Electronic Signature System with Small Number of Private Keys

Ahto Buldas
ahtbu@cyber.ee
Tallinn Technical University,
Tartu University, and Cybernetica AS

Märt Saarepera
marts@neoteny.com
Independent

Abstract

We propose a simple server-based electronic signature system in which a small number of common private keys are used. The motivation of such a system is to escape the scalability and complexity problems that arise if a large-scale Public Key Infrastructure (PKI) is used. We argue that the assumption of personal private keys is the main reason for those problems and high cost of electronic signature systems. We conclude that elimination of personal private keys is justified and further argue that this does not reduce the security.

1 Introduction

Remembering and proving events of the past is an important characteristic of the human civilization. Oral testimonies were used before the literary language was invented, and they are still in use today. Due to increasing complexity of business and public relations, people started to use written documents as external memory. Numerous measures have been developed to protect the content integrity of written documents. For example special inks, paper, seals etc. are used. Further, in order to bind the content of a document with a person responsible for it, handwritten signature is used.

Today, written documents as well as all kinds of data in general are processed, transmitted, and preserved in digitized electronic form (mostly referred to as *electronic content*). Currently, cryptographic means are used to protect the integrity of electronic content and to bind content with persons responsible for it. Cryptographic checksums computed by using asymmetric cryptography are often viewed as electronic analogues of handwritten signatures. Signatures are created by using *private keys* and verified by using *public keys*.

Cryptographers have been studying electronic signature technologies for decades since the discovery of one-way functions [4]. Several electronic signature schemes are (mathematically) proved to be secure under some complexity theoretical assumptions (see [12] for an overview).

In numerous countries, including the USA, electronic signatures are legally admissible. Considering the advantages of electronic data management (creation, transfer, storage) over the traditional paper-based one we could expect that there is an obvious need for electronic signatures in the society. However, electronic signatures are still not widely used. In our opinion, it is for two reasons:

(i) *Security concerns*. Leakage of private keys may cause unlimited risk, because of the number of (possibly forged) signatures cannot be limited.

Secure key management is too complicated for general public.

(ii) *Technical complexity and Cost.* Private key management as well as massive authentic distribution of public keys is costly.

Rapid growth of a technology where the main concerns are related to security and cost – *Internet banking* – suggests that these concerns can be solved for electronic signatures as well. In the Internet banking, (a) the risks are always limited (at least to the amount of money in the user's account), and (b) existing infrastructure (like web browsers) provides simple and user-friendly interface to customers. Why not to design an electronic signature system in a similar way?

In most electronic signature systems it is assumed that private keys are distributed among the users. At the same time, there are electronic signature systems that use a very small number of keys. For example, in the system presented by Asokan et al [2] personal private keys are eliminated and the signature function is delegated to a server. The server authenticates clients and creates electronic signatures in their name by using one single private key. It seems to be a common opinion that electronic signature systems where the private keys are distributed among users are of the highest possible security. We do not think this opinion is sufficiently argued. Moreover, we claim that personal private keys are the main reason for high cost and technical complexity of the systems. Eliminating personal keys may lead to considerably more cost-efficient electronic signature systems, which in addition may be more secure than the previous systems.

In this paper, we show (using an elementary risk analysis methodology) that server-based electronic signature systems may be as secure as those with personal private keys. Moreover, in server-based systems it is easy to restrict the number of signatures, which is a necessary feature to limit risks. In server-based systems, possible abuses are more easy to inspect – the server can log all events, while personal private keys can be abused off-line in an unrestricted way. Hence, even if each (signed) transaction has limited value, an attacker (who abuses a personal key) can still create large numbers of low-value transactions.

We propose a new scalable electronic signature system that uses even a smaller number of keys than the system of Asokan et al [2]. In our system, the signature servers themselves use meta-level signature services to create their signatures so that only few public (and private) keys are needed for the whole service. Due to the small number of keys, it is easy to preserve the validity of signatures in long-term scale.

The paper is organized as follows. In Section 2, we give a general description of electronic signature systems and present two special cases: PKI-based electronic signature, and server-based electronic signature. We point out their relative advantages and drawbacks. In Section 3, we analyze the practical security of electronic signatures, considering both signer's and verifier's view points. In Section 4, we describe techniques to improve the scalability and security of electronic signature systems. In Section 5, we outline a technically simple and efficient server-based solution to electronic signatures.

2 Electronic Signature Systems

In the most general setting, an *electronic signature* [1] is authentic and reliable information that answers the question "Who signed What and When?". In order to use electronic signatures, one has to organize a system that satisfies the following security requirements from both *signer's* and *verifier's* view-point:

[1] We intentionally use the term *electronic signature* instead of *digital signature*, because the latter is mostly used in association with signature schemes that use asymmetric cryptography.

(i) Signers are able to sign messages only in their own name.

(ii) Potential verifier can check the validity of a signature. The verifier is provided with methods that ensure that valid signatures cannot be denied or invalidated later.

In the following, we describe two (totally different) electronic signature systems. The first system has a maximum number of keys – every user has her own private (signature) key. The second system has a minimum number of private keys – there is a single private key that is maintained by a server, which identifies users and creates signatures in their names. We show what are the security and cost concerns in these systems for a signer and for a verifier. The analogies for these two systems in the paper-world are *personal handwritten signature* and *notarized or delegated signature*. We analyze the history of these signature systems and the security concerns in both systems. Massive use of personal handwritten signatures became possible only when literacy became a common skill. Notarized signatures were used way before. We claim that considering the "electronic literacy" of general public, the society is not yet ready to use "personal" electronic signatures. Moreover, we are not able to imagine how "electronic literacy" will become a common skill in the near future.

2.1 PKI-Based Signatures

Each user A has a private key sk_A, which is assumed to be under a sole control of A, and a public key pk_A, an authentic copy of which is assumed to be available to all potential verifiers. To sign a message M, A applies a signature function $\textsc{Sig}$ to a pair (sk_A, M) of the private key and the message. To verify a digital signature $s = \textsc{Sig}(\mathsf{sk}_A, M)$ one has to apply a verification function $\textsc{Ver}$ to a triple (pk_A, s, M), which returns Yes if the signature is correct.

The mechanism for authentic distribution of public keys depends on particular systems. The users may themselves distribute their keys, which is suitable if each user has a small number of communication partners. An example of such system is PGP [14]. In order to simplify the distribution of authentic public keys, a trusted party – Certification Authority CA – is introduced. As any other user, also the CA has its private key $\mathsf{sk}_{\mathsf{CA}}$ and its public key $\mathsf{pk}_{\mathsf{CA}}$. To bind the identity ID_A of A and the public key, the CA issues *public-key certificate* $c = \textsc{Sig}(\mathsf{sk}_{\mathsf{CA}}, (\mathsf{ID}_A, \mathsf{pk}_A))$. A complete signature of A on M consists of two parts: the signature s and the certificate c. To verify such a signature, one needs to have an authentic copy of $\mathsf{pk}_{\mathsf{CA}}$.

For several reasons, we also have to add time to an electronic signature. In order to prove the time when the signature was created, another trusted party – Time Stamping Authority (TSA) – is introduced [17]. By a time stamp for a signature s we mean a signed statement $ts = \textsc{Sig}(\mathsf{sk}_{\mathsf{TSA}}, (s, t))$, where t is a time value. Hence, the signature is a triple

$$\underbrace{\textsc{Sig}(\mathsf{sk}_A, M)}_{s},\ \underbrace{\textsc{Sig}(\mathsf{sk}_{\mathsf{CA}}, (\mathsf{ID}_A, \mathsf{pk}_A))}_{c},\ \underbrace{\textsc{Sig}(\mathsf{sk}_{\mathsf{TSA}}, (s, t))}_{ts}, \quad (1)$$

for the verification of which we need authentic copies of two public keys $\mathsf{pk}_{\mathsf{CA}}$ and $\mathsf{pk}_{\mathsf{TSA}}$. Note that the scheme presented above is considerably simplified compared to its real implementations. However, the simplified description is completely sufficient for the goals of this paper.

The installation procedures of private keys, their protection mechanisms, authentic distribution of public keys and their status checking mechanisms make large-scale PKI systems very costly [5]. The main threat for A is that someone abuses her private key. The main threat for a verifier is that the signature (1) becomes invalid, which may happen due to exposure of the keys $\mathsf{sk}_{\mathsf{CA}}, \mathsf{sk}_{\mathsf{TSA}}$ or due to the cryptographic algorithm $\textsc{Sig}$ becoming insecure. Note that in practice, the signatures can potentially be denied by alleged signers, which is also a threat for the verifier. However, considering the highly non-technical nature of this threat, we do not discuss it here. For example, there may be several different ways of solving “fantom withdrawal” cases between banks and their clients, depending on the contracts (between banks and clients) and the legal environment in which the contracts have been made. We only consider the threats that cause the signature (1) becoming *technically incorrect.*

2.2 Server-Based Signatures

We have a single private/public key pair $\mathsf{sk}_S/\mathsf{pk}_S$ in the system that is maintained by a server S. Every user has means to authenticate herself to the server, in order to create electronic signatures. The exact way how the authentication is performed is not important. The server maintains a database of signature events described as triples (ID_A, M, t). Each such triple means a statement ”ID_A signed M at time t”. In order to sign a message M, a user A sends S a request which comprises M (or its cryptographic digest). After verifying the identity of A (e.g. via password), S creates and stores a triple (ID_A, M, t), where t is the current time. The verification of a signature is either *server-aided* or *off-line*.

In the case of *server-aided verification*, (a) a verifier B sends M to S, (b) S makes a query to its database and finds all triples of the form $(*, M, *)$ and sends all of them to B. From the technical side, such a scheme is extremely simple and does not require digital signature schemes at all. Though, it has been proved by Halevi and Krawczyk [8] that in password-based authentication protocols (under certain security assumptions) asymmetric cryptography is still necessary.

In the case of *off-line verification*, the server (instead of storing triples in its database) signs a triple (ID_A, M, t) by using its private signature key sk_S and communicates the signature

$$\textsc{Sig}(\mathsf{sk}_S, (\mathsf{ID}_A, M, t)), \qquad (2)$$

back to A. It is not hard to notice that a server-based signature (2) is much simpler than a PKI-based signature (1). Both the installation costs at the user side and the public key distribution costs are lower. The main threat for A is that someone impersonates her during the identity check procedure, which may be possible due to a leakage of passwords etc. The main difference from PKI-based signatures is that the service provider S itself is able to create signatures without users’ intent. Hence, S must be absolutely trustworthy. In the next subsection, we argue that trust assumptions in these two systems are only seemingly different.

2.3 Personal and Delegated Signatures: Historical Metaphor

In the case of a hand-written signature, the main skills needed from a person are: (a) knowledge of written language because the signer has to know what she is signing for; and (b) understanding and controlling the functionality of a pen. The signer has to be convinced that the pen cannot sign anything by itself, without user’s intent. The ”pens” for electronic signatures are much more complicated. The users who really like to have control over their private keys must be well educated in electronics, hardware design, operating system design, the software etc. Even if the signer has all the knowledge necessary to understand electronic signatures, she still does not know whether the signature device really behaves as specified. Trapdoors in software and even in hardware are not just science fiction but are rather common practice. Today, the assumption that people may have sole control over their private signature keys is thereby just an illusion, and probably will stay an illusion in the near future. No single person (and most of the institutions and companies) is able to control her signature device. At present, the methods and devices to reliably control private keys are affordable to very few institutions in the world. When using electronic signatures, most of us have to trust technology and hence also the providers of technology. In this sense, we are in the role of ”illiterate” people.

But also illiterate persons can sign documents: they just write X-s at the bottom of the document in the presence of a *trusted notary* who confirms that the X-s are written intentionally. In the past when overall literacy was not yet established, numerous contracts were signed that way. In some sense, any present-day electronic signature is just a confirmation created by the providers of the technology, who are in the role of ”notaries”. We cannot eliminate trust by adding technological security measures (like providing users with personal private keys) to the system.

Trust assumptions are only one aspect that affects security. In order to show that server-based signature systems are practically not less secure than the PKI based system, we have to use more precise definitions of practical security. In the next section, we present a practical security analysis that uses a commonly accepted method of practical security evaluation – *risk analysis*.

3 Practical Security of Signature Systems

Theoretical cryptography focuses on preventing particular threats. In practical security, the primary goal is rather to reduce risk. It is possible that preventing a threat does not reduce the overall risk. In this section, we first present the basic principles of risk analysis that are used in later analysis of electronic signature systems. Then, we analyze and compare the security of PKI-based electronic signature systems and server-based systems. We use the so called *attack tree* method[15] that has been successfully used in several practical security-critical systems.

3.1 Threats, Risks, Attacks

Risk is commonly defined as mathematical expectation of loss. This definition is, however, somewhat inconvenient to use when the threats are related to attacks. The reason is that it is often impossible to estimate directly the probabilities of attacks. But attacks are the most important threats if we estimate the security of electronic signature systems.

One of the most methodical approaches to attack analysis is the *attack tree* method [15, 18]. An attack tree is a graph that represents the decision-making process of a well-informed attacker. The roots of the tree represent the main threats, which are the main goals of attackers. Each node represents an attack. The graph has two types of nodes: AND nodes and OR nodes. The child nodes of an OR node represent a list of conditions (sub-attacks) each of which is sufficient for the attack being successful. The child nodes of AND node represent a list of conditions (sub-attacks) each of which is necessary for the attack being successful. The leaves of the tree represent ”atomic” attacks the costs (and other characteristics) of which are known.

3.2 Security of Signer

In order to compare the security of PKI-based and server-based electronic signature systems, we use a generic model that simultaneously describes both systems. The model consists of the following parts:

(1) *Client workstation*, which is the signer's interface to the system,

(2) *Technology providers* that produce or sell all kinds of technology used in electronic signature systems,

(3) *Signature server* that participates in the signature creation process (not present in PKI-based signature systems), and

(4) *Signature service* that runs the signature server (not present in PKI case).

In a PKI-based system, Client workstation computes client's signature by using the private key of the Client. The key may be stored in the memory of a workstation or in an IC-card. In a server-based system, Client workstation is connected securely to a Signature server (via a secure SSL connection etc.). After succesfully authenticating the signer (by using passwords etc.) the Server creates an electronic signature in signer's name.

We assume that attackers' main goal (root of the attack tree) is to forge a signature. We consider four general sub-attacks, each of which is sufficient for the goal of the attacker:

(a) *Attack client workstation* - steal the key/password, insert a Trojan horse, etc.

(b) *Bribe an employee of a technology provider* - bribed employees may add vulnerabilities to the system. Yung and Young [19] proved that trapdoors can easily be inserted even into cryptographic algorithms.

(c) *Bribe an employee of the signature service provider* - bribed employees may add vulnerabilities to the system or create forged signatures.

(d) *Attack signature server in a technical way* - try to attack the server and abuse the signature key.

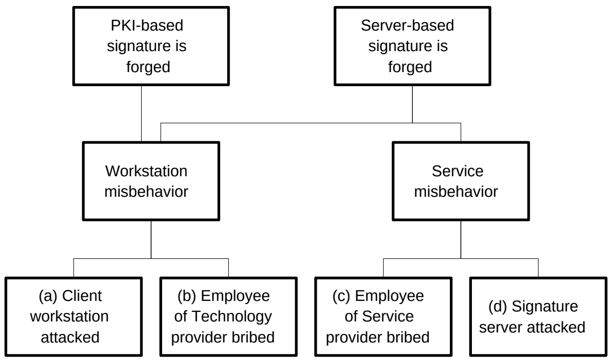
PKI-based signature is forged
Server-based signature is forged
Workstation misbehavior
Service misbehavior
(a) Client workstation attacked
(b) Employee of Technology provider bribed
(c) Employee of Service provider bribed
(d) Signature server attacked

Figure 1: Merged attack trees for PKI- and server-based signatures.

What we claim is that the components (3) and (4) of an electronic signature system do not add additional risks to the system and hence server-based systems are at least as secure as PKI-based systems. This claim is a consequence of the following assumptions:

A0: *A well-informed attacker always chooses the easiest (least costly) attack.*

A1: *It is easier to attack a client workstations than to attack signature servers*: $\mathsf{Cost}[a] < \mathsf{Cost}[d]$, because service providers are commonly more experienced to protect their computers than general public. The above is true for the attacks by outsiders. It may actually be easier for an inside attacker to attack the server. However, once we assume that trusted services may be compromised by insiders, we should also agree that personal keys would not help much. For example, if Microsoft on-line client service is compromised, it would be able to suitably "update" client software in almost any networked client workstation and thereby also to get access to personal keys. Most of the average-skill users do not protect their computers enough to prevent web services from running malicious code in their computers.

A2: *The costs of bribing employees of Technology providers and of the Signature service are comparable*: $\mathsf{Cost}[b] \approx \mathsf{Cost}[c]$. At first glance, this assumption may be doubtful – to bribe scientists and technology experts seems much harder than to bribe a "minimum wage guy" who guards the server room. However, the term *technology provider* in this paper has a wider meaning than in common language. For example, also the shops that sell computers are viewed as technology providers, because they have an influence on the behavior of the computers they sell.

A3: *By using security measures of moderate cost (firewall, etc.) it is possible to make technical attacks to the signature server more costly than bribing an employee of the service provider*: $\mathsf{Cost}[c] < \mathsf{Cost}[d]$.

These assumptions imply that in a PKI-based system (where only (a) and (b) are meaningful attacks) as well as in the server-based system (where all attacks (a),(b),(c),(d) must be considered) either (a) or (b) has the lowest possible cost and hence the attacker always chooses one of them. Thereby, if we have reasonable cryptographic measures used in the signature server, and reasonable organizational means used by the signature service provider, then the attacks (c) and (d) simply do not increase the overall risk of the system.

3.3 Security of Verifier

The most important threat for the verifier is that an accepted valid signature becomes invalid. We only consider the case of off-line verification in both types

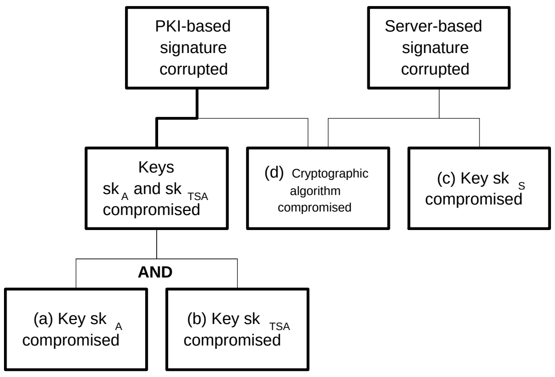
PKI-based signature corrupted
Server-based signature corrupted
Keys sk_A and sk_TSA compromised
(d) Cryptographic algorithm compromised
(c) Key sk_S compromised
AND
(a) Key sk_A compromised
(b) Key sk_TSA compromised

Figure 2: Merged attack trees for signature verification.

of systems. We do not consider the attacks that target verifier's workstation because these attacks are equivalent in both types of signature systems. The most important (threats) attacks to consider are the following:

(a) Private key sk_A of the signer becomes compromised: either because of attacker or intentionally by the signer (in order to escape from liability).

(b) Private key $\mathsf{sk}_{\mathsf{TSA}}$ of a time-stamping service becomes compromised.

(c) Private key sk_S of a signature server becomes compromised.

(d) Cryptographic algorithm becomes compromised.

We assume that the Signer and the CA have a mutual written contract, which states that the Signer possesses (and agrees to use) a particular key. This contract can be used as evidence in later disputes, if the Signer tries to deny having been related to the key. Hence, the compromise of CA key alone does not affect the validity of PKI-based electronic signatures, because the certificate is just a copy of a written contract.

If we compare a PKI-based signature with a server-based signature, we notice that the validity of them relies on the validity of cryptographic algorithms and keys. At first sight, it may seem that server-based signature is more easily corrupted because its validity depends on the validity of one single key (pk_S), while the PKI-based signature becomes corrupted only if two keys (pk_A and $\mathsf{pk}_{\mathsf{TSA}}$) are compromised. Note, however, that in most signature systems, users are allowed to revoke their signature keys. Once A decides to deny her signature, she may try to make her signature technically invalid by immediately revoking her key. Hence, also the validity of PKI-based signature depends on a single key - $\mathsf{sk}_{\mathsf{TSA}}$ - and hence, from the viewpoint of the verifier, there is no difference between the security of the said two signature systems.

4 Techniques to Improve Efficiency and Security

As shown above, we can reduce the cost of electronic signatures by eliminating personal private keys and the related PKI. In this section, we describe some state of the art techniques to improve the efficiency and security of electronic signature systems. Batch signatures [13] is a solution to efficiency and multi-component signatures increase the security of electronic signature system. We also discuss the *randomly chosen servers* approach that was proposed by Haber et al [7] and observe that in the context of our electronic signature system this approach is impractical because of large signature size.

4.1 Batch Signatures

One of the main problems of server-based electronic signature systems is their low scalability. The reason of the problem is that asymmetric cryptography is slow. *Batch signatures* [13] is a method that allows one to sign a multitude of messages at the time and thereby to speed up the signature process. The most efficient batch signature scheme [13] is based on Merkle hash trees [9, 10]. It was first proposed by Micali [11] but was later ”rediscovered” by several researchers [13, 6, 1]. For creating a (Merkle tree based) batch signature for a list of messages $M_1, \ldots, M_\ell$, a signer first composes the messages using a Merkle tree [9]. The resulting hash value d is then signed by an ordinary signature scheme. Each message M_i is then provided with a pair of (a) the ordinary signature on d, which is common for all messages, and (b) an authentication path $\Pi_i = \Pi_i(M_1, \ldots, M_\ell)$, which proves that M_i took part of the computation of d. If the number of messages is large, we achieve up to thousand-fold speedup in computations. It is argued in [13] why this scheme is as secure as ordinary digital signature schemes.

Batch signatures are not recommended for end users – the number of signatures is not limited and hence the risk is indefinite. For service providers, batch signatures could be the basic mechanism to achieve scalability.

4.2 Multi-Component Signatures

In a server-based signature scheme the server must be ultimately trusted. There is no way to prevent the server from creating signatures in users' name. One way of reducing the trust assumption is to use *threshold trust*. Suppose, we have a multitude of servers $N_1, \ldots, N_n$, each N_i possessing a private signature key sk_i with the corresponding private key pk_i. For signing a message M, a user P authenticates itself to all servers and sends M to each server. By a *multi-component signature* on a message M given by a user A, we mean a sequence of digital signatures

$$\textsc{Sign}[M] = (\textsc{Sig}_{\mathsf{sk}_1}(M, \mathsf{ID}_A), \ldots, \textsc{Sig}_{\mathsf{sk}_n}(M, \mathsf{ID}_A)).$$

The signature of A on M is defined as *valid* if at least $t > 1$ servers have signed (M, ID_A). If the servers are controlled by independent parties then the risk of simultaneous misbehavior of these servers is quite low. In particular, no single server can sign in A's name. Different threshold signature schemes are extensively researched [16].

As shown in Section 3, unconditionally trusted server is not the hardest security problem for large majority of users. We can conclude that applying threshold trust at end user level is hardly practical. However, *multi-component signatures are still useful to the service providers*, who are able to guarantee sufficient level of security in their servers.

Multi-component signatures remain valid even if one of the (component) signatures is corrupted, because the other components still protect the authenticity of electronic signature. It the components are created by using different signature schemes then the signature resist the breakage of signature schemes.

Another (more complex) approach is to use *shared signature schemes* [16]. The key is shared between a multitude of servers so that only a coalition of t servers are able to produce a valid signature. The main advantage of such approach is that only one ordinary digital signature is produced, and hence, the size of a signature is smaller than in the multi-component signature approach. Main drawback of shared signatures is that they do not withstand the breakage of a signature scheme. Hence, we prefer the use of multi-component signatures.

4.3 Randomly Chosen Servers

Haber et al [7] proposed a method how to use smaller threshold values t, so that the system would still be relatively secure against attacks performed by $p > t$ colluding servers. The main idea is to use a public pseudo-random function G, which given as input a message M outputs a list of t servers the signatures of whose are necessary for the multi-component signature on M being valid. Their solution may seem very attractive but it leads to impractically large signatures.

If there are n servers in total and a set S of p malicious servers try to forge a signature on M. Attacker chooses slight modifications M' of M, so that the meaning of M' stays almost the same (by rewording sentences or changing numerical values etc.). The goal of this attack is to find M' such that $G(M') \subseteq S$. For each M', the probability that $G(M') \subseteq S$ is $\pi = \left(\frac{p}{n}\right)^t$. The probability of success after N trials is $\pi_N = 1 - (1-\pi)^N$. Note that in most cases, there is no problem to generate a large number of modifications of M. For example, if there are 30 words in M that have at least one synonym then the number of modifications is 2^{30}. If we have two numerical values each having at least one thousand modifications then we have one million modifications. Hence, one may first fix the words and numerical values that may be changed and then use a computer to generate all combinations M' one at a time and check whether $G(M') \subseteq S$. Hence, the number N of trials must be sufficiently large. From the assumptions $\pi_N \leq 1/2$ and $N = 10^k$ we conclude that $t \approx k \cdot \frac{\ln 10}{\ln \frac{n}{p}}$. If one third of the servers are corrupted ($p/n \approx 1/3$) then $t \approx 2k$. Hence, if $N = 10^{20} \approx 2^{66}$ then $t \approx 40$ which is clearly impractical. If again $t = 10$ (which correspond to a reasonable signature size) then an adversary must try about 10^5 random modifications, which is feasible to almost any computer. Hence, for this method to increase practical security, the size of multi-component signature must be few hundred K bytes. For this reason, we do not use the pseudo-random choice method in our system.

5 Electronic Signature Service

Our goal is to design a server-based signature system that is capable of serving billions of clients. The cost of the system must be much lower than PKI-based solutions, while the security must be comparable or better.

As our goal is to minimize the number of private keys in the system, we use two layers of servers (Fig. 3). The front-end *Proxy servers* authenticate clients and process signature requests. The back-end *Notary servers* sign the processed requests.

Each Notary server can serve up to one thousand Proxy servers. Since we use multi-component signatures, each Proxy uses at least two Notary servers. Consequently, in a system with few thousand Proxy servers we need about ten Notary servers. Such a service would potentially be capable of serving the whole on-line Internet community. Users of such system would need just a web browser to sign or verify messages. User authentication could be carried out with tools already incorporated in web browsers (including PKI-based authentication).

As there are about ten key-pairs in total the system has potentially enough resources to guarantee sufficient protection of private keys. The Public Key Infrastructure related to authentic distribution of public keys is very small and could be efficiently implemented. All keys could be stored in browsers' code and hence their use could be completely transparent to end users. Even if Notary keys are changed annually, all the history of keys would still fit into the code of web browsers for hundred years.

5.1 Authenticating a User

User authentication is one of the most costly parts in electronic signature systems. To create a reliable database for user authentication, we probably need face to face communication with all clients. Assuming that only 15 minutes is spent for each user, we deduce that to create a database for one million users, we need at least 1300 man months in total. However, we mostly do not have to start systems from scratch – there are many client bases already developed. For example, numerous banks have internet banking systems with several hundred thousands clients. Though the authentication methods used are different, it would still be reasonable to reuse the existing authentication systems rather than built new ones from the scratch – that would reduce the overall costs. In server-based electronic signature system, the use of a variety of different authentication methods does not affect the simplicity and uniformity of electronic signatures, because only the result of the authentication is included into the signature.

5.2 Signing a Message

For signing a message M, a user A authenticates itself to a Proxy server P and sends M to P. Proxy server immediately replies with electronic signature, which can be verified in client's browser. For users, electronic signature system is just a web service.

5.3 Creating a Signature

After successful authentication of a user, a Proxy server composes a signature statement (M, ID_A) that includes the message M to be signed and a representation of user's identity. The statement may comprise other information, like signature policy, liability constraints, time/date, etc. Proxy server does not sign each signature statement separately, but instead works in rounds and signs the signature statements

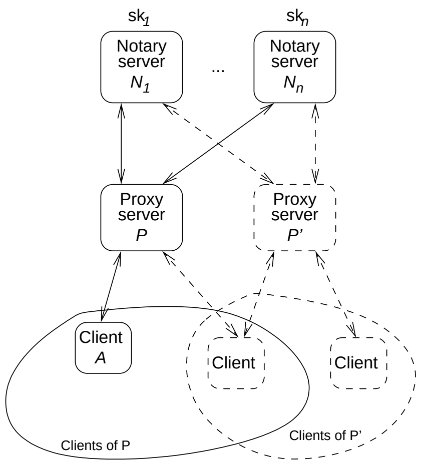

Figure 3: Main structure of the signature system.

in "batch mode". During each round, it collects signature statements. At the end of a round, Proxy server computes a cryptographic digest d of all statements of this round and sends d to n Notary servers $N_1, \ldots, N_n$.

Each Notary server N_i authenticates P and signs a triple (d, ID_P, t_i), where t_i is the current time, and ID_P is a representation of P's identity. Strong cryptographic authentication, like Message Authentication Codes [12] can be used to make impersonation of Proxies very difficult. Having received digital signatures $\textsc{Sig}_{\mathsf{sk}_1}(d, \mathsf{ID}_P, t_1)$, ... , $\textsc{Sig}_{\mathsf{sk}_n}(d, \mathsf{ID}_P, t_n)$, the Proxy P composes complete electronic signatures for all clients who sent their requests during the round. An electronic signature of A on message M is of the form

$$[\mathsf{ID}_A, \Pi, \textsc{Sig}_{\mathsf{sk}_1}(d, \mathsf{ID}_P, t_1), \ldots, \textsc{Sig}_{\mathsf{sk}_n}(d, \mathsf{ID}_P, t_n)], \tag{3}$$

where Π is *authentication path* – a set of hash values which proves that (M, ID_A) participated in the computation of d (the root of Merkle tree [9]).

5.4 Verifying a Signature

To verify a signature (3) one has to possess authentic copies of public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_n$. Verification consists of the following steps:

(i) The root of the Merkle tree is recomputed by using M and the authentication path Π. If the recomputed root hash d' does not coincide with d then the result of the verification is $\mathsf{Invalid}$. Otherwise, the verification continues with the next step.

(ii) The signatures

$$\textsc{Sig}_{\mathsf{sk}_1}(d, \mathsf{ID}_P, t_1), \ldots, \textsc{Sig}_{\mathsf{sk}_n}(d, \mathsf{ID}_P, t_n)$$

are verified using the public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_n$ and the verification procedure $\textsc{Ver}$. If at least t of those signatures are valid, then the result of the verification is Valid. Otherwise, the signature (3) is $\mathsf{Invalid}$.

Note that if all authentication procedures are omitted from the signature creation process, we obtain a *time stamp* [7] instead of electronic signature. Hence, the same service can be used to obtain timestamps.

Time stamps are needed for long-term preservation of electronic signatures. In case one of the component-signatures of s is broken, or if one of the hash functions (either the function h used to create the hash of the message or the one used to create the Merkle tree) used is suspected of getting broken soon, it is sufficient to take a new and secure hash function H and obtain a time stamp for a message $(s, H(M))$ just as described by Haber and Stornetta [3]. The time stamp is added to the signature in order to preserve its validity.

6 Conclusions

Personal private keys do not necessarily mean higher security. They certainly mean high cost and complexity of electronic signature systems. Elimination of personal private keys could considerably simplify the system, and as we have shown, not at the price of security.

Personal private keys were introduced in order to solve the problems with trust. We claim that no technology – personal private keys or any other measure – can solve problems with trust. Trust relations cannot be imposed by technology. They evolve in natural ways.

In some sense, our society is still in the stage of ”electronic illiteracy” – blind trust to technology is inevitable – and it is hard to see how this situation will change in the near future. Nevertheless, electronic signatures could still be used massively. We have shown that electronic signature service can provide a sufficiently secure solution to electronic signatures.

We presented an electronic signature system that is capable of covering the needs for electronic signatures for the whole Internet community. All the components and primitives we used in our system are well known. The new system is extremely simplified, but still remains as secure as any other electronic signature system known to date.

Acknowledgements

We would like to thank Margus Freudenthal, Jaan Priisalu and Viljar Tulit for fruitful discussions on practical aspects of security. We are also grateful to anonymous referees for their helpful comments.

References

[1] Arne Ansper, Ahto Buldas, Meelis Roos and Jan Willemson. Efficient long-term validation of digital signatures. In *Public Key Cryptography - PKC’2001*, Cheju Island, Korea. Feb. 13-15, 2001. LNCS 1992, 402-415. Springer-Verlag, 2001.

[2] A.Asokan, G.Tsudik, M.Waidner. Server-supported digital signatures. In proceedings of ESORICS’96, Rome, Italy, Sept. 25-27, 1996.

[3] Dave Bayer, Stuart Haber, W. Scott Stornetta. Improving the Efficiency and Reliability of Digital Time-Stamping. In *Sequences II: Methods in Communication, Security, and Computer Science*, eds. R. Capocelli, A. DeSantis, and U. Vaccaro, pp. 329-334. Springer-Verlag, 1993.

[4] W.Diffie and M.E.Hellman. New directions in cryptography. IEEE Trans. Inform. Theory, IT-22, 6, 1976, pp.644-654.

[5] Ford, W. A Public-Key Infrastructure for U.S. Government Unclassified but Sensitive Applications. Produced by Nortel and Bell-Northern Research for NIST, September 1995.

[6] I. Gassko, P. S. Gemmell, and P. MacKenzie. Efficient and fresh certification. In *International Workshop on Practice and Theory in Public Key Cryptography – PKC’2000*, LNCS 1751, pp. 342–353, Melbourne, Australia, 2000. Springer-Verlag, Berlin Germany.

[7] Stuart Haber, W. Scott Stornetta. How to time-stamp a digital document. *Journal of Cryptology*, 3(2):99–111, 1991.

[8] S. Halevi and H. Krawczyk. Public-key cryptography and password protocols. In Proceed-

ings of the Fifth Annual Conference on Computer and Communications Security, pages 122–131, 1998.

[9] Ralph C. Merkle. Protocols for Public Key Cryptosystems. In *Proceedings of the 1980 IEEE Symposium on Security and Privacy*, pp. 122-134, 1980.

[10] United States Patent 4,309,569. Ralph Merkle. Method of providing digital signatures. January 5, 1982.

[11] United States Patent 6,097,811. Silvio Micali. Tree-based certificate revocation system. August 1, 2000.

[12] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. Handbook of applied cryptography. CRC Press series on discrete mathematics and its applications. CRC Press, 1996. ISBN 08493 -8523-7.

[13] Chris Pavlovski and Colin Boyd. Efficient Batch Signature Generation using Tree Structures. *International Workshop on Cryptographic Techniques and E-Commerce – CrypTEC'99*, City University of Hong Kong Press, pp.70–77.

[14] `http://www.pgp.com`

[15] Bruce Schneier. Attack Trees: Modeling security threats *Dr. Dobb's Journal*, December 1999.

[16] Victor Shoup. Practical threshold signatures. In *Advances in Cryptology – EUROCRYPT'2000*, LNCS 1807, pp. 207–220. Springer-Verlag, Berlin, 2000.

[17] RFC 3161. Time-Stamp Protocol

[18] John Viega, Gary McGraw. *Building Secure Software: How to Avoid Security Problems the Right Way.* Addison Wesley Professional, 2001. ISBN: 0-201-72152-X. (Chapter 6) `http://www.sdmagazine.com /documents/s=818/sdm0208a/`

[19] Adam Young and Moti Yung. Kleptography: Using Cryptography Against Cryptography. In *Advances in Cryptology – Eurocrypt'97,* LNCS 1233, pp. 62–74. Springer-Verlag. ISBN 3-540-62975-0