional bandwidth is requested during call setup the CER, CLR and CMR have several effects depending on the used AAL type.

A cell error does not result in a error propagation as long as the cell error does not occur in a SKC cell. In this case all cells up to the next re-synchronization cell cannot be decrypted correctly. In an AAL-1 or AAL-3/4 connection, a corrupted sequence number in the ATM payload would lead to the same result.

In case of an used AAL-1 or AAL-3/4 ATM connection a cell loss has an error propagation if the SKC cell is lost. Again all cells up to the next re-synchronization cell cannot be decrypted correctly. For an AAL-5 connection a cell loss means that all cells up to the next End-of-Message cell (EOM) or SKC cell are errored. In case of a lost SKC or EOM the synchronization is lost up to the next SKC cell.

For all AAL types a misinserted ATM cell results in an errored cell stream up to the next SKC cell. The ATM Counter Mode does only have an impact on the Cell Error Ratio and not on the CLR and CMR. Table 2 shows all error and re-synchronization events.

6.3 Impact of the Statistical Counter Mode

As the Statistical Counter Mode for ATM is based on the ATM Counter Mode, the error expansion is similar. This concerns the impact of cell losses and cell errors of user cells not containing the re-synchronization pattern as well as misinserted cells. Differences only occur if the cell containing the re-synchronization pattern is affected. This may happen if this cell has an errored re-synchronization

Table 2. Re-Synchronization in ATM Counter Mode

	AAL-1 & 3/4	**AAL-5**
Cell Error (SKC)	SKC cell	SKC cell
Errored sequence number	SKC cell	—
Cell Loss	—	SKC or EOM cell
Cell Loss (SKC)	SKC cell	SKC cell
Cell Loss (EOM)	—	SKC cell
Misinsertion	SKC cell	SKC or EOM cell

pattern, errored starting value or if this cell is lost. The same problem arises if a cell is errored in that way that a valid re-synchronization pattern is created. In all cases synchronization is lost until the next BPC is detected and processed. Tab. 3 shows all errors and the events that reestablish synchronization.

Table 3. Re-Synchronization in Statistical Counter Mode for ATM

	AAL-1 & 3/4	**AAL-5**
Cell Error (BPC)	BPC cell	BPC cell
Cell Error (generated bit-pattern)	BPC cell	BPC cell
Errored sequence number	BPC cell	—
Cell Loss	—	BPC or EOM cell
Cell Loss (BPC)	BPC cell	BPC cell
Cell Loss (EOM)	—	BPC cell
Misinsertion	BPC cell	BPC or EOM cell

7 Conclusions and Outlook

In this contribution the mostly used and specified modes of operation for high-speed networks are described and new modes, the Statistical Counter Mode for ATM and the Statistical Self-Synchronization for SDH/SONET networks are presented.

The Statistical Self-Synchronization is the only mode that provides self-synchronization, efficient encryption speed and allows parallelization of multiple encryption modules. However, the Statistical Self-Synchronization shows security weaknesses compared to the OFB mode, because of the re-synchronization process. This problem can be prevented by changing the keys more frequently.

For ATM networks, various modes of operations are applicable. The CBC mode is self-synchronizing but has an error propagation. The ATM Counter Mode has less error propagation but requires bandwidth for synchronization purposes. The new Statistical Counter Mode overcomes this disadvantage using a statistical method. Therefore, for ATM the mode has to be chosen in accordance to the desired application and quality parameters.

An overview of the error propagation and the QoS parameters in ATM and SDH/SONET networks has been given. The impairment of quality parameters are tolerated to gain the benefit of data confidentiality. Further research has to be done in the area of the simulation models and the analysis of the simulation results in comparison to the existing ATM security device SEDAN 155 developed at the Institute for Data Communications Systems [13].

References

[1] ATM Forum. *ATM Security Specification Version 1.1 (Final Ballot)*, October 2000.

[2] ISO/IEC. *10116 – Modes of Operation for an n-bit block cipher algorithm*, 1997.

[3] ISO/IEC. *JTC 1/SC 27 N 2531, Summary of NB contributions to the periodical review of International Standards IS 8372, IS 10116, IS 9798-1, ISs 13888-1/3, and TR 13335-2 (SC 27 N 2489)*, April 2000.

[4] ISO/IEC. *JTC 1/SC 27 N 2711, Text for 1st Working Draft 10116, Information technology - Security techniques - Modes of operation for an n-bit block cipher algorithm (revision)*, December 2000.

[5] ITU-T. *Recommendation I.321 – B-ISDN protocol reference model and its application*, April 1991.

[6] ITU-T. *Recommendation G.707 – Network node interface for the synchronous digital hierarchy (SDH)*, March 1996.

[7] ITU-T. *Recommendation I.432 – B-ISDN user-network interface - Physical layer specification: General characteristics*, August 1996.

[8] ITU-T. *Recommendation G.826 – Error performance parameters and objectives for international, constant bit rate digital paths at or above the primary rate*, February 1999.

[9] ITU-T. *Recommendation G.783 – Characteristics of synchronous digital hierarchy (SDH) equipment functional blocks*, October 2000.

[10] ITU-T. *Recommendation G.829 – Error performance events for SDH multiplex and regenerator sections*, March 2000.

[11] ITU-T. *Recommendation I.356 – B-ISDN ATM Layer Cell Transfer Performance*, February 2000.

[12] Oliver Jung and Christoph Ruland. Encryption with statistical self-synchronization in synchronous broadband networks. In *Chryptographic Hardware and Embedded Systems*, number 1717 in Lecture Notes in Computer Science. Springer, 1999.

[13] Sven Kuhn, Christoph Ruland, and Kai Wollenweber. ATM Encryption with 155 Mbit/s. In *IEEE ATM Workshop '99*, Proceedings. IEEE, 1999.

Improving the Availability of Time-Stamping Services

Arne Ansper[1], Ahto Buldas[1,2⋆], Märt Saarepera[3], and Jan Willemson[1,2⋆]

[1] Cybernetica; Akadeemia 21, Tallinn, Estonia; Lai 36, Tartu, Estonia,
{arne,ahtbu,jan}@cyber.ee
[2] Tartu University Department of Mathematics; Liivi 2, Tartu, Estonia
[3] Tokyo University, RCAST,
marsa@hal.rcast.u-tokyo.ac.jp

Abstract. We discuss the availability questions that arise when digital time stamps are used for preserving the evidentiary value of electronic documents. We analyze the time-stamping protocols known to date and point out some weaknesses that have not been addressed so far in scientific literature. Without addressing and solving them, any advantage of the linkage-based protocols over the hash-and-sign time-stamping would be questionable. We present several new techniques and protocols for improving the availability of both the hash-and-sign and the linkage-based time-stamping services. We introduce fault-tolerant linking as a new concept to neutralize fault-sensitivity as the main weakness of linkage-based time-stamping.

1 Introduction

Time stamp is an attestation that a digital document was created at a certain time. Time stamps are essential tools for relying parties to preserve the evidentiary value of electronic data (particularly, digital signatures). Due to their responsible mission, Time-Stamping Authorities (TSAs) must be reliable – trustworthy, and available when needed. Availability threats may be as harmful as potential attacks by network hackers or dishonest behavior of other parties (repudiation etc.). For example, if TSA's server is destroyed, a large number of time stamps may get unverifiable, and therefore, relying parties may suffer from considerable monetary losses because some important documents (agreements, bills etc.) lost their evidentiary value. This seems unfair from the view-point of an interested party who has no control of the procedures running in a time-stamping server. It would thereby be reasonable if no party could affect the validity of time stamps except the relying party itself.

Regardless of their importance, availability questions have almost never been discussed in scientific literature. This paper is intended to be a contribution to filling this gap. We discuss several techniques for improving the availability of time-stamping services. Particularly, we propose protocols for using multiple

⋆ Supported by the Estonian SF, grant no. 4150

time-stamping servers and argue what kind of benefits such approach may offer for both the hash-and-sign and the linkage-based systems. We also discuss methods for fighting against occasional errors in TSA's database which turn out to be the most serious threats in linkage-based time stamping systems.

In Section 2, we outline the objectives of time-stamping, the general model of time-stamping, and point out the main threats to availability. In Section 3, we analyze the time-stamping systems known to date and point out their advantages and weaknesses. In Section 4, we discuss how multiple servers can be used to improve the availability of service. In Section 5, we introduce a new concept of fault-tolerant linking – a technique against fault-sensitivity of the one-way hash computations used in linking schemes.

2 Time-Stamping: Objectives, Model, and Threats

Let $[t, t']$ be a time-interval and x, y be bit-strings. There are three basic statements that time stamps should prove:

- *Freshness* (of x at t) – x was created after t.
- *Existence* (of y at t') – y was created before t'.
- *Order* (of y and x) – y was created before x.

We call the time stamps intended to prove these statements as: (1) *freshness token*, (2) *existence token* (or *stamp*), and (3) *order token*, respectively. Freshness tokens are needed to avoid replay attacks in authentication protocols. Possibly, there exist no reliable ways of proving that x was created precisely at t. Existence tokens (or stamps) are necessary for proving that a digital signature was created before the corresponding key-identity relation was revoked. In some cases we may need to prove more than one of these statements.

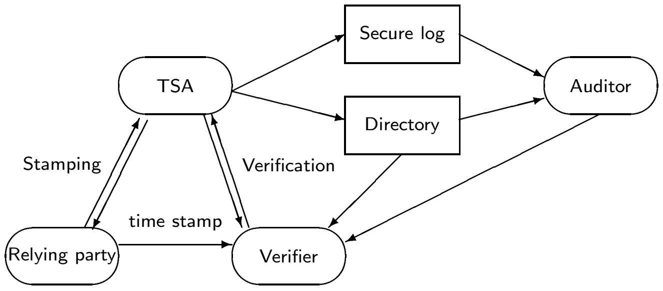


Fig. 1. General model of time-stamping.

Time-stamping is a service used by the *Relying party* to prove temporal relations to the *Verifier* (such as judge and alike). The relying party obtains time stamps from the TSA (and also takes care of them later) by using the

Stamping protocol. The Verifier uses the Verification protocol (which may require communication with the TSA) to check the correctness of time stamps presented by the relying party (Fig. 1). The TSA may also use secure logs and a public directory to enable the Auditor to audit the TSA's work. Audit reports are made available to the verifier and to the relying party. The presence of regular audit gives some additional assurance to time-stamping services.

As referred to in the Introduction, it will be fair if the evidentiary value of time stamps does not depend on third parties (other than the relying party). This is the motivation for the *compactness of evidence* principle: Relying parties possess a compact and time-proof evidence the value of which depends neither on other parties' actions nor on the events which the relying party has no sufficient control of. We discuss three such events:

[A] Broken cryptography and compromised keys. If the cryptographic mechanisms for protecting the authenticity of time stamps are compromised, there should still be a mechanism to distinguish between time stamps (1) issued by the TSA before the compromise, and those (2) created by an attacker, using compromised cryptography. Hence, all the time stamps issued so far can be called into question and cannot further be used as evidence.

[B] Service unavailability. Time-stamping service itself gets unavailable for a while. Relying parties are not able to obtain time stamps for documents they want to preserve as evidence. Such an accident may be causal, for example, in a stock market computer system where time stamps are used to arrange stockbrokers' requests. Unavailability is often caused by the denial of service attacks which are possible if the communication (or security) protocols are poorly designed.

[C] Loss of server's data. A portion of data in a time-stamping server is destroyed and a fraction of time stamps becomes unverifiable. This means that a large number of documents may lose their evidentiary value, and therefore, relying parties may suffer from considerable monetary losses. This type of unavailability may be caused by occasional errors in TSA's server. The most important reason that makes this threat more serious than the previous one is that neither the server nor the relying parties may notice that errors have occurred and the server uselessly continues its work.

In this paper, we analyze which of those threats are encountered in each time-stamping system. We also propose new techniques for overcoming these threats.

3 Time-Stamping Systems: Overview

Preliminaries and notation. By $\mathrm{Sig}_A\{X_1, \ldots, X_m\}$ we mean a digital signature created by A on the ordered list of messages $X_1, \ldots, X_m$. Sometimes, we inherently assume that A uses a signature scheme with message recovery. By a collision-free hash function h, we mean a polynomial-time function family such that it is computationally infeasible to find two arguments $x_1 \neq x_2$ such that $h(x_1) = h(x_2)$. Exact mathematical definition of a hash function is unnecessary for understanding the subject of this paper and is omitted.

Technical assumptions. We only deal with time-stamping systems with one or more central authorities, though there exist protocols without central authorities [1,2]. We also assume that, in order to prevent unreasonable communication, documents are always hashed before they are included into time-stamping requests.

3.1 Absolute (Hash-and-Sign) Time Stamps

Absolute (hash-and sign) time stamps are tokens (signed by the TSA) which comprise a document (or a hash of the document) and a date/time represented as a number. Security of this scheme is based on the assumption that the TSA has a precise enough clock device and is completely trustworthy.

To obtain a freshness token $\mathcal{H}$, a client A (Alice) sends a request to the TSA. The TSA signs the current time t and sends $\mathcal{H} = \mathrm{Sig}_{\mathrm{TSA}}\{t\}$ back to A. For example, given a message $\sigma = \mathrm{Sig}_A\{X, \mathcal{H}\}$, Bob is able to verify that X was signed by Alice after t. Note that even if the TSA is trusted and trustworthy, the freshness token does not prove that σ was created precisely at t.

To obtain a stamp for a bit-string x (for example, $x = \sigma$), a client B (Bob) sends x to the TSA. The TSA adds the current time and date t' to x and sends $\mathcal{T} = \mathrm{Sig}_{\mathrm{TSA}}\{x, t\}$ back to B. The triple $(\mathcal{H}, \sigma, \mathcal{T})$ proves that A signed X during the interval $[t, t']$.

Main concern: key compromise. In systems with a single TSA there seems to be no efficient solution to this problem. The best solution seems to be that the TSA stores all time stamps it ever issues. If then the key is compromised, the TSA signs all time stamps with a new key. This is impractical because of high storage and computational complexity. Tamper-proof hardware may be used to prevent the key compromise. The hardware module may even generate a key-pair for the TSA and never let the private component outside the module. However, this is neither a completely usable solution because the signature scheme itself may be broken and the key length insufficient. In Surety's white paper [8] they even conclude that using keyed cryptography for time-stamping is extremely flawed and only the keyless cryptography can do the job. However, we do not completely agree with this categorical statement. We will show in this paper that the key change is practical in a multi-server case.

3.2 Auditable Relative Time-Stamping

There is a substantive relationship between absolute time stamps and trust. People have unconsciously accepted the concept of time as a number and they hardly realize that actually the relation between physical time and numbers is almost as artificial as the relation between people and their public keys (which are numbers as well!). Thereby, the relationship between time and numbers cannot be fixed reliably without using trusted third parties. However, this does not mean there is no temporal measure which is independent of trusted third parties! Indeed, under some assumptions about computational intractability (one-wayness of hash functions), we have relative temporal measure that uses no trust assumptions.

Let h be a collision resistant one-way hash function and y be a bit-string which was published in a newspaper on February 20, 2001. If we are sent a bit-string x satisfying the relation $y = h(x)$ then we are convinced (because of one-wayness) that x was known to somebody (or stored into a computer) before y was created. Therefore, we also know for sure that x was created before February 20, 2001. More generally, if

$$y = \ell(x, x_1, \ldots, x_n), \tag{1}$$

where ℓ is an arbitrary hash formula (e.g., $h(h(x, x_1), h(x_2, x_3)))$, then $(x_1, \ldots, x_n)$ is a proof that x was created before y. Let $x = \mathrm{Sig}_A\{X\}$ be Alice's signature on X and $\sigma = \mathrm{Sig}_B\{Y, y\}$ be Bob's signature on Y which also comprises a bit-string y, such that equation (1) holds. Then $(x_1, \ldots, x_n)$ is an undeniable proof that Alice signed X before Bob signed Y. Note that the proof itself uses only keyless cryptography and its validity is thereby not affected by the key compromise. This is the main idea of linking first proposed by Haber et al [7].

The TSA maintains a secure log file $(\ell_0, \ell_1, \ldots, \ell_n, \ldots)$ created by using a collision-free hash function h with k-bit output. After each request x_n the TSA computes a new value of ℓ_n using the following recursive formula:

$$\ell_n = h(x_n, \ell_{n-1}) \tag{2}$$

The most important property of the linking scheme is that the value of each log item ℓ_n depends in one-way manner on all the previous items $\ell_0, \ldots, \ell_{n-1}$. If ℓ_n was published in a newspaper on February 20, 2001, then: (1) the previous values cannot be modified without the possibility of detection by an Auditor; and (2) $\ell_0, ..., \ell_n$ may be used as existence tokens for $x_0, \ldots, x_n$, respectively.

To obtain a freshness token, Alice sends a request to the TSA. The TSA sends back the most recent ℓ_m. To obtain a stamp for a bit-string x_n, an interested party B (Bob) sends x_n to the TSA. The TSA computes a new value for ℓ_n using (2). Finally, the TSA sends ℓ_{n-1} back to Bob in order to make him able to compute ℓ_n from x_n, and optionally, uses a short-term signature key to authenticate ℓ_n. So, an existence token is obtained through the following protocol:

$$\boxed{\begin{array}{ll} \text{1. } B \to \text{TSA:} & x_n \\ \text{2 TSA computes:} & \ell_n = h(x_n, \ell_{n-1}) \\ \text{3. TSA} \to B\text{:} & \ell_{n-1},\ [\,\mathrm{Sig}_{\mathrm{TSA}}\{\ell_n\}\,]. \end{array}} \tag{3}$$

Here and further, the square brackets mean that the signature is optional in this protocol. From time to time (say, weekly), the TSA publishes the most recent ℓ_n in the Directory (Fig. 1). After that, the TSA and no-one else is able to modify the chain $\ell_0, \ldots, \ell_{N-1}$ of previous log items. The secure log may also be made widely public. For verifying the order of ℓ_m and ℓ_n ($m < n$), the Verifier obtains a list $\mathcal{T}_{m,n} = (x_{m+1}, x_{m+2}, \ldots, x_n)$ and performs $n - m$ hash-steps (Fig. 2). The list $\mathcal{T}_{m,n}$ is an undeniable proof that ℓ_m was issued before ℓ_n.

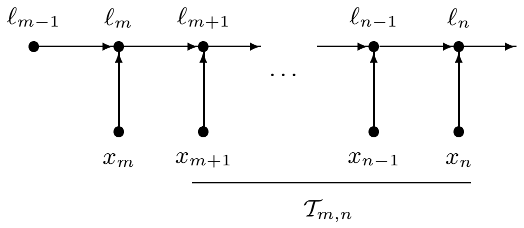


Fig. 2. Linear linking scheme.

Remark: What do signatures add to the linked service? At first sight, the TSA's signature on ℓ_n in Protocol (3) seems redundant since the signature is unnecessary for the comparison of time stamps. However, the TSA's signature is still valuable from the view-point of availability as a method of authenticating the TSA by clients. A malicious party may act as the TSA and thereby, the interested party (Bob) cannot be sure about the validity of the time stamp. Another reason to use signatures is that a malicious TSA may manipulate with the temporal order before it publishes a log item in the Directory. As the value of ℓ_n comprises the whole previous history, the signature of the TSA on ℓ_n can be taken as a temporary commitment. Any modifications in the list $(\ell_0, \ldots, \ell_{n-1})$ can be detected and proved by Bob, even if no ℓ_N ($N \geq n$) has been published yet.

Main drawback: verification cost. Reliable (trust-free) verification of temporal relations may require a large amount of computation and storage, because both are linear functions in $\mid n - m \mid$. The Verifier should download a large amount of data for each verification which may cause huge traffic on the Internet. If a trusted server is used to perform the computation, we have almost the same trust problems as in the absolute time-stamping case. We have very much the same "trust versus communication" problem here as in the public key infrastructure – revocation lists (CRL) require communication, whereas on-line status protocol (OCSP) requires trust. Regular audit may, to some extent, increase reliability of the service. However, an assumption about a trusted Auditor is very much the same as an assumption about a trusted TSA.

The compactness of evidence principle is, thereby, almost unachievable, because it would require a huge amount of storage in the relying party's side. For the service being practical, the Verifier needs communication with the TSA and hence the evidentiary value of time stamps depends on the availability of TSA's service.

3.3 Time Certificates

Time certificates approach, which was first proposed by Pinto and Freitas [10] and independently by Buldas, Laud, Lipmaa and Villemson [4] tries to reduce the communication by saving a part of the linkage data as a meta-part $\tau(X)$ (called *time certificate*) of the time-stamped document X. The purpose of a time certificate $\tau(X)$ is to fix the temporal position of X in a reliable way, so that $(\tau(X), \tau(Y))$ is always a compact piece of evidence for temporal comparison of X and Y (Fig. 3).

We now briefly describe how to use linear linking scheme to create time certificates. Let ℓ_N be the most recent log item which is published, and $x_0, x_1, \dots, x_N$ be the bit-strings time-stamped so far. It is easy to see that we may use a pair $\tau(x_n) = (\ell_{n-1}, \mathcal{T}_{n,N})$ as a time certificate for x_n. Indeed, let $m < n$ and $\tau(x_m) = (\ell_{m-1}, \mathcal{T}_{m,N})$ be another certificate. Then, by the definition,

$$\mathcal{T}_{m,n} = (x_{m+1}, \dots, x_n) \subseteq (x_{m+1}, \dots, x_N) = \mathcal{T}_{m,N}$$

which means that the pair $(\tau(x_m), \tau(x_n))$ indeed comprises a proof that x_m was time-stamped before x_n.

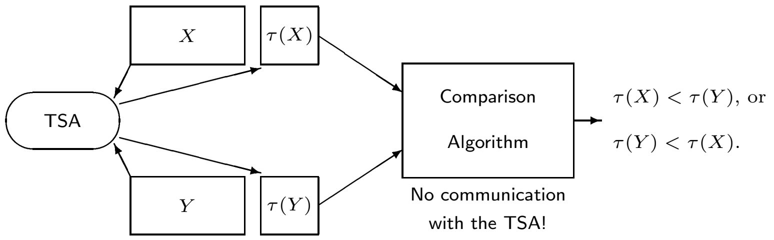


Fig. 3. Off-line comparison of time certificates.

As we mentioned, the linear linking scheme (2) is not practical for using time certificates because of certificates' size (linear in the total number of time stamps). More complex linking schemes presented by Buldas et al [4,3,5,6] make certificates practical because their size in those schemes is logarithmic in the total number of time stamps.

In general, the protocols are almost the same as in the linear scheme, except that after each request x_n the TSA computes two new values: (1) ℓ_n is a bit-string the length of which is equal to the output length k of h; (2) $\mathcal{H}_n$ is a sequence of bit-strings of length k. Computations use the following recursive formulae:

$$\ell_n = \mathcal{L}(x_n, \mathcal{H}_{n-1}), \qquad \mathcal{H}_n = \mathcal{H}(\ell_n, \mathcal{H}_{n-1}), \tag{4}$$

where $\mathcal{H}$ and $\mathcal{L}$ are polynomial-time algorithms consisting of one-way hash computations. Note that by taking $\mathcal{H}_n = \{\ell_n\}$ and $\ell_n = h(x_n, \ell_{n-1})$ we get the linear scheme. The freshness token in the general scheme is $\mathcal{H}_n$. Existence token can be obtained through the following protocol:

$$\boxed{\begin{array}{ll} \text{1. } B \to \text{TSA:} & x_n \\ \text{2 TSA computes:} & \ell_n = \mathcal{L}(x_n, \mathcal{H}_{n-1}), \quad \mathcal{H}_n = \mathcal{H}(\ell_n, \mathcal{H}_{n-1}) \\ \text{3. TSA} \to B\text{:} & \mathcal{H}_{n-1},\ [\,\mathrm{Sig}_{\mathrm{TSA}}\{\ell_n\}\,]. \end{array}} \tag{5}$$

Time certificate for x_n is a pair $\tau(x_n) = (\mathcal{H}_{n-1}, \mathcal{T}_{n,N})$, where $\mathcal{T}_{n,N}$ is a set of hash values, comprising a proof that the value of ℓ_N depends on ℓ_n. Linking algorithms $\mathcal{L}$ and $\mathcal{H}$ can be chosen so that for any $m < n$ the certificates $\tau(x_m) = (\mathcal{H}_{m-1}, \mathcal{T}_{m,N})$ and $\tau(x_n) = (\mathcal{H}_{n-1}, \mathcal{T}_{n,N})$ together are enough to construct a verifiable hash-chain from ℓ_m to ℓ_n. At the same time, the size of a certificate is logarithmic in N. One such linking scheme – *threaded tree* [5] – is described in Appendix A.

One important property of time certificates is that they can be extended, i.e. for any $N_1 > N$ the Relying party (say Bob) may request the TSA for a proof $\mathcal{T}_{N,N_1}$ and then extend the certificate $\tau(x_n)$ from $\tau(x_n) = (\mathcal{H}_{n-1}, \mathcal{T}_{n,N})$ to $(\mathcal{H}_{n-1}, \mathcal{T}_{n,N_1})$. For extension, Bob sends TSA a request which contains two numbers (N, N_1). The TSA answers with $\mathcal{T}_{N,N_1}$ which comprises a one-way link from ℓ_N to ℓ_{N_1}. The answer may also be completed with the TSA's signature $\mathrm{Sig}_{\mathrm{TSA}}\{\ell_{N_1}\}$. Therefore, the extension goes as follows:

$$\boxed{\begin{array}{l} \text{1. } B \to \text{TSA: } (N, N_1) \\ \text{2. TSA} \to B\text{: } \mathcal{T}_{N,N_1},\ [\,\mathrm{Sig}_{\mathrm{TSA}}\{\ell_{N_1}\}\,] \end{array}} \tag{6}$$

We mention without giving a proof that in the threaded tree linking scheme [5] described in Appendix A, we can define an easily-computable composition operation $\circ$, so that

$$\mathcal{T}_{n,N_1} = \mathcal{T}_{n,N} \circ \mathcal{T}_{N,N_1}.$$

The most important thing here is that Bob can extend a time stamp several times, whereas its size will stay logarithmic. This is not the case if we just concatenate $\mathcal{T}_{n,N}$ and $\mathcal{T}_{N,N'}$, because doing so for numerous times, we end up in the linear certificate size.

Therefore, if a set of time stamps are extended to the same published time stamp their temporal order can be determined without the TSA. This is an important property from the view-point of availability. For example, the TSA may require that all the time stamps older than January 1, 2000 were extended to N which was the first time stamp issued on that date. If the relying parties indeed do so, the values $\ell_0, \ldots, \ell_{N-1}$ may be deleted from the TSA's server to save storage space.

Main advantages of the time certificates approach over the previous ones are that (1) Time certificates may always be extended to the most recently published log values. Thereby, users are able to audit the TSA by themselves. (2) Time certificates can be used to verify temporal order of documents off-line, without communicating with the TSA.

Main drawback: increased fault-sensitivity. Computation of log items ℓ_n is (and should be!) extremely fault sensitive. Hence, there is always a concern that errors in the log file will propagate to the future. As a result, the hash-chain splits into two parts, whereas time stamps in different parts would be incomparable. If we compare the formulas of linear linking (2) and general linking (4), we notice that linear linking is less susceptible to error propagation because the value of ℓ_n depends only on ℓ_{n-1}, whereas in the general linking scheme, $\mathcal{H}_{n-1}$ may comprise relatively old ℓ_m, with $m \ll n$. Therefore, if something happens to ℓ_m in the interval between creation of ℓ_m and ℓ_{n-1}, computing ℓ_n in the next step leads us to erroneous log file. So far, almost no attention has been paid in scientific literature to this concern. In Section 5 of this paper we propose a method of fault-tolerant linkage which significantly reduces the danger of fault propagation.

3.4 Usage Example: Time Stamps for Digital Signatures

Time-stamping of digital signature begins with obtaining a *freshness token* from the TSA. The signer A (Alice) sends a request to the TSA and receives the most recent freshness token $\mathcal{H}_m$. In the hash-and-sign scheme, $\mathcal{H}_m = \mathrm{Sig}_{\mathrm{TSA}}\{t\}$, where t is the current time. Alice first concatenates $\mathcal{H}_m$ with the message she wants to sign and then applies the signature algorithm to the concatenation. As a result, she has a signature $\sigma = \mathrm{Sig}_A\{X, \mathcal{H}_m\}$. Suppose B (Bob) is a relying party who received σ and wants that σ would retain its provable authenticity. Usually, this means that B should be able to prove that σ was created *before* the Alice's public key was revoked. To obtain a time stamp, Bob computes a hash $x = h(\sigma)$ and sends x to the TSA as a time stamp request.

A time-stamped signature consumes both the freshness token $\mathcal{H}$ and the stamp $\mathcal{T}$ issued by the TSA. In naive time-stamping systems, the stamp $\mathcal{T}$ is computed as follows:

$$\mathcal{T} = \mathrm{Sig}_{\mathrm{TSA}}\{\mathrm{Sig}_A\{X, \mathcal{H}_t\}, t'\},$$

where $\mathcal{H}_t$ is either $\mathrm{Sig}_{\mathrm{TSA}}\{t\}$, if Alice obtained a freshness token, or t, if Alice did not use the freshness token and added time/date to the message herself. In certain applications this would be sufficient. Naturally, in this case t should be interpreted as the time/date Alice *declares* she signed X at. Note that t (as opposed to $\mathrm{Sig}_{\mathrm{TSA}}\{t\}$) under Alice's signature does not prove that the signature was actually created at t (or after).

In a linkage-based time-stamping system, the TSA sends back the most recent linking item ℓ_n. The pair $(\mathcal{H}_m, L_n)$ is called a preliminary time stamp for σ. For extension of this time stamp, Bob sends TSA the request which contains two numbers (n, N), where $N > n$. The TSA answers with $\mathcal{T}_{n,N}$ (and optionally, with the signature $\mathrm{Sig}_{\mathrm{TSA}}\{L_N\}$). The triple $(\mathcal{H}_m, \mathcal{T}_{n,N}, \mathrm{Sig}_{\mathrm{TSA}}\{L_N\})$ is called a *time certificate* for σ.

One must be careful in time-stamping digital signatures. Signature algorithms behave almost like one-way functions. However, if the key-space is also considered as part of the input space, the signature scheme as a function is not collision-free. Massias et al showed [9] that an attacker may generate a weak RSA key and back-date signatures created with that key. Note that this attack would work in both the hash-and-sign and in the linkage-based time-stamping. To prevent this attack, it is sufficient to time stamp a hash $h(X, \mathrm{Sig}_A\{X\})$ instead of time-stamping a pure signature $\sigma = \mathrm{Sig}_A\{X\}$,.

4 Time-Stamping with Multiple Servers

Using multiple servers is an obvious approach when fighting against service unavailability and loss of data. In this section, we discuss what kind of benefits this approach would give in the case of absolute time-stamping and in linkage-based time-stamping. It turns out that the motivations of using multiple servers are somewhat different in those cases.

As we show in this section, using multiple servers is both (1) a prevention measure and a (2) recovery measure. It helps to keep time stamping services available to clients, and in some cases, to restore the evidentiary value of time stamps if the keys are compromised (in absolute time-stamping) or if TSA's database is lost (in linkage-based time-stamping).

Assumptions. Suppose we have s mutually trusting time-stamping servers TSA_1, ..., TSA_s. We emphasize that it is not assumed that the servers are completely trusted by all clients. To obtain a time stamp, the relying party may interact with all of these servers. Servers' interaction is assumed to be invisible to clients and be protected using a standard authentication protocol (such as SSL). We assume that at every moment, there is at least one server available to clients.

4.1 Absolute Time-Stamping with Multiple Servers

It turns out to be relatively easy to improve the availability of absolute time-stamping just by putting two or three servers together. We need servers' interaction only to synchronize their clocks and in key change scenarios. Using multiple TSAs would enable us to solve the key change problem without storing all the time stamps in servers.

To obtain a freshness token, Alice (A) sends a request to all three servers. The TSA_i signs the current time t_i and sends $\mathcal{H}^i = \mathrm{Sig}_{\mathrm{TSA}_i}\{t_i\}$ back to A:

$$\boxed{\begin{array}{ll} 1.\ \forall i: & A \to \mathrm{TSA}_i\text{: request}_i \\ 2.\ \forall i: & \mathrm{TSA}_i \to A\text{: } \mathcal{H}^i = \mathrm{Sig}_{\mathrm{TSA}_i}\{t_i\} \end{array}} \qquad (7)$$

A s-tuple $\mathcal{H} = (\mathcal{H}^1, \ldots, \mathcal{H}^s)$ is a freshness token. If Alice signs a message X together with freshness token $\mathcal{H}$ then Bob can verify that the signature $\sigma = \mathrm{Sig}_A\{X, \mathcal{H}\}$ was created after $t = \max\{t_1, \ldots, t_s\}$.

To obtain a stamp for a bit-string x (for example, $x = \sigma$), Bob (B) sends x to all s servers. The TSA_i adds the current time t'_i to x, signs the result and sends it back to B:

$$\boxed{\begin{array}{ll} 1.\ \forall i\colon & B \to \mathrm{TSA}_i\colon x \\ 2.\ \forall i\colon & \mathrm{TSA}_i \to B\colon \mathcal{T}^i = \mathrm{Sig}_{\mathrm{TSA}_i}\{x, t_i\} \end{array}} \tag{8}$$

A s-tuple $\mathcal{T} = (\mathcal{T}^1, \ldots, \mathcal{T}^s)$ is the stamp of x. $\mathcal{T}$ proves that σ was created before $t' = \min\{t'_1, \ldots, t'_s\}$. Accordingly, the triple $(\mathcal{H}, \sigma, \mathcal{T})$ proves that A signed X during $[t_i, t'_i]$, assuming that the signature key of TSA_i was valid and the TSA_i itself is trustworthy.

Key change. If the signature key of TSA_1 is compromised, the components $\mathcal{H}_1$ and $\mathcal{T}_1$ signed with this key are no more reliable. The TSA_1 generates a new key and publishes it in a reliable and widely witnessed way. After that, clients may use a *renewal* protocol for replacing the components of $\mathcal{T}$ signed with the old key with components signed with the new key. The renewal protocol runs as follows:

$$\boxed{\begin{array}{ll} 1.\ B \to \mathrm{TSA}_1\colon & \mathrm{Sig}_{\mathrm{TSA}_1,\mathsf{old}}\{x, t_1\}, \mathrm{Sig}_{\mathrm{TSA}_2}\{x, t_2\}, \ldots \mathrm{Sig}_{\mathrm{TSA}_s}\{x, t_s\} \\ 2.\ \mathrm{TSA}_1\colon & \text{verifies signatures } \mathrm{Sig}_{\mathrm{TSA}_2}\{x, t_2\}, \ldots, \mathrm{Sig}_{\mathrm{TSA}_s}\{x, t_s\} \\ & \text{if the signatures are valid and given on the same } x\colon \\ 3.\ \mathrm{TSA}_1 \to B\colon & \mathrm{Sig}_{\mathrm{TSA}_1,\mathsf{new}}\{x, \min\{t_2, \ldots, t_s\}\}. \end{array}} \tag{9}$$

Therefore, TSA_1 is able to renew time stamps without storing all the previously issued time stamps.

Note also that signing old time stamps with a new key is not usable for freshness tokens added to digitally signed documents as signed attributes. To explain this, suppose that we have a signature $\sigma = \mathrm{Sig}_A\{X, \mathcal{H}\}$ with a freshness token $\mathcal{H}$ and the signature key of the TSA is compromised. We cannot just replace the key and sign $\mathcal{H}$ again with a new key because then σ would not be verifiable any more. Signatures are one-way operations and we cannot modify the content of a signed document without violating the signature. Freshness tokens signed with a compromised key are equivalent with plain time stamps created by the signer herself. As we will see later, relative time stamps do not suffer from this concern.

4.2 Linking with Multiple Servers

As linking-based time stamps are keyless, the key compromise is not a motivation of using multiple servers. Moreover, there are many wide-spread techniques for increasing the reliability of storage on the hardware level (e.g. RAID). Thereby, designing special protocols for time-stamping with multiple servers may seem unnecessary. Indeed, we may just use s identical copies of a time-stamping server that uses a linking scheme, such that the clients would even not know that actually there are s servers. However, if the motivation of using multiple servers is to increase the availability of service, there should be multiple processes (running in separate machines) for creating new linking items when the request from a client is received. Just holding several copies is insufficient. One should also guarantee that each request x_n from a client is transmitted to all servers identically. If anything here goes wrong, the servers end up with different linking chains. Considering the nature and purpose of time-stamping services, we cannot completely exclude the possibility of incoherence between the servers because its probability increases as we consider large time-frames. In this paper, we do not assume that each request x_n is certainly received by all servers identically. Therefore, the linking chains (log files) maintained by the servers may be different. The protocol we describe is simple and does not require any additional means of reliable storage. Therefore, we increase the reliability. However, we pay the price in the time certificate size which increases approximately s times. The protocol may be of interest, if the documents time-stamped have a relatively high monetary value. Otherwise, standard tools (RAID etc.) would do their job well.

Assumptions and notation. Each server uses a linking scheme described by general formulae (4). The n-th log item created by the i-th server TSA_i will be denoted as ℓ^i_n. Similarly, we use $\mathcal{H}^i_m$ and $\mathcal{T}^i_{n,N}$ to denote, respectively, the freshness tokens and the relative proofs generated by TSA_i.

Freshness tokens and stamps. To obtain a freshness token, Alice sends each of the servers a request and obtains answers from each of them independently. Each answer consists of s hash formulae. For example, an answer from TSA_1 is

$$(\mathcal{H}^1_{m_{11}}, \mathcal{H}^2_{m_{12}}, \ldots, \mathcal{H}^s_{m_{1s}}),$$

where $\mathcal{H}^j_{m_{1j}}$ is the freshness token issued by TSA_j which TSA_1 knows are the most recent ones. These values are distributed between the servers using protocols which we describe later. If a server TSA^i is down and does not answer, we set $m_{i1} = m_{i2} = \ldots = m_{is} = 0$. The freshness token protocol runs as follows:

$$\boxed{\begin{array}{lll} 1.\ \forall i: & A \rightarrow \mathrm{TSA}_i: & \mathrm{request}_i \\ 2.\ \forall i: & \mathrm{TSA}_i \rightarrow A: & (\mathcal{H}^1_{m_{i1}}, \mathcal{H}^2_{m_{i2}}, \ldots, \mathcal{H}^s_{m_{is}}) \\ 3.\ \forall j: & A \text{ computes: } & m_j = \max\{m_{1j}, m_{2j}, \ldots, m_{sj}\} \\ 4.\ A \text{ computes:} & & \mathcal{H} = (\mathcal{H}^1_{m_1}, \mathcal{H}^2_{m_2}, \ldots, \mathcal{H}^s_{m_s}) \end{array}} \qquad (10)$$

Existence tokens are obtained almost in the same way as in the one-server case. The Relying party sends time stamp requests to all servers and obtains time stamps from all of them except those being inaccessible at the moment.

$$\boxed{\begin{array}{ll} 1.\ \forall i: & B \rightarrow \mathrm{TSA}_i: x \\ 2.\ \forall i: & \mathrm{TSA}_i \rightarrow B: \ell^i_{n_i-1}, [\sigma_i = \mathrm{Sig}_{\mathrm{TSA}_i}\{\ell^i_{n_i}\}] \end{array}} \qquad (11)$$

Time-certificate $\tau(x)$ for x is a s-tuple $((\mathcal{H}^1_{n_1}, [\sigma_1]), (\mathcal{H}^2_{n_2}, [\sigma_2]), \ldots, (\mathcal{H}^s_{n_s}, [\sigma_s]))$ the components of which are ordinary time stamps (already familiar to us from the one-server case).

Extension of a time-certificate is performed on a component-wise basis. Each component is extended using the same protocol as introduced in the one-server case. For extension of the i-th component of a time-certificate, the relying party

(Bob) sends TSA$_i$ the request which contains two numbers (n_i, N_i) with $N_i > n_i$. The TSA$_i$ answers with $\mathcal{T}^i_{n_i,N_i}$ and, optionally, with the signature $\text{Sig}_{\text{TSA}_i}\{\ell^i_{N_i}\}$.

$$\boxed{\begin{array}{l} 1.\ B \to \text{TSA}_i\text{: } (n_i, N_i) \\ 2.\ \text{TSA}_i \to B\text{: } \mathcal{T}_{n_i,N_i},\ [\text{Sig}_{\text{TSA}_i}\{\ell^i_{N_i}\}] \end{array}} \tag{12}$$

Boot. If TSA$_1$ wakes up, it checks whether it has stored values of $\mathcal{H}^1_{n_{11}}$, $\mathcal{H}^2_{n_{12}}$, ... , $\mathcal{H}^s_{n_{1s}}$. These values may be missing either because the server has never been up before or because the linkage data has been destroyed in the last crash. In that case, TSA$_1$ sets $n_{11} = n_{12} = \ldots = n_{1s} = 0$. After that, the TSA$_1$ obtains (using protocol (10)) freshness tokens form other servers and starts linking from the latest value of $\ell_{n_{s1}}$ the other servers know.

$$\boxed{\begin{array}{lll} 1.\ \forall j > 1\text{:} & \text{TSA}_1 \to \text{TSA}_j\text{:} & \text{request} \\ 2.\ \forall j > 1\text{:} & \text{TSA}_j \to \text{TSA}_1\text{:} & (\mathcal{H}^1_{n_{j1}}, \mathcal{H}^2_{n_{j2}}, \ldots, \mathcal{H}^s_{n_{js}}) \\ 3.\ \forall j\text{:} & \multicolumn{2}{l}{\text{TSA}_1 \text{ computes: } n_{1j} := \max\{n_{1j}, n_{2j}, \ldots, n_{sj}\}} \end{array}} \tag{13}$$

Let $n = n_{11}$. If the database was indeed destroyed during the last crash, TSA$_1$ sets $x_{n+1} = h(\mathcal{H}^2_{12}, \ldots, \mathcal{H}^s_{1s})$ and creates ℓ^1_{n+1} and $\mathcal{H}^1_{n+1}$ using formulae (4). If time-certificates older than ℓ^1_n are then completed with the list $(\mathcal{H}^2_{12}, \ldots, \mathcal{H}^s_{1s})$, then we are again able to compare (off-line) time stamps issued by TSA$_1$ with older time stamps issued by other servers before the crash.

New linking item. If the server TSA$_i$ creates a new linking item $L_{n_{ii}+1}$, it computes $\mathcal{H}^i_{n_{ii}+1}$ and sends it immediately to other servers. For example, if $i = 1$ the following protocol is completed:

$$\boxed{\begin{array}{lll} \multicolumn{2}{l}{1.\ \text{TSA}_1 \text{ computes:}} & n_{11} := n_{11} + 1 \\ 2.\ \forall j > 1\text{:} & \text{TSA}_1 \to \text{TSA}_j\text{:} & \mathcal{H}^1_{n_{11}} \\ 3.\ \forall j > 1\text{:} & \text{TSA}_j \text{ computes:} & \mathcal{H}^1_{n_{j1}} := \mathcal{H}^1_{n_{11}} \end{array}} \tag{14}$$

5 Fault Tolerant Linking

As mentioned above, the linking items ℓ_n may get corrupted, either because of occasional errors in hardware or bugs in programs running in the same computer with the TSA software. In linking schemes such errors would propagate to the future and affect the correctness of a large number of time stamps. This threat is even more dangerous than a complete destruction of the TSA's database because the TSA is unaware of the disaster and the service may stay unavailable for a long time. Therefore, some detection measures would be desirable. Error detection codes are the most widespread means against the loss/corruption of data. However, instead of including additional data fields into the linking scheme, we may use the linking scheme itself to detect and correct occasional losses of data. The crucial idea is that a collision-resistant hash function itself is a very efficient error detection code. According to equations (4), we define error detection codes for ℓ_n and for $\mathcal{H}_n$ as follows:

$$\mathsf{Code}(\ell_n) = (x_n, \mathcal{H}_{n-1}), \qquad \mathsf{Code}(\mathcal{H}_n) = (\ell_n, \mathcal{H}_{n-1}). \tag{15}$$

Verifying the code just means verifying the equations (4). Before computing the values of ℓ_n and $\mathcal{H}_n$ by formulae (4), the TSA checks the code $\mathsf{Code}(\mathcal{H}_{n-1})$. If the code is not OK (i.e. if $\mathcal{H}_{n-1} \neq \mathcal{H}(\ell_{n-1}, \mathcal{H}_{n-2})$), then the TSA concludes that the database is corrupted.

Note that the codes will work only if the errors are occasional, i.e. are not caused intentionally by an attacker. For example, if an attacker modifies ℓ_{n-1} and $\mathcal{H}_{n-2}$ and computes a new value for $\mathcal{H}_{n-1}$ using (4), then the error detection procedure we described would not detect any changes because $\mathsf{Code}(\mathcal{H}_{n-1})$ is correct. Therefore if errors are occasional, any error which affects future computations is detected at the very next linkage step. Indeed, equations (4) show that if the value of $\mathcal{H}_{n-1}$ is correct and the request x_n is obtained correctly, then the values ℓ_n, $\mathcal{H}_n$ can also be computed correctly. Any error which does not change $\mathcal{H}_{n-1}$ has no influence on future computations.

Note also that, in principle, the TSA may also try to correct errors by using the codes recursively. This idea needs further research and is not discussed in this paper.

6 Conclusions

This paper indicates that practical problems related to the reliability of digital time-stamping services are far from being completely solved. One of such problems is availability which has not been sufficiently addressed in scientific literature so far. We discussed the availability concerns of both the hash-and-sign and the linkage-based time-stamping systems. We showed how the use of multiple servers eliminates one of the most important threats in hash-and-sign time-stamping – the TSA key compromise. We pointed out a new weakness in linkage-based time-stamping – fault-sensitivity – which arises in its full strength in binary linking schemes [4,3,5]. To overcome the fault-sensitivity concern, we proposed a new approach – fault-tolerant linking – which in our opinion deserves future research.

Acknowledgements

The authors are grateful to Helger Lipmaa and to anonymous referees whose comments helped to improve the final version of this paper.

References

1. Josh Benaloh and Michael de Mare. Efficient broadcast time-stamping. Technical Report 1, Clarkson University Department of Mathematics and Computer Science, August 1991.
2. Josh Benaloh and Michael de Mare. One-way accumulators: A decentralized alternative to digital signatures. In *Advances in Cryptology – Eurocrypt'93*, volume 765 of *LNCS*, pages 274–285. Springer-Verlag, 1994.

3. Ahto Buldas and Peeter Laud. New linking schemes for digital time-stamping. In *Proc. 1st International Conference on Information Security and Cryptology – ICISC'98*, pages 3–13, Seoul, Korea, December 1998.
4. Ahto Buldas, Peeter Laud, Helger Lipmaa, and Jan Villemson. Time-stamping with binary linking schemes. In *Advances in Cryptology – CRYPTO'98*, volume 1462 of *LNCS*, pages 486–501, Santa Barbara, 1998. Springer-Verlag.
5. Ahto Buldas, Helger Lipmaa, and Berry Schoenmakers. Optimally efficient accountable time-stamping. In *Public Key Cryptography – PKC'2000*, volume 1751 of *LNCS*, pages 293–305, Melbourne, Australia, January 2000. Springer-Verlag.
6. Helger Lipmaa. Secure and Efficient Time-Stamping Systems. PhD thesis. *Dissertationes Mathematicae Universitatis Tartuensis.* Tartu University Press, 1999.
7. Stuart Haber and W.Scott Stornetta. How to time-stamp a digital document. *Journal of Cryptology*, 3(2):99–111, 1991.
8. *Secure Time/Date Stamping in a Public Key Infrastructure.* Available at `http://www.surety.com/index-nn.html`
9. Henri Massias, Xavier Serret, and Jean-Jaques Quisquater. Timestamps: Main issues on their use and implementation. In *Proceedings of IEEE 8th International Workshops on enabling Technologies: Infrastructure for Collaborative Enterprises - Fourth International Workshop on Enterprise Security*, pages 178–183, June 1999. ISBN 0-7695-0365-9.
10. Fernando Pinto and Vasco Freitas. Digital time-stamping to support non repudiation in electronic communications. In *Proc. SECURICOM'96 – 14th worldwide Congress on Computer and Communications Security and Protection*, CNIT, pages 397–406, Paris, June 1996.

7 Appendix A: Linking Scheme

For the readers interested in details, we present as an example the linking scheme construction given in [5]. In Fig. 4, we see a fragment of this scheme with only five vertices numbered with 0–4. This fragment is created using the following formulae:

$$\begin{aligned}
&\ell_0 = h(x_0),\\
&\ell_1 = h(x_1, \ell_0) \quad \ell_{01} = h(\ell_0, \ell_1)\\
&\ell_2 = h(x_2, \ell_{01})\\
&\ell_3 = h(x_3, \ell_{01}, \ell_2) \quad \ell_{23} = h(\ell_2, \ell_3) \quad \ell_{0123} = h(\ell_{01}, \ell_{23})\\
&\ell_4 = h(x_4, \ell_{0123}).
\end{aligned}$$

To define this linking scheme in general case, we use binary codes to enumerate linking items. For example, we denote L_2 with `010`, i.e. with binary representation of 2. The non-leaf vertices are numbered by using additional symbol $*$. For example, ℓ_{23} is denoted as `01*` because it represents (is a parent-vertex of) a pair of leaves $\{\texttt{010}, \texttt{011}\}$. For similar reasons, ℓ_{0123} is represented by codeword $\texttt{0}**$ because it represents a pair of vertices $\{\texttt{00}*, \texttt{01}*\}$. Let $b_{k-1} \dots b_0$ be binary representation of n. Then $\mathcal{H}_{n-1}$ is a set of codewords:

$$\mathcal{H}_{n-1} = \{b_{k-1} b_{k-2} \dots b_{i+1} 0 * \dots * \mid 0 \le i < k,\ b_i = 1\}.$$

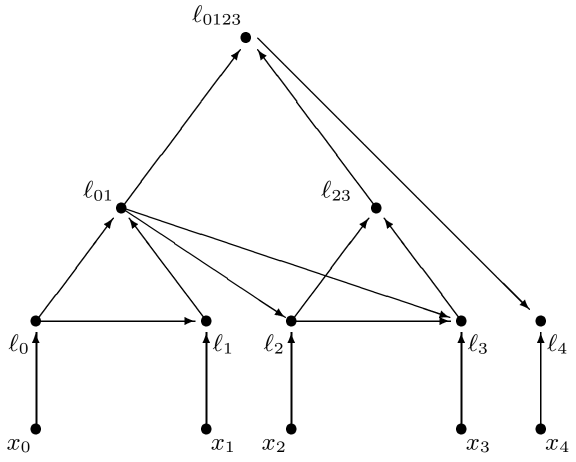


Fig. 4. The threaded-tree linking scheme [5].

For example, in the scheme depicted in Fig. 4, we have $\mathcal{H}_3 = \{\texttt{010}, \texttt{00}*\} = \{\ell_2, \ell_{01}\}$. Let $n' > n$ be another k-bit integer with binary representation $b'_{k-1} \dots b'_0$. Let m be the smallest index such that

$$b_{k-1} = b'_{k-1},\ b_{k-2} = b'_{k-2}, \dots, b_m = b'_m,$$

i.e. $b_{m-1} \neq b'_{m-1}$ and therefore, considering that $n' > n$, we know that $b_{m-1} = 0$ and $b'_{m-1} = 1$. In that case,

$$\mathcal{T}_{n,n'} = \mathcal{H}_{n'-1} \cup \{x_{n'}\} \cup \{b_{k-1} \dots b_m b_{m-1} \dots b_{i+1}\overline{b_i} * \dots * \mid 0 \leq i < m\}.$$

It can be easily proved that having two pairs

$$(\mathcal{H}_{m_1}, \mathcal{T}_{n_1,N}) \text{ and } (\mathcal{H}_{m_2}, \mathcal{T}_{n_2,N})$$

such that $n_1 \leq m_2 \leq N$ then $\mathcal{T}_{n_1,N}$ and $\mathcal{H}_{m_2}$ together are sufficient to compute a one-way link from L_{n_1} to L_{m_2}.

Randomness Required for Linear Threshold Sharing Schemes Defined over Any Finite Abelian Group*

Brian King

University of Wisconsin-Milwaukee, Milwaukee,WI 53201 USA
brking@execpc.com

Abstract. Some secret sharing schemes can be used with only certain algebraic structures (for example fields). Group independent linear threshold sharing (GILTS) refers to a t out of n linear threshold secret sharing scheme that can be used with any finite abelian group. Although group independent secret sharing schemes have long existed, here we formally introduce the definition of group independent linear threshold sharing. Using tools developed by [18], we develop some new necessary conditions for a GILTS. In addition, we develop lower bounds concerning the amount of randomness required within a GILTS.

keywords: *Threshold secret sharing, Linear secret sharing, Randomness requirements.*

1 Introduction

Secret sharing is important in the cases where a secret needs to be distributed over a set of n participants so that only authorized subsets of participants can recover the secret. A setting where the authorized sets consists of all subsets of t or more is called a t out of n threshold secret sharing scheme. Some threshold schemes are constructed for certain algebraic structures, such as fields, groups, semi-groups, etc. Shamir's scheme [31] provides an efficient way to construct t out of n threshold sharing over a field. However, in many cases the setting of the secret space is not a field, for example in RSA [30]. In some cases when the share space is not a field, it is possible to embed the secret within a field and share it using Shamir's secret sharing scheme. The importance of threshold cryptography, especially in the context of threshold signature sharing was noted in [9,15]. Within a threshold signature scheme, the participants are not recovering the secret but a function of the secret (i.e. a signature). In such cases, one cannot embed the secret within a field, and must use the algebraic structure for which the secret space resides. When developing RSA threshold signature schemes an alternative to Shamir's scheme must be used. Some of these alternatives rely on tailoring the scheme to this algebraic setting, some examples of this

* This work was partially funded by NSF grant CCR-9508528

approach include [21,22,23]. Other alternatives introduced the concept of developing threshold schemes which can be used over any finite abelian group, see [13,17]. Another alternative described how threshold sharing can be achieved over any finite group [16], even non abelian groups, this can as well be used with abelian groups. The solutions in [13,17] provided zero-knowledge threshold sharing. Thus these methods achieve zero-knowledge RSA threshold sharing. Further in [24], it has been noted that under certain conditions, the group independent threshold scheme by Desmedt and Frankel [17], can be more efficient than Shoup's threshold signature scheme [32]. Consequently, the importance of threshold sharing over any finite abelian group has been discussed in detail. This paper will provide a formal definition of what we call group independent linear threshold secret sharing (GILTS), and build off the concepts and tools developed in [18]. Moreover, we establish bounds on the amount of randomness required by the dealer to create a t out of n group independent linear threshold scheme (GILTS).

The organization of this paper is as follows: first we provide background and define a GILTS, we discuss the tools developed by Desmedt and Jajodia in [18], we recall some of the work concerning dealer randomness within secret sharing schemes, and highlight those that impact our discussion, lastly we develop our lower bounds concerning randomness R in a GILTS. In all cases, these bounds are lower bounds when working with groups $\mathcal{K}$ of exponent 2. The implication is that if the scheme is a GILTS, and for example, we have determined that R must satisfy a lower bound when $\mathcal{K}$ has exponent 2, then it must satisfy that lower bound if it is a GILTS. Of course, such logic would not be possible in the derivation of upper bounds. A point we wish to make is that once we adopt a group $\mathcal{K}$ with a specific exponent then the R that we are referring to is $R_{\mathcal{K}}$, the randomness needed in a t out of n threshold scheme of that given exponent. In an effort to simplify notation we will refer $R_{\mathcal{K}}$ as R, when there is no ambiguity.

2 Definitions and Notation

$M_{m \times n}(\mathbf{Z})$ represents the set of all m by n matrices with integer entries. If A is a matrix, it's transpose will be denoted by A^T. A row (column) operation of type I is a row (column) interchange. A row (column) operation of type II is a row (column) multiplied by a nonzero constant. A row (column) operation of type III is a row (column) multiplied by a nonzero constant added to another row (column) leaving this result in the second row (column). The rank of a matrix is the number of linearly inde[endent rows within the matrix. If $\boldsymbol{x}_1 \ldots \boldsymbol{x}_n$ are vectors then a linear combination is $\sum_{i=1}^{n} \lambda_i \boldsymbol{x}_i$ where $\lambda_i \in \mathbf{Z}$. $GL_n(\mathbf{Z})$ will denote the group with respect to matrix multiplication of all $n \times n$ nonsingular integer matrices (see [29]). All row vectors will be denoted as $\boldsymbol{x}$. Column vectors will be denoted by $\overline{x}$. The exponent of a group G is the smallest positive integer a such that g^a is the identity for all elements $g \in G$.[1]

[1] If G is an additive group then the exponent is the smallest positive integer a such that $ag = e$ for all $g \in G$.

Definition 1. *[2,31] A perfect t out of n threshold scheme is such that given a secret k, any set of at least t participants can compute k, and any subset of $t-1$ or less participants gain no information about k. If $\overline{s}_1, \ldots, \overline{s}_n$ represent the shares distributed to the n participants, then the security conditions are:*
(i) $Prob(\mathbf{k} = k | \overline{\mathbf{s}}_{i_1} = \overline{s}_{i_1}, \ldots, \overline{\mathbf{s}}_{i_t} = \overline{s}_{i_t}) = 1$
(ii) $Prob(\mathbf{k} = k | \overline{\mathbf{s}}_{i_1} = \overline{s}_{i_1}, \ldots, \overline{\mathbf{s}}_{i_{t-1}} = \overline{s}_{i_{t-1}}) = Prob(\mathbf{k} = k)$
The set Γ of all sets of t or more participants is called the access structure.

Because every set of t or more participants contains a set of precisely t participants, we will use Γ_0 to represent those sets which contain exactly t participants.

Definition 2. *[17,18] A linear secret sharing scheme is one in which for each set $B = \{i_1, \ldots i_t\}$ of t participants, the secret k can be written as $k = \sum_{j=1}^{t} \psi_{B,i_j}(\overline{s}_{i_j})$ where ψ_{B,i_j} is a homomorphism from participant P_{i_j}'s share space $\mathcal{S}_{i_j}(+)$ to the keyspace $\mathcal{K}(+)$.*

This can be algebraically represented as:

$$\begin{bmatrix} \psi_{B_1,1} & \psi_{B_1,2} & \cdots & \psi_{B_1,n} \\ \psi_{B_2,1} & \psi_{B_2,2} & \cdots & \psi_{B_2,n} \\ \vdots & \vdots & & \vdots \\ \psi_{B_{|\Gamma|},1} & \psi_{B_{|\Gamma|},2} & \cdots & \psi_{B_{|\Gamma|},n} \end{bmatrix} \begin{bmatrix} \overline{s}_1 \\ \overline{s}_2 \\ \vdots \\ \overline{s}_n \end{bmatrix} = \begin{bmatrix} k \\ k \\ \vdots \\ k \end{bmatrix}$$

where $\psi_{B_i,j} \cdot \overline{s}_j$ represents $\psi_{B_i,j}(\overline{s}_j)$.

There are many examples of threshold schemes which are linear, including [17,3,27,31]. Further, in schemes [17,3], the key space can be interpreted as a module with left scalars from a commutative ring. In the case of [17], and others, this ring is $\mathbf{Z}$ (the set of integers). Thus the scheme itself can be written in the form of a matrix, with integer entries. The number of rows of this matrix would be at least $\binom{n}{t}$.

2.1 Definition of a t out of n Group Independent Linear Threshold Sharing Scheme

Definition 3. *Let $\mathbf{K} = \{\mathcal{K} | \mathcal{K}$ is a finite abelian group$\}$. A group independent t out of n linear threshold scheme is an ordered pair $(\Psi, \mathcal{S})$ such that:*
(1) For each $\mathcal{K} \in \mathbf{K}$ and for each $i = 1, \ldots, n$ there corresponds a sharespace $S_{i,\mathcal{K}}$. We write $\mathcal{S}_i = \{S_{i,\mathcal{K}} : \mathcal{K} \in \mathbf{K}\}$ and $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_n)$.
(2) For all $B \in \Gamma_0$ and for all i there exists a function $\psi_{B,i}$ such that for all $\mathcal{K} \in \mathbf{K}$, $\psi_{B,i} : S_{i,\mathcal{K}} \longrightarrow \mathcal{K}$ is a homomorphism. Further, for all $k \in \mathcal{K}$, shares $\overline{s}_i$ belonging to $S_{i,\mathcal{K}}$ are distributed to participant P_i such that $\forall B \in \Gamma_0$, $k = \sum_{i \in B} \psi_{B,i}(\overline{s}_i)$,
(3) $Prob(\mathbf{k} = k | \overline{\mathbf{s}}_{i_1} = \overline{s}_{i_1}, \ldots, \overline{\mathbf{s}}_{i_{t-1}} = \overline{s}_{i_{t-1}}) = Prob(\mathbf{k} = k)$, and
(4) $Prob(\mathbf{k} = k | \overline{\mathbf{s}}_{i_1} = \overline{s}_{i_1}, \ldots, \overline{\mathbf{s}}_{i_t} = \overline{s}_{i_t}) = 1$.
We represent $[\psi_{B,i}]_{i=1,\ldots,n; B \in \Gamma_0}$ by Ψ.

Example 1. [20] A group independent 2 out of 3 threshold scheme.
Let k represent an indeterminate for which once a group $\mathcal{K}$ is chosen, will be replaced by the secret. Let r_1 and r_2 represent indeterminates for which once $\mathcal{K}$ is fixed, will be selected randomly from $\mathcal{K}$ to achieve Definition 3 (3) and (4). For all $\mathcal{K} \in \mathbf{K}$, define $S_{1,\mathcal{K}} = \mathcal{K}$ and $S_{i,\mathcal{K}} = \mathcal{K} \times \mathcal{K}$ for i=2, 3. There are 3 sets which belong to Γ_0. Define $\psi_{\{1,2\},1}(x) = x$, $\psi_{\{1,3\},1}(x) = x$, $\psi_{\{1,2\},2}(x_1, x_2) = -x_1$, $\psi_{\{2,3\},2}(x_1, x_2) = x_2$ $\psi_{\{1,3\},3}(x_1, x_2) = -x_1$, and $\psi_{\{2,3\},2}(x_1, x_2) = -x_2$. . For $i \notin B$, $B \in \Gamma_0$, define $\psi_{B,i}(x) = 0$.

Shares will be distributed as follows: P_1 has 1 subshare with $\overline{s}_1 = k + r_1$, P_2 has 2 subshares with $\overline{s}_2 = [r_1, k + r_2]^T$, and P_3 has 2 subshares with $\overline{s}_3 = [r_1, r_2]^T$. Once a group $\mathcal{K}$ is fixed, and the secret k is selected, assuming that the distribution of the secrets is uniform over $\mathcal{K}$, we randomly and independently select r_1 and r_2 from $\mathcal{K}$. We satisfy Definition 3 due to the one-time pad.
Of course there is a simpler representation of Ψ so that $\Psi[\overline{s}_1, \overline{s}_2, \overline{s}_3]^T = [k, k, k]^T$. That is,

$$\left[\begin{array}{c|cc|cc} 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 \end{array}\right] \begin{bmatrix} \overline{s}_1 \\ \overline{s}_2 \\ \overline{s}_3 \end{bmatrix} = \begin{bmatrix} k \\ k \\ k \end{bmatrix} .$$

More examples are provided in the appendix.

2.2 Our Assumption on Group Independent Linear Threshold Sharing

For each set B, of t participants, and for each $i \in B$, we would like $\psi_{B,i}$ to be group independent. That is, the threshold scheme can be used with any finite abelian group such that the reconstruction algorithm is independent of the group. The only method known so far to achieve this is to have $S_{i,\mathcal{K}}$ be the direct product $\mathcal{K}^{a_i}$ (here $S_{i,\mathcal{K}}$ denotes participant P_i's share space and $\mathcal{K}$ is the keyspace, a_i is some positive integer), and to have $\psi_{B,i}$ be a row matrix (with a_i columns) of integers (i.e. P_i possesses subshares which belong to the keyspace). Such a threshold scheme can be described by an integer matrix Ψ such that for each set B there corresponds a row $\boldsymbol{\psi}_B$ of Ψ. That is, the row $\boldsymbol{\psi}_B = [\boldsymbol{\psi}_{B,1}\boldsymbol{\psi}_{B,2} \cdots \boldsymbol{\psi}_{B,n}]$, where for each i, $\boldsymbol{\psi}_{B,i}$ is a row vector of integers of length a_i. (see the above example as a reference). In general, whenever $i \notin B$, then $\boldsymbol{\psi}_{B,i}$ would be a row of zeros.[2]

For each i, $\overline{s}_i$ denotes the share distributed to P_i. This share consists of a_i subshares, the j^{th} subshare of participant P_i will be denoted by $s_{i,j}$ and we will write $s_{i,j} \in \overline{s}_i$. We will assume that all subshares are used in Ψ. As $s_{i,j}$ must be used in Ψ there exists a row $\boldsymbol{\psi}$ in Ψ such that the coefficient in the column for $s_{i,j}$ of row $\boldsymbol{\psi}$ is nonzero.

[2] There is <u>no</u> requirement that the integers in Ψ be chosen from 1, -1, or 0.

2.3 The Basic Model

Let Ψ describe a group independent t out of n threshold scheme. Then Ψ is a matrix belonging to $M_{\mu \times \sum a_i}(\mathbf{Z})$. Shares are distributed to the n participants (which we collectively represent by $\overline{s}$) such that

$$\Psi \overline{s} = \overline{k} \text{ where } \overline{k} = [\alpha_1 k, \alpha_2 k, \ldots, \alpha_\mu k]^T, \tag{1}$$

and α_i is either 0 or 1. When $\alpha_i = 1$, this row describes how the set of participants (those with nonzero entries) can compute k. When $\alpha_i = 0$, this describes a linear dependence between the subshares (those with nonzero entries).[3] We can represent Ψ by $[A_1|A_2|\cdots|A_n]$, where the submatrices A_i denote the blocks which pertain to participant P_i. Since there exists a reconstruction algorithm which is independent of the group, we see that for each $B = \{i_1, \ldots, i_t\}$ there exists a row ψ_B of Ψ such that: there are nonzero entries in those blocks which pertain to participants $P_{i_1}, \ldots, P_{i_t}$, for all other blocks the coefficients are zero, and $\psi_B \overline{s} = k$. Thus this row (call it the j^{th} is such that $\alpha_j = 1$. Therefore $|\{i : \alpha_i = 1\}| \geq \binom{n}{t}$.

3 A Representation of $\overline{s}$

The goal of this section is to provide equations which define $\overline{s}$. Some of the material in this section is due to [18]. The following treatment is reminiscent of van Dijk's [34]. Except that in [34], the setting is a field, whereas we are working with finite abelian groups. The tools that can be used when working with a field are much broader than the tools that we can employ. For example, in a field all square matrices of full rank are nonsingular. Here, we are working with finite abelian groups. One must be careful, for an integer scalar applied to a group element represents repeated computations with the group element. Thus in our treatment, all scalars must be integers. We require that all matrices have integer entries. Further an invertible matrix, should satisfy that its inverse has integer entries. This restricts row/column operations on matrices to those of type I and/or III. It is permitted to perform row (column) operation of type II as long as we restrict ourselves to multiplying the row (column) by the scalar -1. If a group $\mathcal{K}$ is fixed then one will be permitted to multiply rows (columns) by other nonzero scalars. (For example, if the fixed group $\mathcal{K}$ has a prime exponent, then the entries in $\Psi_\mathcal{K}$, Ψ where entries are reduced modulo the exponent, belong to a field. Then one can use any nonzero field element as a scalar and perform row/column operations of type II.) Lastly, there will be occasions when we do use row operations of type II with integer scalars, but in those cases, the inverse of the matrix will not be relevant and will not be used.

[3] The threshold scheme Ψ defines dependencies between shares. To reduce the amount of randomness needed, one may want to introduce additional dependencies in Ψ. Such a dependency is introduced when an $\alpha_i = 0$. To illustrate this we provide two examples in the appendix, see Examples 2 and 3.

Many of our results for a GILTS (group independent linear threshold scheme) are developed using familiar mathematical tools like reduction to Smith-normal form, reduction to Gauss-Jordan form, etc. Often these tools are applied to the general representations of Ψ, at other times these tools are applied to the scheme where a group $\mathcal{K}$ has been adopted.[4] Note that any lower bound required by a group $\mathcal{K}$ implies a lower bound for the GILTS. It is important to realize that the consequence of the implementation of these tools, for example reduction to Smith-normal form, may be different dependent on whether it is applied to the general Ψ or $\Psi_{\mathcal{K}}$, the result of adopting the group $\mathcal{K}$. We always use a subscript $\mathcal{K}$ to indicate that the matrices will be reduced modulo the exponent of $\mathcal{K}$.

3.1 Reduction to Smith-Normal Form

An important tool will be the reduction to Smith-normal form on a matrix (for more information see [25,26,1]). Its use in this contest for secret sharing was first made in [18], as far as we know. In this section, we survey the results given in [18].

Suppose that Ψ is reduced to Smith-normal form. Then there exists $U \in GL(\mu, \mathbf{Z})$ and $V \in GL(\sum a_i, \mathbf{Z})$ such that $U\Psi V = D$ where

$$D = \begin{bmatrix} d_1 & 0 & & 0 & 0 \cdots 0 \\ 0 & d_2 & & & 0 \cdots 0 \\ & & \ddots & & \\ 0 & & & d_l & 0 \cdots 0 \\ 0 & \cdots & & 0 & 0 \cdots 0 \\ \vdots & & & \vdots & \vdots \quad \vdots \\ 0 & \cdots & & 0 & 0 \cdots 0 \end{bmatrix}.$$

U and V are nonsingular matrices which have integer entries, the invariant factors d_i of Ψ are integers and satisfy $d_i | d_{i+1}$ and l is the rank of Ψ. Observe that U can be interpreted as a series of row operations of types I and/or III, and V can be interpreted as a series of column operations of type I and/or III that are performed on Ψ to reduce it to D. Since the ring $\mathbf{Z}$ is a principal ideal domain, the invariant factors of Ψ are unique, up to sign, so we may assume without loss of generality that all invariant factors are positive.

Then $U\Psi VV^{-1}\overline{s} = U\overline{k}$. Hence $DV^{-1}\overline{s} = U\Psi VV^{-1}\overline{s} = U\overline{k}$. Consider the first l rows of the column matrix $U\overline{k}$. Each row can be interpreted as an integer $\sum_j \alpha_j u_{ij}$ applied to k. It follows then that $d_i | (\sum_j \alpha_j u_{ij})$. Since $d_i | \sum_{j=1}^{\mu} \alpha_j u_{ij}$ (for $i = 1, \ldots, l$), we can divide each of the first l rows by the corresponding d_i and still retain the form of an integer matrix. It follows then that we have

$$\begin{bmatrix} I_{l\times l} & 0_{l\times(\sum a_i - l)} \\ 0_{(\mu-l)\times l} & 0_{(\mu-l)\times(\sum a_i - l)} \end{bmatrix} V^{-1}\overline{s} = [k\frac{\sum_{i=1}^{\mu} \alpha_i u_{1i}}{d_1}, \ldots, k\frac{\sum_{i=1}^{\mu} \alpha_i u_{li}}{d_l}, 0, \ldots, 0]^T.$$

[4] In such cases we will use a subscript of $\mathcal{K}$, for example $\Psi_{\mathcal{K}}$ would represent the scheme Ψ when group $\mathcal{K}$ has been adopted.

Set $R = \sum a_i - l$, and let $r_1, \ldots, r_R$ be chosen uniformly at random from denote $\mathcal{K}$. Then

$$V^{-1}\overline{s} = [k\frac{\sum_{i=1}^{\mu} \alpha_i u_{1i}}{d_1}, \ldots, k\frac{\sum_{i=1}^{\mu} \alpha_i u_{li}}{d_l}, r_1, \ldots, r_R]^T.$$

Therefore

$$\overline{s} = V[\frac{\sum_{i=1}^{\mu} \alpha_i u_{1i}}{d_1}k, \ldots \frac{\sum_{i=1}^{\mu} \alpha_i u_{li}}{d_l}k, r_1, \ldots, r_R]^T. \tag{2}$$

Represent V as $V = [X|Y]$, where X is a $\sum a_i \times l$ matrix (which is formed by using the first l columns of V). Then $\overline{s}$ can be represented as

$$\overline{s} = C[k, r_1, \ldots, r_R]^T, \tag{3}$$

where $C = \left[X \cdot [\frac{\sum_{i=1}^{\mu} \alpha_i u_{1i}}{d_1}, \ldots, \frac{\sum_{i=1}^{\mu} \alpha_i u_{li}}{d_l}]^T \,|\, Y\right]$. Consequently the total number of subshares $\sum a_i$ can be expressed as $R + l$. R is the number of random elements required, and l is the rank of Ψ.

3.2 Some Necessary Conditions of a GILTS

Let $\Gamma' = \{A | A \text{ is a set of } t-1 \text{ participants } \}$ and let $B' \in \Gamma'$. $\overline{s}_{B'}$ represent all the (sub)shares used by the participants in B', and $C_{B'}$ represent the corresponding rows of C used to form $\overline{s}_{B'}$. Further let $X_{B'}$ represent the first column of $C_{B'}$ and $Y_{B'}$ the remaining R columns of $C_{B'}$. i.e. $C_{B'} = [X_{B'}|Y_{B'}]$. In [18] the authors established $\overline{s}_{B'} = kX_{B'} + Y_{B'}[r_1, \ldots, r_R]^T$ where $kX_{B'}$ represents a scalar operation of k with $X_{B'}$.

Due to space limitations we omit many of the proofs, and will provide them in the final version of the paper.

Theorem 4. *If a GILTS satisfies Definition 3 then for all $B' \in \Gamma'$, the rank of $C_{B'} \le R$.*

Theorem 5. *If a GILTS satisfies Definition 3 then for all $B' \in \Gamma'$, rank of $C_{B'}$ = rank of $Y_{B'}$.*

Theorem 6. *For each $i = 1, \ldots, n$, either the rank of A_i equals the size of $\overline{s}_i$ or participant P_i can reduce his share size to the rank of A_i. That is, participant P_i can form share $\overline{s''}_i$, which is of size equal to rank of A_i, and P_i can form a new submatrix A''_i of rank equal to the rank of A_i such that $A_i\overline{s}_i = A''_i\overline{s''}_i$.*

For each $i = 1, \ldots, n$, C_{P_i} denotes those rows of C which pertain to participant P_i. That is, $\overline{s}_i = C_{P_i}[k, r_1, \ldots, r_R]^T$.

Theorem 7. *For each i, either the rank of C_{P_i} equals a_i or it is possible to replace $\overline{s}_i$ by a share $\overline{s}'_i$ whose size is equal to rank of C_{P_i}.*

The matrix A_i represents the manner in which participant P_i computes a partial secret using his subshares. The matrix C_{P_i} represents the manner in which the distributor (or dealer) forms the subshares for participant P_i. What we see is that the rank of both matrices can be assumed to be equal, otherwise there exists some dependency either in the way P_i computes partial secrets or in the way the distributor forms the subshares. In this case, it is possible to reduce share size, which removes the dependency. (Hence an agreement in the ranks of these matrices). In [19], it was noted by Desmedt et. al. that within the Desmedt-Frankel scheme[17], there exists dependencies in the matrix which described the manner in which the partial secret were computed. Hence the authors were able to reduce the share size by one-half.

Definition 8. *We say that P_i possesses independent subshares provided the share size is equal to the rank of A_i = rank of C_{P_i}*

Observe that all group independent t out of n threshold schemes can be reformed so that all participants possess independent subshares. Further observe that the participants may reform their subshares (as described in these theorems) and not affect other participants. That is, it is possible for a participant to do this independent of the participation of the others. The notion of independent subshares was defined for the general scheme, once a specific group is adopted, independent subshares may become dependent.

4 Bounds on Randomness

Randomness plays an important role in protecting information, it's role within cryptography has been studied extensively. Randomness represents a computational requirement. A large requirement of randomness represents a burden on computational resources. Further the generation of random elements can be expensive. There has been a considerable amount of investigation in the area of randomness required in secret sharing schemes, see [4,6,7,8,12,14,28]. In the cases of [6,12] it was to develop randomness bounds for secret sharing schemes, that were not necessarily threshold sharing schemes. In [7,8], bounds were developed for multisecret sharing scheme and/or dynamic threshold scheme. The work that most closely impacts our discussion is [4,14] where bounds were developed for the amount of randomness required in ramp schemes and multiparty ramp schemes, because threshold schemes are special cases of ramp schemes.

4.1 Some Background in Randomness Requirement – within Ramp Schemes

Ramp schemes are useful for developing secure multi-party communications in a fault-tolerant model. A (c, t, n) ramp schemes in set S is a protocol to distribute a secret s chosen in set S among a set of n participants $\mathcal{P}$ in such a way that: (1) sets of cardinality greater than or equal to t can compute the secret; (2) sets of cardinality less than or equal to c have no information on s; and (3) sets of cardinality greater than c but less than t may have some information about s. They described the requirements using entropy

1. for all $A \subseteq \mathcal{P}$ with $|A| \geq t$ it holds that $H(S|A) = 0$
2. for all $A \subseteq \mathcal{P}$ with $|A| \leq c$ it holds that $H(S|A) = H(S)$

Note: If $c = t + 1$, then a (c, t, n) ramp scheme is a (t, n) threshold scheme.

Let Σ be a ramp scheme and Π_S be the probability distribution on S. Blundo, DeSantis, and Vaccaro defined the dealer's randomness as

$$\mu_r(c, t, n, \Pi_S, \Sigma) = H(P_1, P_2, \ldots, P_n|S)$$

and in [4] derived the following.

Theorem 9. *[4] Let the number of secrets be* $|S| = 2^\nu > n + t - c$ *for some positive* ν*. The optimal number of random bits to set up a* (c, t, n) *ramp scheme is equal to*

$$\mu_r(c, t, n, U_S, \Sigma) = \frac{c}{t - c} \log_2 |S|.$$

Here U_S denotes the uniform probability distribution on the set of secrets S. Notice that Blundo et. al. are discussing the optimal number of random bits, i.e. a lower bound on the amount of random bits needed. When we discuss R we are discussing the amount of random elements. That is, to compare our work with their bound, we must compare $R \cdot \log_2 |S|$ with $\frac{c}{t-c} \log_2 |S|$, or compare R with $\frac{c}{t-c}$. This comparison only makes sense when talking about t out of n threshold schemes. To make a comparison, we set $t = c + 1$. Thus $R \geq \frac{t-1}{t-(t-1)} = t - 1$. Further Blundo, DeSantis, and Vaccaro 's randomness bound can be interpreted as:

Theorem 10. *[4] The number of random bits used in a* (t, n) *threshold scheme must be* $(t - 1) \log_2 |S|$.

5 Bounds on Randomness in a GILTS

The bound derived by Blundo, DeSantis, and Vaccaro showed that $R \geq t - 1$ show that Shamir's scheme is optimal, in the sense that it requires the minimum amount of randomness generated by the dealer.

Any lower bound on randomness in a threshold sharing scheme over a field is a lower bound of randomness in a GILTS. Thus $R \geq t - 1$ in a GILTS. In the 2 out of 3 GILTS described in Example 1, $R = 2$ which is greater than $t - 1$. In the 2 out of 4 GILTS described in Example 2, $R = 2$ which is also greater than $t-1$. In both of these examples we will find that a minimal amount of randomness will be required by the dealer, and so we see that the Blundo, DeSantis, and Vaccaro randomness lower bound does not effectively state the needed randomness within a GILTS.

In the GILTS described within [17], the amount of randomness R satisfies $R \geq n(t - 1)$. So we see a great difference between the Blundo et. al. bound and the randomness needed by the scheme in [17]. We now derive bounds which better describe the randomness required in a GILTS.

Theorem 11. *If Ψ is a t out of n group independent linear threshold scheme, with independent shares, then for all i the randomness required satisfies $R \geq a_i + (t-2)$.*

Of course every participant must be given a share. Hence $|a_i| \geq 1$. So we see that $R \geq 1 + (t-2) = t-1$ which is the same bound, as Blundo, DeSantis, and Vaccaro had derived. Therefore if any participant is given 2 or more subshares, our bound on the amount of randomness is an improvement on this bound. Note that any bounds concerning participant share size a_i and total share size $\sum a_i$ provide three bounds: one concerning the amount of information that needs to be passed from dealer to participant, two, memory requirements for the participants, and lastly, they provide a lower bound on randomness required. Some examples concerning constructing bounds on share size include [3,5,10,11,27,33].

Recall that we have used Γ' to denote all sets of participants of cardinality $t-1$. For each set $A \subseteq \{P_1, \ldots P_n\}$, let l_A denote the rank of C_A. Further let U_A and V_A to denote the matrices belonging to $GL(\sum_{P_i \in A} a_i, \mathbf{Z})$ and $GL(R+1, \mathbf{Z})$, respectively, which reduces C_A to Smith-normal form. Observe that $U_A C_A$ has been reduced so that the only nonzero rows occur in the first l_A rows. Represent $U_A C_A$ by

$$U_A C_A = [\boldsymbol{\zeta}_{1,A}, \ldots, \boldsymbol{\zeta}_{l_a,A}, 0, \ldots, 0]^T . \tag{4}$$

Define

$$KNOW(A) = \{\sum_{i=1} \lambda_i(\boldsymbol{\zeta}_{i,A} \cdot \omega) \ : \ \lambda_i \in \mathbf{Z}\}. \tag{5}$$

where $\omega = [k, r_1, \ldots, r_R]^T$. $KNOW(A)$ can be thought of as the span of the “information” collectively held by the participants in A.[5]

Whenever a group $\mathcal{K}$ is adopted to be used within the threshold scheme, we will assume that we would perform operations (Smith-normal form reduction, etc.) using the group $\mathcal{K}$. One effect is that entries would be reduced modulo the exponent of $\mathcal{K}$. However there may be other effects, for example the exponent of the group maybe prime and so all entries of the matrices belong to a field. The consequence is that we can use row/column operations of type II, and all square matrices of full rank are units. If $\mathcal{K}$ is the group adopted, then we will use $U_{A,\mathcal{K}}$ and $V_{A,\mathcal{K}}$ which place $C_{A,\mathcal{K}}$ into Smith-normal form. We will also use $KNOW(A, \mathcal{K})$ to denote the knowledge given by the shares of A when working with group $\mathcal{K}$.

Theorem 12. *Let Ψ be a GILTS. For all finite abelian groups $\mathcal{K}$ and $A_1, A_2 \in \Gamma'$ with $A_1 \neq A_2$ $KNOW(A_1, \mathcal{K}) \neq KNOW(A_2, \mathcal{K})$.*

Theorem 13. *For all GILTS the randomness R required satisfies $2^{R(R+1)} \geq 1 + \binom{n}{t-1}$.*

[5] $KNOW(A)$ is a subgroup of the free abelian group generated by $< k, r_1, \ldots, r_R >$

Proof. Let $\mathcal{K}$ be a finite abelian group with exponent 2. For all $B', B'' \in \Gamma'$ with $B' \neq B''$ we have $KNOW(B', \mathcal{K}) \neq KNOW(B'', \mathcal{K})$. Since $KNOW(B', \mathcal{K})$ is formed from $C_{B'\mathcal{K}}$ we see that $U_{B',\mathcal{K}}C_{B',\mathcal{K}} \neq U_{B'',\mathcal{K}}C_{B'',\mathcal{K}}$. Observe that each $U_{B',\mathcal{K}}C_{B',\mathcal{K}}$ is a $\sum_{i\in B'} a_{i,\mathcal{K}} \times (R+1)$ matrix. [6] However the rank of $C_{B',\mathcal{K}}$ is $\leq R$. Therefore

$$U_{B',\mathcal{K}}C_{B',\mathcal{K}} = \begin{bmatrix} & T_{B',\mathcal{K}} & \\ 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{bmatrix},$$

where $T_{B',\mathcal{K}}$ is a $R \times (R+1)$ matrix. Further, for $B', B'' \in \Gamma'$, with $B' \neq B''$, we have $T_{B',\mathcal{K}} \neq T_{B'',\mathcal{K}}$. Since entries in a $T_{B',\mathcal{K}}$ consist of only 0 or 1's, there are exactly $2^{R(R+1)}$ to determine a $R \times R+1$ matrix. As there are $\binom{n}{t-1}$ elements in Γ_0 we conclude $2^{R(R+1)} \geq \binom{n}{t-1}$.

Theorem 14. *For all GILTS the randomness R satisfies*

$$R+1 \geq \sqrt{\log_2(1+\binom{n}{t-1})}.$$

Theorem 14 is an improvement on the Blundo et.al. bound whenever $2^{t^2} \leq 1+\binom{n}{t-1}$. Notice that there are infinitely many t and n which satisfy the above inequality. For example, the above inequality is true whenever $t \leq \sqrt[4]{n}$.

Suppose Ψ is a GILTS, such that every participant had independent subshares. Despite the fact that everyone possesses independent subshares, once a group is adopted, shares may become dependent or shares may become irrelevant (no longer needed to compute the secret). A subshare will be called degenerate if the subshare is not used to compute k. In Theorem 6 and Theorem 7 we described how to generate independent subshares. If a share is degenerate we assume the participant will throw it away. In some instances it is preferred that a set of participants belonging to the access structure can compute the secret key k but also have the ability to compute all other shares. That is, they would together possess all information within the system. We will formally describe the requirements for this model.

Model 5.1 *A t out of n GILTS is called "all-revealing" provided for any finite abelian group $\mathcal{K}$ and for all $k \in \mathcal{K}$*

(1) $Prob(k|\overline{s}_{i_1}, \ldots, \overline{s}_{i_{t-1}}) = Prob(k)$

(2) $Prob(k|\overline{s}_{i_1}, \ldots, \overline{s}_{i_t}) = 1$

(3) $Prob(\overline{s}_j|\overline{s}_{i_1}, \ldots, \overline{s}_{i_t}) = 1$ *for all j, where $\overline{s}_j$ represents the nondegenerate subshares belonging to P_j used in the threshold scheme when working with group $\mathcal{K}$.*

[6] $a_{i,\mathcal{K}}$ represents the number of subshares participant uses within the threshold scheme $\Psi_{\mathcal{K}}$. Thus $a_{i,\mathcal{K}} \leq a_i$. Further $a_{i,\mathcal{K}}$ is the rank of $A_{i,\mathcal{K}}$.

Observe that when the group $\mathcal{K}$ has exponent 2, then all the invariant factors in the reduced $C_{\mathcal{K}}$ are 1. We see that in the all-revealing model, all sets belonging to Γ cannot only determine all shares but compute all random elements. That is, they can determine r_i for $i = 1, \ldots, R$. The following illustrates that all-revealing schemes naturally exist.

Theorem 15. *Every $n-1$ out of n GILTS is all-revealing.*

As we will soon see, the model for all-revealing schemes provides the tools to improve the result given by Theorem 14.

5.1 Some Observations Concerning All-Revealing Schemes when $\mathcal{K}$ Has Exponent 2

Consider the secret sharing scheme $\Psi_{\mathcal{K}}$ where we adopt a group $\mathcal{K}$ with exponent 2. For each $B' \in \Gamma'$, let $L_{B'}$ denote the matrix belonging to $GL(\sum_{P_i \in B'} a_i, \mathbf{Z}_2)$ such that $L_{B'}C_{B',\mathcal{K}}$ is in Gauss Jordan form. Recall that $C_{B',\mathcal{K}}$ has rank $\leq R$. The number of nonzero rows is $\leq R$. Observe that for different $B' \in \Gamma'$ the number of rows in $L_{B'}C_{B',\mathcal{K}}$ may differ. To make effective comparison between $L_{B'}C_{B',\mathcal{K}}$ and $L_{B''}C_{B'',\mathcal{K}}$ for $B', B'' \in \Gamma'$ we use precisely $R+1$ rows of $L_{B'}C_{B',\mathcal{K}}$, if $L_{B'}C_{B',\mathcal{K}}$ lacks that many rows insert enough zero rows. If $L_{B'}C_{B',\mathcal{K}}$ has more than $R+1$ rows delete enough zero rows to achieve a matrix with $R+1$ rows. Although we may have formed a new matrix, the nonzero rows are the same, we will use $\Upsilon_1(L_{B'}C_{B',\mathcal{K}})$ to denote this new matrix. Second, note that $\Upsilon_1(L_{B'}C_{B',\mathcal{K}})$ is still in Gauss Jordan form. To achieve an even more consistent representation between all the $\Upsilon_1(L_{B'}C_{B',\mathcal{K}})$ for different B', we perform one more manipulation. For each nonzero row $\boldsymbol{z}$ in $\Upsilon_1(L_{B'}C_{B',\mathcal{K}})$ if the leading nonzero term in $\boldsymbol{z}$ occurs in column i interchange rows so that $\boldsymbol{z}$ is rotated to row i. We use $\Upsilon_2(L_{B'}C_{B',\mathcal{K}})$ to represent this matrix.

For each $B' \in \Gamma'$, let us represent each row i of $\Upsilon_2(L_{B'}C_{B',\mathcal{K}})$ by $\boldsymbol{\beta}_{i-1,B'}$ for $i = 1, \ldots, R+1$. Then $\Upsilon_2(L_{B'}C_{B',\mathcal{K}}) = [\boldsymbol{\beta}_{0,B'}, \ldots, \boldsymbol{\beta}_{R,B'}]^T$ where each $\boldsymbol{\beta}_{i,B'}$ is a $(R+1)$-tuple. Therefore an equivalent representation of $KNOW(B', \mathcal{K})$ is :

$$KNOW(B', \mathcal{K}) = \{\sum_{i=0}^{R} \theta_i(\boldsymbol{\beta}_{i,B'} \cdot \omega) \; : \; \theta_i \in \mathbf{Z}_2\}. \tag{6}$$

That is, it is trivial to show that Definition (6) is equivalent to Definition (4). Secondly, observe that $(\sum_{i=0}^{R} \theta_i \boldsymbol{\beta}_{i,B'}) \cdot \omega = \sum_{i=0}^{R} \theta_i(\boldsymbol{\beta}_{i,B'} \cdot \omega)$.

Denote $\Theta_{B'}(\boldsymbol{\theta}) = \sum_{i=0}^{R} \theta_i \boldsymbol{\beta}_{B'}$, where $\boldsymbol{\theta} = [\theta_0, \ldots, \theta_r]$. Observe that $\Theta_{B'}(\boldsymbol{\theta})$ is a $R+1$ tuple, and that for distinct $B' \in \Gamma'$ we get distinct $\Theta_{B'}(\boldsymbol{\theta})$. For each $B' \in \Gamma'$, we have $\Theta_{B'}(\boldsymbol{\theta}) = (\xi_0, \ldots, \xi_R)$ where ξ_j is some linear combination of θ_i's. Further, θ_i cannot be used in the linear combination that represents ξ_j whenever $i > j$. This follows from our construction of $\Upsilon_2(L_{B'}C_{B',\mathcal{K}})$.

Define $\Lambda_0 = \{\Theta_{B'}(\boldsymbol{\theta}) \; : \; B' \in \Gamma' \text{ and } \xi_0 \neq \theta_0\}$. Inductively, for $i = 1, \ldots, R$ we define $\Lambda_i = \{\Theta_{B'}(\boldsymbol{\theta}) \; : \; B' \in \Gamma' \;, \xi_0 = \theta_0, \ldots, \xi_{i-1} = \theta_{i-1} \text{ and } \xi_i \neq \theta_i\}$.

Lemma 16. *For $i = 0, \ldots, R$,*

$$|\Lambda_i| \leq 2^i.$$

Theorem 17. *For all GILTS (group independent linear threshold schemes) which satisfy the all-revealing model, the amount of randomness R satisfies*

$$2^{R+1} \geq 1 + \binom{n}{t-1}.$$

Proof. Let $\mathcal{K}$ be a group of exponent 2. What we will establish is that the amount of randomness required in such a group exceeds $\log_2(1 + \binom{n}{t-1})$, this will establish the result.

Recall that for each $B' \in \Gamma'$ we have defined $\Theta_{B'}(\boldsymbol{\theta})$ such that for distinct B' we get distinct $\Theta_{B'}(\boldsymbol{\theta})$. Therefore we see that $|\{\Theta_{B'}(\boldsymbol{\theta}) : B' \in \Gamma'\}| \geq \binom{n}{t-1}$. Some observations:

– for $i, j = 0, \ldots, R$, $i \neq j$ we see that by definition $\Lambda_i \cap \Lambda_j = \emptyset$, and
– for each $\Theta_{B'}(\boldsymbol{\theta}) = (\xi_0, \ldots, \xi_R)$, if there θ_i is used in some linear combination for ξ_j, then $j \geq i$.

The second observation implies that each $\Theta_{B'}(\boldsymbol{\theta})$ belongs to some Λ_i (where $0 \leq i \leq R$). This follows from the argument that if $\Theta_{B'}(\boldsymbol{\theta})$ does not belong to Λ_i for all i, then $\xi_0 = \theta_0$, $\xi_1 = \theta_1, \ldots, \xi_R = \theta_R$. Consequently the original matrix $L_{B'}C_{B'}$ has rank $R+1$, which of course implies that $C_{B'}$ has rank $R+1$, contradicting Theorem 4.

The first observation implies that $|\bigcup_{i=0}^{R} \Lambda_i| = \sum_{i=0}^{R} |\Lambda_i|$. Using Lemma 16, we see that $\sum_{i=0}^{R} |\Lambda_i| \leq 1 + 2 + \cdots + 2^R = 2^{R+1} - 1$. Now using both observations we see that $|\{\Theta_{B'}(\boldsymbol{\theta}) : B' \in \Gamma'\}| \leq \sum_{i=0}^{R} |\Lambda_i| = 2^{R+1} - 1$. Hence $\binom{n}{t-1} \leq 2^{R+1} - 1$. Therefore, $2^{R+1} \geq 1 + \binom{n}{t-1}$.

5.2 Remarks

Let us say that if a GILTS requires the minimal amount of randomness, then it is *efficient*. Observe that the 2 out of 3 GILTS described in Example 1 is all-revealing, and it is efficient (requires a minimal amount of randomness) in that $R = 2$. This last remark follows from the fact that this scheme possess a minimal amount of subshares. (This is due to a result in [3], which implies that for a 2 out of n scheme, $\sum a_i \geq n \log_2 n$), and Theorem 11 which implies $R \geq a_i + (t-2)$. Thus for all 2 out of 3 schemes $R \geq 2$. Observe that Theorem 17 fails to give us the true minimal bound, since $R + 1 \geq \log_2(1 + \binom{3}{1}) = 2$.

The 2 out of 4 GILTS given in Example 3 is also an example of an efficient GILTS (minimal amount of randomness required), again $R = 2$. This follows from the same argument as above. First $\sum a_i \geq 4 \log_2 4 = 8$, by [3]. Secondly, by Theorem 11, $R \geq a_i + (t-2) = 2$. It is trivial to show that this scheme is all-revealing. In this example we find that Theorem 17 provides a tight bound for the 2 out of 4 schemes. That is $R + 1 \geq \log_2(1 + \binom{4}{1}) > 2$. That is, integer $R + 1$ must exceed 2, hence $R \geq 2$.

6 Conclusion

We have formalized the definition of group independent linear threshold sharing and introduced new lower bounds for randomness in a GILTS. In addition, we have provided some examples of cases when these bounds would be tight. We still see that there exists an enormous gap between our bounds on randomness and the randomness required by the scheme [17]. That is, $R+1 \geq \sqrt{\log_2(1+\binom{n}{t-1})}$ and in [17] we have $R \geq n(t-1)$. Future work may include examining this gap. We have also introduced randomness bounds, which incorporate share size. That is, Theorem 11. It would be worthwhile to develop bounds on share size within a GILTS. That is, the bounds on 2 out of n threshold schemes within [3], helped us to show that Examples 1 and 3 were efficient. Lastly, we have found that both of these efficient GILTS were all-revealing, is it true that all efficient GILTS must be all-revealing? That is, if a scheme requires a minimal amount of randomness does this imply that it must be all-revealing?

References

1. W. Adkins and S. Weintrab. *Algebra, an approach via module theory.* Springer-Verlag, NY, 1992.
2. G. Blakley. “Safeguarding cryptographic keys.” In *Proc. Nat. Computer Conf. AFPIPS Conf. Proc., 48* pp. 313-317, 1979.
3. S. Blackburn, M. Burmester, Y. Desmedt, and P. Wild. “Efficient Multiplicative Sharing schemes”. In *Advances in Cryptology - Eurocrypt ’96, LNCS 1070*, pp. 107-118, Springer-Verlag, 1996.
4. C. Blundo, A. De Santis, and U. Vaccaro. “Randomness in Distribution Protocols”. *Inform. Comput.* pp. 111-139, 1996.
5. C. Blundo, A. De Santis, R. De Simone,, and U. Vaccaro. “Tight Bounds on the Information rate of secret Sharing Schemes”. In *Design, Codes and Cryptography*, 11, pp. 107-122, 1997.
6. C. Blundo, A.G. Gaggia, and D. R. Stinson. “On the Dealer’s randomness Required in Secret Sharing Schemes”. In *Design, Codes and Cryptography*, 11, pp. 235-259, 1997.
7. C. Blundo and B. Masucci. Randomness in Multi-Secret Sharing Schemes. In *Journal of Universal Computer Science,* Vol. 5, No. 7, 1999, pp. 367–389.
8. C. Blundo and B. Masucci. A note on the Randomness in Dynamic Threshold Scheme. In *Journal of Computer Security*, Vol. 7, No. 1, 1999, pp. 73–85.
9. C. Boyd, Digital Multisignatures, *Cryptography and coding*, Clarendon Press, 1989, pp 241-246.
10. R.M. Capocelli, A. De Santis, L. Gargano, and U. Vaccaro, “On the Size of Shares for secret Sharing Schemes” In *Journal of Cryptology*, 6, pp. 157-167, 1993.
11. L. Csirmaz. “The Size of a Share Must Be large”. In *Journal of Cryptology*, 10, pp. 223-231, 1997.
12. L. Csirmaz. The dealer’s random bits in perfect sharing schemes In *Studia Sci. Math. Hungar.* 32(1996) pp.429-437.
13. A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung. “How to share a function”. In *Proceedings of the twenty-sixth annual ACM Symp. Theory of Computing (STOC)*, pp. 522-533, 1994.

14. A. De Santis, and B. Masucucci. "Multiple Ramp Schemes". In *IEEE Transns. on Inform. Theory*, 45, no. 5, pp. 1720-1728, 1999.
15. Y. Desmedt. Society and group oriented cryptography: a new concept. In *Advances of Cryptology- Crypto '87*
16. Y. Desmedt, G. Di Crescenzo, and M. Burmester. "Multiplicative non-abelian sharing schemes and their application to threshold cryptography". In *Advances in Cryptology - Asiacrypt '94, LNCS 917.* pp. 21-32, Springer-Verlag, 1995.
17. Y. Desmedt and Y. Frankel. "Homomorphic zero-knowledge threshold schemes over any finite abelian group". In *Siam J. Disc. Math. vol 7, no. 4* pp. 667-679, SIAM, 1994.
18. Y. Desmedt and S. Jajodia. Redistributing secret shares to new access structures and its applications. Tech. Report ISSE-TR-97-01, George Mason University, July 1997 ftp://isse.gmu.edu/pub/techrep/97.01.jajodia.ps.gz
19. Y. Desmedt, B. King, W. Kishimoto, and K. Kurosawa, "A comment on the efficiency of secret sharing scheme over any finite abelian group", In *Information Security and Privacy*, ACISP'98 (Third Australasian Conference on Information Security and Privacy), LNCS 1438, 1998, 391-402.
20. Y. Frankel, Y. Desmedt, and M. Burmester. " Non-existence of homomorphic general sharing schemes for some key spaces", in *Advances of Cryptology- Crypto '92*, 740 ,1992 pp 549-557
21. Y. Frankel, P. Gemmel, P. Mackenzie, and M. Yung. "Proactive RSA". In *Advances of Cryptology-Crypto '97*, 1997, LNCS 1294, Springer Verlag, 1997, p. 440-454.
22. Y. Frankel, P. Gemmel, P. Mackenzie, and M. Yung. "Optimal-Resilience Proactive Public-key Cryptosystems". In *Proc. 38th FOCS*, IEEE, 1997, p. 384-393.
23. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. "Robust and efficient sharing of RSA functions". In *Advances of Cryptology-Crypto '96,* LNCS 1109, Springer Verlag, 1996, p. 157-172.
24. B. King. "Improved Methods to Perform Threshold RSA". In *Advances in Cryptology - ASIACRYPT 2000*, LNCS 1976, Springer Verlag, 2000,p. 359-372 .
25. H.L. Keng. *Introduction to Number Theory.* Springer Verlag, NY 1982
26. T. Hungerford. *Algebra.* Springer-Verlag, NY, 1974.
27. E. Karnin, J. Greene, and M. Hellman. "On secret sharing systems." In *IEEE Trans. Inform. Theory, 29(1),* pp. 35-41, 1983.
28. E. Kushilevitz and A. Rosen. A Randomness Rounds Tradeoff in Private Computation. In *Advances in Cryptology - CRYPTO '94*, LNCS 839, 1994 pp. 397 - 410.
29. R. Lidl and G. Pilz. *Applied Abstract Algebra.* Springer Verlag, NY 1984
30. R. Rivest, A. Shamir, and L. Adelman, A method for obtaining digital signatures and public key cryptosystems, *Comm. ACM,* 21(1978), pp 294-299.
31. A. Shamir, How to share a secret, *Comm. ACM*, 22(1979), pp 612-613.
32. V. Shoup. "Practical Threshold Signatures". In *Advances in Cryptology - EUROCRYPT 2000*, LNCS 1807, Springer Verlag 2000, p. 207-220.
33. D. Stinson. *Cryptography, Theory and practice.* CRC Press, NY, 1995
34. M. van Dijk. "A Linear Construction of secret Sharing Schemes". In *Design, Codes and Cryptography* 12, pp. 161-201, 1997.

7 Appendix

Example 2. [16] Consider the following 2 out of 4 scheme, such that each participant is given 2 shares. Participant P_i is given share $\overline{s}_i$ such that s_{i1} and s_{i2} are defined by the following table.

s_{11}	s_{12}	s_{21}	s_{22}	s_{31}	s_{32}	s_{41}	s_{42}
$k - r_2$	$k - r_1$	$k - r_2$	r_1	r_2	$k - r_3$	r_2	r_3

.

For all $\mathcal{K} \in \mathbf{K}$, let $S_{i,\mathcal{K}} = \mathcal{K}^2$. k represents the secret, r_1, r_2, r_3 represent three random elements that will be chosen from the finite abelian group uniformly random. Here $-r$ represents the inverse of r. Each row represents one of the $\binom{4}{2}$ sets B (sets of cardinality 2) and indicates how that set can compute the secret. The corresponding Ψ will be

$$\Psi = \begin{bmatrix} 1&0&0&0&1&0&0&0 \\ 1&0&0&0&0&0&1&0 \\ 0&0&1&0&1&0&0&0 \\ 0&0&1&0&0&0&1&0 \\ 0&1&0&1&0&0&0&0 \\ 0&0&0&0&0&1&0&1 \end{bmatrix}$$

So we have $\Psi\overline{s} = [k, k, \ldots, k]^T$, and $\sum a_i = 8$.

Example 3. [16]

One can easily see that to have two distinct random elements r_2 and r_3 is not necessary. We can choose $r_1 = r_3$ and still achieve Definition 3.

s_{11}	s_{12}	s_{21}	s_{22}	s_{31}	s_{32}	s_{41}	s_{42}
$k - r_2$	$k - r_1$	$k - r_2$	r_1	r_2	$k - r_1$	r_2	r_1

It can easily be established that this scheme is as secure as the first example. The share size is the same as before, but to create a more *efficient* scheme (i.e. reduce the amount of randomness), we have increased the rank of Ψ (this was done by introducing an additional dependency between shares). The corresponding Ψ will be

$$\Psi = \begin{bmatrix} 1&0&0&0&1&0&0&0 \\ 1&0&0&0&0&0&1&0 \\ 0&0&1&0&1&0&0&0 \\ 0&0&1&0&0&0&1&0 \\ 0&1&0&1&0&0&0&0 \\ 0&0&0&0&0&1&0&1 \\ 0&0&0&1&0&0&0&-1 \end{bmatrix},$$

where $\Psi\overline{s} = [k, k, k, k, k, k, 0]^T$. This last row of Ψ is needed. This row infers that the second share of P_2 is the same as the second share of P_4. (i.e. $r_1 = r_3$.)

Democratic Systems

Hossein Ghodosi[1] and Josef Pieprzyk[2]

[1] School of Information Technology,
James Cook University, Townsville, Qld 4811, Australia,
hossein@cs.jcu.edu.au
[2] Department of Computing,
Macquarie University, Sydney, NSW 2109 Australia,
josef@ics.mq.edu.au

Abstract. Digital signature is a breakthrough of modern cryptographic systems. A (t, n) threshold digital signature allows every set of cardinality t or more (out-of n) co-signers to authenticate a message. In almost all existing threshold digital signatures the threshold parameter t is fixed. There are applications, however, in which the threshold parameter needs to be changed from time to time. This paper considers such a scenario, in order to discuss relevant problems, and proposes a model that solves the related problems.

1 Introduction

In democratic organizations a majority of members rules. So if an organization has n members, any collection of t, where $t \geq \lfloor \frac{n}{2} \rfloor + 1$, has to have the power to act on behalf of the organization. Examples of such organizations include legislative bodies governing countries (parliaments, senates, etc.), cities (city councils), companies and other democratic institutions. Cryptography developed a (t, n) threshold digital signature which can be generated by any group of t members. The fact that a (democratic) group successfully generated a signature means that a majority of its members agreed on the form of the signed message (that may be a piece of legislation). In general, (t, n) threshold signature scheme allows a group of n participants to collectively hold a group secret key. Each participant, using his share from the secret key, can generate a partial signature such that

– any collection of t or more partial signatures enables the group to compute a valid signature of the group,
– any collection of $t - 1$ or less partial signatures is not enough to yield the valid group signature.

Additionally, we require that the knowledge of partial signatures does not allow either the group secret or the shares of the group secret held by participants to be recovered. The group signature is computed

1. *with no help of a combiner* – a collection of currently active participants pool their partial signatures and recover collectively the group signature (each active participant acts as a combiner),

2. *with the help of a combiner* – a collection of active participants submit their partial signatures to a trusted *combiner*, who performs the computation for them and generates the group signature.

Threshold signatures must apply secret sharing schemes. It is worth noting that a threshold secret sharing scheme is typically *one-time* or once an authorised set of participants has reconstructed the secret, both the secret and all shares have become known to everyone within the group. For a threshold signature, however, a collection of participants reveal their partial signatures without compromising their shares so the signature scheme can be used a (polynomial) number of times.

The concept of group signatures was invented by Boyd [4] who demonstrated how to adapt the RSA system to implement a $(2, 2)$ multisignature and a $(2, n)$ threshold RSA signature. A general (t, n) threshold RSA signature scheme has been proposed by Desmedt and Frankel [6]. Implementations of threshold and multisignature schemes based on the ElGamal and its variants (the DSS signature schemes) were the subject of extensive investigations (see, for example, [10], [14], [11], [12] and [16]).

In general, a (t, n) threshold signature is set up by a trusted dealer or key authentication center (KAC), who generates the secret key K, designs a (t, n) threshold secret sharing scheme and distributes the shares of the secret key among the participants. Note that the underlying secret sharing is static so the threshold parameter t is fixed for all signatures generated by the scheme.

There are, however, some applications for which the threshold parameter of the signature needs to be changed from time to time [13]. For example, consider a legislative body (parliament) with 100 members. If all members are present, then any majority with at least 51 is able to pass a bill. To support the voting on the bill by using cryptography in these circumstances, it is enough to design a $(51, 100)$ threshold signature. In practice, however, the presence of the whole house is a rare event and most of the time the number of members present is smaller than 100. There is also a lower bound imposed on the number of members below which all bills passed do not have any legal significance. The quorum is the smallest group of members whose decisions are still legally binding. In our example the cardinality of the smallest quorum is 51. So if there are 51 sitting members voting on a bill, then the bill becomes a law if there are 26 members voting for it. To support this scenario cryptographically, one would need to implement a $(26, 51)$ threshold signature. It is obvious by now that to enable members of parliament to generate valid signatures, the threshold parameter must be adjusted depending on the number of currently sitting members. The range of possible thresholds spans from 26 to 51.

Hence, we would need a dynamic (t_i, n_i) threshold signature scheme which allows us to transfer *signing power* from a set of t members to a set of t_i $(t > t_i)$ members. Note that, the transfer of power must be temporary and applicable for signing a single message, and must not be useful for signing any other message. Thus, a solution which allows us to go from a (t, n) threshold scheme to a (t_i, n_i) threshold scheme (such as, [7]) on the underlying threshold scheme of the threshold signature scheme is not acceptable.

1.1 The Scenario

Consider a parliament with n members. A quorum must have at least $\lfloor \frac{n}{2} \rfloor + 1$ members to result in legally binding proceedings. Sessions are typically run by a chair-person or a committee and there may also be secretaries and even an audience. The secretaries and audience are playing a passive role. The chair-person puts forward the motion (a proposed bill, or simply the message) for voting. The voting can be conducted in two ways, either by casting public or secret ballot. Public voting is in many cases considered to be undemocratic as the voters may be subject to an undue pressure (such as the party discipline). Secret ballot is preferred in these situations where the way participants have voted is to remain secret. The ballot in the voting is binary "yes/no" – the members may agree or disagree with the proposed motion. The motion is passed if the number of "Yes" votes is bigger than $\lfloor \frac{n_i}{2} \rfloor$, where n_i is the number of members present in the session; otherwise it is rejected. The result, whatever it is, will be recorded by secretaries and the documentation keeps track of events in sessions. Note that, the motion should not be known to participants prior to the beginning of the session. Otherwise participants may want the motion to be delayed by walking out of the session and making a quorum unachievable.

1.2 Requirements

The acceptance or rejection of a motion in a democratic group with n participants can be supported by an electronic system based on a threshold signature which needs to satisfy the following conditions.

1. Initially, we need to design a $(\lfloor \frac{n}{2} \rfloor + 1, n)$ threshold signature scheme. In order to sign a message during the i-th session with the presence of n_i participants ($n_i \geq \lfloor \frac{n}{2} \rfloor + 1$), the power to sign a message must be transfered to t_i participants ($t_i = \lfloor \frac{n_i}{2} \rfloor + 1$), where t_i is a new threshold.
2. Transfer of the power to sign must be
   – for a single message only – partial signatures generated by active participants during the i-th session, do not compromise security of threshold signatures generated during other sessions (and vice versa),
   – for a duration of the i-th session only – the validity of partial signatures generated during the i-th session is limited for the duration of the session. This is to say that "if a message is not signed at the i-th session (because the majority has not agreed), then later the message cannot be signed (even if the circumstances have dramatically changed)." To force this requirement, we let participants generate valid group signatures only with the help of a trusted combiner. Note that the chair-person can play the role of the combiner.
3. Voting for/against the motion is to be conducted using a secret ballot so nobody (currently active participants and other participants) is able to discover how the participants voted apart from the knowledge that the motion has been successfully moved (and the corresponding message signed) or that

the motion failed. The trusted combiner is, however, allowed to know how the participants voted.
4. Any minority of dishonest participants cannot create bogus sessions. This is another reason why we use the trusted combiner.

2 The Model

The necessary cryptographic components include the following

– A (t, n) threshold signature – this is the basic signature scheme to be used repeatedly for all consecutive sessions. The parameter $t = \lfloor \frac{n}{2} \rfloor + 1$ specifies the size of a smallest quorum which still has the power to run legally binding sessions.
– A cryptographic protocol used to distribute partial signatures of a message (blinded by the combiner) from the current quorum with n_i participants into a (t_i, n_i) threshold scheme. The resulting one-time threshold signature is unblinded – the message which corresponds to the threshold signature is known.
– The combiner must be active in all legally binding sessions. To give the combiner the required power, the secret key K is split into two parts (K_1, K_2) such that $K = K_1 + K_2$. The first part K_1 is distributed among the participants and the second one K_2 is assigned to the combiner. The valid signature can be generated only if both a majority of the current quorum and the combiner collaborate. One may argue that assigning a portion of the secret key to the combiner is not reasonable – what happens if the combiner loses the key? This problem can be solved by putting a committee in place of a single combiner and distributing shares of K_2 among the members of the committee.
– Generation of a signature for a motion can be blocked by the combiner who may refuse to collaborate in signing. The combiner later may claim that there was not enough support for the motion. To avoid these problems, we assume that the participants are voting for or against the motion by signing either "yes" message M or "no" message M'. Hence, at the end of the session one and only one of these messages will be signed, which indicates the decision of the members (not the combiner) regarding the motion.

3 Components of the System

This section considers the basic tools which we will use for the implementation of our system.

3.1 Communication Channel

Each member and the combiner is connected to a common broadcast medium with the property that messages sent to the channel instantly reach every party connected to it. We assume that the broadcast channel is public, that is, everybody can listen to all information communicated via the channel, but cannot modify them. We assume also there exists private channels between every pair of members, with the property that nobody neither can listen to nor can modify the messages sent via these private channels.

3.2 Threshold Scheme

Threshold secret sharing schemes were introduced in [19,3]. Due to its nice algebraic structure, the Shamir scheme [19] is frequently applied in society-oriented cryptographic systems. The Shamir (t, n) threshold scheme is based on polynomial interpolation. Let the secret be an element of a finite field, that is, $K \in \mathbb{Z}_p$, where p is a prime number. Shamir suggests the following algorithm for constructing a (t, n) threshold scheme.

1. The dealer, $\mathcal{D}$, chooses n distinct and non-zero elements of $\mathbb{Z}_p$, denoted $x_1, \ldots, x_n$ and sends x_i to P_i via a public channel.
2. $\mathcal{D}$ secretly chooses (independently at random) $t-1$ elements of $\mathbb{Z}_p$, denoted $a_1, \ldots, a_{t-1}$ and forms the polynomial
$$f(x) = K + \sum_{i=1}^{t-1} a_i x^i.$$
3. For $1 \leq i \leq n$, the dealer computes s_i, where
$$s_i = f(x_i) \pmod{p}.$$
4. $\mathcal{D}$ sends (via private channel) share s_i to participant P_i; $i = 1, \ldots, n$.

At the reconstruction phase of the secret, every set of at least t participants can apply the Lagrange interpolation formula to reconstruct the polynomial and hence recover the secret. Alternatively, participants could give their shares to a trusted authority, called the *combiner*, to perform the computation for them.

3.3 Threshold Signature

The first solution for democratic systems was proposed by Desmedt and King [8] and it was based on the RSA threshold signature scheme. In [15] Li, Hwang and Lee have argued that a (t, n) threshold signature scheme does not only require that less than t users must not be able to generate a correct signature, but also a particular set of t participants should not be forged by another set of t participants. They also pointed out that the Desmedt-Frankel's [6] (t, n) threshold RSA signature is subject to the *conspiracy attack*. That is, if t (or more) participants conspire, then the group secret key and all participants' shares will be revealed. Once the shares are revealed, the set of collaborating participants can impersonate another set of shareholders to sign a message without holding the responsibility of the signatures, and can deny having signed a message though in fact they have signed it.

This is a common shortcoming of almost all threshold signature schemes. We design our system using an ElGamal-type threshold signature scheme that removes this shortcoming. In our system, even if all members collaborate, they can get K_1 which is useless without knowing K_2. On the other hand, the combiner (or the members of committee) cannot obtain anything more than K_2. That is, the group secret key can be recovered if and only if the participants and the combiner (the committee members) collaborate.

3.4 Verifiable Transfer of Signature Shares

The main assumption of threshold schemes is that the dealer is trustworthy. In our model, the transfer of power to sign a message requires that a subset of members (with cardinality equal to or greater than t) applies a threshold scheme and distributes its partial signatures on the motion among all present members in a session. Since the honesty of members is not guaranteed, the correctness of the shares given to each member could be questionable. A solution to these sorts of problems has been discussed in *verifiable secret sharing* schemes (see, for example, [2], [17], [9]).

Verifiable secret sharing schemes allow the honest participants to ensure that their shares are correct (related to the secret) and thus in the secret reconstruction phase they will recover the original secret. Stadler [18] proposed a *publicly verifiable* secret sharing scheme in which not only the participants but also an outsider can verify that the shares are correctly distributed. The underlying verifiable secret sharing scheme which we will use in our implementation is due to Pedersen [17]. We will propose a protocol that convinces the participants, the combiner, and even outsiders about the correctness of the signature and relevant shares.

4 Implementation

In this section, we implement the proposed democratic scheme that satisfies the requirements of its corresponding real-life application.

4.1 Initialization

We employ the Harn [10] (t, n) threshold digital signature. This system utilises the Shamir threshold scheme and a modified version of the ElGamal signature. A dealer or trusted key authentication centre (KAC) selects the system parameters as,

- p, a prime modulus, where $2^{511} < p < 2^{512}$;
- q, a prime divisor of $p - 1$, where $2^{159} < q < 2^{160}$;
- a random integer $K_2 \in \mathbb{Z}_q$ as the secret key of the combiner (or a committee that plays the role of combiner);

- compute K_1, such that $K = K_1 + K_2 \pmod{q}$, where K is the secret key associated with the organization;
- a polynomial $f(x) = K_1 + a_1 x + \cdots + a_{t-1} x^{t-1} \bmod q$, where K_1 is the secret associated with the members and a_i are random integers in $\mathbb{Z}_q$;
- $g = h^{(p-1)/q} \pmod{p}$, where $h \in GF(p)$ is a primitive element (g is an element of order q in $GF(p)$);

The parameters p, q and g are public, but K, K_1, K_2, and $a_1, \ldots, a_{t-1}$ are secret values.

The KAC uses the Shamir (t, n) threshold scheme to share the secret K_1 among the set $\mathcal{P} = \{P_1, \ldots, P_n\}$ of the group members. That is, it assigns $s_i = f(x_i)$ to participant P_i $(1 \leq i \leq n)$. It assigns K_2 to the combiner. For simplicity, we assume the system has a single combiner. It is not difficult to show how it works if a committee runs the sessions. The KAC also publishes $k = g^K \bmod p$ as the group public key and $u_i = g^{f(x_i)} \bmod p$ as the public key of participant P_i $(1 \leq i \leq n)$.

4.2 Opening a Session

The combiner/chair-person sends an announcement via the public communication channel. Those participants who want to take part in the session will respond to the announcement. If the number of participants is large enough (i.e., satisfies the quorum) then the session can be started. Note that, at this stage everyone knows the identity of all other members present in the session.

A common practice in generating digital signatures is to apply an appropriate hash function [5] on the message and then sign the hash value. Fundamental characteristics of all such hash functions are that they are one-way and collision free. Obviously, knowing the hash value of a message gives no knowledge about the message itself.

Let $\{P_1, P_2, \ldots, P_{n_i}\}$ be the set of present members in the i-th session ($n_i \geq t$). The combiner/chair-person, who is supposed to collect the member's vote to motion m_i, applies the hash function on the motion and sends the hash value of the motion, M_i, via the communication channel.

4.3 Transfer of Signature Shares

In order to sign the message M_i, each participant P_i of an authorised set $\mathcal{A}$ ($|\mathcal{A}| \geq t$) chooses a random value r'_i $(1 < r'_i < q-1)$ and computes a public value r_i, as

$$r_i = g^{r'_i} \pmod{p}$$

and makes r_i publicly available through the broadcast channel. The combiner also chooses a random value r'_c $(1 < r'_c < q-1)$ and computes a public value r_c, as

$$r_c = g^{r'_c} \pmod{p}$$

Once r_c and all r_i are available, collaborating participants of the set $\mathcal{A}$ compute,

$$r = r_c \times \prod_{P_i \in \mathcal{A}} r_i \pmod{p}.$$

Participant $P_i \in \mathcal{A}$ uses his secret $f(x_i)$ and his chosen *one-time* random r'_i, to sign the message M_i as,

$$\sigma_i = f(x_i) \times M_i \times \left(\prod_{\substack{P_i \in \mathcal{A} \\ i \neq j}} \frac{-x_j}{x_i - x_j} \right) - r'_i \times r \pmod{q}.$$

In order to distribute partial signatures among the members present such that every subset of cardinality t_i (out-of n_i members present in the i-th session) is able to reconstruct the signature, participant P_i generates a polynomial $f_i(x) = \sigma_i + a_{i1}x + \cdots + a_{it_i-1}x^{t_i-1} \bmod q$, where a_{i_j} are random integers in $\mathbb{Z}_q$. Then P_i publishes g^{σ_i}, $g^{a_{i\ell}} (1 \leq \ell \leq t_i - 1)$ and privately sends shares $s_{i_j} = f_i(x_j)$ $(1 \leq j \leq n_i)$ to all present participants in the i-th session (P_i keeps his share, s_{i_i}). Two verifications can be done at this stage:

1. Every participant, P_j, can verify that the share s_{i_j} given by P_i is in fact relevant to the partial signature σ_i, by using

$$g^{s_{i_j}} \stackrel{?}{=} g^{\sigma_i} \times \prod_{\ell=1}^{t_i-1} (g^{a_{i\ell}})^{x_j{}^{\ell}} \pmod{q}.$$

2. Simultaneously, every participant, the combiner, and even an outsider can verify the correctness of the partial signature of participant P_i using,

$$(u_i^{M_i})^{\left(\prod_{\substack{P_i \in \mathcal{A} \\ i \neq j}} \frac{-x_j}{x_i - x_j}\right)} \stackrel{?}{=} r_i^r \times g^{\sigma_i} \pmod{p}.$$

   If the equation holds true, the partial signature (r_i, σ_i) of message M_i generated by participant P_i is valid.

Note that, in spite of the verifiability of the partial signatures, nobody can get any information about the partial signatures and therefore about the signature itself (considering the fact that discrete logarithm is a *hard* problem). Also note that dishonest members can be detected and removed from the system at this stage.

4.4 Signing Motion Messages

Once all partial signatures of an authorised set are distributed and verified, the combiner discloses the messages m_i and m'_i (note that partial signatures are generated as discussed previously for two yes and no messages).

Active participants are making up their minds as to which motion they support – they have to be either for or against the motion. Abstention is not an option. There is always a motion that has attracted the support of at least t_i members present in the session. They submit their partial signatures to the combiner and thus they collectively compute the valid signature for either the message M_i or M_i'.

In particular, participant P_i keeps a share $s_{i_i} = f_i(x_i)$ corresponding to the partial signature σ_i generated by himself. P_i also has received at least $t-1$ shares from other co-signers relevant to partial signatures σ_j $(1 \leq j \leq t, \quad j \neq i)$. If P_i adds all these shares (computation is done modulo q), then the resulting value is his share from the secret $\sum_{i=1}^{t} \sigma_i$ (according to the $(+,+)$-homomorphism property of Shamir's threshold scheme –see [1] for further details). That is, every participant's share is derived from the following polynomial

$$F(x) = A_0 + A_1 X + A_2 X^2 + \cdots + A_{t_i-1} X^{t_i-1} \pmod q$$

when $A_0 = \sum_{i=1}^{t} \sigma_i$ is the signature of M_i corresponding to the partial secret K_1, and A_i $(1 \leq i \leq t_i - 1)$ are random integers in $\mathbb{Z}_q$.

Each member transmits his share (on messages M_i or M_i' as "Yes" or "No" vote) to the combiner. The combiner either is able to generate the valid signature of M_i (if at least t_i partial signatures are available) or to generate the valid signature of the message M_i' (if at least t_i partial signatures of M_i' are available). For example, if a majority of members present have given their votes to the motion m_i, the combiner then uses his secret K_2 and his chosen *one-time* random r_c', to sign the message M_i as,

$$\sigma_c = M_i \times K_2 - r_c' \times r \pmod q.$$

Finally, the group signature on the message M_i can be computed using

$$\sigma = \sigma_c + \sum_{P_i \in \mathcal{A}} \sigma_i \pmod q.$$

To verify the signature, every one who knows the group public key can check,

$$k^{M_i} \overset{?}{=} r^r \times g^\sigma \pmod p.$$

If the equation holds true, the group signature (r, σ) is valid. If a single combiner is replaced by a committee, then the committee performs a similar interaction (as performed by participants) in order to generate their signature on the message and to compute the group signature.

Note that, although the Shamir scheme is susceptible to cheating, the participants cannot cheat the combiner by giving false shares because, the combiner can easily verify the correctness of shares of each participant and thus be able to detect any possible cheating. In fact, if participant P_j submits S_j as his share then the combiner can verify the correctness of S_j using

$$g^{S_j} \overset{?}{=} \prod_{P_i \in \mathcal{A}} \left(g^{\sigma_i} \times \prod_{\ell=1}^{t_i-1} (g^{a_{i\ell}})^{x_i{}^{\ell}} \right) \pmod q.$$

5 Security Analysis

The security of underlying cryptographic tools has been discussed in relevant papers. The security of the entire system will be considered in the final version of the paper.

Note that, the major effort in our design (similar to the design of any other electronic schemes) is to satisfy the requirements imposed by a real-life system. One of the assumptions was the chair-person/combiner is unbiased and simply wants to get the decisions of the members on a motion. This is not the case in all real-life applications. There are situations in which a distrusted organization/government wants to get its members'/citizens' vote in its favour. In such cases, the result of voting is questionable (since the combiner can swap "Yes" and "No" votes). A commonly used solution is to get the help of some arbiters, such as an international committee to control the voting procedure.

In our scheme, even if the combiner is biased it cannot change the result of voting as long as the underlying signature scheme is unforgeable. However, the proposed scheme can be corrupted if the basic assumption in almost all secret sharing schemes –that considers an honest dealer distributes the shares of a secret key among all parties in the system– fails. Being more realistic, experiences have shown that having such a trusted authority in crucial and disputable situations is not an easy task. In order to avoid this problem, in the final version of this paper, we will consider the case where the dealer is removed from the system.

References

1. J. Benaloh, "Secret Sharing Homomorphisms: Keeping Shares of a Secret Secret," in *Advances in Cryptology - Proceedings of CRYPTO '86* (A. Odlyzko, ed.), vol. 263 of *Lecture Notes in Computer Science*, pp. 251–260, Springer-Verlag, 1987.
2. M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness Theorem for Non-Cryptographic Fault-Tolerant Distributed Computation," in *20th Annual Symposium on the Theory of Computing (STOC)*, pp. 1–10, 1988.
3. G. Blakley, "Safeguarding cryptographic keys," in *Proceedings of AFIPS 1979 National Computer Conference*, vol. 48, pp. 313–317, 1979.
4. C. Boyd, "Digital Multisignatures," in *Cryptography and Coding* (H. Beker and F. Piper, eds.), pp. 241–246, Clarendon Press, 1989.
5. D. Denning, "Digital Signatures with RSA and Other Public-Key Cryptosystems," *Communications of the ACM*, vol. 27, no. 4, pp. 388–392, 1984.
6. Y. Desmedt and Y. Frankel, "Shared generation of authenticators and signatures," in *Advances in Cryptology - Proceedings of CRYPTO '91* (J. Feigenbaum, ed.), vol. 576 of *Lecture Notes in Computer Science*, pp. 457–469, Springer-Verlag, 1992.
7. Y. Desmedt and S. Jajodia, "Redistributing secret shares to new access structures and its applications," tech. rep., George Mason University, ISS-TR-97-01, ftp://isse.gmu.edu/pub/techrep/97_01i_jajodia.ps.gz, 1997.
8. Y. Desmedt and B. King. Verifiable Democracy. In *Secure Information Networks, IFIP TC6/TC11 Joint Working Conference on Communications and Multimedia Security (CMS'99)*, pages 53–70. Kluwer Academic Publishers, September 1999.

9. R. Gennaro and S. Micali, "Verifiable Secret Sharing as Secure Computation," in *Advances in Cryptology - Proceedings of EUROCRYPT '95* (L. Guillou and J.-J. Quisquater, eds.), vol. 921 of *Lecture Notes in Computer Science*, pp. 168–182, Springer-Verlag, 1995.
10. L. Harn, "Group-oriented (t, n) threshold digital signature scheme and digital multisignature," *IEE Proc.-Comput. Digit. Tech.*, vol. 141, pp. 307–313, Sept. 1994.
11. P. Horster, M. Michels, and H. Petersen, "Meta-Multisignature schemes based on the discrete logarithm problem," in *Information Security - the Next Decade,* (J. H. Eloff and S. H. Solms, eds.), IFIP/Sec '95, pp. 128–142, Proceedings of IFIP TC11 eleventh international conference on information security, Chapman and Hall, 1995.
12. S. Langford, "Threshold DSS Signatures without a Trusted Party," in *Advances in Cryptology - Proceedings of CRYPTO '95* (D. Coppersmith, ed.), vol. 963 of *Lecture Notes in Computer Science*, pp. 397–409, Springer-Verlag, 1995.
13. M. Reiter, "Secure Agreement Protocols: reliable and atomic group multicast in Rampart," in *Proceedings of the 2nd ACM Conference on Computer and Communications Security*, pp. 68–80, 1994.
14. C. Li, T. Hwang, and N. Lee, "Threshold-Multisignature Schemes where Suspected Forgery Implies Traceability of Adversarial Shareholders," in *Advances in Cryptology - Proceedings of EUROCRYPT '94* (A. Santis, ed.), vol. 950 of *Lecture Notes in Computer Science*, pp. 194–204, Springer-Verlag, 1995.
15. C. Li, T. Hwang, and N. Lee, "Remark on the Threshold RSA Signature Scheme," in *Advances in Cryptology - Proceedings of CRYPTO '93* (D. Stinson, ed.), vol. 773 of *Lecture Notes in Computer Science*, pp. 413–419, Springer-Verlag, 1994.
16. C. Park and K. Kurosawa, "New ElGamal Type Threshold Digital Signature Scheme," *IEICE Trans. Fundamentals*, vol. E79-A, pp. 86–93, Jan. 1996.
17. T. Pedersen, "Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing," in *Advances in Cryptology - Proceedings of CRYPTO '91* (J. Feigenbaum, ed.), vol. 576 of *Lecture Notes in Computer Science*, pp. 129–140, Springer-Verlag, 1992.
18. M. Stadler, "Publicly Verifiable Secret Sharing," in *Advances in Cryptology - Proceedings of EUROCRYPT '96* (U. Maurer, ed.), vol. 1070 of *Lecture Notes in Computer Science*, pp. 190–199, Springer-Verlag, 1996.
19. A. Shamir, "How to Share a Secret," *Communications of the ACM*, vol. 22, pp. 612–613, Nov. 1979.

Efficient and Unconditionally Secure Verifiable Threshold Changeable Scheme

Ayako Maeda*, Atsuko Miyaji, and Mitsuru Tada

School of Information Science,
Japan Advanced Institute of Science and Technology (JAIST),
Asahidai 1-1, Tatsunokuchi, Nomi, Ishikawa 923-1292, JAPAN.

Abstract. In this paper, we describe how to construct an efficient and unconditionally secure verifiable threshold changeable scheme, in which any participants can verify whether the share given by the dealer is correct or not, in which the combiner can verify whether the pooled shares are correct or not, and in which the threshold can be updated plural times to the values determined in advance. An optimal threshold changeable scheme was defined and given by Martin et. al., and an unconditionally secure verifiable threshold scheme was given by Pedersen. Martin's scheme is based on Blakley's threshold scheme whereas Pedersen's is based on Shamir's. Hence these two schemes cannot directly be combined. Then we first construct an *almost* optimal threshold changeable scheme based on Shamir's, and after that using Pedersen's scheme, construct a unconditionally secure verifiable threshold scheme in which the threshold can be updated plural times, say N times. Furthermore, our method can decrease the amount of information the dealer has to be publish, comparing with simply applying Pedersen's scheme N times.

1 Introduction

In a secret sharing scheme, a *secret* is broken into several pieces so that certain subsets of those pieces can reconstruct the secret. In a protocol, *a dealer* has a secret, and breaks it into several pieces called *shares*. An entity given a share is called *a participant*, *a shareholder* or *a member* simply. In this paper, we adopt the term *participant*. The entity to gather shares and recover the secret, is called *the combiner*. Basically, a secret sharing is regarded as a strategy for some important data protection. On the other hand, it is useful also for multiparty computation, for example, electronic auction, electronic voting, and so on.

As the most popular secret sharing schemes, we can see Shamir's polynomial-based scheme [Sha79] and Blakley's geometry-based scheme [Bla79]. In that scheme, a secret is broken into n pieces so that the secret can reconstruct with any t ($\leq n$) pieces, and not so that any $(t-1)$ pieces can determine the secret. Such a t is called *the threshold* of the scheme. Also we call such secret sharing schemes with the property given above (t, n)-*threshold schemes*.

* Current affiliation of the first author: Alpha systems corporation.

There are some threshold schemes in which the threshold can be changed without reconstructing the system [TTO99, MPSW99]. In this paper, we generically call such schemes *threshold changeable schemes*. In [TTO99] and in the first part of [MPSW99], after the initial setting, no secure channels is required, and the schemes before and after the threshold is changed are set to be *perfect*. However the required share size, precise to say the entropy of each share, has to be equal to or greater than the twice of that of the secret. Hence if we construct a scheme in which the threshold can be changed N times, the required share size is equal to or greater than $(N+1)$ times of that of the secret. On the other hand, in the latter part of [MPSW99], an optimal (t,n)-threshold scheme that is threshold changeable to t' $(> t)$ is defined, and a concrete construction is actually given. (As described later, we write as a $(t \to t', n)$-threshold changeable scheme instead of writing as a (t,n)-threshold scheme that is threshold changeable to t'.) In that kind of a threshold changeable scheme, the scheme after the threshold change sacrifices the perfect security, but is an optimal $(t-1, t', n)$-ramp scheme. Furthermore the scheme requires only the share size coinciding with the secret size. Even in changing the threshold N times, this scheme requires the same size share as the secret size.

In Section 3, we define *a* $(t \to \mathbf{t}, n)$*-threshold changeable scheme*, in which the threshold can be changed $N (\geq 1)$ times, where $\mathbf{t} = (t_1, t_2, \ldots, t_N)$ with $t < t_k$ for $1 \leq k \leq N$. Note that in case $N = 1$, that scheme has already been defined by [MPSW99]. Each t_k is the threshold after the threshold is changed k times. The optimal $(t \to t', n)$-threshold changeable scheme given by [MPSW99] can easily be extended to be a $(t \to \mathbf{t}, n)$-threshold changeable scheme.

In this paper, we discuss to make a $(t \to \mathbf{t}, n)$-threshold changeable scheme verifiable. By the technique by [Ped92], we can make a scheme non-interactive and unconditionally secure. The optimal $(t \to t', n)$-threshold changeable scheme given by [MPSW99] is unfortunately based on [Bla79]. Since Pedersen's scheme [Ped92] is based on Shamir's one [Sha79], it cannot directly be applied to that optimal $(t \to \mathbf{t}, n)$-threshold changeable scheme. Then we first construct, based on [Sha79], an *almost optimal* $(t \to \mathbf{t}, n)$-threshold changeable scheme. After that we contrive to make such a scheme verifiable so that the whole scheme required the dealer to publish much less information including the commitment than we simply construct a $(t \to \mathbf{t}, n)$-threshold changeable scheme by combining a $(t \to t_1, n)$-threshold changeable, a $(t \to t_2)$-threshold changeable scheme, ..., and a $(t \to t_N, n)$-threshold changeable scheme, and apply, to the whole scheme, the technique by [Ped92] $(N+1)$ times for the (t,n)-threshold scheme and for each (t_k, n)-threshold scheme $(1 \leq k \leq N)$.

2 Preliminaries

First of all, we review some definitions on secret sharing schemes after giving our notations. Let s be a secret belonging to a set S. The secret s is broken into

n shares $s_1, \ldots, s_n$. Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be the set of participants. We assume that each share s_i is securely distributed to the i-th participant P_i. Let P_i denote also the set of possible shares for the participant P_i. Similarly, we denote, by $\mathcal{A}$, the set of the shares the participants in $\mathcal{A} \subset \mathcal{P}$ hold. We say that a set $\mathcal{A} \subset \mathcal{P}$ of shares can recover the secret s if $H(S|\mathcal{A}) = 0$, where $H(*)$ denotes Shannon's entropy function. Such an $\mathcal{A}$ is called *an access set*. The set consists of all access sets is called *the access structure (of a secret sharing scheme)*.

2.1 Threshold Scheme

A secret sharing scheme which has n participants, and whose access structure is of the form $\{\mathcal{A} \subset \mathcal{P} \mid \#\mathcal{A} \geq t\}$ for some $t(\leq n)$, is called *a (t,n)-threshold (secret sharing) scheme*. In a (t,n)-threshold scheme, we, in general, have the following properties: $H(S|\mathcal{A}) = 0$ if $\#\mathcal{A} \geq t$ and $H(S|\mathcal{A}) > 0$ otherwise.

Definition 1. A (t,n)-threshold scheme is said to be *perfect*, if $H(S|\mathcal{A}) = H(S)$ holds for any set $\mathcal{A} \subset \mathcal{P}$ such that $\#\mathcal{A} < t$. A perfect threshold scheme is said to be *ideal*, if $H(P_i) = H(S)$ holds for any i $(1 \leq i \leq n)$.

We can easily see that Shamir's scheme [Sha79] is perfect and ideal. The following theorem states that there exists the lower bound for the share size in a perfect threshold scheme.

Theorem 2 (in [Sti95]). In a perfect (t,n)-threshold scheme, for any i $(1 \leq i \leq n)$, $H(P_i) \geq H(S)$ holds.

2.2 Ramp Scheme

As we can see in Theorem 2, in a perfect threshold scheme, there exists the lower bound for the share size. That means if $H(P_i) < H(S)$ holds for some i, then the threshold scheme cannot be perfect. As a compromise between security and efficiency, *a ramp scheme* is introduced in [MPSW99].

Definition 3. A (t,n)-threshold scheme is said to be *a (c,t,n)-ramp scheme* if it satisfies the following properties:

$$\begin{cases} H(S|\mathcal{A}) = 0, & \text{if } \#\mathcal{A} \geq t\ ; \\ 0 < H(S|\mathcal{A}) < H(S), & \text{if } c < \#\mathcal{A} < t\ ; \\ H(S|\mathcal{A}) = H(S), & \text{if } \#\mathcal{A} \leq c. \end{cases}$$

In a ramp scheme, each share size can be smaller than the secret size. However the smaller the share size gets, the more the information on the secret is disclosed.

Definition 4. A (c,t,n)-ramp scheme is said to be *optimal*, if it has the property that $H(S|\mathcal{A}) = \dfrac{t-r}{t-c} H(S)$ holds for any $\mathcal{A} \subset \mathcal{P}$ such that $\#\mathcal{A} = r$ and $c \leq r \leq t$.

It is shown by [JM96], that a (c,t,n)-ramp scheme with the property that $H(P_i) = \dfrac{H(S)}{t-c}$ holds for each i $(1 \le i \le n)$ is optimal.

2.3 Threshold Changeable Scheme

In a secret sharing scheme, it often occurs that the access structure should to be changed before the secret is reconstructed. Furthermore the dealer may often be suspended after distributing shares. This is why we need a threshold scheme in which the threshold can be changed without any dealer assistance, and hereafter call such a scheme *a threshold changeable scheme*.

Here in a threshold changeable scheme, the first (t,n)-threshold scheme is denoted by Π, and the derived (t',n)-threshold scheme is denoted by Π'. The whole scheme is denoted by $\langle \Pi, \Pi' \rangle$.

As seen in Definition 5 given above, for a subset $\mathcal{A} \subset \mathcal{P}$, we denote the set of the images of respective elements by h_* by $\mathcal{H}(\mathcal{A})$. That is, for $\mathcal{A} = \{P_{i_1}, \ldots, P_{i_\ell}\} \subset \mathcal{P}$, we define $\mathcal{H}(\mathcal{A})$ as follows:

$$\mathcal{H}(\mathcal{A}) := h_{i_1}(P_{i_1}) \times h_{i_2}(P_{i_2}) \times \cdots \times h_{i_\ell}(P_{i_\ell}).$$

Definition 5 (in [MPSW99]). We say that a perfect (t,n)-threshold scheme is called *threshold changeable to* t', if there exist known functions h_i for $1 \le i \le n$, such that $H(S|\mathcal{H}(\mathcal{A})) = 0$ for any $\mathcal{A} \ge t'$, and $H(S|\mathcal{H}(\mathcal{A})) > 0$ for any $\#\mathcal{A} < t'$ where $\mathcal{A} \subset \mathcal{P}$. (In this paper, we simply write as *a perfect* $(t \to t', n)$*-threshold changeable scheme* instead of a perfect (t,n)-threshold scheme that is threshold changeable to t'.)

In the definitions given above, each known function h_i has to satisfy the property that for any P_i $(1 \le i \le n)$, $H(P_i|h_i(P_i)) > 0$ holds not so that s_i can uniquely figured out from s'_i. In this paper, we call each s_i *a full share* (or *share* simply), and each $h_i(s_i)$ *a subshare*.

Though [TTO99] presents an efficient way to derive Π' from Π both of which are perfect, in that scheme, the functions $\{h_i\}$ do not satisfy the property given above. Hence when the threshold is changed, the corresponding secret also has to be simultaneously changed. Since we need to change not the secret but the threshold, we focus the methods given by [MPSW99]. The method given by the first part of [MPSW99] presents a threshold changeable scheme in which both Π and Π' are perfect. But that method requires each share of a threshold changeable scheme to be quite large. Concrete to say, letting α and β denote the secret size and the share size, respectively, we have $\beta \ge 2\alpha$ holds in such a threshold changeable scheme. Hence as described in the following section, if we extend a threshold changeable scheme so that the threshold can be changed plural times, say $N(\ge 2)$ times, then the required share size β is equal to or greater than $(N+1)$ times of the secret size, i.e. $\beta \geq (N+1)\alpha$. For efficiency of the whole scheme, we aim at a perfect threshold changeable scheme in which Π is ideal as the latter part of [MPSW99] even if the perfect security is lost.

We can easily see that a perfect $(t \to t', n)$-threshold changeable scheme $\langle \Pi, \Pi' \rangle$, in which Π is a (t, n)-threshold scheme and Π' is a (t', n)-threshold scheme, has the property that $H(S|\mathcal{H}(\mathcal{A})) = 0$ if $\#\mathcal{A} \geq t'$ and $H(S|\mathcal{H}(\mathcal{A})) = H(S)$ if $\#\mathcal{A} < t$, since $\#\mathcal{A} < t$ implies $H(S) \geq H(S|\mathcal{H}(\mathcal{A})) \geq H(S|\mathcal{A}) = H(S)$.

2.4 Efficiency Measure

Let $\langle \Pi, \Pi' \rangle$ be a perfect $(t \to t', n)$-threshold changeable scheme. Then the efficiency of such a scheme can be measured by the followings:

(1) The maximum and average size of the share which needs to be stored by participants, and which is denoted by $H(P_i)$ for $1 \leq i \leq n$;
(2) The amount of information which needs to be derivered for reconstruction of the secret at the pooling time, and denoted by $\sum_{i \in \mathcal{A}} H(h_i(P_i))$ for $\mathcal{A} \subset \mathcal{P}$ where $\#\mathcal{A} = t'$;
(3) The size of shares after update of the threshold denoted by $H(h_i(P_i))$ for $1 \leq i \leq n$.

Theorem 6 (in [MPSW99]). Let $\langle \Pi, \Pi' \rangle$ be a perfect $(t \to t', n)$-threshold changeable scheme using functions $\{h_i\}_{1 \leq i \leq n}$. Then the followings hold:

(1) $H(P_i) \geq H(S)$ holds for each i $(1 \leq i \leq n)$;
(2) $\sum_{i \in \mathcal{A}} H(h_i(P_i)) \geq \dfrac{t'}{t'-t+1} H(S)$ holds for every $\mathcal{A} \subset \mathcal{P}$ with $\#\mathcal{A} = t'$;
(3) $\mathsf{max}_{1 \leq i \leq n}\{H(h_i(P_i))\} \geq \dfrac{1}{t'-t+1} H(S)$ holds.

Note that $\mathsf{max}_{1 \leq i' \leq n}\{H(h_{i'}(P_{i'}))\} = H(h_i(P_i))$ for each i $(1 \leq i \leq n)$, if $\{h_i\}$ is common among the participants, and if all P_i's come from the same domain with the same probability.

Definition 7 (in [MPSW99]). We say that a perfect $(t \to t', n)$-threshold changeable scheme that is threshold changeable to t' is *optimal*, if each bound in Theorem 6 is met with equality.

Corollary 8. If a perfect $(t \to t', n)$-threshold changeable scheme $\langle \Pi, \Pi' \rangle$, is optimal, then Π is ideal and then Π' is an optimal $(t-1, t', n)$-ramp scheme.

In addition to the definition given above, we define the slightly loose property of a threshold changeable scheme.

Definition 9. Let $\langle \Pi, \Pi' \rangle$ be a perfect $(t \to t', n)$-threshold changeable scheme using functions $\{h_i\}_{1 \leq i \leq n}$. Then the whole scheme is defined to be *almost optimal* if the following holds:

(1) $H(P_i) = H(S)$ holds for each i $(1 \leq i \leq n)$;
(2) $0 \leq \sum_{i \in \mathcal{A}} H(h_i(P_i)) - \dfrac{t'}{t'-t+1} H(S) \leq c_1$ holds for every $\mathcal{A} \subset \mathcal{P}$ with $\#\mathcal{A} = t'$ and some $c_1 \geq 0$ independent of $H(S)$ or n;
(3) $0 \leq \mathsf{max}_{1 \leq i \leq n}\{H(h_i(P_i))\} - \dfrac{1}{t'-t+1} H(S) \leq c_2$ holds for some $c_2 \geq 0$ which does not depend upon $H(S)$, t, t' or n.

From the definition, we can immediately see that an optimal threshold changeable scheme is an almost optimal one in a special case $c_1 = c_2 = 0$.

2.5 Verifiable Secret Sharing Scheme

A verifiable secret sharing scheme enables each participant to check whether her share given by the dealer is indeed correct, or not, and also the combiner to check whether each pooled share is indeed correct, or not. A verifiable secret sharing scheme is applied as tools for secure multi-party computation and for key management. In this paper, we extend our proposed threshold changeable scheme to be verifiable using the method given by [Ped92], since it provides unconditional security and non-interactivity among the dealer and the participants.

3 Threshold Scheme with N-time Threshold Changeability

In this section, we first extend a perfect $(t \to t', n)$-threshold changeable scheme $\langle \Pi, \Pi' \rangle$ to a perfect $(t \to \mathbf{t}, n)$-threshold changeable scheme $\langle \Pi, \Pi_1, \ldots, \Pi_N \rangle$, where $\mathbf{t} = (t_1, \ldots, t_N)$ with $t < t_k$ for each k $(1 \leq k \leq N)$ and with $t_k \neq t_{k'}$ for $k \neq k'$. In such a scheme, without the dealer assistance, the threshold can be changed one after another, that is, from t to t_1, from t_1 to t_2, and so on[1], under the assumption that the secret has not been recovered before the threshold is changed, and that no share has been pooled. We name each derived (t_k, n)-threshold scheme Π_k. The dealer publishes a set of functions $\{h_i^{(k)}\}_{1 \leq k \leq N}$ so that the participants can compute their subshare for $\{\Pi_k\}_{1 \leq k \leq N}$ by themselves. For a participant P_i given a share s_i, her subshare for Π_k is computed as $h_i^{(k)}(s_i)$. For a set $\mathcal{A} \subset \mathcal{P}$, the set of their subshares for Π_k is denoted by $\mathcal{H}^{(k)}(\mathcal{A})$, that is, we define $\mathcal{H}^{(k)}(\mathcal{A})$ as follows:

$$\mathcal{H}^{(k)}(\mathcal{A}) := h_{i_1}^{(k)}(P_{i_1}) \times h_{i_2}^{(k)}(P_{i_2}) \times \cdots \times h_{i_\ell}^{(k)}(P_{i_\ell}),$$

where $\mathcal{A} = \{P_{i_1}, P_{i_2}, \ldots, P_{i_\ell}\}$. Note that the thresholds $(t_1, \ldots, t_N)$ have to be determined in advance, since we assume that the dealer is suspended after the initial setting of the scheme. Formally, a $(t \to \mathbf{t}, n)$-threshold changeable scheme is defined as follows.

[1] We may regard this kind of scheme as one in which the threshold can be changed to an arbitrary values among $\{t_1, t_2, \ldots, t_N\}$ each of which is, in advance, determined.

Definition 10. Let $\mathbf{t}$ be $(t_1, \ldots, t_N)$ with $t_k > t$ for each k $(1 \leq k \leq N)$. *A* $(t \to \mathbf{t}, n)$*-threshold changeable scheme* is a (t, n)-threshold scheme, in which for $1 \leq i \leq n$ and $1 \leq k \leq N$, there exist known functions $h_i^{(k)}$ such that $H(S|\mathcal{H}^{(k)}(\mathcal{A})) = 0$ for any $\mathcal{A} \geq t_k$, and $H(S|\mathcal{H}^{(k)}(\mathcal{A})) > 0$ for any $\#\mathcal{A} < t_k$ where $\mathcal{A} \subset \mathcal{P}$.

The properties of "*optimal*" and "*almost optimal*", can be defined also for a perfect $(t \to \mathbf{t}, n)$-threshold changeable scheme.

Definition 11. A perfect $(t \to \mathbf{t}, n)$-threshold changeable scheme $\langle \Pi, \Pi_1, \ldots, \Pi_N \rangle$ is said to be *optimal* (or *almost optimal*), if each threshold changeable scheme $\langle \Pi, \Pi_k \rangle$ $(1 \leq k \leq N)$ is optimal (or almost optimal, respectively), and if for distinct k and k', Π_k and $\Pi_{k'}$ are independent of each other, that is, if it holds that $I(h_i^{(k)}(P_i); h_i^{(k')}(P_i)) = 0$ for any k and k' with $k \neq k'$.

The equation $I(h_i^{(k)}(P_i); h_i^{(k')}(P_i)) = 0$ means that the subshare for Π_k gives no information on the subshare for $\Pi_{k'}$. In the following, we construct a perfect $(t \to \mathbf{t}, n)$-threshold changeable scheme $\langle \Pi, \Pi_1, \ldots, \Pi_N \rangle$ based on [Sha79], in which Π is ideal. From now on, we omit the subscript of $h_i^{(k)}$ and write as $h^{(k)}$, since in this paper, $\{h_i^{(k)}\}$ is common among the participants for each k $(1 \leq k \leq N)$. By defining $\{h^{(k)}\}$ as in Section 3.2, the following $(t \to \mathbf{t}, n)$-threshold changeable scheme can be shown to be almost optimal.

3.1 Construction of a Perfect Threshold Changeable Scheme with N-time Threshold Changeability

Let n be the number of participants. For simplicity, we assume that the thresholds t and $\mathbf{t} = (t_1, \ldots, t_N)$ with $t < t_k \leq n$ for $1 \leq k \leq N$, satisfy $(2 \leq) t < t_1 < t_2 < \cdots < t_N \leq n$. Let q be a prime of the length L such that L is a multiple of $\mathtt{lcm}(t_1 - t + 1, \ldots, t_N - t + 1)$. Note that the prime q satisfies $q = 2^L - \varepsilon$ with $\varepsilon < 2^{L-1}$. Then for the secret $s \in \mathbb{Z}_q$, the dealer constructs a perfect $(t \to \mathbf{t}, n)$-threshold scheme as follows:

(i) First the dealer constructs Shamir's (t, n)-threshold scheme for the secret $s \in \mathbb{Z}_q$. That means, the dealer chooses a degree at most $(t-1)$ polynomial $f(x) = a_{0,1}x + a_{0,2}x^2 + \cdots + a_{0,t-1}x^{t-1} \in \mathbb{Z}_q[x]$ with $f(0) = s$. Each (full) share s_i for P_i is defined to be $f(i) \pmod q$.

(ii) The dealer provides N public function $\{h^{(k)}\}_{1 \leq k \leq N}$ such that for all i and k, $H(h^{(k)}(P_i)|P_i) = 0$ and $H(P_i|h^{(k)}(P_i)) > 0$, $(1 \leq i \leq n,\ 1 \leq k \leq N)$. (A concrete example of the set $\{h^{(k)}\}$ is given the following subsection.) For each participant P_i, her subshare $s_i^{(k)}$ for the (t_k, n)-threshold scheme Π_k, is defined by $h^{(k)}(s_i)$.

(iii) To construct Π_1 from Π, the dealer figures out the polynomial $f_1(x)$ for a (t_1, n)-threshold scheme Π_1 using $f(x)$. $f_1(x)$ is of the form:

$$f_1(x) = f(x) + a_{1,t}x^t + a_{1,t+1}x^{t+1} + \cdots + a_{1,t+n-1}x^{t+n-1},$$

where each coefficient $a_{1,j}$ $(t \le j \le t+n-1)$ is found by the n equations $f_1(i) = h^{(1)}(s_i)$ $(1 \le i \le n)$. Here we define as follows:

$$\begin{aligned} f_1^{\mathrm{s}} &:= f(x) + a_t x^t + \cdots + a_{t_1-1} x^{t_1-1}; \\ f_1^{\mathrm{p}} &:= f_1(x) - f_1^{\mathrm{s}}(x). \end{aligned}$$

Then if the polynomial $f_1^{\mathrm{p}}(x)$ is open, the (secret) polynomial $f_1(x)$ can be disclosed by any t_1 subshares from $\{h^{(1)}(i)\}_{1 \le i \le n}$.

(iv) For k $(1 \le k \le N-1)$, to construct Π_{k+1} from Π_k, the dealer figures out the polynomial $f_{k+1}(x)$ for Π_{k+1} using $f_k^{\mathrm{s}}(x)$. $f_{k+1}(x)$ is of the form:

$$f_{k+1}(x) = f_k^{\mathrm{s}}(x) + a_{k+1,t_k} x^{t_k} + \cdots + a_{k+1,t_k+n-1} x^{t_k+n-1},$$

where the n coefficients $a_{k+1,j}$ $(t_k \le j \le t_k+n-1)$ are found by the n equations $f_{k+1}(i) = h^{(k+1)}(s_i)$ $(1 \le i \le n)$. Here we similarly define as follows:

$$\begin{aligned} f_{k+1}^{\mathrm{s}}(x) &:= f_k^{\mathrm{s}}(x) + a_{k+1,t_k} x^{t_k} + \cdots + a_{k+1,t_{k+1}-1} x^{t_{k+1}-1}; \\ f_{k+1}^{\mathrm{p}}(x) &:= f_{k+1}(x) - f_{k+1}^{\mathrm{s}}(x). \end{aligned}$$

(v) The dealer securely distributes each s_i to P_i, and publishes N polynomials $f_1^{\mathrm{p}}(x), \ldots, f_N^{\mathrm{p}}(x)$ and the N functions $h^{(1)}, \ldots, h^{(N)}$ which derive the subshares from shares.

If no threshold changing has happened, the combiner recovers the secret s by gathering any t (full) shares s_{i_j} $(1 \le j \le t)$ as well as in Shamir's scheme. On the other hand, in case that the combiner attempts to recover the secret in the scheme Π_k $(1 \le k \le N)$, she gathers any t_k subshares $s_{i_\ell}^{(k)}$ $(1 \le \ell \le t_k)$. Then the secret s can be figured out by the following formula which resembles so-called Lagrange polynomial interpolation:

$$s = \sum_{j=1}^{t_k} \left(s_{i_j}^{(k)} - f_k^{\mathrm{p}}(i_j) \right) \prod_{\substack{1 \le \ell \le t_k \\ \ell \ne j}} \frac{\ell}{\ell - j}.$$

Note that in the scheme given above, Π_1 is constructed using Π, and each Π_k $(2 \le k \le N)$ is constructed using Π_{k-1}. On the other hand, we can also construct $\langle \Pi, \Pi_1, \ldots, \Pi_N \rangle$ by the way that every Π_k $(1 \le k \le N)$ is constructed using Π, not using the previous Π_{k-1}. Such a scheme is, however, less efficient in the viewpoint of the amount of information the dealer has to publish, than the scheme we have just constructed in this subsection. We show the detail in Section 5.

3.2 Example of the Functions $\{h^{(k)}\}$

As far as we construct the scheme given in the previous subsection, we cannot make any $\langle \Pi, \Pi_k \rangle$ $(1 \le k \le N)$ exactly optimal. If we constructed the scheme on a field $\mathbb{Z}_{q'}^{\alpha}$ with a prime q' and α being a multiple of $\mathtt{lcm}(t_1-t+1,\ldots,t_N-t+1)$, then we could make each $\langle \Pi, \Pi_k \rangle$ exactly optimal. But in that case, we cannot efficiently apply the technique by [Ped92] to that threshold changeable scheme. In a $(t \to \mathbf{t}, n)$-threshold changeable scheme, if Π is ideal, then the possible frequency N of threshold changing is restricted as the following proposition states:

Proposition 12. In a $(t \to \mathbf{t}, n)$-threshold changeable scheme $\langle \Pi, \Pi_1, \ldots, \Pi_N \rangle$, if Π is ideal, and if the whole scheme is (almost) optimal, then the possible frequency N of the possible thresholds satisfy $\sum_{k=1}^{N} 1/(t_k - t + 1) \le 1$.

Proof. Since Π is ideal, we have the following:

$$\begin{aligned} H(S) &= H(P_i) \;\ge\; H(P_i^{(1)}) + \cdots + H(P_i^{(N)}) \\ &\ge \left(\frac{1}{t_1 - t + 1} + \cdots + \frac{1}{t_N - t + 1} \right) H(S), \end{aligned}$$

for each i $(1 \le i \le n)$, which is what we claim. ∎

For example in case $t_1 = t+1$, $t_2 = t+2$ and $t_3 = t+5$, since $\sum_{k=1}^{3} 1/(t_k - t+1) = 1$, the correlation yields among $\{P_i^{(k)}\}$ if the threshold is changed more than four times. Hereafter we implicitly assume that for the set of the thresholds $\{t, t_1, \ldots, t_N\}$ and the number N of the threshold changing satisfy the statement of the previous proposition.

Now we define the functions $h^{(k)}$ $(1 \le k \le N)$ as follows. Note that q is of the length L and that L is a multiple of $\mathtt{lcm}(t_1 - t + 1, \ldots, t_N - t + 1)$.

- For an element $x \in \mathbb{Z}_q$, $h^{(1)}(x)$ is the substring of x from the first (rightmost) bit to the $(L/(t_1 - t + 1))$-th bit. That is, for $x \in \mathbb{Z}_q$, we define $h^{(1)}(x) := x \pmod{2^{L/(t_1-t+1)}}$.
- Define T_k to be $\sum_{\ell=1}^{k} 1/(t_\ell - t + 1)$. For an element $x \in \mathbb{Z}_q$ and k $(2 \le k \le N)$, $h^{(k)}(x)$ is the substring of x from the $(1 + LT_{k-1})$-th bit to (LT_k)-th bit. That is, for $x \in \mathbb{Z}_q$ and k $(2 \le k \le N)$, we define $h^{(k)}(x) := \left\lceil \frac{x}{2^{LT_{k-1}}} \right\rceil \pmod{2^{L/(t_k-t+1)}}$.

In the following, we show that the proposed $(t \to \mathbf{t}, n)$-threshold changeable scheme using functions $\{h^{(k)}\}$ given above, is almost optimal.

Proposition 13. The $(t \to \mathbf{t}, n)$-threshold changeable scheme $\langle \Pi, \Pi_1, \ldots, \Pi_N \rangle$ in Section 3.1, is almost optimal, if it uses the functions $\{h^{(k)}\}$ given above.

Proof. We will prove that our scheme satisfies the conditions in Definition 11, that is, the conditions (1), (2) and (3) in Definition 9 and the condition that for k and k' with $k \ne k'$, Π_k and $\Pi_{k'}$ are independent of each other.

The first condition (1) follows immediately from the fact that Π is just Shamir's scheme.

Next we show the third condition. Since we suppose that q is a prime such that $q = 2^L - \varepsilon$ with $\varepsilon < 2^{L-1}$, then we have the following:

$$H(S) = \log(2^L - \varepsilon) \geq \log(2^L - 2^{L-1}) = L - 1.$$

Furthermore from $H(h^{(k)}(P_i)) \leq L/(t_k - t + 1)$ for each k $(1 \leq k \leq N)$, we can get the following:

$$0 \leq H(h^{(k)}(P_i)) - \frac{H(S)}{t_k - t + 1} \leq \frac{L}{t_k - t + 1} - \frac{L-1}{t_k - t + 1} = \frac{1}{t_k - t + 1} \leq \frac{1}{2}.$$

Hence the third condition is satisfied. The second one is immediately obtained from the third one.

Finally, the last condition that $I(h^{(k)}(P_i); h^{(k')}) = 0$ for each k, k' with $k \neq k'$, comes from the fact that for any $x \in \mathbb{Z}_q$, the strings $h^{(k)}(x)$ and $h^{(k')}(x)$ are indeed disjoint. ∎

4 Efficient VSS for $(t \rightarrow \mathbf{t}, n)$-Threshold Changeable Scheme

In this section, we make the $(t \rightarrow \mathbf{t}, n)$-threshold scheme $\langle \Pi, \Pi_1, \ldots, \Pi_N \rangle$ verifiable. Denote, by Π^v, the verifiable (t, n)-threshold scheme derived by making Π verifiable. Also for each k $(1 \leq k \leq N)$, denote, by Π_k^v, the verifiable (t_k, n)-threshold scheme derived by making Π_k verifiable. To provide the unconditional security and non-interactivity among the entities for verification, we adopt Pedersen's technique [Ped92]. Of course, by constructing Π and Π_k's independently and by applying that technique to Π and each Π_k, we can accomplish our purpose, but here we contrive to make the amount of information the dealer has to publish, by applying [Ped92] to the very $(t \rightarrow \mathbf{t}, n)$-threshold changeable scheme given in the previous section.

How to set up the parameters q, t, t_k $(1 \leq k \leq N)$, N and $\{h^{(k)}\}$ $(1 \leq k \leq N)$ is exactly the same as the previous section. In addition to those parameters, we let p be a prime such that q divides $p - 1$ and such that $q^2 < p$ holds[2], and let α and β be order-q elements in $\mathbb{Z}_p^*$. Those two bases α and β should be randomly picked up by the dealer, or should be chosen by some trusted third party, not so that $\log_\alpha \beta$ may be known to any entities joining the scheme. Note that for s and u belonging to $\mathbb{Z}_q$, the dealer can find another pair $(s', u') \in \mathbb{Z}_q \times \mathbb{Z}_q$ such that $\alpha^s \beta^u = \alpha^{s'} \beta^{u'} \pmod p$ if and only if she knows the discrete logarithm $\log_\alpha \beta$ under the modulo p.

In the following, we describe how to construct an almost optimal $(t \rightarrow \mathbf{t}, n)$-threshold changeable scheme with verifiability.

[2] Usually we let p and q be a 1024-bit prime and a 160-bit prime, respectively. Hence this assumption $q^2 < p$ restricts quite little for p and q.

(i) First the dealer constructs a perfect and verifiable (t, n)-threshold scheme Π just like [Ped92]. That means for a secret $s \in \mathbb{Z}_q$, the dealer randomly picks up a degree at most $(t-1)$ polynomial $f(x) \in \mathbb{Z}_q[x]$ such that $f(0) = s$, and also picks up a random $u \in \mathbb{Z}_q$ and a degree at most $(t-1)$ polynomial $g(x) \in \mathbb{Z}_q[x]$ such that $g(0) = u$. The full share for P_i is defined by $f(i)$. Also u_i is defined by $g(i)$ and called *a twin share* for P_i. Here let $f(x) = s + a_{0,1}x + \ldots + a_{0,t-1}x^{t-1}$ and $g(x) = u + b_{0,1}x + \ldots + b_{0,t-1}x^{t-1}$. The commitments $E_0, E_1, \ldots, E_{t-1}$ for $(s, u), (a_{0,1}, b_{0,1}), \ldots, (a_{0,t-1}, b_{0,t-1})$ are defined by $E_0 := E(s, u)$ and $E_j := E(a_{0,j}, b_{0,j})$ $(1 \leq j \leq t-1)$, where for $x, y \in \mathbb{Z}_q$, $E(x, y) := \alpha^x \beta^y \pmod{p}$.

(ii) For each i and k $(1 \leq i \leq n, 1 \leq k \leq N)$, the dealer computes $s_i^{(k)}$ and $u_i^{(k)}$ defined by $h^{(k)}(s_i)$ and $h^{(k)}(u_i)$, respectively. Each $s_i^{(k)}$ and each $u_i^{(k)}$ are called a subshare and *a twin subshare*, respectively.

(iii) To construct Π_1^v from Π^v, the dealer figures out the polynomials $f_1(x)$ and $g_1(x)$ of the form:

$$\begin{aligned} f_1(x) &= f(x) + a_{1,t}x^t + \cdots + a_{1,t+n-1}x^{t+n-1}; \\ g_1(x) &= g(x) + b_{1,t}x^t + \cdots + b_{1,t+n-1}x^{t+n-1}, \end{aligned}$$

where the n coefficients $a_{1,j}$ and the n coefficients $b_{1,j}$ $(t \leq j \leq t+n-1)$ are determined by the n equations $f_1(i) = s_i^{(1)}$ and by the n equations $g_1(i) = u_i^{(1)}$, respectively. Here we define as follows:

$$\begin{aligned} f_1^{\mathrm{s}}(x) &:= f(x) + a_{1,t}x^t + \cdots + a_{1,t_1-1}x^{t_1-1}; \\ f_1^{\mathrm{p}}(x) &:= f_1(x) - f_1^{\mathrm{s}}(x). \end{aligned}$$

Similarly we define $g_1^{\mathrm{s}}(x) := g(x) + b_{1,t}x^t + \cdots + b_{1,t_1-1}x^{t_1-1}$ and $g_1^{\mathrm{p}}(x) := g_1(x) - g_1^{\mathrm{s}}(x)$. For each j $(t \leq j \leq t_1 - 1)$, the commitment E_j for $(a_{1,j}, b_{1,j})$ is defined by $E(a_{1,j}, b_{1,j})$.

(iv) For k $(1 \leq k \leq N-1)$, to construct Π_{k+1}^v from Π_k^v, the dealer figures out the polynomials $f_{k+1}(x)$ and $g_{k+1}(x)$ using $f_k^{\mathrm{s}}(x)$ and $g_k^{\mathrm{s}}(x)$, respectively. $f_{k+1}(x)$ and $g_{k+1}(x)$ are of the form:

$$\begin{aligned} f_{k+1}(x) &= f_k^{\mathrm{s}}(x) + a_{k+1,t_k}x^{t_k} + \cdots + a_{k+1,t_k+n-1}x^{t_k+n-1}; \\ g_{k+1}(x) &= g_k^{\mathrm{s}}(x) + b_{k+1,t_k}x^{t_k} + \cdots + b_{k+1,t_k+n-1}x^{t_k+n-1}, \end{aligned}$$

where the n coefficients $a_{k+1,j}$ and the n coefficients $b_{k+1,j}$ $(t_k \leq j \leq t_k + n - 1)$ are determined by the n equations $f_{k+1}(i) = s_i^{(k+1)}$ and by the n equations $g_{k+1}(i) = u_i^{(k+1)}$, respectively. Here we define as follows:

$$\begin{aligned} f_{k+1}^{\mathrm{s}}(x) &:= f_k^{\mathrm{s}}(x) + a_{k+1,t_k}x^{t_k} + \cdots + a_{k+1,t_{k+1}-1}x^{t_{k+1}-1}; \\ f_{k+1}^{\mathrm{p}}(x) &:= f_{k+1}(x) - f_{k+1}^{\mathrm{s}}(x). \end{aligned}$$

Similarly we define $g^{\mathrm{s}}_{k+1}(x) := g^{\mathrm{s}}_k(x) + b_{1,t_k}x^t + \cdots + b_{k+1,t_{k+1}-1}x^{t_{k+1}-1}$ and $g^{\mathrm{p}}_{k+1}(x) := g_{k+1}(x) - g^{\mathrm{s}}_{k+1}(x)$. For each j $(t_k \le j \le t_{k+1} - 1)$, the commitments E_j for $(a_{k+1,j}, b_{k+1,j})$ are defined by $E(a_{k+1,j}, b_{k+1,j})$.

(v) The dealer securely distributes each (s_i, u_i) to P_i, and publishes the $2n$ polynomials $\{f^{\mathrm{p}}_k(x)\}_{1\le k\le N}$ and $\{g^{\mathrm{p}}_k(x)\}_{1\le k\le N}$, $\{h^{(k)}\}_{1\le k\le N}$ and the commitments $\{E_j\}_{0\le j\le N}$.

Each participant P_i given (s_i, u_i) can verify whether her share and twin share are correct, or not, by the following verification:

$$E(s_i, u_i) = \prod_{j=0}^{t-1} E_j^{i^j} \pmod{p},$$

and also can, for each k $(1 \le k \le N)$, verify whether each pair $(s_i^{(k)}, u_i^{(k)})$ of her subshares and twin subshares is correct, or not, by the following verification:

$$E(s_i^{(k)} - f^{\mathrm{p}}_k(i), u_i^{(k)} - g^{\mathrm{p}}_k(i)) = \prod_{j=0}^{t_k-1} E_j^{i^j} \pmod{p}.$$

In recovering the secret, the combiner can similarly verify whether the full shares or the subshares she has gathered, are correct, or not, by the verification given above.

5 Efficiency of the Proposed Scheme

In this section, we estimate the efficiency of the proposed verifiable threshold changeable scheme with N-time threshold changeability. For simple description, we name the various types of the schemes as follows:

`Scheme-I`: A verifiable $(t \to \mathbf{t}, n)$-threshold changeable scheme $\langle \Pi^v, \Pi^v_1, \ldots, \Pi^v_N \rangle$ in which Π and all Π_k $(1 \le k \le N)$ are independently constructed by using Shamir's method, and in which Pedersen's technique is independently applied to Π and each Π_k.

`Scheme-II`: A verifiable $(t \to \mathbf{t}, n)$-threshold changeable scheme $\langle \Pi^v, \Pi^v_1, \ldots, \Pi^v_N \rangle$ in which each $(t \to t_k, n)$-threshold changeable scheme $\langle \Pi, \Pi_k \rangle$ $(1 \le k \le N)$ is independently constructed, and in which Pedersen's technique is independently applied to each $\langle \Pi, \Pi_k \rangle$.

`Scheme-III`: The proposed verifiable $(t \to \mathbf{t}, n)$-threshold changeable scheme $\langle \Pi^v, \Pi^v_1, \ldots, \Pi^v_N \rangle$ we have constructed in Section 4.

In the following, we show the efficiency for the dealer. Precisely, we, in Figure 1, show the amount of information she has to securely distributed and the amount of information she has to publish, in `Scheme-I`, in `Scheme-II` and in `Scheme-III`, respectively. As seen in Figure 1, to be sure that `Scheme-I` is su-

Scheme	By-SC $(\times H(S))$	COP $(\times \log q)$	Commitment $(\times \log p)$	Security
I	$2(N+1)$	0	$t+\sum_{k=1}^{N}(t_k-1)$	Π : perfect (t,n)-TS Π_k : perfect (t_k,n)-TS
II	2	$2\sum_{k=1}^{N}(n-t_k+1)$	$t+\sum_{k=1}^{N}(t_k-1)$	Π : perfect (t,n)-TS Π_k : $(t-1,t_k,n)$-RS
III	2	$2(nN+t-t_N)$	$\mathsf{max}_{1\le k\le N} t_k$	Π : perfect (t,n)-TS Π_k : $(t-1,t_k,n)$-RS

`By-SC` : The amount of information per one participant, which the dealer has to distribute by some secure channel.
`COP` : The amount of information of the coefficients of the open polynomials $\{f_k^{\mathrm{P}}(x)\}$ and $\{g_k^{\mathrm{P}}(x)\}$, which the dealer has to publish to control the thresholds.
`Commitment` : The amount of information of the commitments, which the dealer has to open for verification of the full shares and the subshare.
`Security` : The security of the schemes $\Pi, \Pi_1, \ldots, \Pi_N$ as threshold schemes. The terms "TS" and "RS" stand for "threshold scheme" and "ramp scheme", respectively.

Fig. 1. Comparison of the efficiency of `Scheme-I,II,III`

perior to the others in view of the security of each Π_k, but that scheme requires much more amount of information to be securely distributed. Since in `Scheme-II` and `Scheme-III`, such amount does not depend upon the number of the frequency of the threshold changing, we discuss `Scheme-II` and `Scheme-III`.

Denote, by A_{II} and A_{III}, the total amount of information the dealer has to publish in `Scheme-II` and in `Scheme-III`, respectively.

Proposition 14. Suppose that $t \geq 2$, $t_k > 2$ $(1 \leq k \leq N)$ and p, q are prime such that $q|(p-1)$ and such that $q^2 < p$. Then $A_{\mathrm{III}} < A_{\mathrm{II}}$ holds. That means `Scheme-III` is more efficient than `Scheme-III` in view of the amount of information the dealer has to publish.

Proof. First note that we may let $\mathsf{max}_{1\le k\le N} t_k = t_N$ without loss of generality. From the definition, we have

$$\begin{aligned} A_{\mathrm{II}} &= \left(t+\sum_{k=1}^{N}(t_k-1)\right)\log p + 2\left(\sum_{k=1}^{N}(n-t_k+1)\right)\log q; \\ A_{\mathrm{III}} &= t_N \log p + 2(nN+t-t_N)\log q. \end{aligned}$$

Then we can get the following:

$$A_{\mathrm{II}} - A_{\mathrm{III}} = \left(t+\sum_{k=1}^{N-1} t_k - N\right)\left(\log p - \log q^2\right),$$

which is necessarily positive, since $p > q^2$ and $t + \sum_{k=1}^{N-1} t_k - N > 2N - N = N > 0$. ∎

6 Conclusion

Remember that a $(t \to \mathbf{t}, n)$-threshold changeable scheme simply constructed by an optimal (t, n)-threshold scheme that is threshold changeable to t' given by [MPSW99], cannot be efficiently made verifiable by the technique [Ped92]. Then in this paper, we have constructed a $(t \to \mathbf{t}, n)$-threshold changeable scheme $\langle \Pi, \Pi_1, \ldots, \Pi_N \rangle$ based on Shamir's threshold scheme. This is an almost optimal $(t \to \mathbf{t}, n)$-threshold changeable scheme, and can be easily made a verifiable $(t \to \mathbf{t}, n)$-threshold changeable scheme with unconditional security and non-interactivity among the entity for verification. As seen in the primitive one (`Scheme-I`) in Figure 1, the perfect security of each Π_k $(1 \leq k \leq N)$ requires much more size full shares to be securely distributed. On the other hand, though in the proposed scheme (that is, `Scheme-III`), each scheme Π_k $(1 \leq k \leq N)$ sacrifices the perfect security, the entropy of the full share does not depend upon the number of the frequency of the threshold changing. Furthermore we decrease the amount of information the dealer has to publish by constructing Π_k using Π_{k-1} $(1 \leq k \leq N)$, where $\Pi_0 := \Pi$. This difference is indicated by the inequality $A_{\text{II}} - A_{\text{III}} > 0$ appearing Proposition 14 in Section 5.

References

[Bla79] G. R. Blakley: "*Safeguarding cryptographic keys*", Proceedings of AFIPS 1979, National Computer Conference, vol.48, pp.313-317, 1979.

[JM96] W. A. Jackson and K. M. Martin: "*A combinatorial interpretation of ramp schemes*", Australasian Journal of Combinatorics 14, pp.51-60, 1996.

[MPSW99] K. M. Martin, J. Pieprzyk, R. Safavi-Naini and H. Wang: "*Changing thresholds in the absence of secure channels*", Proceedings of ACISP'99, Lecture Notes in Computer Science 1587, pp.177-191, 1999.

[Ped92] T. P. Pedersen: "*Non-interactive and information theoretic secure verifiable secret sharing*", Advances in cryptology - Crypto'92, Lecture Notes in Computer Science 576, pp.129-140, 1992.

[Sha79] A. Shamir: "*How to share a secret*", Communications of ACM, pp.612-613, 1979.

[Sti95] D. R. Stinson: Cryptography - Theory and Practice, The CRC press series on discrete mathematics and its applications, CRC Press, 1995.

[TTO99] Y. Tamura, M. Tada and E. Okamoto: "*Update of access structure in Shamir's (k, n) threshold scheme*", Proceedings of The 1999 Symposium on Cryptography and Information Security (SCIS'99), vol.I, pp.469-474, 1999.

Provably Secure Distributed Schnorr Signatures and a (t, n) Threshold Scheme for Implicit Certificates

Douglas R. Stinson[1,2] and Reto Strobl[3]

[1] Certicom Corporation
5520 Explorer Drive
Mississauga ON, L4W 5L1, Canada
[2] Department of Combinatorics and Optimization
University of Waterloo
Waterloo Ontario, N2L 3G1, Canada
dstinson@cacr.math.uwaterloo.ca
[3] Department of Computer Science
Swiss Federal Institute of Technology
CH-9478 Zurich, Switzerland
rstrobl@iiic.ethz.ch

Abstract. In a (t, n) threshold digital signature scheme, t out of n signers must co-operate to issue a signature. We present an efficient and robust (t, n) threshold version of Schnorr's signature scheme. We prove it to be as secure as Schnorr's signature scheme, i.e., existentially unforgeable under adaptively chosen message attacks. The signature scheme is then incorporated into a (t, n) threshold scheme for implicit certificates. We prove the implicit certificate scheme to be as secure as the distributed Schnorr signature scheme.

1 Introduction

Threshold Cryptography. Threshold cryptography addresses the issue of performing cryptographic tasks such as signing, encrypting/decrypting etc. in a distributed way. For example, in a (t, n) threshold signature scheme, any set of t players can issue a signature for an arbitrary message while any set of less than t players cannot issue a signature at all (see [8] for a threshold DSS signature scheme). We refer to [4, 9] for a detailed survey of threshold cryptography.

Certificates. Certificates provide a way to authenticate data, usually a public key. They can be realized using different techniques.

Traditional certificates contain an identity string, the public key, and a signature on these values. To issue a traditional certificate, a Certification Authority (CA) first verifies the authenticity of the public key and the identity string and then issues a digital signature on it. The certificate is therefore as secure as the signature scheme: Certificates cannot be forged because signatures cannot be forged.

Implicit certificates contain an identity string, but neither an explicit public key nor a signature. Instead, they contain public reconstruction data. The public key itself must be computed from the public reconstruction data and the public key of the *CA* who issued the certificate. The advantage of implicit certificates is their size: Besides the identity string, they only contain public reconstruction data, whereas traditional certificates contain a public key *and* a digital signature. This is particularly useful in bandwidth constrained environments such as wireless communication and digital postmarks. A survey of various types of implicit certificates and applications is given in [13].

In contrast to traditional certificates, where the security lies directly on the underlying signature scheme, there are special security issues concerning implicit certificates. In particular, any public reconstruction data and identity string together with a *CA*'s public key, would yield a public key. However, it should be hard to choose the public reconstruction data and compute the private key corresponding to the implied public key, without knowing the *CA*'s private key. Another issue is that — since one usually uses a slightly modified signature scheme to issue a certificate — one has to make sure that no information about the *CA*'s or the user's private key is leaked.

Our Work. We first present a distributed Schnorr signature scheme and prove it to be as secure as the non distributed version, i.e., existentially unforgeable under adaptively chosen message attacks. Second, this scheme is incorporated into the construction of a distributed implicit certificate scheme.

In all proofs, we assume the random oracle model as described in [1]. For all protocols we assume a synchronous communication model, where all players are connected via private channels and a global broadcast channel.

Organization. Our digital signature threshold scheme is based on Pedersen's Verifiable Secret Sharing scheme [12] and a multi-party protocol to generate a random shared secret [7]. These primitives are briefly discussed in Section 2. In Section 3 we recall Schnorr's signature scheme [15]. Then we propose in Section 4 a (t, n) threshold version of this signature scheme. We prove the security of the scheme in Section 5, adapting the proof techniques used in [10]. The non-distributed implicit certificate scheme is introduced in Section 6. The (t, n) threshold version of this scheme is presented in Section 7, and a security proof is presented in Section 8.

2 Secret Sharing Schemes

2.1 Parameters

We use elliptic curve notation for the discrete logarithm problem. Suppose q is a large prime and G, H are generators of a subgroup of order q of an elliptic curve E. We assume that E is chosen in such a way that the discrete logarithm problem in the subgroup generated by G is hard, so it is infeasible to compute the integer d such that $G = dH$.

2.2 Shamir's Secret Sharing Scheme

In a (t, n) secret sharing scheme, a dealer distributes a secret s to n players $P_1, \ldots, P_n$ in such a way that any group of at least t players can reconstruct the secret s, while any group of less than t players do not get any information about s. In [16], Shamir proposes a (t, n) threshold secret sharing scheme as follows. In order to distribute $s \in Z_q$ among $P_1, \ldots, P_n$ (where $n < q$), the dealer chooses a random polynomial $f(\cdot)$ over Z_q of degree at most $t-1$ satisfying $f(0) = s$. Each player P_i receives $s_i = f(i)$ as his share.

There is one and only one polynomial of degree at most $t-1$ satisfying $f(i) = s_i$ for t values of i. Therefore, an arbitrary group $\mathcal{P}$ of t participants can reconstruct the polynomial $f(\cdot)$ by Lagrange's interpolation as follows:

$$f(u) = \sum_{i \in \mathcal{P}} f(i)\omega_i(u) \ , \ \text{where} \ \ \omega_i(u) = \prod_{\substack{j \in \mathcal{P} \\ j \neq i}} \frac{u-j}{i-j} \mod q.$$

Since it holds that $s = f(0)$, the group $\mathcal{P}$ can reconstruct the secret as follows:

$$s = f(0) = \sum_{i \in \mathcal{P}} f(i)\omega_i \ , \ \text{where} \ \ \omega_i = \omega_i(0) = \prod_{\substack{j \in \mathcal{P} \\ j \neq i}} \frac{j}{j-i} \mod q.$$

Each ω_i is non-zero and can be easily computed from public information. Note that the constant term of a polynomial of degree at most $t-1$ is not defined through $t-1$ equations of the form $f(i) = s_i$. Furthermore, since the dealer chooses $f(\cdot)$ uniformly at random, every value for the constant term is equally probable. A coalition of $t-1$ players can therefore neither compute the secret nor get any information about it.

2.3 Verifiable Secret Sharing Scheme

A Verifiable Secret Sharing scheme (VSS) prevents the dealer from cheating. In a VSS scheme, each player can verify his share. If the dealer distributes inconsistent shares, he will be detected. Pedersen presented a VSS scheme in [11] which we will use in this paper. His scheme is as follows.

Assume the dealer has a secret $s \in Z_q$ and a random number $s' \in Z_q$, and is committed to the pair (s, s') through public information $C_0 = sG + s'H$. The secret s can be shared among $P_1, \ldots, P_n$ as follows.

The dealer performs the following steps

1. Choose random polynomials

$$f(u) = s + f_1 u + \cdots + f_{t-1}u^{t-1} \text{ and } f'(u) = s' + f'_1 u + \cdots + f'_{t-1}u^{t-1}$$

where $s, s', f_k, f'_k \in Z_q$ for $k \in \{1, \ldots, t-1\}$. Compute $(s_i, s'_i) = (f(i), f'(i))$ for $i \in \{1, \ldots, n\}$.
2. Send (s_i, s'_i) secretly to player P_i for $i \in \{1, \ldots, n\}$.
3. Broadcast the values $C_k = f_k G + f'_k H$ for $k \in \{1, \ldots, t-1\}$.

Each player P_i performs the following steps

1. Verify that
$$s_i G + s'_i H = \sum_{k=0}^{t-1} i^k C_k. \tag{1}$$
If this is false, broadcast a *complaint* against the dealer.
2. For each complaint from a player i, the dealer defends himself by broadcasting the values $f(i), f'(i)$ that satisfy (1).
3. Reject the dealer if
   - he received at least t complaints in step 1, or
   - he answered to a complaint in step 2 with values that violate (1).

Pedersen proved that any coalition of less than t players cannot get any information about the shared secret, provided that the discrete logarithm problem in E is hard.

2.4 Generating a Random Secret

For the key generation phase of our scheme, it is necessary to generate a random shared secret in a distributed way. The early protocol proposed by Feldman [5] has been shown to have a security flaw, and a secure protocol has been proposed in [7], which we will use for our schemes. We recall it in the following.

Suppose a trusted dealer chooses r, r' at random, broadcasts $Y = rG$ and then shares r among the players P_i using Pedersen's VSS scheme. We would like to achieve this situation without a trusted dealer. This can be achieved by the following protocol.

Each player P_i performs the following steps

1. Each player P_i chooses $r_i, r'_i \in Z_q$ at random and verifiably secret shares (r_i, r'_i), acting as the dealer according to Pedersen's VSS scheme. Let the sharing polynomials be $f_i(u) = \sum_{k=0}^{t-1} a_{ik} u^k, f'_i(u) = \sum_{k=0}^{t-1} a'_{ik} u^k$, where $a_{i0} = r_i, a'_{i0} = r'_i$, and let the public commitments be $C_{ik} = a_{ik}G + a'_{ik}H$ for $k \in \{0, \ldots, t-1\}$.
2. Let H_0 be the index set of players not detected to be cheating at step 1. The distributed secret value r is not explicitly computed by any player, but it equals $r = \sum_{i \in H_0} r_i$. Each player P_i sets his share of the secret as $s_i = \sum_{j \in H_0} f_j(i) \bmod q$, and the value $s'_i = \sum_{j \in H_0} f'_j(i) \bmod q$.
3. Extracting $Y = \sum_{j \in H_0} r_j G$: Each player in H_0 exposes $Y_i = s_i G$ via Feldman's scheme:
   (a) Each player P_i for $i \in H_0$ broadcasts $A_{ik} = a_{ik}G$ for $k \in \{0, \ldots, t-1\}$.
   (b) Each player P_j verifies the values broadcast by the other players in H_0. In particular, every player P_i for $i \in H_0$, P_j checks if
$$f_i(j)G = \sum_{k=0}^{t-1} j^k A_{ik}. \tag{2}$$

If the check fails for an index i, P_j *complains* against P_i by broadcasting the values $f_i(j), f'_i(j)$ that satisfy (1) but do not satisfy (2).

(c) For players P_i who received at least one valid complaint, i.e., values which satisfy (1) but do not satisfy (2), the other players run the reconstruction phase of Pedersen's VSS scheme to compute $r_i, f_i(\cdot), A_{ik}$ for $k = 0, \ldots, t-1$ in the clear[1]. All players in H_0 set $Y_i = r_iG$.

After executing this protocol, the following equations hold:

$$\begin{aligned} Y &= rG \\ f(u) &= r + a_1u + \cdots + a_{t-1}u^{t-1}, \text{ where } a_k = \sum_{j \in H_0} a_{jk} \text{ for } k \in \{1, \ldots, t-1\} \\ f(j) &= s_j \text{ for } j \in H_0. \end{aligned}$$

In [7] this scheme has been proven to be robust under the assumption that $t \leq \frac{n}{2}$, i.e., if less than t players are corrupted, the values computed by the honest players satisfy the above equations.

For convenience, we introduce the following notation for this protocol:

$$(s_1, \ldots, s_n) \overset{(t,n)}{\longleftrightarrow} (r|Y, a_kG, H_0), \ k \in \{1, \ldots, t-1\}.$$

This notation means that s_j is player P_j's share of the secret r for each $j \in H_0$. The values a_kG are the public commitments of the sharing polynomial $f(\cdot)$ (they can be computed using public information), and (r, Y) forms a *key pair*, i.e., r is a private key and Y is the corresponding public key. The set H_0 denotes the set of players that have not been detected to be cheating. In the further protocols, we do not need the values Y_j, C_{jk}, s'_j, r' for $j \in H_0, k \in \{0, \ldots, t-1\}$ and therefore we omit these values in the short notation.

3 Schnorr's Signature Scheme

In [14], Schnorr introduced the following signature scheme. Let (x, Y) be a user's key pair, let m be a message, let $h(\cdot)$ be a one-way hash function, and let G be a generator of an elliptic curve group having prime order q. Then a user generates a Schnorr signature on the message m as follows.

1. Select $e \in Z_q$ at random
2. Compute $V = eG$
3. Compute $\sigma = e + h(m||V)x \bmod q$
4. Define the signature on m to be (V, σ)

A verifier accepts a signature (V, σ) on a message m if and only if $\sigma \in Z_q$ and

$$\sigma G = V + h(m||V)Y.$$

[1] Every player in H_0 simply reveals his share of r_i. Each player can then compute r_i by choosing t shares that satisfy (1).

In [14], Schnorr signatures were shown to be existentially unforgeable under adaptively chosen message attacks in the random oracle model, using the forking lemma, provided that the discrete logarithm problem is hard in the group generated by G.

4 A (t, n) Threshold Signature Scheme

In this section, we propose a robust and efficient (t, n) threshold digital signature scheme for Schnorr signatures. We use the primitives presented in Section 2.

Our protocol consists of a key generation protocol and a signature issuing protocol. Let $P_1, \ldots, P_n$ be the set of players issuing a signature and let G be a generator of an elliptic curve group of order q.

4.1 Key Generation Protocol

All n players have to co-operate to generate a public key, and a secret key share for each P_j. They generate a random shared secret according to the protocol presented in Section 2.4. Let the output of the protocol be

$$(\alpha_1, \ldots, \alpha_n) \overset{(t,n)}{\longleftrightarrow} (x|Y, b_kG, H_0),\ \ k \in \{1, \ldots, t-1\}.$$

For each $j \in H_0$, α_j is the secret key share of P_j and will be used to issue a partial signature for the key pair (x, Y).

4.2 Signature Issuing Protocol

Let m be a message and let $h(\cdot)$ be a one-way hash function. Suppose that the players with index set $H_1 \subseteq H_0$ wants to issue a signature. They use the following protocol:

1. If $|H_1| < t$, stop. Otherwise, the subset H_1 generates a random shared secret as described in Section 2.4. Let the output be

$$(\beta_1, \ldots, \beta_n) \overset{(t,n)}{\longleftrightarrow} (e|V, c_kG, H_2),\ \ k \in \{1, \ldots, t-1\}.$$

2. If $|H_2| < t$, stop. Otherwise, each P_i for $i \in H_2$ reveals

$$\gamma_i = \beta_i + h(m||V)\alpha_i.$$

3. Each P_i for $i \in H_2$ verifies that

$$\gamma_j G = V + \sum_{k=1}^{t-1} c_k j^k G + h(m||V)\left(Y + \sum_{k=1}^{t-1} b_k j^k G\right) \text{ for all } j \in H_2. \quad (3)$$

Let H_3 be the index set of players not detected to be cheating at step 3.

4. If $|H_3| < t$, then stop. Otherwise, each P_i for $i \in H_3$ selects an arbitrary subset $H_4 \subseteq H_3$ with $|H_4| = t$ and computes σ satisfying $\sigma = e + h(m||V)x$, where

$$\sigma = \sum_{j \in H_4} \gamma_j \omega_j \text{ and } \omega_j = \prod_{\substack{l \neq j \\ l \in H_4}} \frac{l}{l-j}.$$

The signature is (σ, V). The signature can be verified as in Schnorr's original scheme:

$$\sigma G = V + h(m||V)Y \text{ and } \sigma \in Z_q.$$

Remarks.

1. The scheme can easily be modified such that a *trusted combiner* calculates the signature, instead of the players. The γ_i's would be sent secretly to the trusted combiner, who proceeds with the verification and the signature generation. In such a scenario, the players would not be able to generate a signature without the combiner.
2. The only property required by the underlying secret sharing scheme is that it must be homomorphic. This signature scheme could therefore be generalized to non-threshold access structures by a suitable linear general access structure secret sharing scheme as for example [17].

4.3 Correctness

We have to show that the signature σ computed in Step 4 is the Schnorr signature on m, i.e., $\sigma = e + h(m||V)x \bmod q$. Let $F_1(\cdot)$ be the sharing polynomial of the key generation protocol ($\alpha_i = F_1(i)$ for $i \in H_0$), and let $F_2(\cdot)$ be the sharing polynomial implied by the generated random shared secret in Step 1 ($\beta_i = F_2(i)$ for $i \in H_1$). Furthermore, let $F_3(\cdot) := F_2(\cdot) + h(m||V)F_1(\cdot)$. Since $\gamma_i = F_3(i)$ for $i \in H_3$, it follows from Lagrange's interpolation formula that the players compute $\sigma = F_3(0)$. We can now argue as follows:

$$\sigma = F_3(0) = F_2(0) + h(m||V)F_1(0) = e + h(m||V)x.$$

4.4 Robustness

We have to show that if less than t players are corrupted, the scheme always produces a valid signature. We assume that $t \leq \frac{n}{2}$.

From the robustness property of the protocol to generate a random shared secret it follows that every honest player P_i computes correct values $\alpha_i, \beta_i, \gamma_i$. Because there are at least t honest players, and because they can identify the correct γ_i by verifying (3), it follows directly by the correctness property that the honest players will always compute a valid certificate.

5 Security

5.1 Notion of Security

In this section, we show that the proposed (t, n) threshold signature scheme is as secure as Schnorr's signature scheme, i.e., existentially unforgeable under adaptively chosen message attacks in the random oracle model.

We define an adaptively chosen message attack against our (t, n) threshold scheme as follows. An adversary $A_{DistSchnorr}$ is allowed to have the signature issuing protocol executed by any t or more signers to compute signatures on messages of his own choice. She also might corrupt up to $t - 1$ arbitrary players. $A_{DistSchnorr}$ then tries to forge a new signature from the signatures she obtained in this way and from her *view*, where the view is everything that $A_{DistSchnorr}$ sees in executing the key generation protocol and the signature issuing protocol.

Let $A_{NormSchnorr}$ be a successful adversary that can break (in the sense of an existential forgery under adaptively chosen message attack) Schnorr's scheme (denoted by $D_{NormSchnorr}$), and let $A_{DistSchnorr}$ be a successful adversary that can break the distributed Schnorr scheme (denoted by $D_{DistSchnorr}$) presented in this paper. To proof the security of our scheme, we will show that given $A_{NormSchnorr}$, one can construct an adversary $A_{DistSchnorr}$, and visa versa. This implies that $D_{DistSchnorr}$ is as secure as $D_{NormSchnorr}$ is.

The idea of how to construct $A_{NormSchnorr}$ given the adversary $A_{DistSchnorr}$, a public key Y and a signing oracle goes as follows. $A_{NormSchnorr}$ simulates the roles of the uncorrupted players during all stages of $D_{DistSchnorr}$, i.e., from the key generation protocol that outputs Y up to the signature issuing protocols for $A_{DistSchnorr}$'s chosen message attack, and lets them interact with $A_{DistSchnorr}$ (see Section 5.3). Because $A_{DistSchnorr}$ cannot distinguish her view during this simulation from her view during a real run of $D_{DistSchnorr}$, she will succeed and output a valid forgery, and therefore so will $A_{NormSchnorr}$. Indistinguishability is used in the sense of the traditional notion of polynomial indistinguishability of two probability distributions as specified in [6].

The next section explains precisely what a view is. We also explain how to build a simulator *SIM* that simulates the honest players during the generation of a distributed random shared secret such that it produces for an arbitrary but given public key Y a view that is indistinguishable for the adversary from a view during a real run of the protocol which outputs Y. This simulator is then used later as a subroutine of a simulator that computes the adversary's entire view of our threshold signature scheme.

5.2 View

During an arbitrary multi-party protocol, a player will choose values on his own, see public broadcast values and receive private values. We define his view of the protocol to consist of all these values. Notice that in order to simulate the view for a player one does not have to simulate the values which the player chooses on his own.

In the following, we will analyze the adversary's view during the generation of a random shared secret. In particular, the goal is to build a simulator *SIM* that succeeds in the following game. Let B be the index set of corrupted players. The corrupted players P_i for $i \in B$ first run the protocol with honest players such that the public value of the random shared secret outputs a random value Y. Now we run the protocol again, but instead of communicating with the honest players, the players P_i for $i \in B$ communicate with the simulator. This simulator will now produce messages exactly as the honest players do, such that the public value of the random shared secret is Y, and furthermore, the adversary controlling players P_i for $i \in B$ cannot distinguish this simulated view from the view resulting from the honest players.

When generating a distributed random shared secret, as explained in Section 2.4, the view of a player P_i would be the following:

the sharing polynomials	$f_i(\cdot), f'_i(\cdot)$,
the temporary shares	$f_j(i), f'_j(i)$ for $j \in H_0$,
the public commitments	C_{jk}, A_{jk} for $j \in H_0, k \in \{0, \ldots, t-1\}$,
answers on a valid complaint against P_l	$f_l(j), f'_l(j)$ for $j \in H_0$,

and the content of his random tape. If an adversary corrupts P_i and P_j, then the adversary's view is $\{\text{view of } P_i\} \cup \{\text{view of } P_j\}$.

Definition 1. *Suppose that a set H_0 of players compute a random shared secret on input (q, G) and produce output Y. Let $\tilde{A}$ be an adversary that corrupts up to $t-1$ players. Let $view(\tilde{A}, G, q, Y)$ denote the view of the adversary for this protocol. Let $VIEW(\tilde{A}, G, q, Y)$ be the random variable induced by $view(\tilde{A}, G, q, Y)$*[2].

Lemma 1. *For any probabilistic polynomial time adversary $\tilde{A}$ there exists a probabilistic polynomial time simulator SIM that can compute a random variable $SIM(G, q, Y)$ which has the same probability distribution as $VIEW(\tilde{A}, G, q, Y)$.*

Proof of Lemma 1. Assume that $\tilde{A}$ corrupts players P_i for $i \in B = \{1, \ldots, t-1\}$. Furthermore, let B' be the index set that denotes the players who publish inconsistent values A_{ik}. Then, $view(\tilde{A}, G, q, Y)$, when generating a random shared secret, is as follows:

1. The content of the random tape of $\tilde{A}$
2. $f_i(\cdot), f'_i(\cdot)$ for $i \in B$
3. $f_j(i), f'_j(i)$ for $j \in H_0, i \in B$
4. C_{jk} for $j \in H_0, k \in \{0, \ldots, t-1\}$
5. A_{jk} for $j \in H_0, k \in \{0, \ldots, t-1\}$
6. $f_i(j), f'_i(j)$ for $j \in H_0, i \in B'$

[2] *view*(.) contains random variables and static values. *VIEW*(.) can be regarded as the interpretation of *view*(.) as one large bit string, so it is a random variable.

We show how to construct a simulator *SIM* that can act in the protocol as the honest players, such that the resulting view has the same probability distribution (we use the same simulator as in [7]). Note that *SIM* does not have to compute the sharing polynomials (2) itself since they are chosen by the adversary. The same holds for the content of the random tape (1) which is part of the adversary's internal state that does not have to be simulated.

1. (3, 4) Perform step 1 of the protocol on behalf of the honest players $P_t, \ldots, P_n$ exactly as specified in the protocol. This includes receiving and processing the information sent privately and publicly from corrupted players to honest ones. After this step, *SIM* knows all polynomials $f_i(\cdot), f_i'(\cdot)$ for $i \in H_0$. In particular, *SIM* knows all the shares $f_i(j), f_i'(j)$, the coefficients a_{ik}, b_{ik} and the public values C_{ik} for $i, j \in H_0$ and $k \in \{0, \ldots, t-1\}$.
2. (5) When extracting the values r_iG, the simulator acts as follows:
   - Compute $A_{ik} = a_{ik}G$ for $i \in H_0 \setminus \{n\}, k \in \{0, \ldots, t-1\}$
   - Compute $A_{n0} = Y - \sum_{i \in H_0 \setminus \{n\}} A_{i0}$
   - Compute $A_{nk} = \lambda_{k0}A_{n0} + \sum_{i=1}^{t-1} \lambda_{ki} f_n(i)G$ for $k \in \{1, \ldots, t-1\}$, where λ_{ki}'s are the Lagrange interpolation coefficients of the set H_0.
   - Broadcast A_{ik} for $i \in H_0, k \in \{0, \ldots, t-1\}$
3. (6) To handle the messages resulting from complaints, *SIM* acts as follows:
   - For each honest player, verify the values A_{ik} for $i \in B$ by checking (2). If the verification fails for some $i \in B, j \in H_0 \setminus B$, broadcast a complaint $f_i(j), f_i'(j)$. Notice that the corrupted players can publish a valid complaint only against one another, and there will be no complaints against an honest player that is simulated by *SIM*.
   - For each valid complaint against P_i, perform the reconstruction phase of Pedersen's VSS scheme to compute r_i and Y_i in the clear.

After step 1, the polynomials $f_i(\cdot), f_i'(\cdot)$ for $i \in H_0 \setminus B$ are chosen at random. All associated values $C_{ik}, f_i(j), f_i'(j), a_{ik}, b_{ik}$ therefore have the exact same probability distribution as in a real run of the protocol.

The broadcasted values A_{ik} are all uniformly random since the corresponding a_{ik} are random. This holds also for the specially computed A_{nk} for $k \in \{0, \ldots, t-1\}$, since for each such coefficient there is at least one random value it depends on. Notice that the fact that these A_{nk}'s are not consistent with the corresponding a_{nk}'s does not appear in the adversary's view: She never sees the a_{nk}'s but only the consistent public commitments of these values.

During the handling of complaints (step 3) there can only be valid complaints against a corrupted server. To reconstruct r_i, *SIM* has to reveal the values $f_i(j), f_i'(j)$ for $j \in H_0 \setminus B$. But *SIM* knows all the polynomials $f_i(\cdot), f_i'(\cdot)$ for $i \in H_0 \setminus B$. Therefore, *SIM* has only to broadcast these values, which will always be consistent with the adversary's view.

A more detailed analysis of the distribution can be found in [7]. The computed view, and the induced random variable $SIM(\tilde{A}, G, q, Y)$ has the same probability distribution as $VIEW(\tilde{A}, G, q, Y)$. $\square$

5.3 Unforgeability

In this section, we will show how to reduce the distributed Schnorr signature scheme to the regular Schnorr signature scheme, and visa versa. This implies that the security of the two schemes is identical.

Definition 2. *Let $A_{NormSchnorr}$ be a probabilistic polynomial time adversary who can ask a signer for valid signatures. By $A_{NormSchnorr}(G, q, Y)$ we denote a random variable which specifies the probability of the event that $A_{NormSchnorr}$ queries $(m_1, m_2, \dots)$ to the signer and outputs $(\tilde{m}, \tilde{\sigma}, \tilde{V})$ (on input (G, q, Y)). The probability is taken over all the coin tosses of $A_{NormSchnorr}$ and the signer.*

Definition 3. *Let $A_{DistSchnorr}$ be a probabilistic polynomial time adversary who can corrupt up to $t-1$ players. He also may have t or more arbitrary signers issue a signature upon his request. By $A_{DistSchnorr}(G, q|Y)$[3] we denote the random variable that has the probability distribution of $A_{DistSchnorr}$ asking for signatures on $(m_1, m_2, \dots)$ (on input (G, q)) and finally computing $(\tilde{m}, \tilde{\sigma}, \tilde{V})$ under the condition that the key generation protocol outputs Y. The probability is taken over all the coin tosses of $A_{DistSchnorr}$ and the signers.*

Theorem 1. *For any adversary $A_{NormSchnorr}$ against $D_{NormSchnorr}$, there exists an adversary $A_{DistSchnorr}$ against $D_{DistSchnorr}$ such that*

$$\begin{aligned} &Pr[A_{DistSchnorr}(G, q|Y) = (m_1, \dots, (\tilde{m}, \tilde{\sigma}, \tilde{V}))] \\ &= Pr[A_{NormSchnorr}(G, q, Y) = (m_1, \dots, (\tilde{m}, \tilde{\sigma}, \tilde{V}))]. \end{aligned}$$

Proof. We show how to construct $A_{DistSchnorr}$ given $A_{NormSchnorr}$. Suppose the key generation protocol of $D_{DistSchnorr}$ generates Y. $A_{DistSchnorr}$ feeds (G, q, Y) and the content of the random tape of $A_{NormSchnorr}$ into $A_{NormSchnorr}$ and starts $A_{NormSchnorr}$. Whenever $A_{NormSchnorr}$ asks for a signature on a message m, $A_{DistSchnorr}$ has some t signers execute the signature issuing protocol for m and returns the signature (σ, V) to $A_{NormSchnorr}$. Thus, $A_{NormSchnorr}$ can perform his chosen message attack. $A_{DistSchnorr}$ outputs $(\tilde{m}, \tilde{\sigma}, \tilde{V})$ if $A_{NormSchnorr}$ outputs $(\tilde{m}, \tilde{\sigma}, \tilde{V})$. □

Theorem 2. *For any adversary $A_{DistSchnorr}$ against $D_{DistSchnorr}$, there exists an adversary $A_{NormSchnorr}$ against $D_{NormSchnorr}$ such that*

$$\begin{aligned} &Pr[A_{NormSchnorr}(G, q, Y) = (m_1, \dots, (\tilde{m}, \tilde{\sigma}, \tilde{V}))] \\ &= Pr[A_{DistSchnorr}(G, q|Y) = (m_1, \dots, (\tilde{m}, \tilde{\sigma}, \tilde{V}))]. \end{aligned}$$

[3] $A_{DistSchnorr}(G, q|Y)$ is different from $A_{DistSchnorr}(G, q, Y)$. It contains not only the values G, q, Y, but also $A_{DistSchnorr}$'s view from the key generation protocol. For $A_{NormSchnorr}$ this view is empty, while for $A_{DistSchnorr}$ this is not the case (since he can corrupt $t-1$ signers)

Proof. We show how to construct $A_{NormSchnorr}$ given $A_{DistSchnorr}$. In particular, we will show how $A_{NormSchnorr}$ can simulate — with the help of a signing oracle (used in the chosen message attack assumption) — the role of the honest players in $D_{DistSchnorr}$ for a given public key Y. Because $A_{DistSchnorr}$ cannot distinguish this simulation, it will be successful and output a forgery which is a forgery in $D_{NormSchnorr}$, too.

Let $\mathcal{B}$ be the index set of players corrupted by $A_{DistSchnorr}$. Using the simulator described in Section 5.2, $A_{NormSchnorr}$ lets *SIM* execute the key generation protocol for the given public key Y. Note that $A_{NormSchnorr}$ knows after this simulation the values α_i for $i \in \mathcal{B}$. Next, $A_{NormSchnorr}$ runs $A_{DistSchnorr}$. Whenever $A_{DistSchnorr}$ requests a signature for a message m_i, $A_{NormSchnorr}$ asks a signer and provides $A_{DistSchnorr}$ with the signature (m_i, σ_i, V_i). $A_{NormSchnorr}$ also has to provide $A_{DistSchnorr}$ with the values she sees during the signature issuing protocol. These values include the view resulting from generating the random shared secret e and the values γ_i for $i \in H_2 \setminus \mathcal{B}$. To compute these values, $A_{NormSchnorr}$ lets *SIM* interact with $A_{DistSchnorr}$ during the generation of the random shared secret e. After this simulation $A_{NormSchnorr}$ knows β_i for $i \in \mathcal{B}$ and can compute γ_i for $i \in \mathcal{B}$. Finally, $A_{NormSchnorr}$ computes γ_j for $j \in H_2 \setminus \mathcal{B}$ as follows. W.l.o.g. we assume that $\mathcal{B} \cap H_2 = t-1$. Then we have for every $j \in H_2 \setminus \mathcal{B}$ (see Section 4.2):

$$\sigma_i = \sum_{k \in \mathcal{B} \cup \{j\}} \gamma_k \omega_k, \text{ where } \omega_k = \prod_{\substack{l \neq k \\ l \in \mathcal{B} \cup \{j\}}} \frac{l}{l-k}.$$

Hence, γ_j is computed as

$$\gamma_j = \frac{\sigma_i - \sum_{k \in \mathcal{B}} \gamma_k \omega_k}{\omega_j}.$$

$A_{NormSchnorr}$ feeds γ_j for $j \in H_2 \setminus \mathcal{B}$ to $A_{DistSchnorr}$. Since $A_{DistSchnorr}$ now has her whole view, she can perform her adaptive chosen message attack. $A_{NormSchnorr}$ outputs $(\tilde{m}, \tilde{\sigma}, \tilde{V})$ if $A_{DistSchnorr}$ outputs $(\tilde{m}, \tilde{\sigma}, \tilde{V})$. □

6 The Implicit Certificate Scheme

To motivate the (t, n) threshold scheme for implicit certificates, we give a short overview of the non-distributed version of this scheme. In [3], security proofs for this scheme in the random oracle model are given.

Assume a *CA* with the key pair (x, Y) issues an implicit certificate to a user, and let $h(\cdot)$ be a one-way hash function. The operation of the scheme is as follows.

1. The user generates a random integer $c \in Z_q$ and computes $V = cG$. Further, he sends V to the *CA*.

2. The CA authenticates the user. Together, the CA and the user determine an identifier string I_u (containing the user's identity and other information such as, for example, a serial number for the certificate).
3. The CA chooses a random integer $e \in Z_q$, and computes $C = V + eG$ and $\sigma = e + h(I_u||C)x$. Further, the CA sends (I_u, C, σ) to the user.
4. The user computes his private key $SK_u = c + s \bmod q$, and verifies the certificate by checking that following equation holds: $SK_u = C + h(I_u||C)Y$.

The user's public key can be computed from the certificate (I_u, C) as follows: $PK_u = C + h(I_u||C)Y$. Note that the equation used to compute σ is exactly Schnorr's signing equation. The only difference from Schnorr's signature scheme is the construction of the point C. Here, this point contains an additive component that the user provides. This is necessary to guarantee that only the user knows his secret key.

7 (t, n) Threshold Scheme for Implicit Certificates

In this section, we incorporate the distributed Schnorr signature scheme into a (t, n) threshold scheme for implicit certificates in the same way as was done in Section 6. In such a scheme, n players $P_1, \ldots, P_n$ represent a CA with public key PK_0. A group of t players together can reconstruct SK_0 and issue an implicit certificate. Any coalition of less than t players do not have any information about SK_0.

Our scheme consists of three steps. First, the players representing the CA have to generate a key pair. Everybody will know the value of PK_0, while only a coalition of at least t players shall be able to recover SK_0 or issue certificates. Second, the players issue a certificate to a user. Finally, the user verifies if the certificate is valid.

7.1 Key Generation Protocol

We would like to generate a random shared secret SK_0 such that each player P_i who follows the protocol holds a share s_i in this key. Moreover, a coalition of less than t players cannot get any information about SK_0.

This situation corresponds exactly to the generation of a shared secret, as described in Section 2.4. Using the notation introduced in Section 2.4, the situation is as follows:

$$(\alpha_1, \ldots, \alpha_n) \overset{(t,n)}{\longleftrightarrow} (SK_0|PK_0, b_kG, H_0), \ k \in \{1, \ldots, t-1\}.$$

7.2 Certificate Issuing Protocol and Public Key Reconstruction

Suppose the players with index set $H_1 \subseteq H_0$ want to issue an implicit certificate.

1. The user selects a random number c_u and sends $V_u = c_uG$ to the players.

2. If $|H_1| < t$, stop. Otherwise, H_1 generates a random shared secret as shown in Section 2.4. Let the public output be
$$(\beta_1, \ldots, \beta_n) \overset{(t,n)}{\longleftrightarrow} (e|V, c_k G, H_2),\ k \in \{1, \ldots, t-1\}.$$
3. If $|H_2| < t$, stop. Otherwise, each P_i for $i \in H_2$ computes $C = V + V_u$ and reveals
$$\gamma_i = \beta_i + h(I_u||C)\alpha_i. \tag{4}$$
4. Each P_i for $i \in H_2$ verifies that
$$\gamma_j G = V + \sum_{k=1}^{t-1} c_k j^k G + h(I_u||C)\left(Y + \sum_{k=1}^{t-1} b_k j^k G\right) \text{ for all } j \in H_2. \tag{5}$$
Let H_3 be the index set of players not detected to be cheating at step 3.
5. If $|H_3| < t$ stop. Otherwise, each P_i for $i \in H_3$ selects an arbitrary set $H_4 \subseteq H_3$ with $|H_4| = t$ and computes σ satisfying $\sigma = e + h(I_u||C)x$ by
$$\sigma = \sum_{j \in H_4} \gamma_j \omega_j, \text{ where } \omega_j = \prod_{\substack{l \neq j \\ l \in H_4}} \frac{l}{l-j}. \tag{6}$$
The implicit certificate is (σ, C), which every player sends to the user.
6. At most $t-1$ of the certificates the user receives may be incorrect. To identify the correct certificates, the user computes his private key SK_u as $SK_u = c_u + \sigma$ and verifies
$$SK_u G = C + h(I_u||C)Y \text{ and } \sigma \in Z_q. \tag{7}$$

The public key of the user can be computed from the implicit certificate as follows:
$$\sigma PK_u = C + h(I_u||C)Y. \tag{8}$$

7.3 Correctness

We have to verify that the private key SK_u computed by the user corresponds to the public key PK_u implied by the implicit certificate, i.e., we have to verify that following holds:
$$SK_u G = C + h(I_u||C)PK_0. \tag{9}$$

Let $\mathcal{P}$ ($|\mathcal{P}| = t$) be a group of players which have not been detected to be cheating when issuing the certificate. Then we have

$$\begin{aligned}
SK_u G &\overset{(6)}{=} (c_u + \sum_{i \in \mathcal{P}} \gamma_i \omega_i) G \\
&\overset{(4)}{=} c_u G + \left(\sum_{i \in \mathcal{P}} (\beta_i + h(I_u || C) \alpha_i) \right) G \omega_i \\
&= V_u + \sum_{i \in \mathcal{P}} (\beta_i \omega_i G + \alpha_i \omega_i h(I_u || C) G) \\
&= V_u + V + h(I_u || C) PK_0 \\
&= C + h(I_u || C) PK_0 .
\end{aligned}$$

7.4 Robustness

We have to show that if less than t players are corrupted, the scheme always produces a valid certificate that is accepted by the user. We assume that $t \le \frac{n}{2}$.

From the robustness property of the protocol to generate a random shared secret it follows that every honest player P_i computes correct values $\alpha_i, \beta_i, \gamma_i$. Because there are at least t honest players, and because they can identify the correct γ_i by verifying (5), it follows directly by the correctness property that the honest players will always compute a valid certificate. Finally, the user can identify a valid certificate by verifying (7).

8 Security Analysis

8.1 Notion of Security

Let (SK_{CA}, PK_{CA}) be the key pair of the CA. An implicit certificate scheme is *secure* if the following two properties hold:

unforgeability It is hard for an adversary who does not know the CA's secret key to forge implicit certificates in such a manner that the adversary knows the corresponding private key

non-impersonating It is hard for the CA to obtain the user's private key provided that the user followed the protocol.

The term “hard” means that there is no polynomial-time adversary who can solve the task with non-negligible probability. These conditions must hold for adversaries defined as follows.

We define a forging adversary A_f as a probabilistic, polynomial-time Turing machine which, on input PK_{CA} does the following:

- It may watch other entities requesting and receiving implicit certificates from the CA.
- It may request implicit certificates from the CA.

– Finally, it produces an implicit certificate and the corresponding private key in time t and with probability p.

We define an impersonating adversary A_i as a probabilistic, polynomial-time Turing machine which, on input (PK_{CA}, SK_{CA}) does the following:

– It may act as a CA and issue implicit certificates to requesting entities.
– It can produce an implicit certificate and the corresponding private key in time t and with probability p.

An adversary A_f (respectively, A_i) is successful if t is polynomial and p is non-negligible.

8.2 Unforgeability

Let (x, Y) be the (SK, PK) key pair of the CA. Let $D_{NormSchnorr}$ denote Schnorr's signature scheme and $A_{NormSchnorr}$ be a successful adversary against it as defined earlier in Section 5.1. We define a successful adversary $A_{DistCert}$ against the implicit certificate scheme $D_{DistCert}$ as a successful forging adversary as defined in Section 8.1.

One can show that a successful adversary $A_{NormSchnorr}$ is equivalent to a successful adversary $A_{DistCert}$, in the sense that each of them can construct the other one. This implies that the distributed implicit certificate scheme is as secure as Schnorr's signature scheme.

The same proof technique as was used for the distributed Schnorr signature scheme can be applied in a straightforward way. That is, one can show how to simulate the view of the given adversary without knowing the private key of the players. Since the adversary cannot distinguish a simulated view from an actual view, she will perform her attack and output a forgery. This forgery can then be used to construct the other adversary.

8.3 Non-impersonating

By proving the unforgeability of our scheme, we implicitly proved that the user does not learn the players' private key shares. We also have to show that the players do not learn the user's private key and impersonate the user. But it follows directly from the scheme that if the players could compute the user's private key, then they could compute discrete logarithms.

8.4 Further Issues

Consider the scenario where a digital signature on a certain message and an implicit certificate authenticating the according verification key are sent to a user. Even though we proved that it is hard to forge an implicit certificate without knowing the CA's secret key such that one knows the corresponding private key, we did not prove that it is hard to forge a digital signature and an implicit certificate such that the public key implied by the certificate just validates the signature.

This is not an issue with traditional certificates. However, whenever implicit certificates are used to authenticate a public key for some application, a specific security proof for the particular application is necessary. For example, in [2], a proof is given in the random oracle model that it is secure to use implicit certificates as authentication for public keys that verify Schnorr signatures.

9 Conclusion

We have presented a (t, n) threshold version of Schnorr's signature scheme, and a (t, n) threshold scheme for implicit certificates. Both schemes are efficient, robust and provably secure in the random oracle model.

From a practical point of view, the implicit certificate scheme has the following drawbacks:

– The scheme itself generates a key pair for the user. Therefore, it cannot be used to generate an implicit certificate for a given key pair of the user.
– The scheme produces a key pair which is defined over the same group as the *CA*'s key pair is. Therefore, the security parameters for the certified public keys are always inherited from the certifying *CA*.

Acknowledgments

This work was done while I was visiting Certicom as an intern during the period August - November, 2000. I would like to thank Certicom for this opportunity. Furthermore, we would like to thank Simon Blake-Wilson for his ideas on the general concepts, Minghua Qu for reviewing the security proofs and for pointing out the special security issues that arise in the context of implicit certificates, and Dan Brown for all the instructive discussions about implicit certificates.

References

[1] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *First Annual ACM Conference on Computer and Communications Security*, pages 62–73, 1993.
[2] D. Brown. Implicitly certifying signatures securely. manuscript.
[3] R. Gallant D. Brown and S. Vanstone. Provably secure implicit certificate schemes. In *Financial Cryptography '01*, to appear.
[4] Y. G. Desmedt. Threshold cryptography. *European Trans. on Telecommunications*, 5(4):449–457, 1994.
[5] P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *28th FOCS*, pages 427–437, 1987.
[6] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. on Computing*, 18/1:186–308, 1989.

[7] R. Gennaro S. Jarecki H. Krawczyk and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *Eurocrypt '99*, pages 295–310, 1999.

[8] S. K. Langford. Threshold DSS signatures without a trusted party. In *Crypto '95*, pages 397–409, 1995.

[9] E. Okamoto, G. Davida, and M. Mambo. Some recent research aspects of threshold cryptography. In *Workshop on Information Security Applications*, 1997.

[10] C. Park and K. Kurosawa. New elgamal type threshold digital signature scheme. *IEICE Trans.*, E79-A:86–93, 1996.

[11] T.P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Crypto '91*, pages 129–140, 1991.

[12] T.P. Pedersen. A threshold cryptosystem without a trusted party. In *Eurocrypt '91*, pages 522–526, 1991.

[13] L. Pintsov and S. Vanstone. Postal revenue collection in the digital age. In *Financial Cryptography '00*, 2000.

[14] D. Pointcheval and J. Stern. Security proofs for signature schemes. In *Eurocrypt '96*, pages 387–399, 1996.

[15] C.P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4:161–174, 1991.

[16] A. Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, 1979.

[17] M. van Dijk. A linear construction of secret sharing schemes. *Designs, Codes and Cryptography*, 12:161–201, 1997.

How to Construct Fail-Stop Confirmer Signature Schemes

Rei Safavi-Naini*, Willy Susilo, and Huaxiong Wang

Centre for Computer Security Research
School of Information Technology and Computer Science
University of Wollongong
Wollongong 2522, AUSTRALIA
{rei, wsusilo, huaxiong}@uow.edu.au

Abstract. In a confirmer signature, verification of a signature requires collaboration of the confirmer. A Fail-Stop Confirmer signature provides protection against an enemy with unlimited computational power. A Fail-Stop Confirmer signature is a combination of Fail-Stop Signature and Confirmer Signature Schemes which was first constructed in [15]. In this paper we discuss security issues that will arise in naive construction of such systems.

1 Introduction

An ordinary digital signature [8] is verifiable by anyone who has access to the correct public key. If only a single recipient is to verify the signature, a zero-knowledge proof [10] can be used. Undeniable signatures [4] are between these two: an undeniable signature can be verified by everyone but requires the help of the signer. The signer is able to reject invalid signatures, but he must not be able to deny valid signatures. If the signer is unavailable or unwilling to cooperate, the signature would no longer be verifiable. To overcome this shortcoming, the notion of *confirmer signatures* [3] is proposed. In confirmer signatures, the ability to verify or deny signatures is transferred to a designated confirmer. A generic construction of a confirmer signature scheme from a ordinary signature scheme is proposed in [2].

Security of traditional signature schemes relies on some computational assumptions. This means that if an enemy can solve the underlying hard problem, he can successfully forge a signature and there is no way for the signer to prove that a forgery has occurred. To provide protection against an enemy with unlimited computational power, Fail-Stop Signature (FSS) schemes have been proposed [25,17]. An FSS scheme is a signature scheme equipped with an algorithm to prove a forgery has happened. To achieve this property, many secret keys match to the same public key and the sender uses a specific one of them. An unbounded enemy can find out the set of all secret keys but cannot determine

* This work is in part supported by Australian Research Council Grant Number $A49703076$

which secret key is actually used. So in the case of forgery, that is generating a signed message that passes the verification test, the sender can use his secret key to generate a second signature for the same message which with overwhelming probability will be different from the forged one. The two signatures on the same message can be used as a proof that the underlying computational assumption is broken and the system must be stopped - hence the name *fail-stop*. Thus, FSS schemes provide information-theoretic security for the signer. However security for the receiver is computational. An FSS in its basic form is a *one-time digital signature* that can only be used for signing a single message. However, it is possible to extend an FSS scheme to be used for signing multiple messages [5,23,1].

A Fail-Stop Confirmer Signature (FSCS) scheme, introduced in [15], combines the confirmer signature property with the fail-stop property. The purpose of FSCS is to provide information-theoretic security for the signer and maintain the confirmer property, so that when the signer is unavailable, the confirmer is able to verify the signature.

In this paper, we propose a model of FSCS scheme and show the difficulties of constructing one.

1.1 Previous Works

Confirmer Signature Scheme is introduced in [3]. Okamoto presented a formal model and proved that the existence of confirmer signature schemes are equivalent to the public-key encryption schemes [16] and presented a practical scheme. However, it is shown [14] that Okamoto's scheme is insecure because the confirmer can forge a signature. Michels and Stadler [14] proposed a solution to Okamoto's problem by introducing a new model. However, as pointed out in [2], their model is vulnerable to an *adaptive signature-transformation* attack (which is similar to security against adaptive chosen-ciphertext attacks [11] for encryption schemes) and that all previous schemes are vulnerable to this attack. Camenisch and Michels presented a generic construction for confirmer signature schemes that does not suffer from the adaptive signature-transformation attack.

Fail-Stop Signature (FSS) schemes protects the signer information theoretically against an unlimited forger. The first construction of FSS [25] uses a one-time signature scheme (similar to [13]) and results in bit by bit signing of the message, which is impractical. In [18] an efficient single-recipient FSS to protect clients in an on-line payment system, is proposed. The main disadvantage of this system is that signature generation is a 3-round protocol between the signer and the recipient which makes it expensive from communication point of view. van Heijst and Pedersen [23] proposed an efficient FSS that uses the difficulty of discrete logarithm problem as the underlying assumption. In the case of a forgery, the presumed signer can solve an instance of the discrete logarithm problem, and prove that the underlying assumption is broken.

In [17,19], a formal definition of FSS schemes is given and a general construction using *bundling homomorphism* is proposed. The important property of this construction is that it is provably secure against the most stringent type of attack on signature schemes, that is adaptive chosen message attack [12]. The proof of forgery is by showing two different signatures on the same message, the forged one and the one generated by the valid signer. To verify the proof of forgery the two signatures are shown to collide under the 'bundling homomorphism'. The scheme by van Heijst and Pedersen [23] is an example of this construction. Heijst, Pedersen and Pfitzmann [24] also gave an example of this construction that uses the difficulty of factoring as the underlying computational assumption of the system [24]. Other works in this area include [23,24,22]. A general construction of FSS from authentication codes has been given in [20] and has been used to construct an efficient FSS to sign long messages [21].

A Fail-Stop Confirmer Signatures (FSCS) combines the property of Confirmer Signature and FSS schemes. The first construction of FSCS was proposed in [15]. The scheme is an extension of an FSS scheme proposed in [23].

1.2 Our Contributions

In this paper, we define a model of FSCS scheme that has *separability* property [2], that is, it allows all parties to independently run their key generation algorithms (cf. [15]). We propose a generic method for converting an FSS scheme into an FSCS scheme while maintaining its security properties. We show that an FSCS can be constructed from an FSS scheme combined with an encryption scheme. We discuss the security issues that arise in the FSCS scheme because of the unbounded enemy. In particular we show that the confirmer does not have any significance from security point of view and is mainly to provide non-transferability for the signatures. This shows that a simple combination of FSS and encryption schemes to construct an FSCS scheme is insecure.

The paper is organized as follows. In the next section, we give a model for FSCS schemes and outline its security requirements. Section 3 proposes a generic construction for FSCS schemes from FSS scheme and a secure encryption scheme. In section 4 we discuss the problem that happens in an FSCS model. Section 5 concludes the paper.

2 FSCS Model

There exists a signer $\mathcal{S}$, a confirmer $\mathcal{C}$ and a signature verifier $\mathcal{V}$ who are polynomially bounded. There is a trusted third party TA whose role is only required during prekey generation (and it can be eliminated by replacing its role with the recipient or the signature verifier). The enemy $\mathcal{E}$ has unlimited computational power.

A Fail-Stop Confirmer Signature (FSCS) scheme consists of the following procedures:

- `Prekey Generation:`
  Let $PKG(k, \sigma, \ell) \rightarrow (x_D, y_D)$ is a probabilistic algorithm where k and σ are the security parameters for the receiver and sender, respectively, and (x_D, y_D) is a secret/public key pair for the TA (or trusted dealer). ℓ is the security parameter of the confirmer.
- `Key Generation:`
  Consists of two probabilistic algorithms: $KGS()$ and $KGC()$, where $KGS()$ is performed by $\mathcal{S}$ and $KGC()$ is performed by $\mathcal{C}$. $KGS(k, \sigma, y_D) \rightarrow (x_S, y_S)$ where y_D is the public key of TA with the same security level (k, σ) obtained from the algorithm PKG, and $KGC(\ell, y_D) \rightarrow (x_C, y_C)$. (x_S, y_S) is a secret/public key pair for the signer $\mathcal{S}$, and (x_C, y_C) is a secret/public key pair for the confirmer $\mathcal{C}$.
- `Signing:`
  A probabilistic algorithm $CSig(m, x_S, y_S, y_C) \rightarrow \delta$ that generates a signature for a message $m \in \{0,1\}^*$.
- `Confirmation and Disavowal:`
  A signature verification protocol $Ver()$ between a confirmer $\mathcal{C}$ and a verifier $\mathcal{V}$. The private input of $\mathcal{C}$ is x_C and their common input consists of m, δ, y_S, y_C. The output of this protocol is either 1 (true) or 0 (false).
- `Proof of Forgery:`
  A probabilistic algorithm $PoF(m, \delta, \tilde{\delta}) \rightarrow \{\eta, \perp\}$ will be performed by $\mathcal{S}$ to generate a proof of forgery in the case of dispute, where δ and $\tilde{\delta}$ denote two signatures that pass $Ver()$. The output of this protocol is either η (the proof of forgery) or $\perp$ (fail). If an enemy has successfully constructed a signature $\tilde{\delta}$ on a message m, in which $Ver()$ outputs 1, then with an overwhelming probability the presumed signer $\mathcal{S}$ can run $PoF(m, \delta, \tilde{\delta})$, where $\delta \leftarrow CSig(m, x_S, y_S, y_C)$ to show that the underlying hard assumption of the system has been broken.
  An probabilistic algorithm $VerPoF(\eta, y_S, y_C) \rightarrow \{0,1\}$ that allows everyone to verify the proof of forgery. It takes as input the proof of forgery η together with the public information (y_S, y_C) and returns 1 if the proof of forgery is valid, or 0 otherwise.
- `Selective Convertibility:`
  An algorithm $CConv(m, \delta, y_S, x_C, y_C) \rightarrow \{s, \perp\}$ that allows a confirmer $\mathcal{C}$ to convert a confirmer signature δ into an ordinary signature, that allows anyone to verify the signature without the help of the confirmer. If the conversion fails, the algorithm outputs $\perp$.
- `Signature Verification (Ordinary):`
  An algorithm $COVer(m, s, y_S) \rightarrow \{0,1\}$ that allows everyone to verify the ordinary signature that is the output of $CConv()$. It takes as input a message m, a signature s and the signer's public key y_S.

Notions of Security

The FSS scheme used in FSCS must be provably secure against *adaptive chosen-message attack* [11]. In this type of attack, the adversary can choose messages and get the corresponding signatures. His task is to sign a different message that has not been signed by the original signer such that the signature is *identical* to the one that should have been produced by the original signer. An algorithm is secure against adaptive chosen-message attack if the probability of the adversary producing such signature is negligible.

Security Requirements
In the following, we define the security requirements for the sender, confirmer and recipient of FSCS schemes.

- `Security for the Sender:`
  Security for the sender $\mathcal{S}$ ensures that the confirmer signature and the converted signatures are unforgeable under an adaptive chosen-message attack. The signer $\mathcal{S}$ is protected information theoretically against an enemy with unlimited computational power, with security level σ. For each message many signatures can be generated that pass verification test. The chance of an unbounded enemy to construct the one produced by the true signer is bounded by $2^{-\tau}$, where τ is the *bundling degree* homomorphism [17] which is the relevant security parameter. In the case of forgery, the presumed signer $\mathcal{S}$ can generate a proof of forgery with an overwhelming probability.
- `Security for the Confirmer:`
  If the confirmer's confirmation is forged, the presumed signer will always be able to generate a proof of forgery with overwhelming probability.
- `Security for the Receiver:`
  The receiver is protected computationally against the sender and the confirmer, which are polynomially bounded. The sender and the confirmer cannot falsely confirm or deny the signature with overwhelming probability. To be more precise, the security level for the receiver against the sender is measured by k and the security level against the confirmer is measured by ℓ. Therefore, for a sufficiently large k and ℓ and $c > 0$, we require that

$$
\begin{aligned}
&P\Big\{\eta \leftarrow PoF(m, \delta, \tilde{\delta})| \\
&(\delta \leftarrow CSig(m, x_S, y_S, y_C)) \wedge (1 \leftarrow Ver(m, y_S, y_C, \tilde{\delta})) \wedge \\
&(\tilde{\delta} \neq \delta) \wedge (\eta \textit{ is valid})\Big\} \leq (\min(k, \ell))^{-c}
\end{aligned}
$$

- `Collusion Attack against the Sender:`
  The strongest attack in FSCS can be performed by an unbounded enemy who is colluding with the confirmer against the sender. In this case, the enemy (or the colluding confirmer) has the knowledge of x_C together with his unbounded ability to solve the hard underlying assumption. Under this attack, we require that the signer is still protected information theoretically from the colluding enemy and confirmer, with appropriate security level (e.g. σ).

An FSCS scheme must satisfy the following security requirements.

- `Unforgeability of Signatures:`
  There exists no polynomial time algorithm which on input y_S, y_C outputs with non-negligible probability an arbitrary correct message-signature pair $(m, \tilde{\delta})$ where $\tilde{\delta} \neq \delta$ for $\delta \leftarrow CSig(m, x_S, y_S, y_C)$ and $\perp \leftarrow PoF(m, \delta, \tilde{\delta})$.
- `Consistency of Verification:`
  If the confirmer is honest, for all $Ver()$ between a confirmer $\mathcal{C}$ and a verifier $\mathcal{V}$ and all (correct and incorrect) message-signature pairs (m, δ) the following equation must hold

$$Ver(m, y_S, y_C, \delta) = \begin{cases} 1 \text{ if } \delta \overset{?}{=} CSig(m, x_S, y_S, y_C) \\ 0 \ otherwise \end{cases}$$

  Informally, this means that the honest confirmer will always confirm correctly.
- `Non-transferability of Verification:`
  The verification protocol $Ver()$ must be a minimum knowledge bi-proof (according to the definition of [9]). Receiving the confirmation from $\mathcal{C}$, the verifier $\mathcal{V}$ cannot reuse this proof to show someone else that the signature is valid.

Definition 1. *A (k, σ, ℓ)-secure FSCS scheme is an FSCS scheme in which the security level for the signer against an unbounded forger is σ, security level for the confirmer is ℓ, and the recipient is protected computationally against the sender and the confirmer with security level* $\min(k, \ell)$.

3 A Generic Construction for FSCS Schemes

In this section, we propose a generic construction for a FSCS scheme from an FSS scheme. This is a variation of the construction proposed in [2].

Let $SIG = (SPKG, SKG, Sig, Ver)$ denote an FSS scheme, where $SPKG$ is the prekey generation algorithm, SKG is the key generation algorithm, Sig is the signing algorithm and Ver is the verification algorithm [17]. Let $ENC = (EKG, Enc, Dec)$ denote a public key encryption scheme. On input a security level, EKG outputs a key pair (x', y') where x' is a secret key and y' is the corresponding public key. On input y' and a message m, Enc outputs a ciphertext c, and on input the secret key x' and a ciphertext c, Dec outputs m. If c is not valid Dec outputs $\perp$.

Given an FSS scheme and a secure encryption scheme, an FSCS can be constructed as follows:

1. The key generators are chosen as
   - $PKG(k, \sigma) \overset{\triangle}{=} SPKG(k, \sigma)$;
   - $KGS(k, \sigma, y_D) \overset{\triangle}{=} SKG(k, \sigma, y_D)$, and
   - $KGC(\ell) \overset{\triangle}{=} EKG(\ell)$.

2. The signer signs a message $m \in \{0,1\}^*$ by constructing $s := Sig(m, x_S, y_S)$ and $\delta := Enc(s, y_C)$. The confirmer signature on m is given by δ.
3. The confirmation and disavowal protocol $Ver()$ between the confirmer and a verifier is as follows:
   Receiving a confirmer signature δ, the confirmer $\mathcal{C}$ decrypts δ to obtain $\tilde{s} := Dec(\delta, x_C)$. If $Ver(m, \tilde{s}, y_S) = 1$, then $\mathcal{C}$ declares the signature valid. This is through a concurrent zero-knowledge [7] protocol between the confirmer and the verifier that proves to the verifier that "$\gamma_1 = Dec(\delta, \gamma_2, y_C)$ and $Ver(m, \gamma_1) = 1$, and γ_2 is the secret key corresponding to y_C". Otherwise, the confirmer declares the signature invalid and proves in concurrent zero-knowledge that "($\gamma_1 = Dec(\delta, \gamma_2)$ and $Ver(m, \gamma_1, y_S) = 0$, where γ_2 is the secret key corresponding to y_C, or decryption fails)".
4. The protocol to prove forgery is run by $\mathcal{S}$ in the case that there is a signature $\hat{\delta}$ on a message m that passes the verification test performed with the confirmer $\mathcal{C}$.
   $\mathcal{S}$ generates his signature on the same message $s := Sig(m, x_S, y_S)$ and publishes it as the proof of forgery.
   The proof of forgery verification can be performed as follows:
   - Verify that $Ver(s, y_S) \stackrel{?}{=} 1$.
   - Compute $\delta = Enc(s, y_C)$ and verify that $\delta \stackrel{?}{\neq} \hat{\delta}$.

   The above conditions show that δ is different from $\hat{\delta}$ and both of the signatures pass the verification test. If the above conditions hold, the proof of forgery is valid and the scheme has to be stopped at this stage.
5. The selective conversion algorithm $CConv(m, \delta, y_S, x_C, y_C)$ outputs $s := Dec(\delta, x_C)$ if $Ver(m, Dec(\delta, x_C), y_S) = 1$. Otherwise, outputs $\perp$.
6. The public verification algorithm for converted signatures is defined as

$$COVer(m, s, y_S) \triangleq Ver(m, s, y_S)$$

3.1 Properties of the Signature and Encryption Schemes

The above construction is based on a generic construction proposed in [2] and uses an FSS that is secure against adaptive chosen-message attack with a deterministic public key encryption scheme.

Unlike [2], we do not require an encryption scheme that is secure against adaptive chosen-ciphertext attacks which demands the encryption scheme (for instance [6]) to be probabilistic. Using a probabilistic public key encryption scheme allows the signer to be able to deny his signature as shown below:

1. The signer constructs $s := Sig(m, x_S, y_S)$ and calculates $\delta = Enc_P(s, y_C, r)$, where Enc_P is a probabilistic encryption scheme and r is randomly selected.
2. Publish δ as an FSCS on m.

The signer can always deny his own signature δ by publishing s. This is because the verifier (who does not know r) will select an $\hat{r}$ which with overwhelming probability will be different from r, and produces a proof of forgery as follows:

- Verify $Ver(s, y_S) \overset{?}{=} 1$, which will be true; and
- Calculate $\tilde{\delta} = Enc_P(s, y_C, \hat{r})$ and check $\tilde{\delta} \overset{?}{\neq} \delta$, which will also be true.

Theorem 1. *The above construction satisfies the security requirements mentioned in section 2.*

Proof (sketch).

- `Security for the Sender:`
  The signature on m, $s := Sig(m, x_S, y_S)$, is obtained from an FSS that is secure against adaptive chosen message attack, with σ as the security level of the signer. In the case of dispute, the proof of forgery PoF can always be generated with an overwhelming probability. The output of the selective conversion algorithm $CConv()$ is an FSS that has the same property as the original FSS signature.
- `Security for the Receiver:`
  In the above construction, with overwhelming probability the sender cannot deny his signature and an honest confirmer cannot falsely confirm a signature. The security level of the system for the receiver against the sender is k and against the confirmer is ℓ. Since the signer is computationally bounded, he cannot find another secret key that matches with his public key and use it to create a signature that could be used for a proof of forgery (hence denying his own signature). In fact the chance of finding such a key is $\leq k^{-c}$ where k is as defined above, and $c > 0$.

4 Security Problems in FSCS

The enemy in FSCS is unbounded and can solve the underlying hard problem(s) of the system. Hence he can always create a signature that will be confirmed. On the other hand this signature can be shown to be a forgery with a very high probability. This means that the confirmer's role is strictly limited to making the signature untransferable and does not have any significance from the security point of view.

On the other hand an unlimited enemy can find the secret key of the confirmer and fully impersonate him, not only generate false signatures but also run a false verification protocol with the recipient of a signature generated by the sender and reject the signature. That is, correctly generated signatures may be rejected.

Both above security flaws exist in the scheme proposed in [15]. It seems that there is no easy way of correcting these problems as they are direct result of assuming the enemy has unlimited computational power.

5 Conclusion

In this paper we defined a model for Fail-Stop Confirmer signature (FSCS) schemes and proposed a generic construction for FSCS schemes using a combination of Fail Stop Signature schemes and encryption schemes. However as discussed above, the resulting system will have security flaws that are not easily correctable. These flaws exist in a construction proposed in [15] and so modelling and constructing a secure FSCS remains an interesting open problem.

References

1. N. Barić and B. Pfitzmann. Collision-Free Accumulators and Fail-Stop Signature Schemes without Trees. *Advances in Cryptology - Eurocrypt '97, Lecture Notes in Computer Science 1233*, pages 480–494, 1997.
2. J. Camenisch and M. Michels. Confirmer signature schemes secure against adaptive adversaries. *Advances in Cryptology - Eurocrypt 2000, Lecture Notes in Computer Science 1807*, 2000.
3. D. Chaum. Designated Confirmer Signatures. *Advances in Cryptology - Eurocrypt '94, Lecture Notes in Computer Science 950*, pages 86 – 91, 1994.
4. D. Chaum and H. van Antwerpen. Undeniable signatures. *Advances in Cryptology - Crypto '89, Lecture Notes in Computer Science 435*, pages 212–216, 1990.
5. D. Chaum, E. van Heijst, and B. Pfitzmann. Cryptographically strong undeniable signatures, unconditionally secure for the signer. *Interner Bericht, Fakultät für Informatik*, 1/91, 1990.
6. R. Cramer and V. Shoup. A Practical Public Key Cryptosystem Provably Secure against Adaptive Chosen Ciphertext Attack. *Advances in Cryptology - Crypto '98, Lecture Notes in Computer Science 1642*, pages 13 - 25, 1998.
7. I. B. Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. *Advances in Cryptology - Eurocrypt 2000, Lecture Notes in Computer Science 1807*, 2000.
8. W. Diffie and M. Hellman. New directions in cryptography. *IEEE IT*, 22:644–654, 1976.
9. A. Fujioka, T. Okamoto, and K. Ohta. Interactive bi-proof systems and undeniable signature schemes. *Advances in Cryptology - Eurocrypt '91, Lecture Notes in Computer Science 547*, pages 243 – 256, 1992.
10. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. *SIAM Journal of Computing*, 18(1):186–208, 1989.
11. S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal of Computing*, 17/2:281–308, 1988.
12. S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal of Computing*, 17:281–308, 1998.
13. L. Lamport. Constructing digital signatures from a one-way function. *PSRI International CSL-98*, 1979.
14. M. Michels and M. Stadler. Generic constructions for secure and efficient confirmer signature schemes. *Advances in Cryptology - Eurocrypt '98, Lecture Notes in Computer Science 1403*, pages 406 – 421, 1998.
15. Y. Mu and V. Varadharajan. Fail-Stop Confirmer Signatures. *Information Security and Privacy, ACISP 2000, Lecture Notes in Computer Science 1841*, pages 368 – 377, 2000.
16. T. Okamoto. Designated confirmer signatures and public-key encryption are equivalent. *Advances in Cryptology - Crypto '94, Lecture Notes in Computer Science 839*, pages 61 – 74, 1994.

17. T. P. Pedersen and B. Pfitzmann. Fail-stop signatures. *SIAM Journal on Computing*, 26/2:291–330, 1997.
18. B. Pfitzmann. Fail-stop signatures: Principles and applications. *Proc. Compsec '91, 8th world conference on computer security, audit and control*, pages 125–134, 1991.
19. B. Pfitzmann. *Digital Signature Schemes – General Framework and Fail-Stop Signatures.* Lecture Notes in Computer Science 1100, Springer-Verlag, 1996.
20. R. Safavi-Naini and W. Susilo. A general construction for Fail-Stop Signature using Authentication Codes. *Proceedings of Workshop on Cryptography and Combinatorial Number Theory (CCNT '99), Birkhäuser*, pages 343–356, 2001.
21. R. Safavi-Naini, W. Susilo, and H. Wang. Fail-Stop Signatures for Long Messages. *The First International Conference on Cryptology in India, Indocrypt 2000, Lecture Notes in Computer Science 1977*, pages 165 – 177, 2000.
22. W. Susilo, R. Safavi-Naini, M. Gysin, and J. Seberry. A New and Efficient Fail-Stop Signature schemes. *The Computer Journal vol. 43 Issue 5*, pages 430 – 437, 2000.
23. E. van Heijst and T. Pedersen. How to make efficient fail-stop signatures. *Advances in Cryptology - Eurocrypt '92*, pages 337–346, 1992.
24. E. van Heijst, T. Pedersen, and B. Pfitzmann. New constructions of fail-stop signatures and lower bounds. *Advances in Cryptology - Crypto '92, Lecture Notes in Computer Science 740*, pages 15–30, 1993.
25. M. Waidner and B. Pfitzmann. The dining cryptographers in the disco: Unconditional sender and recipient untraceability with computationally secure serviceability. *Advances in Cryptology - Eurocrypt '89, Lecture Notes in Computer Science 434*, 1990.

Signature Schemes Based on 3rd Order Shift Registers

Chik How Tan, Xun Yi, and Chee Kheong Siew

Information Communication Institute of Singapore
School of Electrical and Electronic Engineering
Nanyang Technological University
Nanyang Avenue, Singapore 639798
{echikhow, exyi, ecksiew}@ntu.edu.sg

Abstract. In this paper, we propose two digital signature schemes based on a third order linear feedback shift register. One of them is a normal signature scheme for signing document and the other is with encryption for intended reciever. These two signature schemes are different from most of the signature schemes which are based on discrete logarithm problem, elliptic curves discrete logarithm problem, RSA or quadratic residues. The efficient computational algorithm for computing k^{th} term of a sequence is also presented. The advantage of these two schemes is that the computation is carried out in the ground field and not in an extension field. We also show that the security of these two signature schemes is equivalent to that of Schnorr signature scheme and Signed-ElGamal encryption scheme respectively.

Key words : Cryptography; digital signature; shift register; discrete logarithm

1 Introduction

1.1 Background and Previous Results

Since the concept of public-key cryptography was first invented by Diffie and Hellman [6] in 1976 which is based on the hardness of discrete logarithm problem (DL), many public key cryptosystems have been proposed and broken.

Most successful unbroken and practical public key cryptosystems are RSA [22] and elliptic curves cryptosystem [10, 16]. RSA was introduced by Rivest, Shamir and Adleman [22] in 1978 and based on the intractibility of factorization. Elliptic curves cryptosystem was discovered independently by Koblitz [10] and Miller [16] in 1985 and based on the hardness of elliptic curve discrete logarithm. All these public-key cryptographic algorithms are believed to be secured based on the assumption that no efficient algorithm has been found to solve the hardness of these problems. During the early eighties, although the public key cryptography was found not suitable for encryption due to their speeds, the concept of public key cryptosystem is risen to a new remarkable notion : digital signature. Since then, many digital signature schemes were proposed based on discrete logarithm problem and RSA, for example, blind signatures [2], ElGamal signature [7], group signature [3], Schnorr signature [24], signature with message recovery [17] and RSA-based undeniable signature [9], etc.

There were also many efforts to design alternative signature scheme that are based on other mathematical problems. For example, Ong-Schnorr-Shamir [19] schemes proposed in 1984 based on low degree polynomials modulo a composite number, namely, $x^2 + ky^2 \equiv m \pmod{n}$. This scheme was subsequently broken by Pollard and Schnorr [21] in 1987. A similar scheme was proposed by Shamir [27] in 1993 and soon was broken by Coppersmith, Stern and Vaudenary [5]. In 1997, Satoh and Araki proposed a signature scheme based on the non-commutative ring of quaternions and this scheme is a generalization of the Ong-Schnorr-Shamir [19] scheme. But, this scheme was also subsequently broken by Coppersmith [4] in 2000. In this paper, we propose an alternative method to design signature schemes based on third order linear feedback shift registers.

1.2 Our Contributions

In this paper, we propose two new digital signature schemes based on third order linear feedback shift register and the hardness of discrete logarithm problem in an extension field $GF(q^3)$. The paper is organised as follows :

In the following section, we discuss the cryptographic properties of a third order linear feedback shift register over $GF(q)$. In section three, we give an efficient computational method for computing k^{th} term of sequences on third order shift registers. In section four, we construct two digital signature schemes based on shift registers. Our signature schemes have the following features :

1. Our two signature schemes are proved to be based on the discrete logarithm problem over extension fields $GF(q^3)$.

2. The security of these two schemes are equivalent to that of Schnorr signature scheme and Signed-ElGamal encryption scheme respectively.

3. The computational complexity to compute $(k+n)^{th}$ term of sequences on third order shift registers is $11H(n) + 3$ multiplications in $GF(q)$, where $H(n)$ is the Hamming weight of integer n represented in binary representation.

In section five, we examine the security of our two digital schemes and show that the security of our schemes are equivalent to that of Schnorr signature scheme and Signed-ElGamal encryption scheme respectively.

2 Third Order Linear Feedback Shift Registers

Let p be an odd prime and e any positive number. Let $GF(q)$ be a finite field where $q = p^e$ and

$$f(x) = x^3 - ax^2 + bx - 1, \ a, b \in GF(q)$$

be an irreducible polynomial over $GF(q)$ of order $Q = q^2 + q + 1$. A sequence $\mathbf{s} = \{s_k\}$ generated by the polynomial $f(x)$ is called the third order linear feedback shift register (LFSR) sequence over $GF(q)$ if the elements of $\mathbf{s}$ satisfy

$$s_k = as_{k-1} - bs_{k-2} + s_{k-3},\ k \geq 3.$$

with the initial values $s_0 = 3$, $s_1 = a$ and $s_2 = a^2 - 2b$.

In order to distinguish the sequence $\{s_k\}$ generated by $f(x) = x^3 - ax^2 + bx - 1$, we also denote s_k by $s_k(a, b)$. Assume that $\alpha_1, \alpha_2, \alpha_3$ are all roots of $f(x)$ in the extension field of $f(x)$ over $GF(q)$, then according to Newton's formula, the elements of $\mathbf{s}$ can be represented by the symmetric k^{th} power sum of the roots as follows :

$$s_k = \alpha_1^k + \alpha_2^k + \alpha_3^k.$$

<u>Remarks</u> :

1. As the roots of $f(x)$ conjugate each other, we have $\alpha_2 = \alpha_1^q$ and $\alpha_3 = \alpha_1^{q^2}$, then we have $s_k = s_{qk} = s_{q^2k}$.

2. The number of k coprime to $q^2 + q + 1$ is $\phi(q^2 + q + 1)$ and is approximately to $\frac{3Q}{2\pi^2}$ where ϕ is a Euler function.

Let $S_k = [s_k, s_{k+1}, s_{k+2}]$ be a vector over $GF(q)$, then $\mathbf{s}$ can also be obtained through matrix representation as follows :

$$S_k = A^k S_0^T,$$

where S_0^T is a transpose of a vector $S_0 = [s_0, s_1, s_2]$ and

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & -b & a \end{pmatrix}.$$

If given an initial state S_k, we can represent a sequence s_{k+n} in the following form :

$$S_{k+n} = A^n S_k^T.$$

Let $Q = q^2 + q + 1$ and for any k such that $(k, Q) = 1$, then we define $f_k(x)$ as follows :

$$\begin{aligned} f_k(x) &= (x - \alpha_1^k)(x - \alpha_2^k)(x - \alpha_3^k) \\ &= x^3 - \sum_{i=1}^{3} \alpha_i^k x + (\sum_{i \neq j}^{3,3} \alpha_i^k \alpha_j^k)x - \prod_{i=1}^{3} \alpha_i^k \end{aligned}$$

By some calculations, we have

$$\sum_{i=1}^{3} \alpha_i^k = s_k,$$

$$\sum_{i \neq j}^{3,3} \alpha_i^k \alpha_j^k = s_{-k},$$

$$\prod_{i=1}^{3} \alpha_i^k = 1.$$

Hence, we have

$$f_k(x) = x^3 - s_k x^2 + s_{-k} x - 1. \qquad (1)$$

Lemma 1. *(Corollary 3.47, [11]) An irreducible polynomial over $GF(q)$ of degree n remains irreducible over $GF(q^k)$ if and only if $(k, n) = 1$.*

With the above Lemma, we know that an irreducible polynomial of degree 3 over $GF(p)$ will be irreducible over $GF(p^2)$. For example, $x^3 + x - 1$ is irreducible over $GF(7)$ and it is also irreducible over $GF(7^2)$ by Lemma 1.

Now, we list the following Theorem that has important properties to construct digital signature schemes :

Theorem 1. *Let $f(x) = x^3 - ax^2 + bx - 1$ be an irreducible polynomial over $GF(q)$ of order Q where $Q = q^2 + q + 1$ and let $\mathbf{s}$ be the sequence generated by $f(x)$. Then,*
(a) For any integer k with $(k, Q) = 1$ and $f_k(x) = x^3 - s_k x^2 + s_{-k} x - 1$, we have
(i) The order of $f_k(x)$ is Q,
(ii) $f_k(x)$ is irreducible iff $f(x)$ is irreducible,
(b) For any positive integers k and d, we have

$$\begin{aligned} s_k(s_d(a,b), s_{-d}(a,b)) &= s_{kd}(a,b) \\ &= s_d(s_k(a,b), s_{-k}(a,b)) \end{aligned}$$

Proof :
(a) (i) Note that α_i^k and α_i are roots of $f_k(x)$ and $f(x)$ respectively and both have the same order iff $(k, Q) = 1$. (ii) follows (i) immediately and the fact that $Q|(q^3 - 1)$ and $Q > q^2$.

(b) Since $s_d(a,b) = \alpha_1^d + \alpha_2^d + \alpha_3^d$ and $s_k(a,b) = \alpha_1^k + \alpha_2^k + \alpha_3^k$, we have

$$\begin{aligned} s_k(s_d(a,b), s_{-d}(a,b)) &= (\alpha_1^d)^k + (\alpha_2^d)^k + (\alpha_3^e)^k \\ &= \alpha_1^{dk} + \alpha_2^{dk} + \alpha_3^{dk} \\ &= s_{kd}(a,b). \end{aligned}$$

We also have the following :

$$\begin{aligned} s_d(s_k(a,b), s_{-k}(a,b)) &= (\alpha_1^k)^d + (\alpha_2^k)^d + (\alpha_3^k)^d \\ &= \alpha_1^{dk} + \alpha_2^{dk} + \alpha_3^{dk} \\ &= s_{kd}(a,b). \end{aligned}$$

Hence, we complete the proof.

We denote a coset leader of a set $\mathcal{S} = \{tq^i \bmod Q \mid 0 \le i \le 2\}$ be a smallest integer in a set $\mathcal{S}$. In fact, a coset leader is closely related to distinct polynomials. We state this property in the following theorem.

Theorem 2. *Let $f(x) = x^3 - ax^2 + bx - 1$ be an irreducible polynomial of order Q over $GF(q)$, where $Q = q^2 + q + 1$. Let k and k' be relatively prime to Q and different coset leaders modulo Q, then*

$$(s_k, s_{-k}) \neq (s_{k'}, s_{-k'}).$$

Proof : If $(s_k, s_{-k}) = (s_{k'}, s_{-k'})$, then the polynomials $f_k(x)$ and $f_{k'}(x)$ are equal and let their roots be α^k and $\alpha^{k'}$ respectively. Then α^k and $\alpha^{k'}$ are conjugate of each other. This means that there exists an integer t where $1 \le t \le 2$ such that $k = k'q^t \bmod Q$. This contradicts the fact that k and k' are different coset leaders modulo Q. Hence, we have the result.

3 Fast Computational Methods

In [8], the authors give an algorithm to calculate the k^{th} term of a third order sequence over $GF(q)$ through the following formula :

$$s_{2n} = s_n^2 - 2s_{-n}$$

$$s_n s_m - s_{n-m} s_{-m} = s_{n+m} - s_{n-2m}$$

Now, we simply extend the algorithm given in [8] to $GF(q)$ as follows :

Algorithm 1 : Given $f(x) = x^3 - ax^2 + bx - 1$ be an irreducible polynomial over $GF(q)$ and k any number, we express $k = \sum_{i=0}^{r} k_i 2^{r-i}$, where $k_i \in \{0,1\}$ and $r = \log q^3$. Let $K_j = k_j + 2K_{j-1}$ with $K_0 = k_0 \neq 0$, then s_k and s_{-k} can be calculated as follows :

1a) For $k_j = 0$, we have

$$s_{K_j-1} = s_{K_{j-1}} s_{K_{j-1}-1} - b s_{-K_{j-1}} + s_{-(K_{j-1}+1)}$$

$$s_{K_j} = s_{K_{j-1}}^2 - 2 s_{-K_{j-1}}$$

$$s_{K_j+1} = s_{K_{j-1}} s_{K_{j-1}+1} - a s_{-K_{j-1}} + s_{-(K_{j-1}-1)}$$

1a) For $k_j = 1$, we have

$$s_{K_j-1} = s_{K_{j-1}}^2 - 2 s_{-K_{j-1}}$$

$$s_{K_j} = s_{K_{j-1}} s_{K_{j-1}+1} - a s_{-K_{j-1}} + s_{-(K_{j-1}-1)}$$

$$s_{K_j+1} = s_{K_{j-1}+1}^2 - 2 s_{-K_{j-1}+1}$$

The author in [8] also further showed that with the above algorithm, to calculate the k^{th} terms of s_k and s_{-k}, we need 9log$k$ modulo q multiplications on the average.

As the algorithm 1 is only suitable for those k which is known and it is not possible to calculate those sequences s_{k+n} for unknown k and known n. We give an algorithm, called Algorithm 2, to calculate such sequence as follows :

Let p be an odd prime and e any positive integer. Let $GF(q)$ be a finite field where $q = p^e$ and

$$f(x) = x^3 - ax^2 + bx - 1,\ a, b \in GF(q)$$

be an irreducible polynomial over $GF(q)$ of order $Q = q^2 + q + 1$. Then, a sequence $\mathbf{s} = \{s_k\}$ can be calculated as follows :

$$S_{n+k} = A^n S_k^T,$$

where $S_k = [s_k, s_{k+1}, s_{k+2}]$ be a vector over $GF(q)$ and S_k^T a transpose of a vector S_k. We know that the complexity of the general method for matrix multiplication is 27 multiplications in $GF(q)$. Furthermore, Blaser [1] showed that the lower bound for multiplicative complexity of nxn matrix multiplication is $\frac{5}{2}n^2 - 3n$ and for $n = 3$, the lower bound is 13.5. Therefore, in this section, we will discuss an efficient method for matrix multiplication and then derive an efficient method to compute A^n.

For any x^n mod $f(x)$, where $n > 2$, we can express it as follows :

$$x^n = c_2 x^2 + c_1 x + c_0 \bmod f(x),$$

where c_i are in $GF(q)$ for $0 \leq i \leq 2$. Then, there is a one-to-one expression for matrix A^n as follows :

$$A^n = c_2A^2 + c_1A + c_0I \bmod f(x),$$

where I is an identity matrix.

Algorithm 2 : Given $f(x) = x^3 - ax^2 + bx - 1,\ a, b \in GF(q)$ be an irreducible polynomial over $GF(q)$ and its companion matrix A. For any integer n, we can calculate A^n as follows :
Step 1 : Let $r = \log_2(q^2+q+1)$, express $n = \sum_{i=0}^{r-1} n_i 2^i$ as a binary representation, where $n_i \in \{0, 1\}$ and $0 \leq i \leq r - 1$.
Step 2 : Compute $x^{2^2}, \cdots, x^{2^{r-1}}$ mod $f(x)$ and stores these values. Also computes A^2 as follows :

$$A^2 = \begin{pmatrix} 0 & 0 & 1 \\ 1 & -b & a \\ a & 1 - ab & a^2 - b \end{pmatrix}.$$

Step 3 : Compute $x^n = \prod_{i=0}^{r-1}(x^{2^i})^{n_i}$ mod $f(x)$ and the result, say,

$$x^n = c_2x^2 + c_1x + c_0 \bmod f(x)$$

Step 4 : Compute A^n as follows :

$$A^n = c_2A^2 + c_1A + c_0I \bmod f(x).$$

Let the two polynomial be $a_2x^2 + a_1x + a_0$ and $b_2x^2 + b_1x + b_0$, where a_i and b_i are in $GF(q)$, where $0 \leq i \leq 2$, then the usual multiplication is

$$\begin{aligned} &(a_2x^2 + a_1x + a_0) * (b_2x^2 + b_1x + b_0) \\ &= a_2b_2x^4 + (a_1b_2 + a_2b_1)x^3 + (a_2b_0 + a_0b_2 + a_1b_1)x^2 + (a_1b_0 + a_0b_1)x + a_0b_0 \end{aligned}$$

This will take 9 multiplications. Now, we do the multiplications as follows :

$$(a_0 + a_1) * (b_0 + b_1) = a_1b_1 + a_0b_1 + a_1b_0 + a_0b_0 \quad (2)$$
$$(a_1 + a_2) * (b_1 + b_2) = a_1b_1 + a_2b_1 + a_1b_2 + a_2b_2 \quad (3)$$
$$(a_0 + a_2) * (b_0 + b_2) = a_0b_0 + a_2b_0 + a_0b_2 + a_2b_2 \quad (4)$$

Then, we have

$$\begin{aligned} a_1b_2 + a_2b_1 &= (3) - a_1b_1 - a_2b_2 \\ a_1b_0 + a_0b_1 &= (2) - a_1b_1 - a_0b_0 \\ a_2b_0 + a_0b_2 + a_1b_1 &= (4) - a_0b_0 - a_2b_2 + a_1b_1 \end{aligned}$$

With the above calculations, we only require to carry out 6 multiplications in $GF(q)$ which is faster than the usual polynomial multiplications. For x^4 and $x^3 \bmod f(x)$ will take 5 multiplications in $GF(q)$. Hence, the total number of multiplications in $GF(q)$ for two polynomials multiplication modulo $f(x)$ is 11.

The complexity of an algorithm of computing s_{k+n} is discussed in the following theorem.

Theorem 3. *Given a matrix A and initial vector $S_k = [s_k, s_{k+1}, s_{k+2}]$, then there is an efficient algorithm that computes $S_{k+n} = [s_{k+n}, s_{k+n+1}, s_{k+n+2}]$ by $S_{k+n} = A^n S_k^T$ with the computational complexity of $(11H(n)+3)$ multiplications in $GF(q)$ and $3(r-2)$ $log_2 q$ bits of memory to store polynomials $x^{2^2}, \cdots, x^{2^{r-1}}$ mod $f(x)$, where $H(n)$ is the Hamming weight of n in the binary representation and $r = \log_2(q^2+q+1)$.*

Proof : In step 2 of the Algorithm 2, it is easily calculated that the number of memory to store $x^{2^i} \bmod f(x)$ for $2 \le i \le r-1$ is about $3(r-2)\log_2 q$. There is no storage required for A^2 as the first row is (0,0,1), the second row is the polynomial representation of $f(x)$ and the third row is the same as that of the coefficient of $x^{2^2} \bmod f(x)$. In step 3, the number of polynomials multiplication required is $H(n)$ and the complexity of two polynomial multiplications take 11 multiplications in $GF(q)$. Hence, we have the result and completed the proof.

Remarks :
1. If $\log q^3 = 1024$, then the memories required to store x^{2^i}, for $2 \le i \le r-1$ is 85 K bytes.
2. The above algorithm is suitable for parallel computing implementation, for example, we can implement three multipliers in $GF(q)$ with less complexity than one multiplier in $GF(q^3)$.

4 Digital Signature Schemes

A digital signature is an electronic version of handwritten signature for digital documents and these are used in many applications, for example, signing document, electronic cash, electronic payment system, electronic voting and electronic auction, etc. We will discuss some of the applications by the different requirements of the signature schemes in the following subsections. A signature scheme normally involves three stages, that is key generation, signature generation and verification of message. We now give a formal definition as follows :

Definition 1. *A signature is defined as :*
- The key generation algorithm $\mathcal{G}$. On input 1^k, where k is the security parameter, the algorithm $\mathcal{G}$ produces a pair (K_P, K_S) of public and secret keys.
- The signing algorithm Σ. Given a message m and a pair of public and secret keys (K_P, K_S), Σ produces a signature σ.
- The verification algorithm $\mathcal{V}$. Given a signature σ, a message m and a public key K_P, $\mathcal{V}$ tests whether σ is a valid signature of m with respect to K_P.

Now, we describe various digital signature schemes in the following subsection:

4.1 A Normal Digital Signature Scheme

This scheme is usually used as a digital signature for signing document to provide non-repudiation and anyone who knows the public key could verify the signature. Among the best known signature schemes were constructed by ElGamal [7] and Schnorr [24]. Both were constructed from finite field over $GF(p)$. In this subsection, we construct a new signature scheme which is based on 3^{th} order shift registers. The construction is described as follows :

Key Generation : Let h be a one way hash function, p an odd prime number and e any positive number. Let $q = p^e$ and $f(x) = x^3 - ax^2 + bx - 1$ be an irreducible polynomial over $GF(q)$ of order $Q = q^2 + q + 1$. A signer, Alice, first chooses a secret key z that satisfies $0 < z < Q$ and $(z, Q) = 1$ and computes her public key $s_z(a, b), s_{-z}(a, b)$ by using Algorithm 1.

Signature Generation : To sign a message m, Alice performs the following to generate a signature for message m :
(S-1) Choose a random number k that satisfies $0 < k < Q$ and $(k, Q) = 1$.
(S-2) Compute $s_k(a, b), s_{k+1}(a, b), s_{k+2}(a, b)$ by using algorithm 1.
(S-3) Compute $h_1 = h(m, s_k, s_{k+1}, s_{k+2})$. If $(h_1, Q) \neq 1$, then go back to S-1 otherwise proceed to S-4.
(S-4) Compute t such that $t + k = h_1 z \bmod Q$.

Then, $(t, s_k, s_{k+1}, s_{k+2})$ is a signature for message m that Alice sends to Bob.

Signature Verification : After Bob receives the signature $(t, s_k, s_{k+1}, s_{k+2})$, he first computes $h_1 = h(m, s_k, s_{k+1}, s_{k+2})$ and checks the following equation :

$$s_{k+t} = s_{zh_1}$$

where s_{k+t} and s_{zh_1} can be computed using Algorithm 2 and Algorithm 1 respectively as follows :

$$S_{k+t} = A^t S_k$$

and

$$s_{zh_1} = s_{h_1}(s_z(a, b), s_{-z}(a, b))$$

If s_{k+t} and s_{zh_1} are equal, then it is a correct signature, otherwise it is incorrect.

The security of this signature scheme will be discussed in section 5.

4.2 A Signed Encryption Scheme

In some cases, we want to encrypt a message with a signature for an intended receiver. This type of signature can only be verified and decrypted by the intended receiver. This is different from the previous signature and could only be verified by intended receiver, which is called signed encryption. One of the best well-known signed encryption scheme is signed ElGamal encryption scheme [26]. In this subsection, we construct a new signed encryption scheme as follows:

Key Generation : Let h be a one way hash function, p an odd prime number and e any number. Let $q = p^e$ and $f(x) = x^3 - ax^2 + bx - 1$ be an irreducible polynomial over $GF(q)$ of order $Q = q^2 + q + 1$. A signer, Alice, first chooses a secret key z that satisfies $0 < z < Q$ and $(z, Q) = 1$ and computes her public key $s_z(a, b), s_{-z}(a, b)$ by using Algorithm 1. If a signer, Alice, wishes to send an encrypted and signed message to Bob, she needs to know Bob's public key. First, Bob chooses a secret key r that satisfies $0 < r < Q$ and $(r, Q) = 1$ and computes his public key $s_r(a, b), s_{-r}(a, b)$ by using Algorithm 1.

Signature Generation : To sign and encrypt a message m, Alice performs the following steps for a signature of message m :
(SS-1) Choose random number u that satisfies $0 < u < Q$ and $(u, Q) = 1$ and compute s_u, s_{-u} and $s_{ur} = s_u(s_r(a, b), s_{-r}(a, b))$ by using Algorithm 1. If $s_{ur} = 0$, then go back to SS-1 otherwise proceed to SS-2.
(SS-2) Choose random number k that satisfies $0 < k < Q$ and $(k, Q) = 1$ and compute s_k, s_{k+1}, s_{k+2} by using Algorithm 1.
(SS-3) Compute $m_1 = ms_{ur}$ and $h_1 = h(s_k, s_{k+1}, s_{k+2}, s_u, s_{-u}, m_1)$. If $(h_1, Q) \neq 1$, then go back to SS-2, otherwise proceed to SS-4.
(SS-4) Compute t such that $t + k = h_1 z \bmod Q$.

Then Alice will send $(s_k, s_{k+1}, s_{k+2}, s_u, s_{-u}, m_1, t)$ to Bob as a signature and encryption of message m.

Signature Verification : Upon receipt of the signature $(s_k, s_{k+1}, s_{k+2}, s_u, s_{-u}, m_1, t)$, Bob first computes $h_1 = h(s_k, s_{k+1}, s_{k+2}, s_u, s_{-u}, m_1)$ and checks the following equation :

$$s_{k+t} = s_{h_1 z},$$

where s_{k+t} and $s_{h_1 z}$ can be calculated using Algorithm 2 and Algorithm 1 respectively as follows :

$$S_{k+t} = A^t S_k$$

and

$$s_{h_1 z} = s_{h_1}(s_z(a, b), s_{-z}(a, b)).$$

If s_{k+t} and $s_{h_1 z}$ are equal, then it is a correct signature, otherwise it is incorrect. For the correct signature, Bob is able to obtain message m as follows:

$$m = m_1 / s_{ur},$$

as Bob knows his own secret key r and he is able to calculate s_{ur} as $s_{ur} = s_r(s_u(a,b), s_{-u}(a,b))$ by using Algorithm 1.

5 Security of Signature Schemes

To cryptanalyse a signature scheme, it is basically done by forgeries. There are three kinds of well known cryptanalysis for forgeries.
a) Total break : An adversary is able to recover the secret key of the signer. This is the most serious attack of the system. It means that the scheme cannot be used totally.
b) Universal forgery : An adversary is able to construct an efficient algorithm which is able to sign any message.
c) Existential forgery : An adversary is able to generate new message-signature pair but the message may not be meaningful.

Based on the above type of attacks, we analyse the security of these signature schemes in the following Theorems. Normal and signed encryption schemes are quite similar except that the later is with encryption. We will analyse the two schemes together. First, we show that the security of these signature schemes are based on the difficulty of solving the discrete logarithm problem in $GF(q^3)$ and in general linear group. Following that, we show that the security of a normal signature scheme and a signed encryption scheme are equivalent to that of Schnorr scheme and Signed-ElGamal encryption scheme.

Definition 2. *The discrete logarithm problem (DL) is the following : given a finite cyclic group $\mathcal{G}$, a generator g of $\mathcal{G}$ and an element a, find an integer x, $1 \leq x \leq \mid \mathcal{G} \mid -1$ such that $a = g^x$ holds.*

Now, we define a similar concept of discrete logarithm problem in shift register as follows :

Definition 3. *The shift register type of discrete logarithm problem (SR-DL) is the following : given (s_k, s_{-k}) and (s_1, s_{-1}), find an integer k, $1 \leq k \leq q^3 - 1$ such that $s_k = s_k(s_1, s_{-1})$ holds.*

Theorem 4. *The SR-DL is equivalent to DL problem.*

Proof :

($\Longrightarrow$) Given (s_k, s_{-k}) and (s_1, s_{-1}), then a polynomial $f_k(x)$ can be formed by Theorem 1 and its roots can be easily calculated. Let α and β be the roots representation of $f(x)$ and $f_k(x)$ in the field $GF(q^3)$ respectively. Then, there exists some i, $0 \leq i \leq 2$ such that $\beta = \alpha^{q^i k}$ in $GF(q^3)$. Now, if there exists an efficient algorithm to find k from (s_k, s_{-k}) and (s_1, s_{-1}), then we can also easily find $q^i k$ such that $\beta = \alpha^{q^i k}$ which is a DL problem. Hence, SR-DL implies DL problem.

($\Longleftarrow$) Given α and β, if we can find k such that $\beta = \alpha^k$, then we have $\beta^{q^i} = \alpha^{q^i k}$ for $0 \leq i \leq 2$ and we can form the polynomial $f_k(x)$. Then, we have (s_k, s_{-k}). Hence, for a given (s_k, s_{-k}) and (s_1, s_{-1}), there is an efficient algorithm to find k such that $s_k = s_k(s_1, s_{-1})$. Therefore, this shows that DL problem implies SR-DL. Hence, we proved that SR-DL is equivalent to DL problem.

In [14], Menezes and Wu showed that the discrete logarithm problem in general linear group $GL(n, q)$, a set of nxn nonsingular matrices over $GF(q)$, can be reduced to the discrete logarithm in the small extension field of $GF(q)$. The discrete logarithm problem in $GL(n, q)$ is to find k, given nxn matrices A and B and in $GF(q)$ such that $B = A^k$. Let the factorization of the characteristic polynomial $p_A(x)$ of A over $GF(q)$ be $p_A(x) = f_1^{e_1} f_2^{e_2} \cdots f_u^{e_u}$, where the degree of f_i is m_i. Menezes and Wu showed the following theorem :

Theorem 5. *[14] The discrete logarithm problem in $GL(n, q)$ is reduced to the discrete logarithm in $GF(q^{m_i})$, $1 \leq i \leq u$.*

If a characteristic polynomial is irreducible and we called its companion matrix an irreducible matrix, then we have the following corollary.

Corollary 1. *The discrete logarithm of irreducible matrix in $GL(n, q)$ is reduced to the discrete logarithm in $GF(q^n)$.*

From the above Theorem 4, we know that it is impossible for adversary to find the secret key with the knowledge of s_z and s_{-z}. Furthermore, it is also difficult to find random number k. From the Corollary 1 and Theorem 4, it is also difficult to find $k + t$ with the knowledge of s_{k+t}. Hence, the two signature schemes can withstand the first attack.

In [20] and [26], the authors have showed that Schnorr signatures over $GF(p)$ is secured against the adaptive chosen message attack using random oracle model. In the following theorem, we show that our normal signature scheme and Schnorr signature scheme are equivalent under the same group of order Q. Let g be a generator of a group of order Q and h is a one-way hash function. Let z be a secret key and public key is g^z. Choose a random number k, then a signature of a message m is generated as follows:

$$h_1 = h(m, g^k), k + t = h_1 z \text{ and } g^{k+t} = g^{h_1 z}.$$

The signature of m is (g^k, t).

Theorem 6. *Our normal signature scheme is equivalent to Schnorr signature scheme.*

Proof :
($\Longrightarrow$) Given our normal signature scheme, we have a signature $(t, s_{k-2}, s_{k-1}, s_k)$ and the following relations :

$$s_{k+t} = s_{h_1 z}$$

and

$$k + t = h_1 z \bmod Q.$$

Hence, we have

$$-(k+t) = -h_1 z \bmod Q$$

and

$$s_{-(k+t)} = s_{-h_1 z}.$$

As we know $(h_1 z, Q) = 1$, where $Q = q^2 + q + 1$, then $(k + z, Q) = 1$. Then, by equation (1), we can form polynomials $f_{k+t}(x)$ and $f_{h_1 z}(x)$. These two equations are equal and their roots can easily be calculated. Let α^{k+t} and $\alpha^{h_1 z}$ be their roots of $f_{k+t}(x)$ and $f_{h_1 z}(x)$ respectively, where α is a root of $f(x)$ and in $GF(q^3)$. Then, we have $\alpha^{k+t} = \alpha^{h_1 z}$ with $k + t = h_1 z$ which is Schnorr signature scheme. Hence, our normal signature scheme is reduced to Schnorr signature scheme.

($\Longleftarrow$) Similarly, given Schnorr signature scheme, we have $\alpha^{k+t} = \alpha^{h_1 z}$ and $k+t = h_1 z$. We have

$$(\alpha^{k+t})^q = (\alpha^{h_1 z})^q$$

and

$$(\alpha^{k+t})^{q^2} = (\alpha^{h_1 z})^{q^2}.$$

Therefore, we have $s_{k+t} = s_{h_1 z}$ and $k + t = h_1 z$, which forms our normal signature scheme. Hence, our normal signature scheme and Schnorr signature scheme are equivalent. This completes the proof.

As our signed encryption scheme is quite the same as a normal signature scheme, except that the first is with encrypted message. Therefore, we have the following theorem.

Theorem 7. *Our signed encryption scheme is equivalent to Signed-ElGamal encryption signature scheme.*

Hence, from Theorem 6 and Theorem 7, we can conclude the following theorem.

Theorem 8. *(i) The security of our normal signature scheme is equivalent to the security of Schnorr signature scheme.*
(ii) The security of our signed encryption scheme is equivalent to the security of Signed-ElGamal encryption scheme.

6 Conclusion

In this paper, we proposed two digital signature schemes based on a 3rd order linear feedback shift register over $GF(q)$. We show a method of implementing these two schemes efficiently. We show that the security of the two schemes relies on the discrete logarithm over $GF(q^3)$. Furthermore, we also show that the security of these two signature schemes is equivalent to that of Schnorr signature scheme and Signed-ElGamal encryption scheme respectively. The methods presented here can lead to many construction of digital signature schemes over $GF(q^3)$ and it is expected that more signatures could be constructed from 3rd order linear feedback shift register. The other advantage of this method is that the computation is carried out in $GF(q)$ and not in $GF(q^3)$.

References

[1] M. Blaser. A $\frac{5}{2}n^2$ - lower bound for multiplicative complexity of nxn-matrix multiplication. In *STACS 2001*, Lecture Notes in Computer Science **2010**, Springer-Verlag, (2001) 99-109.

[2] D. Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology - Crypto'82*, Plenum Press, New York, (1983) 199-203.

[3] D. Chaum and E. V. Heyst. Group signatures. In *Advances in Cryptology - Crypto'91*, Lecture Notes in Computer Science **547**, Springer-Verlag, (1991) 257-265.

[4] Don Coppersmith. Weakness in quaternion signatures. *Journal of Cryptology*, **14**, No. 2, (2001) 77-85.

[5] Don Coppersmith, J. Stern and S. Vaudenay. Attacks on the birational permutation signature schemes. In *Advances in Cryptology - Crypto'93*, Lecture Notes in Computer Science **773**, Springer-Verlag, (1993) 207-221.

[6] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, **22** (1976) 644-654.

[7] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, **31** (1985) 469-472.

[8] G. Gong and L. Harn. Public key cryptosystems based on cubic finite field extensions. *IEEE Transactions on Information Theory*, **45** (1999) 2601-2605.

[9] R. Gennaro, H. Krawczyk and T. Rabin. RSA-based undeniable signatures. *Journal of Cryptology*, Vol **13**, No. 4, (2000) 397-416.

[10] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, **48**, No. 177, (1987) 203-209.

[11] R. Lidl and H. Niederreiter. *Introduction to Finite Fields and Their Applications.* Cambridge University Press, 1986.

[12] U. M. Maurer and S. Wolf. The relationship between breaking the Diffie-Hellman protocol and computing discrete logarithms. *SIAM J. COMPUT*, Vol **28**, No 5, (1999) 1689-1721.

[13] A. J. Menezes, P. C. van Oorschot and S. A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1997.

[14] A. J. Menezes and Y. H. Wu. The discrete logarithm problem in GL(n,q). *Ars Combinatoria*, **47**, (1998) 23-32.

[15] R. C. Merkle and M. E. Hellman. Hiding information and signatures in trapdroor knapsacks. *IEEE Transactions on Information Theory*, **24** (1978) 525-530.

[16] V. S. Miller. Use of elliptic curves in cryptography. In *Advances in Cryptology - Crypto'85*, Lecture Notes in Computer Science **218**, Springer-Verlag, (1986) 417-426.

[17] K. Nyberg and R. A. Rueppel. Message recovery for signature schemes based on the discrete logarithm problem. In *Advances in Cryptology - Eurocrypt'94*, Lecture Notes in Computer Science **547**, Springer-Verlag, (1994) 175-190.

[18] A. Odlyzko. Discrete logarithms : The past and the future. *Designs, Codes and Cryptography*, **19**, (2000) 129-145.

[19] H. Ong, C. P. Schnorr and A. Shamir. An efficient signature scheme based on quadratic equations. In *16th ACM Symposium Theory of Computation*, (1984) 208-216.

[20] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, **13**, No. 3, (2000) 361-396.

[21] J. M. Pollard and C. P. Schnorr. An efficient solution of the congruence $x^2+ky^2 \equiv m \pmod{n}$. *IEEE Transactions on Information Theory*, **33**, (1987) 702-709.

[22] R. L. Rivest, A. Shamir and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. *Communications of the ACM*, **21** (1978) 120-126.

[23] T. Satoh and K. Araki. On construction of signature scheme over a certain non-commutative ring. *IEICE Transactions on Fundamentals*, Vol **E80-A**, No. 1, (1997) 40-45.

[24] C. P. Schnorr. Efficient signature generation for smart cards. *Journal of Cryptology*, **4**, (1991) 161-174.

[25] C. P. Schnorr and M. Jakobsson. Security of discrete log cryptosystems in the random oracle + generic model. In *Conference on The Mathematics of Public-Key Cryptography*, The Fields Institute, Toronto, Canada. 1999.

[26] C. P. Schnorr and M. Jakobsson. Security of signed ElGamal encryption. In *Advances in Cryptology - Asiacrypt'00*, Lecture Notes in Computer Science **1976**, Springer-Verlag, (2000) 73-89.

[27] A. Shamir. Efficient signature schemes based on birational permutations. In *Advances in Cryptology - Crypto'93*, Lecture Notes in Computer Science **773**, Springer-Verlag, (1993) 1-12.

[28] V. Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in Cryptology - Eurocrypt'97*, Lecture Notes in Computer Science **1233**, Springer-Verlag, (1997) 256-266.

Anonymous Statistical Survey of Attributes

Toru Nakanishi and Yuji Sugiyama

Department of Communication Network Engineering,
Faculty of Engineering, Okayama University,
3-1-1 Tsushimanaka, Okayama 700-8530, Japan
{nakanisi,sugiyama}@cne.okayama-u.ac.jp

Abstract. A distributor of digital contents desires to collect users' attributes. This is because the distributor can grasp the image of users, and work out the marketing strategy. On the other hand, the users do not desire to offer the attributes owing to the privacy protection. For anonymous surveys, a protocol to generate statistical results of the attributes is previously proposed, where the extra information is not released beyond the statistical results. However, in the simple application of this protocol to the surveys, the correctness of the statistical results is not assured, since the users do not necessarily offer the correct attributes. In this paper, under the assumption that some trusted third parties exist, an anonymous statistical survey system of attributes with the correctness is proposed.

Keywords: Statistical survey of attributes, Anonymity, Group signature scheme, Shuffle, Threshold cryptosystem

1 Introduction

Recently, digital contents have been distributed on the computer network for the commercial purpose, where the distributor sends users the digital contents including texts, images and sound, while the users watch advertisements or pay the distributor the money. It is desirable that these services are conducted anonymously, since otherwise the distributor can collect the history that indicates which contents a user utilizes and furthermore the distributor may leak the history to others. By contrast, the distributor wants to grasp the image of users, since the distributor can work out the strategy according to the image. This may also benefit the users, since they may obtain the more suitable contents. One method to grasp the image is to collect the attributes of the users, which are concretely the gender, age, job and so on. However, even if offering the attributes during the services is conducted anonymously, offering many attributes may help the distributor to trace the identity of the user. Only the statistical results of the attributes may give the distributor useful information to grasp the image.

In [1], Sako proposes a protocol executed among several trusted third parties (TTPs) in order to generate statistical results of attributes for anonymous survey systems. The merit of this protocol is that it releases no extra information beyond the statistical results. Each TTP is in charge of each attribute type, and the TTP obtains only the information of the corresponding attribute type. The TTP's input is the set of the ciphertexts encrypted with the TTP's public key from attribute values on the corresponding attribute type. The output is only the statistical result of the attribute type. By using this protocol, the anonymous survey system of attributes is simply constructed as follows: A user sends the distributor the ciphertexts of the user's attribute values, and the distributor collects them. After collecting a certain amount of ciphertexts, the distributor gives the TTPs them to execute this protocol. Then, the distributor can obtain the statistical results of all attribute types without the extra information. This protocol has a mechanism to detect a TTP that does not obey the protocol, and thus it is assured that the results are correct if the inputs are correct, that is, the attribute values in the ciphertexts are correct. However, in the above simple survey system, since the user does not necessarily send the correct attribute values, it is not assured that the statistical results are correct.

This paper proposes an anonymous statistical survey system of attributes with the correctness. In this system, a set of multiple TTPs, called trustees, is in charge of every attribute type. Unless a quorum of the trustees is corrupted, the statistical results are generated without releasing any extra information to even each trustee. In addition, to verify that a user commits the correct attribute values, this system introduces an attribute authority as the additional TTP. The attribute authority assures the correspondence between the user's genuine attribute values and the committed values. In the proposed system, extensions of the group signature scheme [2], verifiable shuffle protocols [3] and the threshold cryptosystem [4] produce the correctness along with the anonymity.

This paper is organized as follows: Section 2 shows a model of the anonymous statistical survey system of attributes. Next, as the cryptographic tools used in the proposed system, signatures based on zero-knowledge proofs of knowledge (SPKs) which are also used in the group signature scheme, a verifiable shuffle protocol and a threshold cryptosystem are reviewed in Section 3. Then, the overview and detailed construction of an anonymous statistical survey system of attributes are described in Section 4 and 5, respectively. Its security is discussed in Section 6. Finally, Section 7 concludes this paper.

2 A Model of Anonymous Statistical Survey System of Attributes

The participants in an anonymous statistical survey system of attributes are an attribute authority, users, a distributor, and trustees. The attribute authority is a TTP, and the authority assures the correspondence between the user's genuine attribute values and the encrypted values which are offered from the user. The trustees are also TTPs, and it is assumed that a quorum of them is not corrupted. The survey system consists of the setup, registration, offering, and generating protocols. In the setup protocol, the secret and public keys of the attribute authority and the trustees are set up. In the registration protocol, a user generates his secret key and public key, and is issued the attribute certificate for a registered value from the attribute authority. In the offering protocol, a user sends the distributor the encrypted values correspondent with user's attribute values, whose validity is assured by the attribute certificate. In the generating protocol, given the encrypted values of many users, a quorum of the trustees outputs the statistical result of every attribute type.

The requirements of anonymous survey system of attributes are as follows:

Correctness: The statistical result is correct if the participants obey the protocols. If a participant disobeys the protocols, it can be detected.

Anonymity: The offering protocol is conducted anonymously. That is, the other party can not identify the user from a transcript of this protocol, and can not also link two transcripts w.r.t. the sameness of the user. Furthermore, the other party can not link a transcript of the offering protocol to the corresponding attribute values.

3 Preliminaries

3.1 Signatures Based on Zero-Knowledge Proofs of Knowledge

The proposed system uses an extension of the group signature scheme in [2]. In this scheme, as primitives to prove the knowledge of secret values without leaking any useful information, signatures based on zero-knowledge proofs of knowledge (SPKs) are used. Since the proposed system also uses some types of SPKs, this subsection reviews the SPKs. These are converted from zero-knowledge proofs of knowledge (PKs) by the so-called Fiat-Shamir heuristic [5]. That is, the prover determines the challenge by applying a collision-resistant hash-function to the commitment and the signed message and then computes the response as usual. The resulting signature consists of the challenge and the response. Such SPKs can be proven to be secure in the random oracle model [6] given the security of the underlying PKs. Let $SPK\{(\alpha, \beta, \ldots) : \mathit{Predicates}\}(m)$ be the signature on message m proving that the signer knows $\alpha, \beta, \ldots$ satisfying the predicates *Predicates*. In this notation, Greek letters denote the secret knowledge and the other letters denote public parameters between the signer and the verifier. The proposed system is based on the hardness of the discrete logarithm problem as well as the group signature scheme [2]. Thus, the relations among the discrete logarithms from cyclic groups are used as the predicates to prove. In the following, let G be a cyclic group with order q. The discrete logarithm of $y \in G$ to the base $z \in G$ is $x \in Z_q$ satisfying $y = z^x$ if such an x exists. This is extended to the representation of $y \in G$ to the bases $z_1, z_2, \ldots z_k \in G$ which is $x_1, x_2, \ldots x_k \in Z_q$ satisfying $y = z_1^{x_1} \cdot z_2^{x_2} \cdots z_k^{x_k}$ if such x_i's exist. The e-th root of the discrete logarithm of $y \in G$ to the base $z \in G$ is $x \in Z_q$ satisfying $y = z^{(x^e)}$ if such an x exists.

The first type of SPK is the signature proving the knowledge of representations of $y_1, \ldots, y_w \in G$ to the bases $z_1, \ldots, z_v \in G$ on message m, and it is denoted as

$$SPK\{(\alpha_1,\dots,\alpha_u) : (y_1 = \prod_{j=1}^{l_1} z_{b_{1j}}^{\alpha_{e_{1j}}})$$
$$\wedge \cdots \wedge (y_w = \prod_{j=1}^{l_w} z_{b_{wj}}^{\alpha_{e_{wj}}})\}(m),$$

where constants $l_i \in \{1,\dots v\}$ indicate the number of bases on representation of y_i, the indices $e_{ij} \in \{1,\dots,u\}$ refer to the elements $\alpha_1,\dots,\alpha_u$ and the indices $b_{ij} \in \{1,\dots,v\}$ refer to the elements $z_1,\dots,z_v$. For example, $SPK\{(\alpha,\beta) : y_1 = z_1^{\alpha} \wedge y_2 = z_1^{\beta} z_2^{\alpha}\}(m)$ is the SPK on m of an entity knowing the discrete logarithm of y_1 to the base z_1 and a representation of y_2 to the bases z_1 and z_2, where the z_2-part of this representation equals the discrete logarithm of y_1 to the base z_1. The second type is the SPK proving the knowledge of the e-th root of the discrete logarithm of $y \in G$ to the base $z \in G$ on m, and is denoted as

$$SPK\{\beta : y = z^{\beta^e}\}(m).$$

The third type is the SPK proving the knowledge of the e-th root of the z_2-part of a representation of $y \in G$ to the bases $z_1, z_2 \in G$ on m, and is denoted as

$$SPK\{(\gamma,\delta) : y = z_1^{\gamma} z_2^{\delta^e}\}(m).$$

The efficient constructions of these types of signatures are concretely described in [2].

3.2 Shuffle and Threshold Cryptosystem

We define a shuffle protocol as the following protocol: Given a list of ciphertexts $(c_1,\dots,c_N)$, multiple parties output a list of permuted ciphertexts $(c'_1,\dots,c'_N)$ satisfying $Dec(c_j) = Dec(c'_{\pi(j)})$ for all j, where Dec is the decryption function and π is a permutation. Furthermore, it is infeasible to determine $\pi(j)$ for any j with non-negligibly better probability unless the parties cooperate, since the permuted ciphertexts are randomized. In [3], a shuffle protocol of the ElGamal encryption is proposed for constructing the Mix-net. Since the parties may disobey the shuffle, the shuffle should be *verifiable*, which is brought by the parties' proving the correctness of their actions without revealing their random factors. In [3] a PK to prove the correctness is also proposed, which is used in this survey system together with the shuffle protocol.

We also use the idea of threshold cryptosystem [4], where a quorum of parties cooperatively decrypts a ciphertext w.r.t. the parties' public key. As well as the above shuffle, since the parties may disobey the decryption, the decryption protocol should be also verifiable, that is, the correctness should be proved without revealing their secret keys. In [3], an ElGamal threshold decryption protocol and a PK to prove the correctness are also proposed.

4 Overview

Before the proposed system is described, the overview is shown. In the system, the group signature scheme [2] is used to assure that a user offers the correct attribute values.

Group signature scheme [2]: The group signature scheme allows a group member to anonymously sign on group's behalf. Furthermore, the anonymity of the signature can be revoked by a revocation manager. The scheme consists of setup, registration, signing, verification, anonymity revocation protocols. The protocols are informally as follows:

Setup protocol: The group manager sets up public and secret keys on a digital signature scheme, and the revocation manager sets up public and secret keys on a public encryption scheme.

Registration protocol: When a user wants to participate in the group, the user sends the group manager $f(x)$ together with his identity, where f is a public one-way function and x is the user's secret key. Then, the manager returns his digital signature on $f(x)$, denoted as $DS(f(x))$, as the membership certificate.

Signing and verification protocol: As the group signature on a message m, a group member computes $d = Enc(f(x))$ and $p = SPK\{(\alpha, \beta) : d = Enc(f(\alpha)) \wedge \beta = DS(\alpha)\}(m)$, where Enc is the encryption function with the revocation manager's public key. Its verification is accomplished by verifying the SPK.

Anonymity revocation protocol: When the anonymity of a signature (d, p) is revoked, the revocation manager decrypts d to obtain $f(x)$. Through the registration transcript including $f(x)$, the identity of the signer is found.

The proposed anonymous statistical survey system uses the group signature scheme, where the group manager is replaced by the attribute authority and the revocation manager is replaced by the trustees. The informal descriptions of protocols are as follows. For the simplicity, the case of one attribute type is only described.

Setup protocol: The setup protocol of the group signature scheme is conducted, where the trustees cooperatively set up keys of the threshold cryptosystem.

Registration protocol: The registration protocol of the group signature scheme is conducted, where the attribute authority preserves the attribute value of the registering user instead of the identity. Thus, each $f(x)$ is correspondent with each attribute value, and $f(x)$ is called the attribute index. The list of the indices and attribute values of all users is made public. Furthermore, the membership certificate plays the role of the attribute certificate.

Offering protocol: To offer the attribute value, the user sends the distributor the user's group signature on a random message. The distributor can check the correctness of the attribute value by verifying the signature. The distributor collects signatures of users.

Generating protocol: To obtain the statistical result of the attribute type, the distributor sends the trustees the received signatures. Each signature includes the ciphertext of the attribute index, that is $Enc(f(x))$. The trustees cooperatively shuffle the ciphertexts. After that, for each ciphertext, the following steps are executed.

1. The trustees cooperatively shuffle all registered attribute indices, where the indices are randomized by the same random factor while the ciphertext is randomized by the same factor.
2. The quorum of trustees decrypts the randomized ciphertext to make the decrypted value correspondent with a randomized attribute index, which indicates only the attribute value, not the attribute index itself.

By counting the attribute values made correspondent with all ciphertexts, the statistical result is computed. In this protocol, furthermore, the correctness of the shuffles and decryption are proved with zero-knowledge.

The correctness of this system is satisfied as follows: The SPK in the group signature assures that the signature includes the ciphertext of the attribute index. The correspondence between the attribute index and the attribute value is assured by the attribute authority through the digital signature. Furthermore, the proofs of the shuffles and decryption assure that the statistical result is correctly computed from the ciphertexts.

The anonymity is satisfied as follows: Since the group signature is anonymous, the transcript of the offering protocol is anonymous. In the generating protocol, the ciphertexts are shuffled, and are made correspondent to the attribute values, from which anyone can compute only statistical result. Furthermore, proving the correctness of the shuffles and decryption reveals no information.

5 An Anonymous Statistical Survey System of Attributes

In this section, the detailed protocols are described. For the simplicity of description, assume that there are two attribute types, A and B. Let U be a user, and let D be a distributor. Let AA be the attribute authority. Let $T_1, \ldots, T_\ell$ be the trustees.

Assume that the communication between the participants is authenticated (e.g., by the digital signature) except the offering protocol. Let $\tilde{0}$ be the empty string. If S is a set, $e \in_R S$ means that e is chosen at random from S according to the uniform distribution.

5.1 Setup Protocol

In this protocol, AA and trustees $T_1, \ldots, T_\ell$ generate the secret and public keys.

1. AA computes an RSA modulus n, two public exponents $e_1, e_2 > 1$, and two integers $f_1, f_2 > 1$. Note that e_1, e_2, f_1 and f_2 must satisfy that solving the congruence $f_1 x^{e_1} + f_2 \equiv v^{e_2} \pmod{n}$ is infeasible. The choices for e_1, e_2, f_1 and f_2 are discussed in [2]. AA chooses a cyclic group G of order n. Then, AA chooses bases $g_A, g_B, h \in G$ in which it is infeasible to compute and compare the discrete logarithms. AA's public key is $(n, e_1, e_2, f_1, f_2, G, g_A, g_B, h)$, and the secret key is the factorization of n.

2. The trustees cooperatively generate their secret keys and the public key, where a secret key $x_T \in_R Z_n^*$ are shared with the trustees by Shamir's threshold scheme. Then, they publish $y_T = h^{x_T}$ as the trustees' public key. Note that the normal (threshold) ElGamal encryption is constructed on a multiplicative subgroup of order prime q in Z_p^* such that $p = 2q + 1$, while the encryption in this system is constructed on a cyclic group of order n that is the RSA modulus, since the underlying group signature scheme [2] uses the encryption.

5.2 Registration Protocol

When a user U participates in this system, U is issued the attribute certificate from AA by this protocol, which is similar to the registration of the original group signature scheme. Assume that AA is convinced of U's attribute values.

1. U chooses $x \in_R Z_n^*$ to compute $y = x^{e_1} \pmod{n}$, $z_A = g_A^y$ and $z_B = g_B^y$. Then, U chooses $r \in_R Z_n^*$ to compute $\tilde{y} = r^{e_2}(f_1 y + f_2) \pmod{n}$ and the following SPKs:

$$V_1 = SPK\{\alpha : z_A = g_A^{\alpha^{e_1}}\}(\tilde{0}),$$
$$V_2 = SPK\{\beta : g_A^{\tilde{y}} = (z_A^{f_1} g_A^{f_2})^{\beta^{e_2}}\}(\tilde{0}),$$
$$V_3 = SPK\{\gamma : z_A = g_A^{\gamma} \wedge z_B = g_B^{\gamma}\}(\tilde{0}).$$

   Note that V_1 and V_2 are the same as the original. V_3 proves the correctness of z_A and z_B.
   U sends $(\tilde{y}, z_A, z_B, V_1, V_2, V_3)$ to AA.
2. If V_1, V_2 and V_3 are correct, AA sends $\tilde{v} = \tilde{y}^{1/e_2} \pmod{n}$ to U.
3. U computes $v = \tilde{v}/r \pmod{n}$ to obtain the attribute certificate (x, v), where $v \equiv (f_1 x^{e_1} + f_2)^{1/e_2} \pmod{n}$. This is the same as the membership certificate of the original.

After the registration, AA publishes (z_A, a), where a is U's value on the attribute type A. Similarly, AA publishes (z_B, b), where b is U's value on the attribute type B. The values z_A and z_B are the attribute indices.

5.3 Offering Protocol

U offers the encrypted attribute indices by the following offering protocol. D collects the encrypted attribute indices of users.

1. D sends a random message m to U.
2. U computes $\tilde{C}_{1A} = y_T^{\tilde{r}_A} g_A^y$ and $\tilde{C}_{2A} = h^{\tilde{r}_A}$ for $\tilde{r}_A \in_R Z_n^*$, and computes $\tilde{C}_{1B} = y_T^{\tilde{r}_B} g_B^y$ and $\tilde{C}_{2B} = h^{\tilde{r}_B}$ for $\tilde{r}_B \in_R Z_n^*$. Note that $(\tilde{C}_{1A}, \tilde{C}_{2A})$ and

$(\tilde{C}_{1B}, \tilde{C}_{2B})$ are the ElGamal encryptions for z_A and z_B with the public key y_T. Furthermore, to prove their validity, U computes the following SPKs:

$$\begin{aligned}\tilde{V}_1 &= SPK\{(\alpha, \beta) : \tilde{C}_{1A} = y_T^{\alpha} g_A^{\beta^{e_1}}\}(m),\\ \tilde{V}_2 &= SPK\{(\gamma, \delta) : \tilde{C}_{1A}^{f_1} g_A^{f_2} = y_T^{\gamma} g_A^{\delta^{e_2}}\}(m),\\ \tilde{V}_3 &= SPK\{(\epsilon, \zeta, \eta) : \tilde{C}_{1A} = y_T^{\epsilon} g_A^{\zeta}\\ &\qquad \wedge \tilde{C}_{2A} = h^{\epsilon} \wedge \tilde{C}_{1B} = y_T^{\eta} g_B^{\zeta}\\ &\qquad \wedge \tilde{C}_{2B} = h^{\eta}\}(m).\end{aligned}$$

Note that $\tilde{V}_1$ and $\tilde{V}_2$ prove that the user knows the attribute certificate (x, v), which is similar to the proof of the knowledge of the membership certificate in the original group signature scheme.

U sends D $(\tilde{C}_{1A}, \tilde{C}_{2A}, \tilde{C}_{1B}, \tilde{C}_{2B}, \tilde{V}_1, \tilde{V}_2, \tilde{V}_3)$.

3. D verifies its correctness by checking $\tilde{V}_1, \tilde{V}_2$ and $\tilde{V}_3$.

5.4 Generating Protocol

Assume that D collects u transcripts of offering protocol. Note that u should be large to some degree. By the following generating protocol, the trustees $T_1, \ldots, T_\ell$ cooperatively compute the statistical result of every attribute type from the transcripts, while the anonymity of users is kept. For the simplicity of description, only the case of the attribute type A is shown, and assume that the values of A are a and a'. The public lists of the attribute indices z_A registered by all users with the value a and a' are denoted as $P_a = (z_1, \ldots, z_N)$ and $P_{a'} = (z'_1, \ldots, z'_{N'})$, respectively.

1. D sends the trustees u transcripts of offering protocol.
2. Each T_i verifies $\tilde{V}_1, \tilde{V}_2$ and $\tilde{V}_3$ in the transcripts, and rejects if they are not valid. The ciphertexts $(\tilde{C}_{1A}, \tilde{C}_{2A})$ in u transcripts are numbered as $\tilde{L} = \{(\tilde{C}_{1,1}, \tilde{C}_{2,1}), \ldots, (\tilde{C}_{1,u}, \tilde{C}_{2,u})\}$.
3. For the list $\tilde{L}$, $T_1, \ldots, T_\ell$ cooperatively shuffle the list to output the list $\hat{L} = \{(\hat{C}_{1,1}, \hat{C}_{2,1}), \ldots, (\hat{C}_{1,u}, \hat{C}_{2,u})\}$ by using the shuffle protocol [3]. Note that this shuffle makes the ciphertexts in the original list unlinkable to the attribute values, which are correspondent with the ciphertexts in the shuffled list. Then, the trustees cooperatively prove to D the correctness of the shuffle by the proof protocol [3].
4. For every ciphertext $(\hat{C}_{1,k}, \hat{C}_{2,k})$ $(1 \le k \le u)$ in $\hat{L}$, $T_1, \ldots, T_\ell$ cooperatively execute the following sub-steps:
   (a) The trustees cooperatively shuffle the public lists P_a and $P_{a'}$ by using a same random factor while the ciphertext $(\hat{C}_{1,k}, \hat{C}_{2,k})$ is also randomized by the factor. The concrete protocol SHUFFLE-SAME-RAND is shown afterward. The outputs of SHUFFLE-SAME-RAND are denoted as $\dot{P}_a = (\dot{z}_1, \ldots, \dot{z}_N)$, $\dot{P}_{a'} = (\dot{z'}_1, \ldots, \dot{z'}_{N'})$ and $(\dot{C}_{1,k}, \dot{C}_{2,k})$. After this protocol, the following relations are satisfied:

$$\dot{z}_j = z^t_{\pi(j)} \text{ for all } 1 \le j \le N,$$
$$\dot{z'}_{j'} = {z'}^t_{\pi'(j')} \text{ for all } 1 \le j' \le N',$$
$$\dot{C}_{1,k} = \hat{C}^t_{1,k}, \text{and}$$
$$\dot{C}_{2,k} = \hat{C}^t_{2,k},$$

for random permutations π, π', and $t \in_R Z^*_n$. Furthermore, the trustees cooperatively prove to D the correctness of the SHUFFLE-SAME-RAND by the protocol SHUFFLE-SAME-RAND-PROOF, which is also shown afterward.

(b) For the output of SHUFFLE-SAME-RAND $\dot{P}_a = (\dot{z}_1, \ldots, \dot{z}_N)$, $\dot{P}_{a'} = (\dot{z'}_1, \ldots, \dot{z'}_{N'})$ and $(\dot{C}_{1,k}, \dot{C}_{2,k})$, a quorum of the trustees cooperatively decrypts the ciphertext by $\dot{z} = \dot{C}_{1k}/\dot{C}^{x_T}_{2k}$. Furthermore, the trustees cooperatively prove to D the correctness. Their concrete protocols are shown in [3]. Note that, if the attribute index z is encrypted into $(\hat{C}_{1,k}, \hat{C}_{2,k})$, $\dot{z}$ should equal z^t, which is in P_a or $P_{a'}$. Therefore, if $\dot{z}$ is in list P_a, the ciphertext $(\hat{C}_{1,k}, \hat{C}_{2,k})$ is proved out to be correspondent with the attribute value a. Otherwise, it is proved out to be correspondent with the value a'.

5. D obtains the statistical result of the attribute type A by calculating the numbers of all attribute values which are correspondent with all ciphertexts in the shuffle list $\hat{L}$.

Next, the protocol SHUFFLE-SAME-RAND is concretely shown.

SHUFFLE-SAME-RAND**:** In this protocol, for inputs $P_a = (z_1, \ldots, z_N)$, $P_{a'} = (z'_1, \ldots, z'_{N'})$ and $(\hat{C}_{1,k}, \hat{C}_{2,k})$, the trustees cooperatively output $\dot{P}_a = (\dot{z}_1, \ldots, \dot{z}_N)$, $\dot{P}_{a'} = (\dot{z'}_1, \ldots, \dot{z'}_{N'})$ and $(\dot{C}_{1,k}, \dot{C}_{2,k})$ such that

$$\dot{z}_j = z^t_{\pi(j)} \text{ for all } 1 \le j \le N,$$
$$\dot{z'}_{j'} = {z'}^t_{\pi'(j')} \text{ for all } 1 \le j' \le N',$$
$$\dot{C}_{1,k} = \hat{C}^t_{1,k}, \text{and}$$
$$\dot{C}_{2,k} = \hat{C}^t_{2,k},$$

for random permutations π, π', and $t \in_R Z^*_n$.
The task of each trustee is as follows. Trustee T_i receives two lists $(z_{i-1,1}, \ldots, z_{i-1,N})$ and $(z'_{i-1,1}, \ldots, z'_{i-1,N'})$, and two values E_{i-1} and F_{i-1}, where $z_{0,1} = z_1, \ldots, z_{0,N} = z_N, z'_{0,1} = z'_1, \ldots, z'_{0,N'} = z'_{N'}, E_0 = \hat{C}_{1,k}$ and $F_0 = \hat{C}_{2,k}$. T_i chooses two random permutations π_i and π'_i and a random factor $t_i \in_R \mathcal{Z}^*_n$. Then, T_i computes

$$z_{i,j} = z^{t_i}_{i-1,\pi_i(j)} \text{ for all } 1 \le j \le N,$$
$$z'_{i,j'} = {z'}^{t_i}_{i-1,\pi'_i(j')} \text{ for all } 1 \le j' \le N',$$
$$E_i = E^{t_i}_{i-1}, \text{and}$$
$$F_i = F^{t_i}_{i-1}.$$

T_i's output consists of $(z_{i,1}, \ldots, z_{i,N})$, $(z'_{i,1}, \ldots, z'_{i,N'})$, E_i and F_i. The next trustee works in the same way, and the process continues up to T_ℓ. The output of this protocol consists of $\dot{P}_a = (\dot{z}_1 = z_{\ell,1}, \ldots, \dot{z}_N = z_{\ell,N})$, $\dot{P}_{a'} = (\dot{z'}_1 = z'_{\ell,1}, \ldots, \dot{z'}_{N'} = z'_{\ell,N'})$ and $(\dot{C}_{1k} = E_\ell, \dot{C}_{2k} = F_\ell)$. □

For above random permutations π_i and π'_i and factors t_i, $t = \prod_{i=1}^{\ell} t_i$, $\pi = \pi_1 \cdots \pi_\ell$ and $\pi' = \pi'_1 \cdots \pi'_\ell$ should hold.

The following protocol is SHUFFLE-SAME-RAND-PROOF, which is derived from the shuffle proof protocol in [3].

SHUFFLE-SAME-RAND-PROOF: In this protocol, the trustees cooperatively prove that they honestly conduct the protocol SHUFFLE-SAME-RAND. The trustees cooperatively conduct the followings σ times, which indicates the error probability $1/2^\sigma$.

1. T_i receives $(\tilde{z}_{i-1,1}, \ldots, \tilde{z}_{i-1,N})$ and $(\tilde{z'}_{i-1,1}, \ldots, \tilde{z'}_{i-1,N'})$, and two values $\tilde{E}_{i-1}$ and $\tilde{F}_{i-1}$, where $\tilde{z}_{0,1} = z_1, \ldots, \tilde{z}_{0,N} = z_N, \tilde{z'}_{0,1} = z'_1, \ldots, \tilde{z'}_{0,N'} = z'_{N'}, \tilde{E}_0 = \hat{C}_{1,k}$ and $\tilde{F}_0 = \hat{C}_{2,k}$. T_i chooses two random permutations λ_i and λ'_i and a random factor $s_i \in_R \mathcal{Z}_n^*$. Then, T_i computes
$$\begin{aligned} \tilde{z}_{i,j} &= \tilde{z}^{s_i}_{i-1,\lambda_i(j)} \text{ for all } 1 \le j \le N, \\ \tilde{z'}_{i,j'} &= \tilde{z'}^{s_i}_{i-1,\lambda'_i(j')} \text{ for all } 1 \le j' \le N', \\ \tilde{E}_i &= \tilde{E}^{s_i}_{i-1}, \text{and} \\ \tilde{F}_i &= \tilde{F}^{s_i}_{i-1}. \end{aligned}$$
T_i sends $(\tilde{z}_{i,1}, \ldots, \tilde{z}_{i,N})$, $(\tilde{z'}_{i,1}, \ldots, \tilde{z'}_{i,N'})$, $\tilde{E}_i$ and $\tilde{F}_i$ to the next trustee T_{i+1}. The last trustee sends $(\tilde{z}_{\ell,1}, \ldots, \tilde{z}_{\ell,N})$, $(\tilde{z'}_{\ell,1}, \ldots, \tilde{z'}_{\ell,N'})$, $\tilde{E}_\ell$ and $\tilde{F}_\ell$ to D and all trustees.
2. D sends $c \in_R \{0,1\}$ to all trustees.
3. If $c = 0$, each T_i computes a commitment $b_i = BC(i, \lambda_i, \lambda'_i, s_i)$ and distributes the commitment to D and all trustees, where BC is a bit commitment scheme. After all commitments are distributed, each T_i opens his commitment by revealing λ_i, λ'_i and s_i. The last trustee T_ℓ computes $\lambda = \lambda_1 \cdots \lambda_\ell, \lambda' = \lambda'_1 \cdots \lambda'_\ell$ and $s = \prod_{i=1}^{\ell} s_i \pmod{n}$. Every trustee verifies that all commitments, λ, λ' and s are correctly made. If this verification fails, this protocol stops.
If $c = 1$, each T_i computes $\varphi_i = \pi_i^{-1}\varphi_{i-1}\lambda_i, \varphi'_i = {\pi'}_i^{-1}\varphi'_{i-1}\lambda'_i$ and $w_i = w_{i-1}s_i/t_i \pmod{n}$, where φ_0, φ'_0 are the identity permutations and $w_0 = 1 \pmod{n}$. The last T_ℓ sends $\varphi = \varphi_\ell, \varphi' = \varphi'_\ell$ and $w = w_\ell$ to D and the other trustees.
4. D and each trustee verify that, if $c = 0$,
$$\begin{aligned} \tilde{z}_{\ell,j} &= z^s_{\lambda(j)} \text{ for all } 1 \le j \le N, \\ \tilde{z'}_{\ell,j'} &= {z'}^s_{\lambda'(j')} \text{ for all } 1 \le j' \le N', \\ \tilde{E}_\ell &= \hat{C}^s_{1,k}, \text{and} \\ \tilde{F}_\ell &= \hat{C}^s_{2,k}, \end{aligned}$$

and if $c = 1$,

$$\begin{aligned} \tilde{z}_{\ell,j} &= \dot{z}^{w}_{\varphi(j)} \text{ for all } 1 \leq j \leq N, \\ \tilde{z'}_{\ell,j'} &= \dot{z'}^{w}_{\varphi'(j')} \text{ for all } 1 \leq j' \leq N', \\ \tilde{E}_{\ell} &= \dot{C}^{w}_{1,k}, \text{and} \\ \tilde{F}_{\ell} &= \dot{C}^{w}_{2,k}. \end{aligned}$$

□

6 Discussion

Before the satisfaction of the requirements is discussed, two lemmas are shown. The following lemma for the protocol SHUFFLE-SAME-RAND holds, since it is infeasible to determine the sameness of the discrete logarithms.

Lemma 1. *Given all $z_{i-1,\pi_i(j)}$, $z_{i,j}$, $z'_{i-1,\pi'_i(j')}$ and $z'_{i,j'}$, no adversary can determine $\pi_i(j)$ for any j or $\pi'_i(j')$ for any j' with non-negligibly better probability.*

The next lemma shows the security of SHUFFLE-SAME-RAND-PROOF. The proof of this lemma is similar to that of the shuffle proof protocol in [3]. The sketch of the proof is shown in the appendix.

Lemma 2. SHUFFLE-SAME-RAND-PROOF *is a honest verifier zero-knowledge proof of knowledge.*

Now, we discuss that the proposed system satisfies the requirements in Section 2. For the simplicity, only the case of the attribute type A is shown.

Correctness: Owing to the soundness of SPKs $\tilde{V}_1, \tilde{V}_2$ and $\tilde{V}_3$, and the protection of the replay attack that is brought by the use of the random message m, it is assured that the user knows the attribute certificate (x, v) such that the ciphertext $(\tilde{C}_{1A}, \tilde{C}_{2A})$ is encrypted from $z_A = g_A^{x^{e_1}}$. Owing to the unforgeability of (x, v) and the soundness of SPKs V_1, V_2 and V_3, it is assured that the user, in advance, registered z_A, which is published with the genuine attribute value a. Therefore, the user can offer only the ciphertext of z_A correspondent with a.

The remain is to show that, in the generating protocol, the ciphertext $(\tilde{C}_{1A}, \tilde{C}_{2A})$ is revealed as only the genuine attribute value a. After the list of the ciphertexts is shuffled, let $(\hat{C}_{1A}, \hat{C}_{2A})$ be the permuted ciphertext of $(\tilde{C}_{1A}, \tilde{C}_{2A})$. Then, since both ciphertexts can be decrypted into the same plaintext, the following is satisfied.

$$\hat{C}_{1A}/\hat{C}_{2A}^{x_T} = \tilde{C}_{1A}/\tilde{C}_{2A}^{x_T}.$$

In Step 4-(a) of the protocol, all users' z_A with the attribute value a, which are denoted z_j, are transformed into $z^{t}_{\pi^{-1}(j)}$ for trustees' random permutation

π and random factor t, and $(\hat{C}_{1A}, \hat{C}_{2A})$ is also transformed into $(\dot{C}_{1A} = \hat{C}_{1A}^t, \dot{C}_{2A} = \hat{C}_{2A}^t)$. Then, the decrypted value $\dot{z}$ satisfies the following:

$$\begin{aligned} \dot{z} &= \dot{C}_{1A}/\dot{C}_{2A}^{x_T} \\ &= \hat{C}_{1A}^t/(\hat{C}_{2A}^t)^{x_T} \\ &= (\hat{C}_{1A}/\hat{C}_{2A}^{x_T})^t \\ &= (\tilde{C}_{1A}/\tilde{C}_{2A}^{x_T})^t \\ &= z_A^t. \end{aligned}$$

Since the value z_A^t should be in the list P_a of $z_{\pi^{-1}(j)}^t$, the ciphertext $(\hat{C}_{1A}, \hat{C}_{2A})$ is made correspondent with the genuine value a. Thus, the original ciphertext $(\tilde{C}_{1A}, \tilde{C}_{2A})$ is revealed as only the genuine attribute value a. Therefore, the correctness of the statistical result is assured.

Anonymity: In the proposed protocols, since the SPKs and PKs release no information on secrets, they are ignored in the following discussion. Similarly, the blinded message $\tilde{y}$ is ignored. Furthermore, note that the ciphertext of the ElGamal encryption does not also release the information on secrets, but it should be only discussed that the ciphertext itself appears in both offering and generating protocols.
The first discussion is to trace the owner's identity from the transcript of the offering protocol. This is possible if the transcript is linked to each attribute index z of which AA knows the owner. In the generating protocol, the transcript is not directly linked to z, but z^t for a random factor t. In addition, the correspondence of z^t with z is concealed through the permutation and randomization, which is shown in Lemma 1. Therefore, the tracing is infeasible.
The second is to link between the transcripts w.r.t. the sameness of the owner. This is possible if the attribute indices z and z' of the transcripts are linkable. However, the transcripts are only linked to z^t and $z'^{t'}$ for different random factors t and t'. And, given z^t and $z'^{t'}$, it is infeasible to determine the sameness of z and z' since it is infeasible to determine the sameness of the discrete logarithms. Thus, the link between the transcripts is infeasible.
The final is to link from the transcript to the attribute value. In the generating protocol, though it is proved that the transcript is correspondent with something of the attribute values, the corresponding value itself is unknown owing the shuffle protocol in Step 3 of the generating protocol. Furthermore, as stated above, it is infeasible to link the transcript to the attribute index. Therefore, this link is also infeasible.

7 Conclusion

In this paper, a statistical survey system of attributes is proposed, where both correctness and anonymity are satisfied. Though the complexity of users' offering their attributes is comparable to the practical group signature [2], the complexity of the trustees' proving the correctness is proportional to the number of all users. This implies the inefficiency when many users join the system in order to obtain the attributes of many users. Thus, our future work is to propose the system overcoming the inefficiency.

References

[1] Sako, K.: Generating Statistical Information in Anonymous Surveys, *IEICE trans. on Fundamentals.*, Vol. E79-A, No. 4, pp. 507–512 (1996).

[2] Camenisch, J. and Stadler, M.: Efficient Group Signature Schemes for Large Groups, *Advances in Cryptology — CRYPTO'97*, LNCS 1294, Springer–Verlag, pp. 410–424 (1997).

[3] Abe, M.: Universally Verifiable Mix-net with Verification Work Independent of the Number of Mix-centers, *Advances in Cryptology — EUROCRYPT'98*, LNCS 1403, Springer–Verlag, pp. 437–447 (1998).

[4] Desmedt, Y. and Frankel, Y.: Threshold Cryptosystems, *Advances in Cryptology — CRYPTO'89*, LNCS 435, Springer–Verlag, pp. 307–315 (1990).

[5] Fiat, A. and Shamir, A.: How To Prove Yourself: Practical Solutions to Identification and Signature Problems, *Advances in Cryptology — CRYPTO '86*, LNCS 263, Springer–Verlag, pp. 186–194 (1987).

[6] Bellare, M. and Rogaway, P.: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols, *Proc. of First Annual Conference on Computer and Communications Security*, Association for Computing Machinery, pp. 62–73 (1993).

Appendix: Sketch of Proof of Lemma 2

In this appendix, the sketch of the proof of Lemma 2 is shown.

Lemma 2. SHUFFLE-SAME-RAND-PROOF *is a honest verifier zero-knowledge proof of knowledge.*

Sketch of proof. The completeness holds as follows. In the case of $c = 0$, it is clear that the verification equations are satisfied if trustees compute the correct values. In the case of $c = 1$,

$$\tilde{z}_{\ell,j} = z^s_{\lambda(j)},$$

for $\lambda = \lambda_1 \cdots \lambda_\ell$ and $s = \prod_{i=1}^{\ell} s_i \pmod n$. On the other hand, from $\dot{z}_j = z^t_{\pi(j)}$, $\varphi = \pi_\ell^{-1} \cdots \pi_1^{-1} \lambda_1 \cdots \lambda_\ell = \pi^{-1}\lambda$ and $w = \prod_{i=1}^{\ell} s_i/t_i = (\prod_{i=1}^{\ell} s_i)/(\prod_{i=1}^{\ell} t_i) = s/t \pmod n$,

$$\begin{aligned} \dot{z}^w_{\varphi(j)} &= (z^t_{\pi\varphi(j)})^w \\ &= (z^t_{\pi\pi^{-1}\lambda(j)})^{s/t} \\ &= z^s_{\lambda(j)}. \end{aligned}$$

Thus,

$$\tilde{z}_{\ell,j} = \dot{z}^{w}_{\varphi(j)}.$$

Similarly, other verification equations of $c = 1$ are satisfied if the trustees compute the correct values.

Next, the soundness is proved as follows. Assume that the trustees correctly answer both $c = 0$ and $c = 1$ cases for the same λ, λ' and $(\tilde{z}_{\ell,1}, \ldots, \tilde{z}_{\ell,N})$, $(\tilde{z'}_{\ell,1}, \ldots, \tilde{z'}_{\ell,N'})$, $\tilde{E}_\ell$ and $\tilde{F}_\ell$. Then, by using φ and λ, one can extract π as $\lambda\varphi^{-1} = \lambda(\pi^{-1}\lambda)^{-1} = \pi$. Similarly, π' is extracted. Furthermore, by using s and w, one can extract t as $s/w = s/(s/t) = t \pmod{n}$. Though it is complex to extract the knowledge of the individual trustee, it can be extracted by the similar way to [3].

Finally, to prove the zero-knowledge, ℓ simulators $S_1, \ldots, S_\ell$ are constructed as well as [3]. First, the simulators cooperatively choose $c \in_R \{0, 1\}$. If $c = 0$, they honestly conduct the protocol. They can accomplish it, since the knowledge is not needed in this case. If $c = 1$, each simulator S_i chooses fake permutations $\tilde{\lambda}_i$ and $\tilde{\lambda}'_i$ and a fake factor $\tilde{s}_i \in_R Z_n^*$. Then, the simulators except the last simulator S_ℓ honestly obey the protocol. In Step 1, the last simulator S_ℓ computes

$$\begin{aligned}
\tilde{z}_{\ell,j} &= \dot{z}^{\tilde{s}_\ell}_{\tilde{\lambda}_\ell(j)} \text{ for all } 1 \le j \le N,\\
\tilde{z'}_{\ell,j'} &= \dot{z'}^{\tilde{s}_\ell}_{\tilde{\lambda'}_\ell(j')} \text{ for all } 1 \le j' \le N',\\
\tilde{E}_\ell &= \dot{C}^{\tilde{s}_\ell}_{1,k}, \text{and}\\
\tilde{F}_\ell &= \dot{C}^{\tilde{s}_\ell}_{2,k},
\end{aligned}$$

and sends them to D and all simulators. In Step 3, S_ℓ sends D and all simulators $\varphi = \tilde{\lambda}_\ell, \varphi' = \tilde{\lambda}'_\ell$ and $w = \tilde{s}_\ell$, which satisfy the verification equations of Step 4. The views of the simulators and trustees (in the real protocol) are indistinguishable except the ℓ-th party in the case of $c = 1$, since they honestly obey the protocol. Consider T_ℓ and S_ℓ in the case of $c = 1$. In Step 1, $\tilde{z}_{\ell,j}$ of T_ℓ is the form g_A^R for a random factor $R \in_R Z_n^*$, since the original z_j is the form and it is raised to the power $s_i \in_R Z_n^*$. On the other hand, from the same reason, $\dot{z}_j$ is also the form, and so is $\tilde{z}_{\ell,j}$ of S_ℓ. Thus, the distributions of $\tilde{z}_{\ell,j}$ of T_ℓ and S_ℓ are the same. It similarly holds for the other values in Step 1. In Step 3, the values w of T_ℓ and S_ℓ distribute uniformly on Z_n^* and so do the permutations φ, φ'. Therefore, the views of them are also indistinguishable.

□

Secure Mobile Agent Using Strong Non-designated Proxy Signature

Byoungcheon Lee[1], Heesun Kim[2]*, and Kwangjo Kim[1]

[1] IRIS (International Research center for Information Security)
ICU (Information and Communications University)
58-4, Hwaam-dong, Yusong-gu, Taejon, 305-732, Korea
{sultan,kkj}@icu.ac.kr
[2] Information Security Technology Division
ETRI (Electronics and Telecommunications Research Institute)
161, Kajong-dong, Yusong-gu, Taejon, 305-600, Korea
sezsez@etri.re.kr

Abstract. It is expected that mobile agent will be widely used for electronic commerce as an important key technology. If a mobile agent can sign a message in a remote server on behalf of a customer without exposing his/her private key, it can be used not only to search for special products or services, but also to make a contract with a remote server. To construct mobile agents, [KBC00] used an RSA-based undetachable signature scheme, but it does not provide server's non-repudiation because the undetachable signature does not contain server's signature.
Mobile agent is a very good application example of proxy signature, and the undetachable signature can be considered as an example of proxy signature. In this paper we show that *secure mobile agent* can be constructed using *strong non-designated proxy signature* [LKK01] which represents both the original signer's (customer) and the proxy signer's (remote server) signatures. We provide RSA-based and Schnorr-based constructions of secure mobile agent, and moreover we show that the Schnorr-based scheme can be used very efficiently in multi-proxy mobile agent situation.

Keywords. Secure mobile agent, strong non-designated proxy signature, multi-proxy signature.

1 Introduction

1.1 Mobile Agent

Mobile agents [FGS96, KKC99, LM99] are autonomous software entities that are able to migrate across different execution environments through network. The characteristics of mobile agents, mobility and autonomy, make them ideal for

* This work was done when she was with ICU.

electronic commerce applications because permanent connections between customers and servers are unnecessary and low-bandwidth connections and asynchronous communications are possible. Furthermore, they provide better support for heterogeneous environments. Mobile agents can be used for electronic commerce in many ways; search and buy special products or services on behalf of a customer, negotiate something with other entities, and sell products on behalf of a shopping mall server.

We consider a scenario that a mobile agent is ordered to search the price of a flight ticket and book it on behalf of a customer. If the mobile agent finds a proper bid presented by a server, the mobile agent will book it by digitally signing the server's bid and the customer's requirement with both customer's and server's keys. To make it possible, the mobile agent must carry in any form the customer's private key and compute with it.

However, mobile agents are vulnerable to several attacks, particularly by malicious hosts. Fundamental problems of executing mobile code in a remote host can be listed as follows [ST97]:

1. Code and execution integrity: Can a mobile agent protect itself against tampering by a malicious server?
2. Code privacy: Can a mobile agent conceal the program it wants to have executed?
3. Computing with secrets in public: Can a mobile agent remotely sign a document without disclosing user's private key?

There have been extensive researches to solve these problems. A reasonable and practical approach is to provide software-based mechanism to prevent any kind of vulnerability actively. Implementing any kind of secure function in mobile agent is difficult because all the code and data of mobile agent are exposed to remote server. One of the best ways to conceal customer's private key and keep the integrity of mobile code is to use cryptographic hard problems such as integer factorization problem or discrete logarithm problem. Undetachable signature scheme is an example.

1.2 Undetachable Signature Scheme

[ST97] introduced the concept of Computing with Encrypted Function (CEF) which tried to conceal the signature function by composing it with encryption function. [KBC00] implemented CEF using an RSA-based undetachable signature scheme. The customer signs his requirement information using RSA signature and builds up an encrypted signature function, and then gives it to mobile agent. Then the server can generate customer's signature on the bid information on behalf of the customer. Customer's private key is hidden in the encrypted signature function and its secrecy is based on the RSA assumption.

Although the undetachable signature scheme of [KBC00] hides customer's private key successfully, it does not provide the fairness of contract. The basic requirement of fair contract is non-repudiations of both parties. The undetachable signature represents only customer's signature and it can be computed by any party, so the server can repudiate his signature generation later. After the booking process of the flight ticket is finished with customer's payment, the server can repudiate his signature generation and refuse to deliver the flight ticket.

A simple solution for this problem is that the server signs his final messages before giving them to the mobile agent, but this is not a neat solution. In Section 4, we propose an efficient strong proxy signature scheme which represents both the customer's and the server's signatures providing the fairness of contract.

The basic concept of undetachable signature scheme is very similar to the delegation of customer's signing capability to unspecified proxy signers. Hereafter we review proxy signature schemes briefly.

1.3 Proxy Signature

Proxy signature is a signature scheme that an original signer delegates his/her signing capability to a proxy signer, and then the proxy signer creates a signature on behalf of the original signer. When a receiver verifies a proxy signature, he verifies the signature itself and original signer's delegation together. The basic methodology of proxy signature is that the original signer creates a signature on delegation information (ID of the proxy signer, or any warrant information) and gives it secretly to the proxy signer, and then the proxy signer uses it to generate a proxy key pair. Because the proxy key pair is generated using original signer's signature on delegation information, any verifier can check original signer's agreement from a proxy signature.

[MUO96] firstly introduced the concept of proxy signature. They classified proxy signatures based on delegation type as full delegation (giving the original signer's private key itself), partial delegation (issuing a new key pair), and delegation by warrant (issuing a certificate stating the delegation information). Partial delegation is further classified as proxy-unprotected and proxy-protected according to protection of proxy signer. They provided various constructions of proxy signature schemes and their security analysis. [KPW97] extended them by using Schnorr signature and including warrant information in partial delegation schemes (partial delegation with warrant). [LKK01] provided several attacks against previous proxy signature schemes and introduced the concept of strong proxy signatures which represent both original signer's and proxy signer's signatures. They also introduced the concept of strong non-designated proxy signature where the original signer does not specify proxy signers in the delegation stage. It is useful when proxy signers cannot be determined in the delegation stage.

Mobile agent is one of the best application areas of proxy signature scheme, because the original signer (customer) has to delegate his/her signing capability to the mobile agent (and to the server) for it to execute any authentic operation on behalf of the original signer. [KBLK01] applied proxy signature scheme to mobile agent and introduced one-time proxy signature to guarantee one-timeness of signature generation. [OSM01] considered multi-proxy situation where plural customers delegate their signing capabilities to a mobile agent and proposed an efficient mobile agent scheme. Multi-proxy signature is also considered in

[YBX00]. But [OSM01] and [YBX00] have used weak version of proxy signature, so they cannot provide non-repudiation of the server.

1.4 Our Contribution

To provide strong undeniability, i.e., non-repudiation of the server, we construct Secure Mobile Agent (SMA) using the Strong Non-designated Proxy Signature (SNPS) [LKK01]. We provide two implementation examples of SMA. Firstly, we construct RSA-based SMA which is an extension of [KBC00] and show that it satisfies all the requirements of SNPS. Secondly, we construct Schnorr-based SMA using [LKK01, KBLK01] and show that it also satisfies all the requirements of SNPS. Moreover, we show that the Schnorr-based SNPS can be used very efficiently in multi-proxy situation providing efficiency in communication and computation.

In Section 2, we describe SNPS briefly with its security requirements. In Sections 3 and 4, we construct Schnorr-based SMA and RSA-based SMA, respectively. In Section 5, we describe multi-proxy SMA using multi-proxy signature. Finally, we conclude in Section 6.

2 Strong Non-designated Proxy Signature

[LKK01] has shown several attacks against previous proxy signature schemes [MUO96, PH97, KPW97]. There are possibilities of proxy signer's repudiation or misuse of the proxy key pair. They classified proxy signatures as strong and weak ones. Strong proxy signatures represent both original signer's and proxy signer's signatures, while weak ones represent only original signer's signature. In real situation, assuming the trustedness of original signer or proxy signer is difficult, specially in distributed environment as mobile agent. So weak versions of proxy signature cannot be used. If the proxy signature scheme is strong, it can be used without designating the proxy signer in delegation stage. We define the Strong Non-designated Proxy Signature (SNPS) as follows.

Definition 1 (Strong Non-designated Proxy Signature). *Let A be an original signer who has authentic key pair (sk_A, pk_A) and B be a proxy signer who has authentic key pair (sk_B, pk_B). Let m_w be A's warrant information for the delegation which does not specify a proxy signer. Let $\sigma_A = S(sk_A, m_w)$ be A's signature on warrant m_w using her private key sk_A. Then SNPS is constructed as the following three algorithms $(\mathcal{PKG}, \mathcal{PS}, \mathcal{PV})$.*

– *$\mathcal{PKG}$ is a proxy key issuing algorithm that takes original signer's signature σ_A and proxy signer's private key sk_B and outputs a proxy key pair (sk_P, pk_P). It is executed by the proxy signer.*

$$(sk_P, pk_P) \leftarrow \mathcal{PKG}(\sigma_A, sk_B).$$

– *$\mathcal{PS}$ is a proxy signing algorithm that takes proxy private key sk_P and message m and outputs proxy signature σ_P. It is executed by the proxy signer.*

$$\sigma_P \leftarrow \mathcal{PS}(sk_P, m).$$

– *$\mathcal{PV}$ is a proxy verification algorithm that takes $(\sigma_P, m, m_w, pk_A, pk_B)$ and outputs either accept or reject. It is executed by any verifier.*

$$\mathcal{PV}(\sigma_P, m, m_w, pk_A, pk_B) \overset{?}{=} accept \text{ or } reject.$$

SNPS should satisfy the following security requirements [LKK01].

R1. Verifiability: From a proxy signature a verifier can be convinced of the original signer's agreement on the signed message.
R2. Strong unforgeability: A proxy signer can create a valid proxy signature for the original signer. But the original signer and any third party cannot create a valid proxy signature with the name of proxy signer.
R3. Strong identifiability: Anyone can determine the identity of the corresponding proxy signer from a proxy signature.
R4. Strong undeniability: Once a proxy signer creates a valid proxy signature on behalf of an original signer, the proxy signer cannot repudiate his signature creation against anyone.
R5. Prevention of misuse: It should be confident that proxy key pair cannot be used for other purposes. In the case of misuse, the responsibility of proxy signer should be determined explicitly.

A proxy signature represents both the original signer's signature (by R1) and the proxy signer's signature (by R2, R3, and R4). Requirement R5 guarantees that the proxy key pair cannot be used for other purposes.

In mobile agent environment, the customer (original signer) cannot determine a proper server (proxy signer) in the delegation stage who will suggest a conforming bid. In this case mobile agent has the role of transferring customer's delegation information to possible proxy signers. To provide fairness of contract, proxy signature scheme should contain proxy signer's signature together with original signer's agreement. Therefore, SNPS is a perfect solution to construct SMA.

Because SNPS represents both the original signer's and the proxy signer's signatures, it can be considered as an efficient integration scheme of two related signatures. As stated in [MUO96] and [KPW97], partially delegated proxy signature is more efficient than that of delegation by warrant which is represented by two signatures. We will discuss the efficiency issue of proxy signatures in more detail in Section 5.

3 Schnorr-Based SMA

We apply the SNPS of [LKK01] to mobile agent situation. Firstly we review Schnorr signature briefly. Let p and q be large primes with $q|p-1$. Let g be a generator of a multiplicative subgroup of Z_p^* with order q. $h()$ denotes a collision resistant cryptographic hash function. Assume that a signer A has a private key x_A and the corresponding public key $y_A = g^{x_A}$. To sign a message m, A chooses a random number $k \in_R Z_q^*$ and computes $r = g^k$, $s = x_A h(m, r) + k$. Then the tuple (m, r, s) becomes a valid signed message. The validity of signature is verified by $g^s \overset{?}{=} y_A^{h(m,r)} r$. Note that the verification of signature requires two modular exponentiations.

Let A be a customer who has an authentic key pair (x_A, y_A) and B be a server who has also an authentic key pair (x_B, y_B). Let ID_A and ID_B denote the identities of A and B, respectively. Let req_A be A's requirement for a purchase (any necessary information such as price range, date, delivery requirement, etc) and bid_B be B's bid information which conforms to req_A.

Preparing the agent (by the customer A):

A chooses a random number $k_A \in_R Z_q^*$ and computes $r_A = g^{k_A}$, $s_A = x_A h(req_A, r_A) + k_A$. The tuple (req_A, r_A, s_A) is A's Schnorr signature on req_A. A gives (req_A, r_A, s_A) to the mobile agent. Note that A does not specify any server in this stage. Mobile agent will migrate to servers through the network.

Executing the agent (by the server B):

B gets the mobile agent and tries to sell the product to A.

- B verifies the validity of the mobile agent by checking $g^{s_A} \overset{?}{=} y_A^{h(req_A, r_A)} r_A$.
- B generates a secure proxy key pair as

$$x_P = s_A + x_B, \quad y_P \equiv g^{x_P} = y_A^{h(req_A, r_A)} r_A y_B.$$

- B generates a bid information bid_B which conforms to req_A. He signs $m = (ID_A, req_A, ID_B, bid_B, r_A)$ with the proxy private key x_P to generate $\sigma_P = S(x_P, m)$ using the Schnorr signature scheme $S()$. He gives the following messages to the agent.

$$(ID_A, req_A, ID_B, bid_B, r_A, \sigma_P).$$

The mobile agent will get back to A with these messages as a receipt for her purchase.

Verifying the signature (by anyone):

When A receives $(ID_A, req_A, ID_B, bid_B, r_A, \sigma_P)$ from the mobile agent, she can verify the validity of her purchase as follows:

1) Verify the signature by $V(y_P, m, \sigma_P) \overset{?}{=} true$ where $y_P = y_A^{h(req_A, r_A)} r_A y_B$ and $m = (ID_A, req_A, ID_B, bid_B, r_A)$.
2) Verify the conformance of bid: $bid_B \overset{?}{\in} \{req_A\}$.

If the signature verification holds, it represents both the validity of signature itself and the authenticity of customer's delegation.

We show that the proposed Schnorr-based SMA satisfies all the security requirements of SNPS.

Theorem 1. *The proposed Schnorr-based SNPS is as secure as the Schnorr signature scheme.*

Proof. We consider two attack scenarios; the first case is that A tries to forge a SNPS with the name of B without B's agreement, and the second case is that B tries to forge a SNPS without A's delegation. Let $\sigma_P = (r, s)$ be a valid Schnorr-based SNPS for the message $m = (ID_A, req_A, ID_B, bid_B, r_A)$ generated by using the proxy private key x_P where $r = g^k$ for a random number $k \in_R Z_q^*$ and $s = x_P h(m, r) + k$. Note that x_P is not known to A and B in both attack scenarios.

1. Forgery by A: Assume that there is a SNPS breaker (oracle) which takes (m, k) and A's delegation as input and outputs a valid proxy signature (σ_P, r_A) which satisfies the verification equation. An attacker A chooses a random number k and computes $r = g^k$. She gives (m, k) and her delegation $s' = x_A h(req_A, r_A) + k_A$ to the SNPS breaker, then it will output a valid SNPS (σ_P, r_A) which satisfies the verification equation $g^s = (y_A^{h(req_A, r_A)} r_A y_B)^{h(m,r)} r$. Because of the group property of discrete logarithm problem,

$$\begin{aligned} s &= (x_A h(req_A, r_A) + k_A + x_B) h(m, r) + k \\ &= (s' + x_B) h(m, r) + k \end{aligned}$$

should hold. Then A can compute

$$x_B h(m, r) + k = s - s' h(m, r)$$

which is B's Schnorr signature on the message m. Using the SNPS breaker, A can forge B's Schnorr signature without knowing x_B.

2. Forgery by B: Assume that there is a SNPS breaker which takes (m, req_A, k) as input and outputs a valid proxy signature (σ_P, r_A) which satisfies the verification equation. An attacker B chooses a random number k and computes $r = g^k$. He gives (m, req_A, k) to the SNPS breaker, then it will output a valid SNPS (σ_P, r_A) which satisfies the verification equation $g^s = (y_A^{h(req_A, r_A)} r_A y_B)^{h(m,r)} r$. Because of the group property of discrete logarithm problem,

$$s = (x_A h(req_A, r_A) + k_A + x_B) h(m, r) + k$$

should hold. Then B can compute

$$x_A h(req_A, r_A) + k_A = (s - k)/h(m, r) - x_B$$

which is A's Schnorr signature on req_A. Using the SNPS breaker, B can forge A's Schnorr signature without knowing x_A.

Therefore the proposed Schnorr-based SNPS is as secure as the Schnorr signature scheme. □

From Theorem 1, the proposed Schnorr-based SMA satisfies all the security requirements of SNPS.

(i) *Verifiability*: A's agreement on req_A is included in y_P. If the proxy signature is verified to be valid, A's agreement is also verified explicitly.
(ii) *Strong unforgeability*: Anyone except the proxy signer B cannot generate a valid proxy key pair under the name of B because it contains proxy signer's private key x_B. Only the legitimate proxy signer can create a valid proxy signature.
(iii) *Strong identifiability*: Identity information of the proxy signer B is included explicitly in a valid proxy signature as a form of public key y_B. So anyone can determine the identity of the corresponding proxy signer.
(iv) *Strong undeniability*: Once the proxy signer B creates a valid proxy signature, he cannot repudiate it because the proxy key pair can be computed only by himself.
(v) *Prevention of misuse*: If the proxy signer B uses the proxy key pair for other purposes that are not specified in req_A, it is his responsibility because he is the only person who can generate it.

4 RSA-Based SMA

In this Section, we propose an RSA-based SNPS scheme and apply it to construct SMA. It is an extension of [KBC00] scheme to include proxy signer's signature.

To generate RSA keys, each participant selects a modulus n which is the product of two large primes p, q and a number e, such that $1 < e < \varphi(n) = (p-1)(q-1)$ and $\gcd(e, \varphi(n)) = 1$. Let d be such that $de = 1 \bmod \varphi(n)$. Let $h()$ denote collision resistant cryptographic hash function.

Let A be a customer who has an authentic RSA key (n_A, e_A, d_A) and B be a server who has an authentic RSA key (n_B, e_B, d_B). Let ID_A and ID_B denote the identities of A and B, respectively. Let req_A be A's requirement for a purchase (any necessary information such as price range, date, delivery requirement, etc) and bid_B be B's bid information which conforms to req_A.

Preparing the agent (by the customer A):

A computes $k = h(ID_A, req_A)^{d_A} \bmod n_A$ which is her RSA signature on (ID_A, req_A). She gives (ID_A, req_A, k) to the mobile agent. Note that A does not specify any server (proxy signer) in this stage. Mobile agent will migrate to servers through the network.

Executing the agent (by the server B):

B gets the mobile agent and tries to sell the product to A.

– B verifies the validity of the mobile agent by checking

$$k^{e_A} \bmod n_A \overset{?}{=} h(ID_A, req_A).$$

- B generates a bid information bid_B which conforms to req_A and computes

$$x = h(ID_A, req_A, ID_B, bid_B)^{d_B} \bmod n_B$$

which is B's RSA signature on $(ID_A, req_A, ID_B, bid_B)$.
- B computes $y = h(ID_A, req_A)^x \bmod n_A$ and $z = k^x \bmod n_A$. He gives following messages to the mobile agent.

$$(ID_A, req_A, ID_B, bid_B, x, y, z).$$

The mobile agent will get back to A with these messages as a receipt for her purchase.

Verifying the signature (by anyone):
When A receives $(ID_A, req_A, ID_B, bid_B, x, y, z)$ from the mobile agent, she can verify the validity of her purchase as follows:

1) Verify B's signature: $x^{e_B} \bmod n_B \overset{?}{=} h(ID_A, req_A, ID_B, bid_B)$.
2) Verify the validity of y: $y \overset{?}{=} h(ID_A, req_A)^x \bmod n_A$.
3) Verify A's signature: $z^{e_A} \bmod n_A \overset{?}{=} y$.
4) Verify the conformance of bid: $bid_B \overset{?}{\in} \{req_A\}$.

The proxy signature is valid only when all the verifications above are passed.

We show that the proposed RSA-based SMA satisfies all the security requirements of SNPS.

Theorem 2. *The proposed RSA-based SNPS is as secure as the RSA signature scheme.*

Proof. We consider two attack scenarios; the first case is that A tries to forge a SNPS with the name of B without B's agreement, and the second case is that B tries to forge a SNPS without A's delegation. Obviously the first attack cannot happen because a valid SNPS contains x which is B's signature for $(ID_A, req_A, ID_B, bid_B)$. Consider the second attack scenario where B tries to forge a SNPS without k.

Assume that there is a SNPS breaker (oracle) which takes $(ID_A, req_A, ID_B, bid_B, x)$ as input and outputs (y, z) which satisfy the verification equations. B prepares a warrant req_A and a conforming bid bid_B and generates his signature $x = h(ID_A, req_A, ID_B, bid_B)^{d_B} \bmod n_B$. He gives $(ID_A, req_A, ID_B, bid_B, x)$ to the SNPS breaker, then it will provide a valid (y, z). $y = h(ID_A, req_A)^x \bmod n_A$ can be verified from the known values (ID_A, req_A, x). To satisfy the third verification equation, the following equation should hold.

$$z = y^{d_A} \bmod n_A = h(ID_A, req_A)^{x d_A} \bmod n_A.$$

Then B can compute

$$z^{1/x} \bmod n_A = h(ID_A, req_A)^{d_A} \bmod n_A = k$$

which is A's RSA signature on message (ID_A, req_A). Using the SNPS breaker, B can forge A's RSA signature without knowing d_A. Therefore the proposed RSA-based SNPS is as secure as the RSA signature scheme. □

From Theorem 2, the proposed RSA-based SMA satisfies all the security requirements of SNPS.

(i) *Verifiability*: Original signer's agreement on the purchase can be verified by the third verification equation.
(ii) *Strong unforgeability*: Only the proxy signer B can generate a valid signature x satisfying the first verification equation.
(iii) *Strong identifiability*: Anyone can determine the identity of the corresponding proxy signer by the first verification equation.
(iv) *Strong undeniability*: Once B creates a valid proxy signature which passes all the verification equations, he cannot repudiate it later against anyone because a valid proxy signature can be generated only by himself.
(v) *Prevention of misuse*: k is A's signature on (ID_A, req_A) and it cannot be used for other purposes which are not stated in req_A. The proxy signature scheme is executed using B's signature x, so any possible misuse of k is B's responsibility.

5 Multi-proxy Mobile Agent

In this Section, we propose an efficient mobile agent scheme when plural customers delegate their signing capabilities to a mobile agent. For example, we consider a situation that a mobile agent is ordered to book flight tickets for plural customers. Using the Schnorr-based SMA scheme where plural customers share the common system parameters p, q, and g, we can build an efficient mobile agent.

[OSM01] considered a similar application, but their scheme is based on the proxy signature of [MUO96] and customer's requirements are not used. So customers delegate their full signing capabilities to unspecified proxy signers and a server can sign any message on behalf of customers. [YBX00] also proposed proxy multi-signature scheme based on [MUO96]. We apply the strong non-designated proxy signature [LKK01] to multi-proxy mobile agent.

5.1 Multi-proxy Mobile Agent Scheme

Let A_i $(i = 1, ..., n)$ denote plural customers who have certified key pairs (x_i, y_i) and requirements req_i. They try to delegate their signing capabilities to unspecified servers through the mobile agent. Let B be a server who has certified key pair (x_B, y_B) and is willing to sell flight tickets to customers. He has to create a proxy signature on behalf of $\{A_1, ..., A_n\}$ under requirements $\{req_1, ..., req_n\}$.

Preparing the agent (by plural customers A_i):

Plural customers A_i $(i = 1, ..., n)$ choose random numbers $k_i \in_R Z_q^*$ and compute $r_i = g^{k_i}$, $s_i = x_i h(req_i, r_i) + k_i$. The tuple (req_i, r_i, s_i) is A_i's Schnorr signature on req_i. A_i gives (req_i, r_i, s_i) to the mobile agent. Mobile agent will migrate to servers through the network with this information.

Executing the agent (by the server B):

The server B gets the mobile agent and tries to sell the product to customers $\{A_1, ..., A_n\}$.

- B verifies the validity of the delegation information by checking $g^{s_i} \overset{?}{=} y_i^{h(req_i, r_i)} r_i$ for $i = 1, ..., n$.
- If this tests have passed, B generates a secure proxy key pair as

$$x_P = s_1 + \cdots + s_n + x_B, \quad y_P = g^{x_P}.$$

- B generates his bid bid_B which conforms to all req_i $(i = 1, ...n)$. He signs on $m = (req_1, \cdots, req_n, bid_B)$ with the proxy private key x_P to generate $\sigma_P = S(x_P, m)$ using the Schnorr signature scheme $S()$. The tuple

$$(bid_B, \sigma_P, req_1, r_1, y_1, ..., req_n, r_n, y_n, y_B)$$

is a valid proxy signature and represents valid flight tickets for $\{A_1, ..., A_n\}$.

Verifying the signature (by anyone):

When plural customers receives the tuple from the mobile agent, they can verify the validity of their tickets as follows:

1) Verify the signature by $V(y_P, m, \sigma_P) \overset{?}{=} true$ where

$$y_P = y_1^{h(req_1, r_1)} r_1 \cdots y_n^{h(req_n, r_n)} r_n y_B, \quad m = (req_1, \cdots, req_n, bid_B).$$

2) Check whether bid_B confirms to $\{req_1, \cdots, req_n\}$.

5.2 Comparison with Multiple Signatures

As stated in [MUO96], proxy signature schemes of partial delegation are more efficient than those of delegation by warrant. Consider a traditional approach of multiple independent signatures that plural customers A_i publish their signatures (req_i, r_i, s_i) and the server B just signs on bid_B with his certified key pair (x_B, y_B). The proposed multi-proxy signature scheme is more efficient than the traditional approach of multiple independent signatures in the following sense.

- A valid signature can be created by the proxy signer himself without any interaction with original signers, while traditional scheme requires n communications with original signers.

– Message size is reduced by $n|q|$ because $(s_1, ..., s_n)$ are not necessary in proposed scheme.
– Verification of signature is more efficient because proposed scheme requires only $n+2$ exponentiations (one signature verification and n exponentiations) while traditional scheme requires $2(n+1)$ exponentiation for $n+1$ signature verifications. Moreover, simultaneous multiple exponentiation with distinct bases can be computed very efficiently [MOV97].

Proposed scheme can be used in a very flexible way because the server can choose different combinations of delegations by himself among n delegations depending on the property of his bid. If he has only $l < n$ flight tickets to sell, he can sell them only to l customers of his choice.

6 Conclusion

We have pointed out the necessity of using SNPS to construct SMA. To provide the fairness of a purchase, the proxy signature should represent both customer's and server's signatures. The validity of bid information is verified by comparing it with customer's requirement. From the observation that the features of undetachable signatures are very similar to those of proxy signatures, we extended [KBC00] to provide an RSA-based SNPS scheme and applied it to mobile agent. Very similarly, we provided a Schnorr-based SMA scheme. In multi-proxy situation, Schnorr-based SNPS can be used in very efficient manner because plural customers can share the same system parameters.

Proxy signatures are very useful tools when one needs to delegate his/her signing capability to other party. But in distributed environment like the Internet, it is very difficult to assume the trustedness of original signer, proxy signer, and the proxy key issuing protocol between them. Because the delegation of signing capability to others can be risky, proxy signature schemes should be designed carefully such that proxy signer's responsibility is determined explicitly and any possibility of misuse is prevented. But if we can delegate signing capabilities safely using strong proxy signature schemes, many cryptographic applications in distributed environment such as electronic commerce and mobile agent can be implemented in more efficient and flexible way.

References

[FGS96] W. Farmer, J. Gutmann and V. Swarup, "Security for Mobile Agents: Authentication and State Appraisal", *Proc. of the European Symposium on Research in Computer Security (ESORICS)*, LNCS 1146, Springer-Verlag, pp.118-130, 1996.

[KBC00] P. Kotzanikolaous, M. Burmester and V. Chrissikopoulos, "Secure Transactions with Mobile Agents in Hostile Environments", *ACISP 2000*, LNCS 1841, Springer-Verlag, pp.289-297, 2000.

[KBLK01] H. Kim, J. Baek, B. Lee, and K. Kim, "Secret Computation with Secrets for Mobile Agent using One-time Proxy Signature", *Proc. of SCIS2001*, pages 845–850, 2001.

[KKC99] P. Kotzanikolaous, G. Katsirelos and V. Chrissikopoulos, "Mobile Agents for Secure Electronic Transactions", *Recent Advances in Signal Processing and Communications*, World Scientific and Engineering Society Press, pp.363-368, 1999.

[KPW97] S. Kim, S. Park, and D. Won, "Proxy Signatures, Revisited", *Proc. of ICICS'97,* Y. Han *et al*(Eds.), LNCS 1334, Springer-Verlag, pages 223-232, 1997.

[LKK01] B. Lee, H. Kim and K. Kim, "Strong Proxy Signature and its Applications", *Proc. of SCIS2001*, pages 603–608, 2001.

[LM99] S. Loureio and R. Molva, "Privacy for Mobile Code", *Proc. of Distributed Object Security Workshop OOPSLA'99*, 6 pages, 1999.

[MOV97] A. Menezes, P. Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*, pages 617 - 618, CRC Press, 1997.

[MS97] J. Merwe and S.H. Solms, "Electronic Commerce with Secure Intelligent Trade Agents", *Proc. of ICICS'97*, Y. Han *et al*(Eds.), LNCS 1334, Springer-Verlag, pp.452-462, 1997.

[MUO96] M. Mambo, K. Usuda, and E. Okamoto, "Proxy Signatures: Delegation of the Power to Sign Messages", *IEICE Trans. on Fundamentals*, Vol. E79-A, No. 9, Sep., pages 1338–1353, 1996.

[OSM01] R. Otomura, M. Soshi, and A. Miyaji, "On Digital Signature Schemes for Mobile Agents", *Proc. of SCIS2001*, pages 851–855, 2001.

[PH97] H. Petersen and P. Horster, "Self-certified Keys – Concepts and Applications", *Proc. Communications and Multimedia Security'97*, pages 102 - 116, Chapman & Hall, 1997.

[PS96] D. Pointcheval and J. Stern, "Security Proofs for Signatures", *Advances in Cryptology: Eurocrypt'96*, pages 387 - 398, Springer, 1996.

[ST97] T. Sander and C. F. Tschudin, "Protecting Mobile Agents Against Malicious Hosts", *Mobile Agent Security*, LNCS 1419, Springer-Verlag, pp.44-60, 1997.

[YBX00] L. Yi, G. Bai and G. Xiao, "Proxy Multi-signature Scheme: A New Type of Proxy Signature Scheme", *Electronics Letters*, Vol.36, No.6, pages 527-528, 16th March 2000.

Elliptic Curve Based Password Authenticated Key Exchange Protocols

Colin Boyd[1], Paul Montague[2], and Khanh Nguyen[2]

[1] Information Security Research Centre,
Queensland University of Technology,
Brisbane 4001, Australia
boyd@isrc.qut.edu.au

[2] Motorola Australia Software Centre,
2 Second Ave, Mawson Lakes, SA 5095, Australia
pmontagu@asc.corp.mot.com, knguyen@asc.corp.mot.com

Abstract. We investigate password authenticated key exchange (PAKE) protocols in low resource environments, such as smartcards or mobile devices. In such environments, particularly in the future, it may be that the cryptosystems available for signatures and/or encryptions will be based on elliptic curves, because of their well-known advantages with regard to processing and size constraints. As a result, any PAKE protocols which the device requires should also preferably be implemented over elliptic curves. We show that the direct elliptic curve (EC) analogs of some PAKE protocols are insecure against partition attacks. We go on to propose a new EC based PAKE protocol. A modified version of the protocol for highly constrained devices, such as smartcards, is also presented.

1 Introduction

A protocol that allows two parties to agree on a shared secret key is commonly known as a key exchange protocol. The protocol is said to be *authenticated* if the protocol authenticates one party to the other during the protocol run. Further, if the means of authentication is by a simple password known by both entities, the authenticated key exchange protocol is said to be *password-based.*

Recently, password authenticated key exchange (PAKE) has received significant interest from the research community. One of the reasons for this is that PAKE protocols can be used to establish an authenticated and secret channel between two parties without relying on the existence of a Public Key Infrastructure (PKI), and provides security against both active and off-line dictionary attacks. This is certainly appealing in many environments where the deployment of a PKI is not possible or would be overly complex.

The first PAKE protocol, known as Encrypted Key Exchange (EKE), was suggested by Bellovin and Merritt [3]. Subsequently many other PAKE protocols have been proposed. Nonetheless, most of the protocols have been proposed for only RSA or Discrete Logarithm (DL) settings so far. It seems that most authors have presumed that the adaptation of DL based protocols to the elliptic curve (EC) environment is straightforward. To the best of our knowledge, no concrete EC based PAKE protocol has been proposed in the literature.

In a low resource environment, the natural choice for cryptographic protocols would be an EC implementation. This is due to the low computation and storage costs of EC based protocols. Since EC primitives may be the only ones available in certain environments, it is important to study the precise adaptation of DL-based PAKE protocols to an EC setting.

In this paper, we investigate EC analogs of PAKE protocols. We show that direct EC analogs of the EKE protocol and its variants are susceptible to partition attacks. We then go on to propose an EC encrypted key exchange protocol that is secure against partition attacks. We further propose a modification of the protocol for low resource (*e.g.* smartcard) applications. Here we stress that the EC analogs of other PAKE protocols such as SPEKE[5] or PAK[4] do not immediately appear to suffer from the partition attack described in this paper.

The remainder of this paper is organized as follows. Section 2 gives an introduction to PAKE. This includes a discussion on the possible attacks against a PAKE protocol. Section 3 gives a detailed description of the EKE protocol. A discussion on how a partition attack can be applied to the protocol is also given in this section. Section 4 reviews the concept of elliptic curves and twisted elliptic curves, establishing some notation and elementary results for the subsequent sections. Section 5 proposes a new elliptic curve encrypted key exchange protocol. In the section, we also justify our solution by showing that trivial solutions are insecure against partition attacks. A modification of the proposed protocol for smartcard applications is also included in the section. This has a DL analog, which is briefly described and discussed. Finally, conclusions are presented in Section 6.

2 Password Authenticated Key Exchange

In its simplest form, a PAKE protocol involves two parties both possessing the same secret, referred to as the password. This password is typically short, since it usually has to be memorised by a human participant. The PAKE protocol allows the involved parties to exchange information from which each party can derive a secret key. This secret key satisfies all requirements of a conventional Diffie-Hellman key exchange protocol. In particular, this shared secret key is not known to any parties not involved in the exchange protocol. Furthermore, the nature of password authentication ensures that a party can follow the protocol correctly (and thus be accepted by the other party) only if the party knows the correct password. During the protocol, a party can always detect whether the other party in the exchange possesses the correct password.

The first PAKE protocol was the EKE protocol proposed by Bellovin and Merritt [3]. The idea of their proposal is to use the password to symmetrically encrypt the protocol messages of a standard Diffie-Hellman key exchange. An attacker could decrypt the symmetric encryption by guessing the password but could not tell whether the decryption results in a valid message. However, Patel subsequently showed [6, 7] that EKE protocols are susceptible to a partition attack (see below for further details). The solution to prevent the partition attack, is to carefully choose the system parameters so that the attack is no longer applicable. Fortunately, such restrictions on the system parameters do not introduce any performance penalty or security issues.

Another well-known PAKE protocol is Simple Password-authenticated Exponential Key Exchange (SPEKE) proposed by Jablon [5]. The idea of the protocol is to involve the password in computing the base used in conventional Diffie-Hellman key exchange. The protocol also introduces two extra steps for authenticating the generated secret key for each party respectively.

Recently two new PAKE protocols with provable security have been proposed. Bellare *et al.* [1] have given a specific variant of EKE which is provably secure in the ideal cipher model. The protocol of Boyko *et al.* [4] is also a variant of EKE in which the symmetric encryption takes the form of multiplication in the Diffie-Hellman group; it is provably secure in the random oracle model.

The security of a standard key exchange protocol may be measured against both *passive attacks* in which the adversary wiretaps valid instances of the key exchange protocol, and against *active attacks* in which the adversary may also masquerade as a valid protocol principal. Attacks may be aimed at obtaining information about the generated secret or about the long-term keying material.

The classical Diffie-Hellman key exchange and its EC analog are known to be secure against passive attacks. However, it is clearly insecure against an impersonation attack, because of the lack of an authentication mechanism being deployed. To prevent such attacks long-term keys are required, which in a PAKE protocol takes the form of the password. The password allows each party in the protocol to authenticate the other party and thus prevents an adversary from impersonating an authorized party in the PAKE protocol run.

However, the low entropy of a typical password opens a new possible attack against a PAKE protocol. This type of attack is often known as the off-line dictionary attack. In such an attack, the adversary tries to interact (even if unsuccessfully) with an honest party and also gathers information from exchanges between two honest parties. The adversary then applies a brute-force attack over the domain of the passwords (i.e. the dictionary) off-line. The attacker is successful if the gathered information can confirm which password in the dictionary is the valid one. A special class of the dictionary attack is the *partition attack*. In a partition attack, the adversary tries to use the gathered information to partition the password space (the dictionary) into feasible and infeasible passwords; if the latter set is large then the adversary may simply search through and eliminate all passwords in this set. Typically the correct password may be recovered after a number of valid sessions have been observed from the intersection of the feasible partition of the passwords for each session.

3 Diffie Hellman Encrypted Key Exchange

In this section, we give an overview of the Diffie-Hellman based encrypted key exchange (DH-EKE) protocol. This is one of the three variants of EKE proposed by Bellovin and Merritt in [3]. We also provide a brief description of how a partition attack can be applied against the protocol. Patel [6] gives further details.

3.1 The DH-EKE Protocol

The DH-EKE protocol involves two parties, Alice - the initiator of the protocol and Bob. Alice and Bob share a secret password P that is not known to any other party. There is also a publicly known prime number p and a publicly known generator g of the field $\mathsf{GF(p)}$. Also a symmetric encryption algorithm $Enc()$, its corresponding decryption algorithm $Dec()$ and a one-way hash function $\mathcal{H}()$ are publicly known. A variant of the DH-EKE protocol is as follows (see also figure

$$\begin{array}{lcr}
\text{Alice} & & \text{Bob} \\
r_A \in_R \mathsf{GF(p)} & & r_B \in_R \mathsf{GF(p)} \\
g_A \leftarrow g^{r_A} \bmod p & & g_B \leftarrow g^{r_B} \bmod p \\
 & \xrightarrow{Enc_P(g_A)} & \\
 & & g_A \leftarrow Dec_P(Enc_P(g_A)) \\
 & & K_B \leftarrow g_A^{r_B} \bmod p \\
 & & Auth_B \leftarrow \mathcal{H}(K_B \| B) \\
 & \xleftarrow{Enc_P(g_B), Auth_B} & \\
g_B \leftarrow Dec_P(Enc_P(g_B)) & & \\
K_A \leftarrow g_B^{r_A} \bmod p & & \\
Auth_A \leftarrow \mathcal{H}(K_A \| A) & & \\
Auth_B \overset{?}{=} \mathcal{H}(K_A \| B) & & \\
 & \xrightarrow{Auth_A} & \\
 & & Auth_A \overset{?}{=} \mathcal{H}(K_B \| A)
\end{array}$$

Fig. 1. A variant of the DH-EKE protocol

1):

1. Alice and Bob generate random numbers r_A and r_B and compute $g_A = g^{r_A} \bmod p$ and $g_B = g^{r_B} \bmod p$ respectively.
2. Alice and Bob encrypt g_A and g_B using the shared password P to generate $Enc_P(g_A)$ and $Enc_P(g_B)$ respectively.
3. Alice sends $Enc_P(g_A)$ to Bob.
4. Upon receiving $Enc_P(g_A)$, Bob computes $g_A = Dec_P(Enc_P(g_A))$. Bob then computes the key $K_B = g_A^{r_B}$. Bob then generates the authenticator $Auth_B = \mathcal{H}(K_B \| B)$ of K_B.

5. Bob sends $Enc_P(g_B)$ and $Auth_B$ to Alice.
6. Upon receiving $Enc_P(g_B)$ and $Auth_B$, Alice applies Dec_P to $Enc_P(g_B)$ to recover g_B. Alice then computes $K_A = g_B^{r_A}$ and verifies that $Auth_B \equiv \mathcal{H}(K_A \| B)$.
7. If the verification passes, Alice accepts $Auth_B$ and sends to Bob $Auth_A = \mathcal{H}(K_A \| A)$.
8. In turn, Bob checks that $Auth_A \equiv \mathcal{H}(K_B \| A)$. If the verification passes, Bob accepts $Auth_A$.

The protocol is successful if both Alice and Bob accept $Auth_B$ and $Auth_A$ respectively. The generated key then is $K = K_A = K_B$. The completeness is due to $K_A = g_B^{r_A} = g^{r_A r_B} = g_A^{r_B} = K_B$.

The protocol described here is not the exact description of the original DH-EKE. It is in fact an instance of the construction described by Bellare and Rogaway [2]. The difference between this protocol and the original DH-EKE protocol is the existence of values $Auth_A$ and $Auth_B$. These two values are to authenticate Alice to Bob and Bob to Alice respectively. In the original DH-EKE, a different authentication mechanism is used. Nonetheless both methods achieve the same goal.

3.2 Variants of the DH-EKE Protocol

The DH-EKE protocol can be modified in many ways. One type of variant changes the construction of the authentication part as shown above.

Another modification that can be made to the protocol is to omit the encryption with the password P by either Bob or Alice (but not both). Then instead of sending the encryption of g_A or g_B, Alice or Bob shall send the plaintext g_A or g_B to the other party. In this variant, the order of authentication between Alice and Bob must also be modified. The rule is that the party who does not perform any encryption in the first part, must initiate the authentication step. This prevents an active adversary using the authentication field to mount an off-line dictionary attack. Patel [6] discusses the security of such omissions in detail.

3.3 A Partition Attack against DH-EKE Family

Patel [7, 6] has shown that the DH-EKE protocol and its variants are susceptible to a partition attack if the values of g and p are not chosen carefully.

If the value g is not a generator of $\mathsf{GF(p)}$ but only a generator of a subgroup of order q over $\mathsf{GF(p)}$, an adversary can mount a partition attack as follows. Firstly the adversary obtains $Enc_P(g_A)$ by wiretapping an exchange between Alice and Bob. Next, the adversary tries to decrypt $Enc_P(g_A)$ using a password P_i. If the password P_i is correct, the decryption will result in a value g_A which is of order q. If P_i is not the correct password, it is likely that the decryption will result in a value g'_A which is not of order q. The probability that the decryption will result in a value of order r, $r|p-1$, for a random P_i is $\frac{\phi(r)}{p}$. The attacker can check, by raising the result to the power q and checking whether 1 is obtained, whether the order divides q. It can then be seen that the probability that 1 is obtained, for an incorrect password, is $\frac{q}{p-1}$. Thus the possible space of valid passwords is reduced by a factor of $\frac{q}{p-1}$, on average, by observing one exchange session. Over a number of sessions the space of valid passwords will be narrowed down to a single password at a logarithmic rate.

To avoid the attack, it is suggested that g has to be a generator of $\mathsf{GF(p)}$ and that if Alice is allowed to choose p and g for the protocol, Bob must check that g is indeed a generator.

Similar partition attacks are possible if the value of p is not chosen carefully. In this case, if trial decryption of $Enc_P(g_B)$ with candidate passwords leads to values equal to or larger than p, then these candidate passwords may be eliminated.

4 Elliptic Curves and Twisted Elliptic Curves

In this section, we review the basic notation and definitions of elliptic curves, as well as the concept of the twist of an elliptic curve. This concept is crucial to the construction of our new protocol.

The use of elliptic curve groups in public key cryptography was first proposed by Koblitz [9] and Miller [8]. Recently there has been much focus on such cryptosystems, with adoption in various standards, such as WAP [10], IEEE P1363 [11] and ANSI X9.62 [12]. This is because public key methods based on elliptic curve groups typically have lower processing requirements, and can achieve the same level of security with considerably shorter key sizes than cryptosystems based on the more traditional RSA and standard discrete logarithm schemes. As such, elliptic curve cryptographic systems are ideal for environments where processing power, time and/or network bandwidth are at a premium. Consequently, there is a drive for the adoption of elliptic curve based protocols in wireless environments and smartcard based applications.

For the sake of simplicity, we consider the case of a curve in a field of characteristic greater than 3. For the case of characteristic two, analogous statements hold true [13]. A more general discussion of twists of elliptic curves is given by Silverman [14].

Following Blake *et al* [13], we consider a curve defined over the field $K = \mathsf{GF(q)}$, where $q = p^m$ and $p > 3$ is a prime. Consider the curve in short Weierstrass form, *i.e.*

$$E_{a,b} : Y^2 = X^3 + aX + b.$$

Set $g(X) = X^3 + aX + b$, so that the equation of the curve becomes $Y^2 = g(X)$. As shown in the literature, we may define an additive (abelian) group on the set of points on this curve (taken together with the point at infinity). It is this group, and the discrete logarithm problem defined therein, which may be used to define cryptographic primitives.

In the following sections, use shall be made of the concept of the twist of an elliptic curve. Consider then the curve $E_{a',b'}$ where $a' = v^2a$ and $b' = v^3b$ for some $v \in K^*$. If we set $g_v(X) = v^3 g(X/v)$, then we have the equation of the curve $E_{a',b'}$ given by $Y^2 = g_v(X)$. However, $E_{A,B}$ is isomorphic to $E_{a,b}$ over K if and only if $A = u^4 a$ and $B = u^6 b$ for some $u \in K^*$. Hence the curve $E_{a',b'}$ is isomorphic to $E_{a,b}$ if v is a quadratic residue. Furthermore, if v is not a quadratic residue, then there is a unique such curve, up to isomorphism over K. This is called the **twist** of $E_{a,b}$. [Note that over $\mathsf{GF(q^2)}$ the original curve and its twist are isomorphic.]

An observation (see *e.g* [13]) that will be of use in the subsequent sections is the following. Consider $X \in K$ for which $g(X) \neq 0$. Then if $g(X)$ is a non-zero quadratic residue, X is the x-coordinate of a point on $E_{a,b}$. Otherwise, $g_v(vX)$ is a quadratic residue, and hence vX is the x-coordinate of a point on $E_{a',b'}$. The following lemma summarises the relevant properties about the connection between a curve and its twist.

Lemma 1. *Let*

$$C = \{X \in K | g(X) \textit{ is a quadratic residue in } K\}$$
$$T = \{X \in K | g_v(vX) \textit{ is a quadratic residue in } K\}$$

Then

1. $C \cap T = \emptyset$. *If* $g(X) \neq 0$ *then* $X \in C \cup T$.
2. $X \in C \iff g(X) \neq 0$ *and* $\exists Y$ *such that* $(X, Y) \in E_{a,b}$.
3. $X/v \in T \iff g(X/v) \neq 0$ *and* $\exists Y$ *such that* $(X, Y) \in E_{a',b'}$.

Note also in the following sections that we will need to represent the points of the elliptic curve in a compressed form. Denoting the points naively, an affine point (X, Y) requires $2n$ bits, where n is the bit length of the underlying field. There is a trivial reduction to $n+1$ bits by observing that, given X, the value of Y is one of the two solutions of a quadratic equation. A single bit may be used to distinguish between these two solutions. While this compressed form is clearly convenient and cost effective (particularly in situations in which transmission bandwidth is limited) in the following we shall see that it is essential in order to obviate simple attacks on the protocols described.

In order for the elliptic curve $E_{a,b}$ to be suitable for use in a cryptosystem, it is required that [13]:

– the group has a subgroup of large prime order,
– the curve is not anomalous ($q = N_{a,b} = p$, where $N_{a,b}$ is the group order),
– the curve satisfies the MOV condition [13, 11] (the smallest value of l such that $q^l = 1 \bmod N_{a,b}$ should be large).

In the protocols we will describe in which both the curve and its twist are used, these properties are also required of the twist. [The order of the group of the twisted curve is given by the relation $N_{a,b} + N_{a',b'} = 2q+2$, in the case that $E_{a',b'}$ is the twist of $E_{a,b}$. Hence there is little additional effort required in verifying these properties hold for any generated curve and its twist over and above for the curve alone.] We shall assume in the following that the elliptic curves we use satisfy these security constraints.

5 Elliptic Curve Encrypted Key Exchange

In this section, we present our elliptic curve encrypted key exchange (EC-EKE) protocol. We further show an unbalanced variant of the protocol. Such a protocol may be utilized in a situation in which one of the parties has limited processing power, such as communication between a smartcard or mobile device and a terminal or server. First, however, we will show that the direct EC analog of DH-EKE protocol is insecure and thus justify our design.

Throughout this section, we shall use the notation of the preceding section, and consider a curve $E_{a,b}$, together with its twist $E_{a',b'}$, for suitably chosen a' and b'. For further simplicity, we restrict to the case $q = p$, *i.e.* consider the curve over $\mathsf{GF(p)}$.

5.1 Trivial Protocols Are Insecure against Partition Attacks

Following the usual methodology of replacing the DL group operations with operations in an EC group, the obvious procedure would be to design the EC-EKE protocol as the direct EC analog of a variant of the DH-EKE protocol.

The principle of any variant of the DH-EKE protocol is that Alice, say, will encrypt and send the encryption of $g_A = g^{r_A}$ to Bob. The direct analog in EC is that Alice will encrypt the point $G_A = r_A{*}G$ and send the encryption $Enc_P(G_A)$ to Bob.

The trivial encryption method is to encrypt (X_A, Y_A) in $Enc_P(G_A)$, where $G_A = (X_A, Y_A)$. In this case, the adversary can simply apply an off-line dictionary attack for a valid $Enc_P(G_A)$ by decrypting $Enc_P(G_A)$ with every password P_i in the password space. Clearly, if P_i is incorrect, the decryption should result in a random pair (X_i, Y_i). Even if X_i, $Y_i \in \mathsf{GF(p)}$, the point (X_i, Y_i) is on the elliptic curve $E_{a,b}$ only if it satisfies

$$Y_i^2 = g(X_i) \bmod p.$$

This happens with a probability of order $1/p$ for a random pair $(X_i, Y_i) \in \mathsf{GF(p)}^2$. Typically the size of the password space is much less than p. Hence the adversary should be able to identify the correct password P given a single valid encryption $Enc_P(G_A)$ using such a dictionary attack.

As discussed in the previous section, an alternative more compact form for the representation of, G_A of the point is its compressed form, in which the y-coordinate is replaced by a single bit. If the adversary applies a dictionary attack on the encryption $Enc_P(G_A)$ in this situation, the adversary will be able to recover the x-coordinate X_i for the password choice P_i and a bit indicating which solution Y_i to choose. If the password P_i is incorrect, X_i will be essentially random. Observe however that a random X_i is a valid x-coordinate only if $g(X_i)$ is a quadratic residue. ¿From Hasse's theorem (see *e.g.* [13]), we know that the number of such values X_i in $\mathsf{GF(p)}$ is in the range $[(p+1)/2-\sqrt{p}, (p+1)/2+\sqrt{p}]$. Thus a random X_i in $\mathsf{GF(p)}$ is a valid x-coordinate of $E_{a,b}$ with a probability in the range $[1/2-O(1/\sqrt{p}), 1/2+O(1/\sqrt{p})]$. Hence the adversary can successfully apply a partition attack by reducing the possible password space by roughly half given a valid $Enc_P(G_A)$. This means the password can be recovered given a number of sessions of the order of the log of the size of the password space.

5.2 An Elliptic Curve Encrypted Key Exchange Secure against Partition Attacks

In the previous subsection, we have shown that the direct EC analog of any variant of the DH-EKE protocol is insecure. In this subsection, we propose a new EC-EKE protocol and show that the protocol is secure against partition attacks.

In order to avoid the elementary attacks described in the previous subsection, the simplest approach would be to ensure that any candidate x-coordinate observed by an adversary is valid. This would then obviate the partition attack.

We recall from Lemma 1 that for $X \in \mathsf{GF(p)}$ for which $g(X) \neq 0$, then if $g(X)$ is a non-zero quadratic residue, X is the x-coordinate of a point on the curve. Otherwise, $g_v(vX)$ is a quadratic residue, and hence vX is the x-coordinate of a point on the twist of the curve. Using this observation, an EC-EKE protocol is designed as follows.

Let $E_{a',b'}$ be a twisted curve of $E_{a,b}$. Let G be a generator point of the curve $E_{a,b}$ and H be a point which generates the curve $E_{a',b'}$. Here we assume that the points in $E_{a,b}$ and $E_{a',b'}$ respectively form a cyclic group. There are two important remarks that should be made in relation to the choice of these parameters.

- Generation of suitable curve/twist pairs will inevitably take considerably longer than when the properties of the twist are irrelevant. However, this is a one time setup cost and a single curve may be re-used for a large number of different users.
- It is common to run Diffie-Hellman exchange in prime order subgroups in order to avoid small subgroup attacks. To preserve the properties which prevent partition attacks we have to use generators of the whole of the curve groups. Therefore we need to additionally require *either* that the curve group and its twist have prime order, *or* (more practically) check that all received values are not in any small subgroup. The latter may be achieved by making a number of simple checks depending on the factorisation of the curve co-factor. We regard this matter as an implementation detail and therefore ignore it in the protocol description.

To generate a password-authenticated shared secret key K, Alice and Bob proceed as follows (see figure 2):

1. Alice randomly selects either the curve $E_{a,b}$ or $E_{a',b'}$ for use in this run of the protocol.
2. Alice chooses a random r_A and computes $G_A = r_A * G$ if the selected curve is $E_{a,b}$ or $H_A = r_A * H$ otherwise.

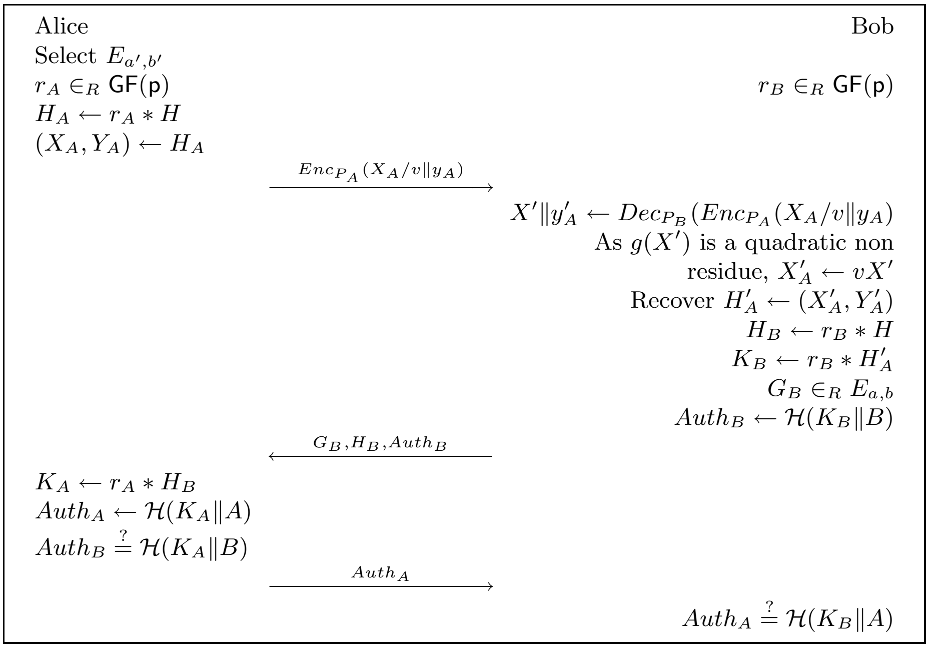

Alice
Bob
Select $E_{a',b'}$
$r_A \in_R \mathsf{GF(p)}$
$r_B \in_R \mathsf{GF(p)}$
$H_A \leftarrow r_A * H$
$(X_A, Y_A) \leftarrow H_A$
$\xrightarrow{Enc_{P_A}(X_A/v \| y_A)}$
$X' \| y'_A \leftarrow Dec_{P_B}(Enc_{P_A}(X_A/v \| y_A)$
As $g(X')$ is a quadratic non residue, $X'_A \leftarrow vX'$
Recover $H'_A \leftarrow (X'_A, Y'_A)$
$H_B \leftarrow r_B * H$
$K_B \leftarrow r_B * H'_A$
$G_B \in_R E_{a,b}$
$Auth_B \leftarrow \mathcal{H}(K_B \| B)$
$\xleftarrow{G_B, H_B, Auth_B}$
$K_A \leftarrow r_A * H_B$
$Auth_A \leftarrow \mathcal{H}(K_A \| A)$
$Auth_B \overset{?}{=} \mathcal{H}(K_A \| B)$
$\xrightarrow{Auth_A}$
$Auth_A \overset{?}{=} \mathcal{H}(K_B \| A)$

Fig. 2. The EC-EKE protocol: the chosen curve is the twisted curve $E_{a',b'}$

3. Alice compresses the point $G_A = (X_A, Y_A)$ (or $H_A = (X_A, Y_A)$) to (X_A, y_A) where y_A is a single bit representing Y_A in the compressed form.
4. If the (untwisted) curve $E_{a,b}$ is chosen, Alice sends $Enc_{P_A}(X_A \| y_A)$ to Bob. Otherwise Alice sends $Enc_{P_A}(X_A/v \| y_A)$ to Bob.
5. Upon receipt of the ciphertext, Bob decrypts it to obtain $X' \| y'_A$. If Bob finds that $g(X')$ is a quadratic residue, then Alice's chosen curve is the untwisted curve and the x-coordinate is $X'_A = X'$. Otherwise, Alice's chosen curve is the twisted curve and the x-coordinate is $X'_A = vX'$.
6. Once Bob has determined the x-coordinate X'_A, Bob recovers the point G'_A (or H'_A) from X'_A and y_A. Bob also chooses a random value r_B. Bob then computes $G_B = r_B * G$ and $K_B = r_B * G_A$ if the untwisted curve is chosen by Alice and chooses a random point H_B on the twisted curve. Otherwise Bob computes the points $H_B = r_B * H$ and $K_B = r_B * H_A$, and chooses a random point G_B on the untwisted curve.
7. Next Bob sends the points G_B and H_B and the authenticator $Auth_B = \mathcal{H}(K_B \| B)$ to Alice.
8. In turn, Alice computes $K_A = r_A * G_B$ if the untwisted curve $E_{a,b}$ is chosen. Otherwise Alice computes $K_A = r_A * H_B$.
9. Alice then verifies $Auth_B \equiv \mathcal{H}(K_A \| B)$. If so, Alice sends $Auth_A = \mathcal{H}(K_A \| A)$ to Bob.
10. Finally, Bob verifies that $Auth_A \equiv \mathcal{H}(K_B \| A)$. If so, the protocol is completed.

The shared secret key K is derived from K_A or K_B using a publicly known algorithm. As this step is not important in our protocol, we omit it here and simply assume that K_A and K_B are Alice's and Bob's copy of the shared secret key respectively.

The completeness of the protocol is as follows (assuming Alice chooses the untwisted curve - clearly a similar result holds in the other case). If $P_A = P_B$, at step 4, Bob will recover $X'_A = X_A$ and thus can determine that Alice chose the untwisted curve at step 1. This means that at step 6, Bob will recover the point $G'_A = G_A$ that is chosen by Alice. Thus we have $K_B = r_B * G'_A = r_B * r_A * G = r_A * G_B = K_A$ and the checks

$$Auth_B = \mathcal{H}(K_B \| B) \equiv \mathcal{H}(K_A \| B)$$

and

$$Auth_A = \mathcal{H}(K_A \| A) \equiv \mathcal{H}(K_A \| A)$$

follow.

5.3 Security

Formally proving the security for this protocol is difficult as this uses a symmetric encryption scheme for which no formal proof model exists. Under passive attacks, the security of this protocol is based on the fact that both $E_{a,b}$ and $E_{a',b'}$ are cyclic and G and H generate $E_{a,b}$ and $E_{a',b'}$ respectively. Thus the triplet $\{G_A, G_B, K\}$ (or $\{H_A, H_B, K\}$) is indistinguishable from a random set in $E_{a,b}$ (or $E_{a',b'}$). This implies that passive attacks are not feasible assuming that $\mathcal{H}()$ leaks no information to the attacker. For active attacks, we consider the protocol under three different types of attacks, namely impersonation attacks, off-line dictionary attacks and partition attacks.

For the impersonation attack:

- If the attacker impersonates Bob, the attacker will need to supply a valid $Auth_B$ to Alice given $Enc_{P_A}(X_A \| y_A)$. Alice will accept $Auth_B$ only if $Auth_B \equiv \mathcal{H}(K_A \| B)$. This requires Bob to know K_A when Bob computes $Auth_B$. This is possible only if Bob can derive the correct point G_A from the encryption $Enc_{P_A}(X_A \| y_A)$. This happens only if Bob knows the correct password.
- If the attacker impersonates Alice, the attacker will have to give Bob the value $Enc_{P_i}(X_A \| y_A)$ for a password P_i of the attacker's choice. Unless $P_i = P_A$, the point G'_A or H'_A that Bob recovers will differ from the point G_A (or H_A) that the attacker has chosen in the first place. Bob accepts the protocol only if Bob accepts $Auth_A$. This happens only if $K_A = K_B$ or that the attacker is able to compute K_B. Assuming that $\mathcal{H}()$ is a random oracle, the attacker will have to compute $K_B = r_B * G'_A$ (or $r_B * H'_A$) from G_A and G_B (or H_A and H_B). This is solvable only if the attacker can solve the EC discrete logarithm problem for G'_A and G_A (or H'_A and H_A).

For dictionary and partition attacks, an attacker can decrypt $Enc_{P_A}(X\|y)$ using a password P_i to obtain $(X_i\|y_i)$ where X_i is random. However, as we have seen above, all $X_i \in \mathsf{GF(p)}$ are valid, with the X where $g(X) = 0$ being sufficiently negligible in practise. Furthermore as $E_{a,b}$ $(E_{a',b'})$ is cyclic, the point G_i (or H_i) is indistinguishable from a random point under the EC Diffie-Hellman assumption. Thus the decryption gives no useful information about the plaintext to the attacker provided the value of p is chosen suitably, exactly as for DH-EKE as discussed in section 3.3 and by Patel [7]. Note that it is essential in this protocol that the points be represented in compressed form prior to encryption. If not, then sufficient information will remain to perform a parition attack.

5.4 An Unbalanced Variant for Smartcards

As we have discussed, the use of EC methods is often advantageous in environments in which there are limited resources for computation. In severely constrained devices, such as smartcards, further optimizations may be required. In this subsection, we describe a suitable protocol for this case.

An example of the sort of situation we have in mind is that a user is required to authenticate to a smartcard by means of a password. The link between the user's password entry device (a terminal) and the smartcard may be insecure. A PAKE protocol is an ideal solution for this particular problem. In this scenario, authentication of the card back to the user is not required. However, it may easily be added to our proposed protocol if required.

The idea of the protocol is to optimize the number of scalar (EC) multiplications that the card has to perform, since this is the operation which is computationally expensive. This is achieved by fixing the value r_A generated by the card for all transactions. The price we pay is the loss of forward secrecy. However if we link this value to the secret stored in the card, the use of which the user authentication is there to protect, then if r_A is compromised the secret in the card is compromised, and thus there is no longer anything to protect. This means that forward secrecy is no longer a significant requirement.

The protocol is as follows. Again, we use the notation of section 4. To set up the protocol, the card chooses either $E_{a,b}$ or $E_{a',b'}$. For the sake of simplicity, let us assume that the card chooses $E_{a,b}$. The card then chooses a random value r_A, preferably linked to the secret stored in the card. It then computes $G_A = r_A * G$. The above process is performed only once. The card then stores r_A and G_A for future use. It also stores a counter c initially set to 0. When a user requires to establish a password-authenticated secret session with the card, the card and the user perform the following variant of the EC-EKE protocol:

1. The card increases the counter $c = c+1$ and sends to the user $Enc_{P_A}(X_A\|y_A)$ where X_A is the x-coordinate of G_A and y_A is the bit representing the y-coordinate Y_A of G_A.
2. The user decrypts $Enc_{P_A}(X_A\|y_A)$ to obtain X'. The user then determines whether $g(X')$ is quadratic residue and thus determines which is the chosen curve. In the case here, the user determines that $E_{a,b}$ is the chosen curve. Then the user constructs the point G'_A from $X'_A = X'$ and y_A.

3. Next the user chooses a random value r_B and computes $G_B = r_B * G$ and $K_B = r_B * G'_A$. Also the user chooses a random point H_B of $E_{a',b'}$. The user also constructs the authenticator $Auth = \mathcal{H}(K_B \| c)$. Then the user sends $Auth$, G_B and H_B to the card.
4. The card computes $K_A = r_A * G_B$ and verifies that $Auth \equiv \mathcal{H}(K_A \| c)$. If so, the card accepts the user and the protocol is completed.

The advantage of this scheme is that the card only needs to perform a single scalar multiplication, a saving of one scalar multiplication compared with the original protocol. This reduces the computational requirement for the card by about a half. There is a potential replay attack on the protocol if an old K_A or K_B value is available to an attacker; this is discussed further below.

It is clear that this protocol is secure against partition and dictionary attacks. A similar argument as for our EC-EKE protocol can be applied here. Impersonating the user would be as difficult as for the original protocol. This is because the tasks that the user has to perform and the card has to verify in regard to the user's information remain unchanged. The value c is introduced to compensate for the fixing of r_A. Thus, each session is different and a straight replay attack is not possible.

5.5 Some Comments on the DL Analog

For completeness, we make a few points regarding the DL analog of the above unbalanced scheme. The relevance is that the following discussion also touches on points of difference between the EC and DL schemes, showing again that the map between protocols is not always straightforward.

There is a naive analog of the above EC protocol as follows.

Let us suppose that we are working over the field $\mathsf{GF(p)}$ with g a generator. As before, the counter c is initialized to zero. In addition, the card chooses a random value r_A, preferably linked to the secret stored in the card. It then computes $g_A = g^{r_A} \bmod p$, and stores r_A and g_A for future use.

The protocol may then be as follows:

1. The card increases the counter $c = c + 1$ and sends to the user $Enc_{P_A}(g_A)$.
2. The user decrypts this to obtain g_A.
3. Next the user chooses a random value r_B and computes $g_B = g^{r_B} \bmod p$ and $K_B = {g'_A}^{r_B} \bmod p$. The user also constructs the authenticator $Auth = \mathcal{H}(K_B \| c)$. Then the user sends $Auth$ and g_B to the card.
4. The card computes $K_A = {g_B}^{r_A} \bmod p$ and verifies that $Auth \equiv \mathcal{H}(K_A \| c)$. If so the card accepts the user and the protocol is completed.

This naive protocol satisfies the same security properties as for the EC case. However, we note that compromise of K_A or K_B would allow an adversary to impersonate the user in subsequent protocol runs by replaying the compromised session. Though these values are temporary (the session key should be derived from them and then they should be deleted) and this vulnerability is therefore not of major significance, it is easily fixed by the following version of the protocol:

1. The card increases the counter $c = c + 1$ and sends to the user $Enc_{P_A}(g_A)$.
2. The user decrypts this to obtain g_A.
3. Next the user chooses a random value r_B and computes $g_B = g^{r_B} \bmod p$ and $K_B = {g'_A}^{r_B} \bmod p$. The user also constructs the authenticator $Auth = \mathcal{H}(K_B)$. Then the user sends $Auth$ and $Enc_{P_B}(c + g_B)$ to the card.
4. The card recovers g_B and computes $K_A = {g_B}^{r_A} \bmod p$ and verifies that $Auth \equiv \mathcal{H}(K_A)$. If so, the card accepts the user and the protocol is completed.

Note that the tying of the counter c to the user's DH value g_B by the password P_B prevents the sort of replay attack noted above should K_A or K_B be compromised. Of course, compromise of r_A is still fatal as discussed above.

It is interesting to note however that there does not seem to be an EC analog of the above protocol. The encryption on the data sent by the user could be mapped, as we have done above, to a protocol using a curve and its twist. However, the card would also have to use the same protocol ideas, and the user and card cannot make an independent choice of curve to use, *i.e.* if the card chooses the twisted curve then so must the user. A partition attack therefore becomes feasible again. We omit the details, but simply note it as an example of the distinction between DL and EC schemes which is not immediately apparent from a naive approach.

6 Conclusion

Motivated both by the recent interest in PAKE protocols and by the uptake of EC cryptographic schemes, we have considered the transition of the DL based PAKE protocol DH-EKE to the EC environment. We have demonstrated that the naive EC analogs of such schemes are vulnerable to partition attacks. Furthermore, we have proposed a secure EC variant, using the concept of the twist of an elliptic curve in order to render the effectiveness of the partition attack negligible. Unbalanced schemes, in both the DL and EC settings, adapted to severely resource limited participants on one side of the protocol, have also been proposed and examined. It is observed throughout that the transition to EC schemes cannot be made naively, and it is suggested that distinct protocols for the DL and EC environments may need to be considered.

Further development of these ideas is the subject of ongoing work. In particular, given that a curve and its twist over $\mathsf{GF(p)}$ are isomorphic over $\mathsf{GF(p^2)}$, it might seem that the problems touched on in section 5.5 regarding introducing encryption of the data sent by both parties may be resolved if the two parties are able to derive the shared secret elliptic curve point in this larger structure.

Note that though PAKE schemes such as SPEKE [5] and PAK [4] do not immediately appear to suffer from the problems encountered above in transitioning to an EC analog, further investigation is required to clarify this statement. We have at the very least demonstrated that the development of EC analogs of PAKE protocols can be non-trivial.

Acknowledgements

This research is part of an ARC SPIRT project undertaken jointly by Queensland University of Technology and Motorola.

References

[1] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In *Advances in Cryptology – Eurocrypt'2000*, pages 139–155. LNCS vol. 1807, Springer-Verlag, 2000.

[2] Mihir Bellare and Phillip Rogaway. The AuthA protocol for password-based authenticated key exchange. In *Submission to IEEE P1363 study group*, 2000.

[3] S. Bellovin and M. Merritt. Encrypted key exchange: Password based protocol secure against dictionary attackes. In *Proceedings of the Symposium on Security and Privacy*, pages 72–84. IEEE, 1992.

[4] V. Boyko, P. MacKenzie, and S. Patel. Provably secure password-authenticated key exchange. In *Advances in Cryptology – Eurocrypt'2000*, pages 156–171. LNCS vol.1807, Springer-Verlag, 2000.

[5] D. Jablon. Strong password-only authenticated key exchange. *ACM Computer Communication Review*, 26(5):5–20, 1996.

[6] S. Patel. Number theoretic attacks on secure password schemes. In *Proceedings of the Symposium on Security and Privacy*, pages 236–247. IEEE, 1997.

[7] S. Patel. Information Leakege in Encrypted Key Exchange. In *Proceedings of the DIMACS Workshop on Network Threats*, 1997.

[8] V. Miller. Use of elliptic curves in cryptography. In *Advances in Cryptology – Crypto 85*, pages 417–426. LNCS vol.218, Springer-Verlag, 1986.

[9] N. Koblitz. Elliptic curve cryptosystems. *Math. Comp.*, 48:203–209, 1987.

[10] *Wireless Application Protocol – Wireless Transport Layer Security Specification*, Wireless Application Forum Ltd, 2000.

[11] *IEEE P1363 Study Group for Future Public-Key Cryptography Standards.* http://grouper.ieee.org/groups/1363/StudyGroup.

[12] *Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA).* American National Standard for Financial Services, X9.62, 1998.

[13] I.F. Blake, G. Seroussi and N.P. Smart. *Elliptic Curves in Cryptography.* LMS Lecture Note Series 265, CUP, 1999.

[14] J.H. Silverman. *The Arithmetic of Elliptic Curves.* Springer-Verlag, GTM 106, 1986.

Elliptic Curve Cryptography on a Palm OS Device

André Weimerskirch[1], Christof Paar[2], and Sheueling Chang Shantz[3]

[1] CS Department, Worcester Polytechnic Institute, USA
weika@wpi.edu
[2] ECE and CS Department, Worcester Polytechnic Institute, USA
christof@ece.wpi.edu
[3] Networking and Security Research, Sun Microsystems Laboratories, USA
Sheueling.Chang@sun.com

Abstract. The market for Personal Digital Assistants (PDA) is growing rapidly and PDAs are becoming increasingly interesting for commercial transactions. One requirement for further growing of eCommerce with mobile devices is the provision of security. We implemented elliptic curves over binary fields on a Palm OS device. We chose the NIST recommended random and Koblitz curves over $GF(2^{163})$ that are providing a sufficient level of security for most commercial applications. Using Koblitz curves a typical security protocol like Diffie-Hellman key exchange or ECDSA signature verification requires less than 2.4 seconds, while ECDSA signature generation can be done in less than 0.9 seconds. This should be tolerated by most users.

Keywords: Elliptic Curves, Koblitz Curves, Binary Fields, Palm OS

1 Introduction

The market for Personal Digital Assistants (PDA) is growing rapidly and PDAs are becoming increasingly interesting for commercial transactions. For eCommerce, provision of security is a must. Since elliptic curve cryptosystems are a promising match for embedded systems because of their short operand lengths and efficient arithmetic, we implemented elliptic curves over binary fields on a Palm OS device to investigate if today's PDAs are sufficient for secure transactions. The most popular PDAs use the Palm Operating System (Palm OS). We used a Handspring Visor model with 2 MB of memory. This device has a Motorola Dragonball CPU that provides eight data registers and seven address registers, all of them 32-bit in size [15]. The processor offers 16-bit and 32-bit operations and runs at 16 MHz. To the author's knowledge there were only two elliptic curve implementations on a Palm Pilot reported yet. In [2] PGP was ported to wireless devices and [3] analyzes electronic commerce applications on a Palm Pilot. However, the first implementation was not optimized for the Palm Pilot while the second one uses a commercial library. Both papers also point out that the popular RSA system is very slow on a Palm Pilot. For most electronic commerce applications the security provided by elliptic curves over $GF(2^{163})$ should be sufficient as long as there are no substantial improvements in solving the elliptic curve discrete logarithm problem (DLP). This bit length is often considered to be security-equivalent to RSA with 1024-bit key length [1]. We chose the NIST recommended random and Koblitz curves over $GF(2^{163})$ and selected binary curves since the integer multiplication unit of the Dragonball processor is very slow. Koblitz curves allow shorter run times while they provide nearly the same level of security according to current knowledge about attacks. Our implementation is mostly based on the algorithms used in a comprehensive software implementation for a PC [4] and Solinas' work about Koblitz curves [19].

2 Arithmetic in $GF(2^m)$

2.1 Field Representation

For our implementation we used a polynomial basis representation. Let $f(x) = x^m + r(x)$ be an irreducible binary polynomial of degree m with small weight, that is a trinomial or pentanomial. The elements of $GF(2^m)$ are represented by the binary polynomials of degree at most $m-1$. Addition and multiplication in $GF(2^m)$ are performed as polynomial operations modulo $f(x)$. An element $a \in GF(2^m)$ is written as the polynomial $a(x) = \sum_{i=0}^{m-1} a_i x^i$ and is stored as binary vector $a = (a_{m-1}, \ldots, a_0)$. We store a in an array A of 16-bit words of size $s = \lceil m/16 \rceil$ and write $A = (A[s-1], \ldots, A[0])$. The rightmost bit of $A[0]$ is a_0 and a_{m-1} is part of $A[s-1]$. The bits left of a_{m-1} are set to zero. For our implementation we used an array of twelve 16-bit words to store an element of $GF(2^{163})$ such that we can also consider A to be an array of $s' = 6$ 32-bit words.

2.2 Addition

Addition over binary fields is performed by a bitwise XOR. Since the Motorola Dragonball CPU performs one 32-bit XOR faster than two 16-bit XORs [15] we add two binary vectors A and B by performing five 32-bit XORs and one 16-bit XOR. We denote this operation by $\oplus$.

2.3 Multiplication

To compute $c = a \cdot b$ we first compute the polynomial $c'(x) = a(x) \cdot b(x)$ and then reduce it to $c(x) \equiv c'(x) \bmod f(x)$.

Polynomial Multiplication Algorithm 1 computes $c' = a \cdot b$ by using a window method [12]. First polynomials $B_u = u \cdot b(x)$ are precomputed for $0 \le u < 2^w$ where w is the window size. In each step of the loop w bits of a are considered. We unrolled the two nested FOR loops completely which resulted in a slight performance gain. By $C\{j\}$ we denote the bit vector $(C[s-1], \ldots, C[j])$. In step

```
Algorithm 1 Comb method with window size $w = 4$
INPUT: Binary polynomials $a(x)$ and $b(x)$ of degree at most $m - 1$.
OUTPUT: The binary polynomial $c'(x) = a(x) \cdot b(x)$.
 1: Compute $B_u(x) = u(x) \cdot b(x)$ for all polynomials $u(x)$ of degree at most 3.
 2: $C' \leftarrow 0$
 3: for $i = 3$ down to 0 do
 4:   for $j = 0$ to $s - 1$ do
 5:     Let $u = (u_3, u_2, u_1, u_0)$, where $u_k$ is bit $(4i + k)$ of $A[j]$.
 6:     $C'\{j\} = C'\{j\} \oplus B_u$
 7:   end for
 8:   if $i \neq 0$ then
 9:     $C' \leftarrow C'x^4$
10:   end if
11: end for
12: Return $c'(x)$
```

6 the m-bit vector B_u is added to C' where the rightmost bit of B_u is added to the rightmost bit of $C'\{j\}$.

We also experimented with the Karatsuba Algorithm [7] as described in Algorithm 2. However, our results were always slower than the above described comb method. We implemented the Karatsuba Algorithm three times recursively and applied the comb method with windows size $w = 3$ to the resulting degree-20 polynomials.

```
Algorithm 2 Karatsuba Algorithm
INPUT: Binary polynomials $a(x)$ and $b(x)$ of degree at most $m - 1$.
OUTPUT: The binary polynomial $c'(x) = a(x) \cdot b(x)$.
1: Write $a(x) = a_1(x)x^{m/2} + a_0(x)$ and $b(x) = b_1(x)x^{m/2} + b_0(x)$
2: $D_0(x) \leftarrow a_0(x)b_0(x)$
3: $D_1(x) \leftarrow a_1(x)b_1(x)$
4: $D_2(x) \leftarrow (a_0(x) \oplus a_1(x))(b_0(x) \oplus b_1(x))$
5: $c'(x) \leftarrow D_1(x)x^m \oplus (D_2(x) \oplus D_0(x) \oplus D_1(x))x^{m/2} \oplus D_0(x)$
6: Return $c'(x)$
```

Polynomial Reduction If $f(x)$ is a trinomial or a pentanomial with middle terms close to each other, reduction of $c'(x)$ modulo $f(x)$ can be efficiently performed one word at a time. Algorithm 3 performs the modulo reduction by $f(x) = x^{163} + x^7 + x^6 + x^3 + 1$. It is based on the fact that

$$\begin{aligned} x^{163} &\equiv x^7 + x^6 + x^3 + 1 \bmod f(x) \\ &\vdots \\ x^{324} &\equiv x^{168} + x^{167} + x^{164} + x^{161} \bmod f(x) \end{aligned}$$

A word $C[i]$ is now reduced by adding $C[i]$ four times to C, with the rightmost bit of C[i] properly aligned as described on the right side of the above congruences. For example, reduction of $C[9]$ is performed by adding $C[9]$ four times to C, with the rightmost bit of $C[9]$ added to the bits 132, 131, 128 and 125 of C. Note that we used 32-bit arithmetic and 32-bit words since XOR and shift operations for 32-bit words have lower runtime than for two 16-bit words. Therefore the array value $C[0]$ describes the bits 0 to 31 of the value c.

Algorithm 3 Modular reduction by $f(x) = x^{163} + x^7 + x^6 + x^3 + 1$

INPUT: A binary polynomial $c(x)$ of degree at most 324.
OUTPUT: $c(x)$ mod $f(x)$.

```
 1: for $i = 10$ down to 6 do
 2:     $T \leftarrow C[i]$
 3:     $C[i-6] \leftarrow C[i-6] \oplus (T << 29)$
 4:     $C[i-5] \leftarrow C[i-5] \oplus (T << 4) \oplus (T << 3) \oplus T \oplus (T >> 3)$
 5:     $C[i-4] \leftarrow C[i-4] \oplus (T >> 28) \oplus (T >> 29)$
 6: end for
 7: $T \leftarrow C[5]$ AND 0xFFFFFFF8
 8: $C[0] \leftarrow C[0] \oplus (T << 4) \oplus (T << 3) \oplus T \oplus (T >> 3)$
 9: $C[1] \leftarrow C[1] \oplus (T >> 28) \oplus (T >> 29)$
10: $C[5] \leftarrow C[5]$ AND 0x00000007
11: Return $(C[5], \dots, C[0])$
```

2.4 Squaring

Squaring in $GF(2^m)$ is a linear operation and much faster than multiplying two arbitrary elements [17]. To square $a(x) = \sum_{i=0}^{m-1} a_i x^i$ we compute $a(x)^2 = \sum_{i=0}^{m-1} a_i x^{2i}$ which is obtained by inserting a 0-bit between consecutive bits of the binary representation of a. The result is reduced modulo $f(x)$. Algorithm 4 describes how this can be done using a precomputed table. As before we used 32-bit words and 32-bit operations.

2.5 Modular Division

Instead of using the Extended Euclidean Algorithm to compute an inversion, we used Algorithm 5 to compute a modular division $\frac{a}{b}$ directly [18]. It has roughly the same running time as the Extended Euclidean Algorithm and therefore saves one multiplication to compute a field division. Note that division by x is accomplished by a right-shift operation. The comparison between two elements $a(x)$ and $b(x)$ is done by considering the bit vectors a and b as integers.

Algorithm 4 Squaring in $GF(2^m)$

```
INPUT: $a \in GF(2^m)$
OUTPUT: $c = a^2 \in GF(2^m)$.
 1: Precompute for each byte $b = (b_7, \ldots, b_0)$ the 16-bit vector $T(b) = (0, b_7, \ldots, 0, b_0)$.
 2: for $i = 0$ to $s' - 1$ do
 3:    Let $A[i] = (A_3[i], A_2[i], A_1[i], A_0[i])$ where $A_j[i]$ are bytes.
 4:    $C'[2i] \leftarrow (T(A_1[i]), T(A_0[i]))$
 5:    $C'[2i+1] \leftarrow (T(A_3[i]), T(A_2[i]))$
 6: end for
 7: $c(x) = c'(x) \bmod f(x)$
 8: Return $c(x)$
```

Algorithm 5 Modular Division in $GF(2^m)$

```
INPUT: $a, b \neq 0 \in GF(2^m)$
OUTPUT: $c = \frac{a}{b} \bmod f(x) \in GF(2^m)$.
 1: $u \leftarrow b, v \leftarrow f(x), c \leftarrow a, d \leftarrow 0$.
 2: while $u \neq v$ do
 3:    if $u \bmod 2 = 0$ then
 4:       $u \leftarrow \frac{u}{x}$
 5:       if $c \bmod 2 = 0$ then
 6:          $c \leftarrow \frac{c}{x}$
 7:       else
 8:          $c \leftarrow \frac{c \oplus f(x)}{x}$
 9:       end if
10:    else if $v \bmod 2 = 0$ then
11:       $v \leftarrow \frac{v}{x}$
12:       if $d \bmod 2 = 0$ then
13:          $d \leftarrow \frac{d}{x}$
14:       else
15:          $d \leftarrow \frac{d \oplus f(x)}{x}$
16:       end if
17:    else if $u > v$ then
18:       $u \leftarrow \frac{u \oplus v}{x}$, $c \leftarrow c \oplus d$
19:       if $c \bmod 2 = 0$ then
20:          $c \leftarrow \frac{c}{x}$
21:       else
22:          $c \leftarrow \frac{c \oplus f(x)}{x}$
23:       end if
24:    else
25:       $v \leftarrow \frac{u \oplus v}{x}$, $d \leftarrow c \oplus d$
26:       if $d \bmod 2 = 0$ then
27:          $d \leftarrow \frac{d}{x}$
28:       else
29:          $d \leftarrow \frac{d \oplus f(x)}{x}$
30:       end if
31:    end if
32: end while
33: Return $c(x)$
```

2.6 Timings

Table 1 displays the timings for one field operation. We spent most time implementing the comb method since the field multiplication is the crucial operation. Reduction and squaring can be implemented efficiently. Division is very expensive and will be avoided where possible.

Table 1. Timings in ms. for one field operation

		time
Multiplication	Comb Method	2.35
	Karatsuba	4.41
Reduction		0.24
Squaring		0.49
Division		38.01

3 Elliptic Curve Basics

3.1 Arithmetic

An elliptic curve over $GF(2^m)$ is defined by the (affine) curve equation

$$E : Y^2 + XY = X^3 + aX + b \tag{1}$$

where $a, b \in GF(2^m)$ and $b \neq 0$. If $a, b \in GF(2)$, i.e., $b = 1$ and $a = 0$ or 1 the curve has special properties that can be used for efficient arithmetic and it is called Koblitz curve. All points $P = (x, y)$ that satisfy (1) and an additional point at infinity $\mathcal{O}$ form a group $E(GF(2^{163}))$. Assume $P_1 = (x_1, y_1) \neq \mathcal{O}, P_2 = (x_2, y_2) \neq \mathcal{O}$ and $P_1 \neq -P_2$. Then $P_3 = (x_3, y_3) = P_1 + P_2$ is computed as follows.

If $P_1 \neq P_2$

$$\begin{aligned}\lambda &= \frac{y_1 + y_2}{x_1 + x_2}\\ x_3 &= \lambda^2 + \lambda + x_1 + x_2 + a\\ y_3 &= (x_1 + x_3)\lambda + x_3 + y_1\end{aligned}$$

If $P_1 = P_2$

$$\begin{aligned}\lambda &= \frac{y_1}{x_1} + x_1\\ x_3 &= \lambda^2 + \lambda + a\\ y_3 &= (x_1 + x_3)\lambda + x_3 + y_1\end{aligned}$$

The group operation requires one division and one multplication in either case. By using projective coordinates as described later the division can be avoided. A scalar or point multiplication is defined as repeated addition via

$$k \cdot P = \underbrace{P + \ldots + P}_{k \text{ times}}$$

There are no efficient attacks known on elliptic curves. The DLP for random curves $E(GF(2^m))$ can be solved on average in $2^{m/2}$ steps, e.g., by using Pollard's Rho method [13]. Therefore a curve over $GF(2^{163})$ is considered appropriate to obtain a secret key for a symmetric cipher with a key length of around 80 bits. An attack on Koblitz curves using the special structure shortens the running time by a factor of $\sqrt{m}$ [20].

3.2 Point Representation

If inversion in $GF(2^m)$ is expensive relative to multiplications it may be more efficient to represent points in projective coordinates. Since a field division is more expensive than 10 multiplications we use projective coordinates as proposed in [10] where the projective point (X, Y, Z) corresponds to the affine point $(X/Z, Y/Z^2)$. The doubling formula $(X_2, Y_2, Z_2) = 2(X_1, Y_1, Z_1)$ for projective coordinates is given by

$$\begin{aligned} Z_2 &= Z_1^2 X_1^2 \\ X_2 &= X_1^4 + bX_1^4 \\ Y_2 &= bZ_1^4 Z_2 + X_2(aZ_2 + Y_1^2 + bZ_1^4) \end{aligned}$$

The projective form of the addition formula is

$$(X_0, Y_0, Z_0) + (X_1, Y_1, Z_1) = (X_2, Y_2, Z_2)$$

For the special case $Z_1 = 1$, i.e., (X_1, Y_1) are affine coordinates this can be computed as follows:

$$\begin{aligned} &A = Y_1 Z_0^2 + Y_0,\ B = X_1 Z_0 + X_0,\ C = Z_0 B \\ &D = B^2(C + aZ_0^2),\ Z_2 = C^2,\ E = AC,\ X_2 = A^2 + D + E \\ &F = X_2 + X_1 Z_2,\ G = X_2 + Y_1 Z_2,\ Y_2 = EF + Z_2 G \end{aligned}$$

In case that $a = 0$ or 1 point doubling requires 4 field multiplications. Mixed Point Addition requires 9 multiplication. We based all point multiplication methods on these projective point doubling and mixed point addition. We only used affine point operations for point precomputations.

4 Random Curves

4.1 Curve Parameters

For our implementation we used a NIST recommended random curve [16] with the parameters

$$a = \mathtt{1}$$
$$b = \mathtt{0x\ 2\ 0A601907\ B8C953CA\ 1481EB10\ 512F7874\ 4A3205FD}$$

and group order

$$\#E(GF(2^{163})) = 2 \cdot 5846006549323611672814742442876390689256843201587$$

NIST also recommends the randomly chosen base point $G = (G_x, G_y)$ where

$$G_x = \mathtt{0x\ 3\ F0EBA162\ 86A2D57E\ A0991168\ D4994637\ E8343E36}$$
$$G_y = \mathtt{0x\ 0\ D51FBC6C\ 71A0094F\ A2CDD545\ B11C5C0C\ 797324F1}$$

4.2 Point Multiplication

There are several methods known to compute $kP \in E(GF(2^m))$ where $k \approx 2^m$. The binary double-and-add method [13] requires m doublings and $m/2$ additions on average. The addition-subtraction method requires only $m/3$ additions on average [14]. It is based on the nonadjacent form (NAF) of the coefficient k. NAF(k) is a unique signed binary expansion with the property that no two consecutive coefficients are nonzero. It has the fewest nonzero coefficients of any signed binary expansion of k, on average $m/3$. Window methods precompute some values and operate on more than one bit of the coefficient k at the same time. Window methods also reduce the number of additions. The sliding window method uses a variable window size. It has an effect equivalent to using fixed windows one bit larger [1]. On average this method requires $m+1$ doublings and $2^{w-1} - 1 + \frac{m}{w+1}$ additions where w is the largest window size. A slight improvement can be gained by using a windowed addition-subtraction method [8]. This is accomplished by using a windowed NAF. A width-w NAF of k is a unique expression $k = \sum_{i=0}^{l-1} k_i 2^i$ where each nonzero k_j is odd and less than 2^{w-1} in absolute value, and among any w consecutive coefficients at most one is nonzero. This method requires $m + 1$ doublings and $2^{w-2} - 1\frac{m}{w+1}$ additions on average. A different approach for point multiplication based on Montgomery's idea was proposed in [11]. It requires $6m$ field multiplications and squarings and does not need any extra memory storage.

If a fixed base point is used we can use precomputed points as done by the fixed base comb method [9]. Using 2^w precomputed points this method requires $d - 1$ doublings and $(d - 1)(2^w - 1)/2^w$ additions where $d = \lceil m/w \rceil$. We implemented this method with a window size of $w = 4$ and $w = 8$. This requires $2^4 = 16$ precomputed points and $16 \cdot 2 \cdot 22 = 704$ bytes, and $2^8 = 256$ precomputed points and $256 \cdot 2 \cdot 22 = 11264$ bytes, respectively. The precomputed points can easily be stored on the Palm device.

4.3 Timings

Table 2 displays the timings for one point multiplication on a Handspring Visor with 2 MB of memory. The implementation was done in C using the Code Warrior IDE. One can see that the differences are relatively small. While the Montgomery method has always the same running time the other methods depend on the coefficient k. The timings were obtained by taking the average time of multiply test runs with random coefficients. When precomputed points can be used the running time is small. A typical key-exchange protocol like Diffie-Hellman or the ECDSA signature verification require one point multiplication by a random point and one point multiplication by a fixed point. This can be done in 3.5 seconds using the Montgomery method and fixed base comb method with $w = 8$. ECDSA signature generation requires a point multiplication by a fixed base point which can be done in 0.8 seconds.

Table 2. Timings in sec. for one point multiplication on random curves

	time
Addition-subtraction	3.31
Sliding windows ($w = 4$)	3.07
Width-w addition-subtraction ($w = 4$)	2.96
Montgomery	2.73
Precomputation (Fixed base comb, $w = 4$)	1.43
Precomputation (Fixed base comb, $w = 8$)	0.79

5 Koblitz Curves

Koblitz curves were first introduced in [6]. All described facts and methods are due to Solinas [19]. The advantage of Koblitz curves is that point multiplication methods can be changed in such a way that point doublings is replaced by the Frobenius map. The Frobenius $\tau : E(GF(p^m)) \to E(GF(p^m))$ is defined as $\tau(x, y) = (x^p, y^p)$. Since $p = 2$ this can be done efficiently using only two field squaring operations. There are two Koblitz curves that use $a = 0$ or $a = 1$. Let $\mu = (-1)^{1-a}$ and τ be the Frobenius map. It is known that $(\tau^2 + 2)P = \mu\tau P$ for all $P \in E(GF(2^m))$. Therefore τ can be expressed as the complex number $\tau = (\mu + \sqrt{-7})/2$. Since $\tau^2 + 2 = \mu\tau$ every integer k can be expressed as $r_1\tau + r_0$ where $r_0, r_1 \in \mathbb{Z}$. The main idea is to replace a coefficient k by a τ-adic number $k' = \sum_{i=0}^{l-1} k'_i\ 2^i$ with $k = k'$ and to compute $k'P$. When computing $k'P = k'_{l-1}\tau^{l-1}(P) + \ldots + k'_0 P$, a point multiplication is reduced to a sequence of point additions without point doublings involved. The τ-adic representation of k has to be computed in such a way that it has short bit length. This is done using modulo reduction in $\mathbb{Z}[\tau]$. Note that this reduction requires multi-precision integer arithmetic.

5.1 Curve Parameters

Again we used a NIST recommended curve [16] with the parameters

$$a = 1, b = 1$$

and group order

$$\#E(GF(2^{163})) = 2 \cdot 5846006549323611672814741753598448348329118574063$$

NIST also provides the base point $G = (G_x, G_y)$ where

$$G_x = \texttt{0x2 FE13C053 7BBC11AC AA07D793 DE4E6D5E 5C94EEE8}$$
$$G_y = \texttt{0x2 89070FB0 5D38FF58 321F2E80 0536D538 CCDAA3D9}$$

5.2 Point Multiplication

Similar to the addition-subtraction method for random curves we implemented a signed binary τ-adic method that uses a reduced τ-adic NAF of k. The τ-adic NAF of k is the unique expression $k = \sum_{i=0}^{l-1} k_i \tau^i$ where $k_i \in \{-1, 0, 1\}$ and no two consecutive coefficients k_i are nonzero. Since the Frobenius map can be computed very efficiently the expected running time is $m/3$ point additions. The cost to compute the NAF of k is much more expensive than for random curves though. The method can be improved by using a window technique. This is called the τ-adic width-w window method. It is based on the τ-adic width-w NAF that is defined very similar as the binary width-w NAF for random curves. This method requires $2^{w-2} - 1 + \frac{m}{w+1}$ additions on average. As before we used mixed projective point addition.

If a fixed base point is used precomputation reduces the time for one point multplication. This is easily achieved by applying the τ-adic width-w window method. Instead of precomputing points for each multiplication the points are only precomputed once such that the window size can be chosen larger.

5.3 Timings

Table 3 displays the values for one point multiplication. Note that the windows τ-adic version with $w = 4$ is faster than with $w = 5$ although the later one has a slightly lower complexity. Also, precomputation using more points is not as efficient as for random curves since the computational overhead to compute the τ-adic representation of the scalar k' increases. The usual time for a key exchange or signature verification is around 2.4 seconds while a signature generation can be done in 0.9 seconds. This is significiantly faster than for random curves.

6 Conclusion

We implemented a NIST recommended random and Koblitz curve over $GF(2^{163})$ on a Palm OS device. A normal transaction such as a key exchange or signature

Table 3. Timings in sec. for one point multiplication on Koblitz curves

	time
τ-adic	1.67
τ-adic width-w ($w = 4$)	1.51
τ-adic width-w ($w = 5$)	1.68
Precomputation ($w = 6$)	1.08
Precomputation ($w = 10$)	0.87

verification can be done in less than 2.4 seconds while signature generation can be done in less than 0.9 seconds. Koblitz curves are particular suitable for these devices since they allow running times that will probably be tolerated by most users.

References

1. I. Blake, G. Seroussi and N. Smart. *Elliptic Curves in Cryptography.* Cambridge University Press, Cambridge, 1999.
2. M. Brown, D. Cheung, D. Hankerson, J. L. Hernandez, M. Kirkup, and A. Menezes. PGP in Constrained Wireless Devices. *Proceedings of the 9th USENIX Security Symposium*, 2000.
3. N. Daswani and D. Boneh. Experimenting with Electronic Commerce on the Palm Pilot. *Financial Cryptography '99*, LNCS 1648, Springer-Verlag, 1-16, 1999.
4. D. Hankerson, J. L. Hernandez and A. Menezes. Software Implementation of Elliptic Curve Cryptography Over Binary Fields. *Cryptographic Hardware and Embedded Systems, CHES 2000*, LNCS 1965, Springer-Verlag, 1-24, 2000.
5. IEEE P1363, *Standard Specifications for Public-Key Cryptography*, 2000.
6. N. Koblitz. CM-curves with good cryptographic properties. *Advances in Cryptology – CRYPTO '91*, LNCS 576, Springer-Verlag, 279-287, 1992.
7. D.E. Knuth. *Seminumerical Algorithms.* Addision-Wesley, 1981.
8. K. Koyama and Y. Tsuruoka. Speeding up elliptic curve cryptosystems by using a signed binary window method. *Advances in Cryptology – Crypto '92*, LNCS 740, Springer-Verlag, 345-357, 1993.
9. C. Lim and P. Lee. More flexible exponentiation with precomputation. *Advances in Cryptology - Crypto '94*, LNCS 839, Springer-Verlag, 95-107, 1994.
10. J. López and R. Dahab. Improved Algorithms for Elliptic Curve Arithmetic in $GF(2^n)$. *Selected Areas in Crytography - SAC '98*, LNCS 1556, Springer-Verlag, 201-212, 1999.
11. J. López and R. Dahab. Fast multiplication on Elliptic Curves over $GF(2^n)$ without Precomputation, *Cryptographic Hardware and Embedded Systems-CHES '99*, LNCS 1717, Springer-Verlag, 316-327, 1999.
12. J. López and R. Dahab. High-Speed Software Multiplication in $\mathbb{F}_{2^m}$, IC Technical Reports, IC-00-09, Institute of Computing, University of Campinas, May 2000, Available from http://www.dcc.unicamp.br/ic-main/publications-e.html.
13. A.J. Menezes, P.C. van Oorschot and S.A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1996.

14. F. Morain and J. Olivos. Speeding up the computations on an elliptic curve using addition-subtraction chains. *Informatique théorique et Applications*, 24, 531-544, 1990.
15. Motorola. M68000 8-/16-/32-Bit Microprocessors User's Manual, Ninth Edition.
16. National Institute of Standards and Technolgy. *Recommended Elliptic Curves for Federal Government Use*, May 1999, available from http://csrc.nist.gov/encryption.
17. R. Schroeppel, H. Orman, S. O'Malley and O. Spatscheck. Fast Key Exchange with Elliptic Curve Systems. *Advances in Cryptology - Crypto '95*, LNCS 963, Springer-Verlag, 43-56, 1995.
18. S. Shantz. From Euclid's GCD to Montgomery Multiplication to the Great Divide, preprint, 2000.
19. J. A. Solinas. Efficient Arithmetic on Koblitz Curves. *Designs, Codes and Cryptography*, 19(2/3), 195-249, 2000.
20. M. Wiener and R. Zuccherato. Faster attacks on elliptic curve cryptosystems. *Selected Areas in Cryptography*, LNCS 1556, Springer-Verlag, 190-200, 1999.

Reducing Certain Elliptic Curve Discrete Logarithms to Logarithms in a Finite Field

Kyungah Shim

KISA (Korea Information Security Agency)
5th FL., Dong-A Tower, Seocho-Dong, Seocho-Gu, Seoul 137-070, Korea
kashim@kisa.or.kr

Abstract. We construct a variant of Weil pairing to reduce the elliptic curve discrete logarithm problem to the discrete logarithm problem in the multiplicative subgroup of a finite field. We propose an explicit reduction algorithm using a new pairing and apply the algorithm to the case of two trace elliptic curves.

Key words : Anomalous curve, supersingular curve, Weil pairing, elliptic curve discrete logarithm.

1 Introduction

The discrete logarithm problem for a general group G can be stated as follows: given $\alpha \in G$ and $\beta \in G$, find an integer x such that $\beta = \alpha^x$, provided that such an integer exists. The integer x is called the *discrete logarithm* of β to the base α. If we replace the group G by the elliptic curve group over a finite field then it is the elliptic curve discrete logarithm problem (ECDLP).

In [4] and [7], Koblitz and Miller independently propose how to use the group of points on an elliptic curve over a finite field to construct public key cryptosystems. The security of these cryptosystems is based upon the presumed intractability of computing logarithms in the elliptic curve group. The best algorithms known for solving this problem are the exponential square root attacks that can be applied to any finite group and have a running time that is proportional to the square root of the largest prime factor dividing the order of the group. In [7], Miller argues that the index-calculus methods, which produced dramatic results in the computation of discrete logarithms in the multiplicative subgroup of a finite field, do not extend to elliptic curve groups. Consequently, if the elliptic curve is chosen so that its order is divisible by a large prime, then even the best attacks take exponential time.

The integrity of ECDLP cryptographic tools would be widely accepted, however, there exist two exceptional families of elliptic curves (i.e., supersingular and anomalous curves) and for each case powerful cryptanalysis method has been invented. But both classes of elliptic curves may be easily avoided in practice.

At first Menezes, Okamoto and Vanstone [6] propose a subexponential time algorithm to solve the ECDLP over a supersingular elliptic curve E defined over a finite field F_q ($q = p^n,\ p > 3$), the so-called MOV algorithm. It employed the Weil pairing to reduce ECDLP to the discrete logarithm problem in a multiplicative subgroup of an extension field F_{q^k} of F_q, $k \leq 6$. By using a variant of the Tate pairing, Frey and Rück [2] gave a generalization of this the discrete logarithm over the divisor class group of curves, we call this algorithm the FR algorithm. Furthermore, Balasubramanian and Koblitz [1] showed that if we choose an elliptic curve at random over a prime finite field F_p whose number of F_p-rational points is prime, then the MOV algorithm on that curve is not effective with overwhelming probability.

Recently, Semaev [9], Smart [11], and Satoh and Araki [8] independently proposed a polynomial time algorithm (SSSA algorithm) for the ECDLP over an anomalous elliptic curve defined over a prime field F_p, i.e., an elliptic curve over F_p whose number of F_p-points is p. It is easy to see that we can also apply the SSSA algorithm to the discrete logarithm problem over the p-part of $E(F_q)$, where q is a power of p. Semaev employs an algebraic geometrical approach, while Smart and Satoh-Araki employ a number theoretical approach to reduce the ECDLP over E to the additive group F_p.

In this paper, we present the following results;

1. We construct a variant of Weil pairing to reduce the ECDLP defined over F_q to the discrete logarithm problem in F_q^*.
2. We propose an efficient reduction algorithm for ECDLP over elliptic curves with trace two, more generally, elliptic curves with even trace under a special condition.

2 Construction of Bilinear Pairing

We want to construct a certain variant of Weil pairing with simple computation. Let E be an elliptic curve defined over a finite field F_q where $q = p^n$ for some prime $p \neq 2, 3$. Let $E(F_q)$ be the group of rational points of E over F_q. Suppose that $E(F_q)$ contains a 2-torsion point and let a be a 2-torsion point in $E(F_q)$. If a divisor $D_1 = div(f)$ is principal, for any $D_2 = \sum_{i=1}^{r} n_i(a_i) \in \mathrm{Div}^0(E)_{F_q}$ such that $\mathrm{supp}(D_1) \cap \mathrm{supp}(D_2) = \emptyset$, we let $f(D_2) \equiv \prod_{i=1}^{r} f(a_i)^{n_i}$ where $\mathrm{Div}^0(E)_{F_q}$ is the group of divisors of degree zero whose components are F_q-rational. This value depends only on f since the constant disappears when taking the product over the points of a divisor of degree zero. We can define a bilinear pairing from $E(F_q) \times E(F_q)$ to F_q^* in the following way,

$$< \cdot, \cdot >_a : E(F_q) \times E(F_q) \rightarrow F_q^*, \; < P, Q >_a \equiv f_{\bar{P}}(\bar{Q}) / f_{\bar{Q}}(\bar{P}). \qquad (1)$$

where $\bar{P}$ is a divisor $(P) - (O)$ corresponding to a point $P \in E(F_q)$ via an isomorphism from E to the divisor class group of E, $\mathrm{Pic}^0(E)$. We can deduce that $\bar{P}_a - \bar{P} = (P + a) - (a) - (P) + (O)$ is a principal divisor, by Cor 3.5 (pp. 67) of [10], namely, there exists a rational function $f_{\bar{P}}$ such that

$$div(f_{\bar{P}}) = (P + a) - (a) - (P) + (O).$$

Similarly, there exists a rational function such that

$$div(f_{\bar{Q}}) = (Q+a) - (a) - (Q) + (O).$$

These rational functions are uniquely determined up to constants. In fact, this pairing is similar to the Weil pairing on the group of m-torsion point of elliptic curve E.

Theorem 1. *For the pairing $< \cdot, \cdot >_a$ given in (1), we have the following properties.*

1. *$< \cdot, \cdot >_a$ depends only on the divisor class.*
2. *It is a bilinear pairing.*
3. *It is alternative, i.e.,$< P, Q >_a = < Q, P >_a^{-1}$.*

Proof. (1) Let $\bar{P}'$ be linearly equivalent to $\bar{P}$. Then we can express as $\bar{P}' = \bar{P} + (g)$ for some rational function g. Thus we get

$$(f_{\bar{P}'}) = \bar{P}'_a - \bar{P}' = \bar{P}_a - \bar{P} + (g)_a - (g) = (f_{\bar{P}}) + (g)_a - (g).$$

The value of $< P, Q >_a$ for $\bar{P}'$ is

$$\frac{f_{\bar{P}'}(\bar{Q})}{f_{\bar{Q}}(\bar{P}')} = \frac{f_{\bar{P}}(\bar{Q}) g(Q-a) g(-a)^{-1} g(0) g(Q)^{-1}}{f_{\bar{Q}}(\bar{P}) f_{\bar{Q}}((g))}.$$

Since a is a 2-torsion point, i.e., $a = -a$,

$$\frac{g(Q-a)g(O)}{g(Q)g(-a)} = \frac{g(Q+a)g(O)}{g(Q)g(a)} = g((f_{\bar{Q}})),$$

and by Weil reciprocity law in Lang [5], pp.172, we have $f_{\bar{Q}}((g)) = g((f_{\bar{Q}}))$. Hence we conclude that

$$\frac{f_{\bar{P}'}(\bar{Q})}{f_{\bar{Q}}(\bar{P}')} = \frac{f_{\bar{P}}(\bar{Q})}{f_{\bar{Q}}(\bar{P})},$$

namely, it is independent of the choice of a divisor in the same divisor class. Similarly, it is well-defined with respect to the second variable. If $\text{supp}((f_{\bar{P}})) \cap \text{supp}(\bar{Q}) \neq \emptyset$, then we can find a divisor $\bar{Q}' = \bar{Q} + (g)$ such that $\text{supp}((f_{\bar{P}})) \cap \text{supp}(\bar{Q}') = \emptyset$. Thus we can avoid the points at which the rational function are not defined.

(2) We show $< P_1 + P_2, Q >_a = < P_1, Q >_a < P_2, Q >_a$. Since $\overline{P_1 + P_2}$ is linearly equivalent to $\bar{P}_1 + \bar{P}_2$, we get

$$(P_1 + P_2) - (0) \sim (P_1) - (0) + (P_2) - (0)$$

by the square theorem [5]. Thus we have

$$\frac{f_{\overline{P_1+P_2}}(\bar{Q})}{f_{\bar{Q}}(\overline{P_1+P_2})} = \frac{f_{\bar{P}_1+\bar{P}_2}(\bar{Q})}{f_{\bar{Q}}(\bar{P}_1+\bar{P}_2)} = \frac{f_{\bar{P}_1}(\bar{Q}) f_{\bar{P}_2}(\bar{Q})}{f_{\bar{Q}}(\bar{P}_1) f_{\bar{Q}}(\bar{P}_2)} = < P_1, Q >_a < P_2, Q >_a .$$

It is also linear with respect to the second variable by the similar way.

(3) It is obvious by definition.

We can easily find our rational functions in computation of the pairing of (1) using the following algorithm.

Algorithm 1

[**Description**] Algorithm for finding a rational function over E with a given divisor.
[**Input**] A divisor of the form $(P+Q)-(P)-(Q)+(0)$ where $P, Q \in E(F_q)$.
[**Output**] A rational function g such that $(P+Q)-(P)-(Q)+(O)=(g)$.

1. Find a line equation $L : f(x,y,z) = ax+by+cz = 0$ in P^2 through P and Q where $a,b,c \in F_q$.
2. Compute R the point of intersection of L with E.
3. Find a line equation $L' : f'(x,y,z) = a'x+b'y+c'z = 0$ in P^2 through R and O where $a',b',c' \in F_q$.
4. Output $g = f'/f$.

How the algorithm yields an easy way to compute the rational function g; for a given divisor $\bar{P} = (P)-(O)$, let $f(x,y,z) = ax+by+cz = 0$ be the line L in P^2 through P and Q. Also let R be the point of intersection of L with E and $f'(x,y,z) = a'x+b'y+c'z = 0$ the line L' through R and O. Then, from the definition of addition on E and the fact that the line $z = 0$ intersects E at O with multiplicity 3, we have

$$div(f/z) = (P)+(Q)+(R)-3(O)$$

and

$$div(f'/z) = (R)+(P+Q)-2(O).$$

Hence

$$(P+Q)-(P)-(Q)+(O) = div(f'/f).$$

The rational function f'/f is the function for which we are looking.

Remark 2.2

1. In fact, for general elliptic curves the image of this pairing is very tiny. Let N be the order of the elliptic curve E then we have

$$1 = < NP, NQ >_a = < P, Q >_a^{N^2} .$$

Hence $< P,Q >_a$ has a order dividing $\gcd(N^2, q-1)$. Thus this technique is meaningful when $N = q-1$ or $\gcd(N, q-1)$ is of size almost that of q, as will be seen in the next section.
2. The similar procedure can be used to compute $< \cdot,\cdot >_a$ in the case J is the Jacobian variety of a hyperelliptic curve.

3 The Reduction

In fact, the condition of the extension degree for the FR algorithm is usually weaker than that for the MOV algorithm, theorem 4.2 in [3] shows that the condition $q^k \equiv 1 \pmod{l}$ is equivalent to the condition $E[\ell] \subset E(F_{q^k})$ if $\ell \not| q-1 \pmod{\ell}$, i.e., the effectiveness of the MOV algorithm is the same as that of the FR algorithm if $q \equiv 1 \pmod{\ell}$. The extension degree k is exponential in $\log q$ when $\ell \, | q-1$. We consider the case of $\ell \, | q-1$.

From a standard cryptographic view point, let $|E(F_q)| = 2 \cdot \ell$, we may assume that ℓ is around q, then it is easy to see $|E(F_q)| = q-1$ when $\ell \, | q-1$. Suppose that ℓ is a prime number. Let $P \in E(F_q)$ be an element of order ℓ and $R \in < P >$. Let $a \neq O$ be a 2-torsion point in $E(F_q)$. With the pairing described in section 2, we can obtain the following theorem.

Theorem 2. *There exists some point $Q \in E(F_q)$ such that the map $\phi_{Q,a}; < P > \longrightarrow G$ defined by $\phi_{Q,a}(R) = < Q, R >_a$ is a group isomorphism where G is a unique cyclic subgroup of F_q^*.*

Proof. There exists a unique 2-torsion point $a = -a$ since ℓ is prime. This we need not worry about the choice of the 2-torsion point. If we take Q be a non 2-torsion point in $E(F_q)$ then we have an one to one homomorphism $\phi_{Q,a} : < P > \longrightarrow F_q^*$ since $< P >$ has a prime order.

We can introduce the following algorithm to reduce the ECDLP when $|E(F_q)| = q-1 = 2 \cdot \ell$ where ℓ is a prime number, to the discrete logarithm problem in the multiplicative subgroup of a underlying finite field.

Algorithm 2

[**Description**] Reduction the discrete logarithm on $E(F_q)$ to the discrete logarithm in F_q^*
[**Input**] An element $P \in E(F_q)$ of order ℓ, $R \in < P >$.
[**Output**] An integer m such that $R = mP$.

1. Find $Q \in E(F_q)$ such that $\alpha = \phi_{Q,a}(P)$ has order ℓ.
2. Compute $\beta = \phi_{Q,a}(R)$.
3. Compute m, the discrete logarithm of β to the base α in F_q^*.

Note that the output of Algorithm 2 is correct since

$$\beta = \phi_{Q,a}(mP) = < Q, mP >_a = < Q, P >_a^m = \alpha^m.$$

Thus, in this case, the reduction step of Algorithm 2 takes polynomial time resulting in a probabilistic subexponential time algorithm for computing elliptic curve discrete logarithms in these curves. Thus, to select a secure elliptic curve, we must avoid elliptic curves of trace 2.

Remark 3.1

1. In our cases, which are important ones for cryptographic reasons, unlike Algorithm 2 in MOV reduction, which can choose a point Q probabilistically, we can determine a point Q easily, because every non 2-torsion point of $E(F_q)$ has a prime order ℓ. Consequently, (1) in Algorithm 2 is independent to the choice of a point Q such that $2Q \neq O$.
2. We can compare our pairing with Weil pairing and Tate-Lichtenbaum pairing. The fast algorithm used to compute the Weil pairing following V. Miller consists in a twofold computation of the Tate-Lichtenbaum pairing. But the Tate-Lichtenbaum pairing is computable in $O(\log q)$ steps, where one step is equivalent to the addition in $E(F_q)$. But the computation of our pairing is much simpler than that of Tate-Lichtenbaum pairing because the computation takes a constant number of multiplications in F_q^*.
3. In [3], Kanayama *et al.*, also proposed a reduction algorithm (KKSU algorithm) for the ECDLP over trace two elliptic curves. Their algorithm differ from the FR algorithm is faster than the FR algorithm. They confirmed that the reduction part of the proposed algorithm was 1.5 times faster than that of the FR algorithm. However the reduction part of our algorithm is faster than the KKSU algorithm since the computation of the pairing used to reduction consists of only a constant number of multiplications in F_q^*.
4. We know that, to resist the MOV attack, one only needs to check that n, the order of point P, does not divide $q^k - 1$ for all small k foe which the DLP in F_{q^k} is tractable- in practice, when $n > 2^{160}$ then $1 \leq k \leq 20$ suffices. More generally, the divisible check rules out all elliptic curves for which the ECDLP can be efficiently reduced to the DLP in some small extension of F_q. These include the elliptic curves of trace 2 as well as supersingular elliptic curves.

We conclude that the we can reduce the discrete logarithm problem on trace two elliptic curves defined over F_q to the discrete logarithm problem on the multiplicative subgroup of a underlying finite field F_q.

References

1. R. Balasubramanian and N. Koblitz, Improbability that an elliptic curve has subexponential discrete logarithm problem under the Menezes-Okamoto-Vanstone algorithm, Journal of cryptology, vol. 2, no. 11 (1998), pp. 141-145.
2. G. Frey and H. G. Rück, A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves, Mathematics of Computation, vol. 62 (1994), pp. 865-874.
3. N. Kanayama, T. Kobayashi, T. Satoh and S. Uchiyama, Remarks on elliptic curve discrete logarithm problem, IEICE Transaction Fundamentals, vol. E83-A, no. 1 (2000), pp. 17-23.
4. N. Koblitz, Elliptic curve cryptosystems, Mathematics of Computation, vol. 48 (1987), pp. 203-209.

5. S. Lang, Abelian varieties, Springer Verlag, New York, 1983.
6. A. J. Menezes, T. Okamoto and S. A. Vanstone, Reducing elliptic curve logarithms in a finite field, IEEE Transaction on Information Theory, vol. 39, no. 5 (1993), pp. 1639-1646.
7. V. Miller, Use of elliptic curves in cryptography, Advances in cryptology; Proceedings of Crypto'85, LNCS, 218 (1986), Springer-Verlag.
8. T. Satoh, K. Araki, Fermat quotients and the polynomial time discrete logarithm algorithm for anomalous elliptic curves, Proc. of algebraic number theory and its related topics, Koukyuuroku vol. 1026 (1998), pp. 139-150.
9. J. A. Semaev, Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curves in characteristic p, Mathematics of Computation, vol. 67 (1998), pp. 353-356.
10. J. H. Silverman, The arithmetic of elliptic curves, Springer-Verlag, 1986.
11. N. P. Smart, The discrete logarithm problem on elliptic curves of trace one, Journal of Cryptology, vol. 12 (1999), pp.193-196.

Author Index