Digital Cash

Mandana Jahanian Farsi

Master's Thesis in Computer Science

Department of mathematics and computing science

Göteborg University

1997

Examiner: Dag Wedelin, Göteborg University

Adviser: Mikael Simovits

Abstract

Need of a payment system which enables the electronic transactions is growing at the same time that the use of Internet is growing in our daily life. Present days electronic payment systems have a major problem they cannot handle the security and the users anonymity at the same time these systems are secure on the cost of their users anonymity. Digital cash is a payment system which enables a secure off-line transaction without revealing the payers identity. Digital cash can be used both as papper cash and electronic money since it keeps its users anonymity, enables off-line transactions, is portable and at the same time offers the ability of electronic transactions. This report explains the concept of digital cash and discusses its properties. This report shows how a digital cash system can be formed by presenting a few of the present days digital cash systems in details, together with an evaluation of how well these systems fulfils the properties of digital cash. We also discuss the implementation and practical usage of digital cash. In the end of this report we will disscuss if an ideal digital cash system exist in practice by discussing implementing problems, user problems and the problems related to the objectives of digital cash system.

Keywords: Cryptography, Digital Cash, Off-line payment, Anonymity

Acknowledgments

Dag Wedelin, my examiner, gave many helpful comments on this report. He also suggested the idea of comparing some digital cash systems. My adviser, Mikael Simovits has been very helpful throughout this project. It is safe to say that it would not have existed without him, since he sparked my interest in cryptography and digital cash in the first place.

CONTENTS

1 Introduction to Digital Cash

This thesis is about digital cash, a way to implement anonymous electronic payments in an environment of mutual mistrust between the bank and the system users.

The present day payment systems fall into two large categories: account-based systems and token-based systems. Token-based systems such as paper cash, pre-paid phone cards or mail stamps, do not identify its users. A pre-paid phone card, for example, does not distinguish one caller from the other. Account-based systems such as checks, credit cards or bank accounts need, by design, to identify the system users and their transactions.

People like to use paper cash because it is easy to carry around, they can make a payment with the received cash and they don't need to ask a third party like a bank to perform their payments. Paper cash can, however, be stolen or lost and no one compensates for the lost or stolen money.

Credit cards reduces risk of lost cash for people, but by using electronic money people are in the risk of losing their privacy. Annually, credit card companies and banks lose large sums of money since they are required to compensate for lost cards and the costs associated with fraud and human error. In light of the explosive increase of electronic services such as Internet, the need for more efficient electronic payments has become an essential fact.

Since anonymity of payments is usually associated with anonymity of paper cash, an anonymous token-based electronic payment system is referred to as digital cash (also known as electronic cash, e-cash, D-cash). Digital cash offers a solution to the problems of paper cash and today's credit cards; it is secure and protects people's privacy. The customer can use digital cash to pay over the Internet without the involvement of a bank during their payments.

The goal of this report is to present a few of the present day digital cash systems, discuss their properties, provide a comparison and determine them together to see which one of them fulfils the properties for digital cash and the required security level. This report also presents an understanding of the method that the chosen system can be applied in practice. In addition this report tries to be a bridge by building a bridge between the gap of research and real-life application.

This report is organized as follows: The remainder of this chapter, presents why today's society needs a digital cash system and the properties of digital cash system are discussed in detail. To fully understand digital cash systems, it's best to begin reading an overview of cryptography protocols which are the building blocks for digital cash protocols; chapter 2 presents this overview. Chapter 3 explains one of the first digital cash systems; one which was introduced by Chaum, Fiat and Naor. In chapter 4, the Ferguson single-term digital cash system is explained. Chapter 5 explains one of the most efficient digital cash systems, which is Brand's system based on the representation problem. In chapter 6, one of the few divisible digital cash systems produced by Okamoto is presented. Chapter 7 is an overview of some products related to digital cash protocols. Finally, chapter 8 provides a discussion regarding the implementation of a digital cash system based on Brand's protocol.

Many people are interested in the business of digital cash systems. Here is the best way to proceed: Read the first chapter with an eye toward understanding the benefits of creating a digital cash system. Read chapter 7 for an overview of the various products. The final chapter, chapter 8, will give you some ideas about how a digital cash system can be built. Read the remaining chapters of interest.

Some people are interested in the research or will implement a digital cash system. Here are some suggestions for achieving the information you will need to do this: Read this chapter as an introduction. Read chapter 2 if you are not familiar with the various encryption algorithms. Study chapter 3, 4, 5 and 6 for details about how to design digital cash systems. Chapter 8, will give you some ideas about how a digital cash system can be implemented and about the problems that will be encountered in real-life implementation.

1.1 General Structure of Digital Cash Transactions

There are three different types of transactions during a digital cash procedure:

a) *Withdrawal*, in which Alice transfers some of her money from her bank account to her wallet (it could be a smart card or a personal computer).

b) *Payment*, in which Alice transfers money from her wallet to Bob's.

c) *Deposit*, in which Bob transfers the money he has received to his bank account.

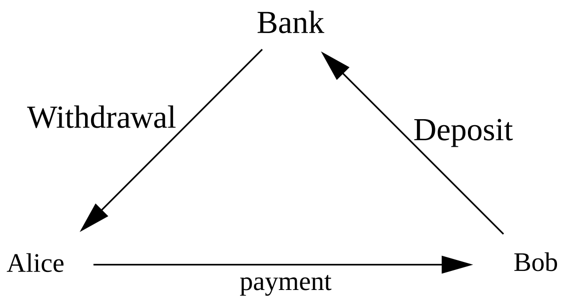


FIGURE 1. Life-cycle of electronic coins

In a digital cash system we have three kind of actors:

- A financial network (The bank).
- A payer or consumer (Alice).
- A payee or a shop (Bob).

1.2 Important Properties of Digital Cash

Digital cash is designed to construct an electronic payment system modelled after our paper cash system. Therefore Digital Cash should have the same features as paper cash like: recognizable hence readily acceptable, transferable, untraceable, anonymous and portable and has the ability to make “change” (some people like Okamoto believe that even the paper cash is undivisable[24]).

Here we present in detail some necessary properties of digital cash [21].

1.2.1 Security

With security we mean that digital cash cannot be copied and reused. Then we have to minimize the risks for forgery and establish a good authenticity system.

Forgery

The most obvious risk with any payment system is forgery or counterfeiting. As with paper cash we have two kinds of forgery in a digital cash system.

- Token forgery: to create a valid-looking coin without making a corresponding bank withdrawal.
- Multiple spending: using the same token over again. Multiple spending is also commonly called re-spending, double-spending, and repeat-spending.

To protect against token forgery, one relies on the usual authenticity functions of user identification and message integrity. To protect against multiple spending, the bank maintains a database of spent electronic coins. Coins already in the database are to be rejected for deposit. If the payments are on-line, this will prevent multiple spending. If off-line, the best one can do is to detect when multiple spending has occurred. To protect the payee, it is then necessary to identify the payer. Thus it is necessary to disable the anonymity mechanism in the case of multiple spending.

Authenticity

As a consequence of the problems of forgery, it becomes necessary to establish various levels of authenticity measures.

- user identification: A user must know with whom he is dealing with.
- message integrity: To be sure that the copy of the message is the as same as it was in the beginning.
- nonrepudiation: To protect against later denial of a transaction.

The authenticity features are attained via key management. Key management is carried out using a certification authority(CA) (see section 2.12), a trusted agent who is responsible for confirming a user’s identity. Without a trusted CA and a secure infrastructure, the security features of digital cash will be practically impossible over an entrusted transmission medium like Internet.

1.2.2 Privacy

The definition of privacy is not really clear. For some people privacy means protection against eavesdropping but for others like David Chaum privacy means anonymity for the payer during payment and untraceability of the payment such that the bank cannot tell whose money was used in a particular payment.

Just as cash is anonymous, digital cash is anonymous in that it cannot be traced back to a particular individual, it is considered to be “unconditionally untraceable”. However, the service provider is assured of its authenticity, all that is missing is the ability to link the transaction with a particular person. If a user’s coin is linkable, we can identify the user by finding a single payment in which the user has identified himself. Then a digital cash system will protect user’s privacy if it is both unlinkable and untraceable.

Digital cash systems that don’t pay attention to privacy are “privacy-invading systems”. Virtually all commercial systems currently being proposed are privacy-invading. They emphasize the bank’s security, but pay little attention to the security of the customer (in terms of protection from financial surveillance).

Anonymity increases the danger with money laundering, illegal purchasing, blackmailing and counterfeiting that are far more serious than with paper cash. Anonymity would increase the danger of these problems. More anonymity means less security and vice versa.

1.2.3 Portablility

The security and use of digital cash is not dependent on any physical location. The cash can be transferred through computer networks into storage devices and vice versa.

1.2.4 Transferability

Transferability allows a user to spend a coin that he has received in a payment without having to contact the bank. A payment is *transfer* if the payee can use the received coin in a payment. A payment system is *transferable* if it allows at least one transfer per coin. We have to notice that the ability to transfer paper cash is very important in our daily life. The life-cycle of an electronic coin in a transferable system looks like:

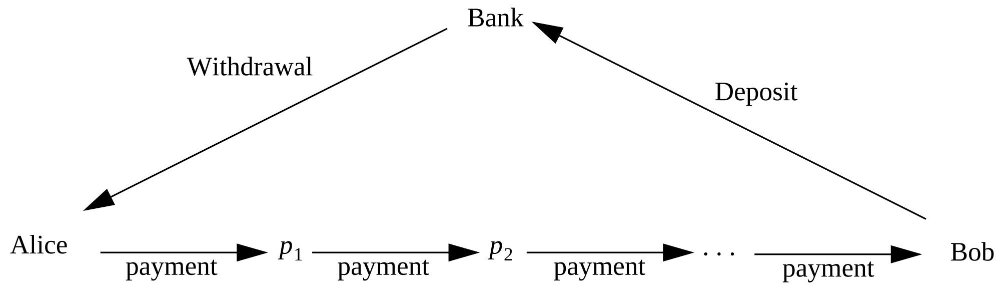


FIGURE 2. Life-cycle of a transferable coin

The problems which appears with transferable systems are:

- Any transferrable electronic cash system has the property that the coin must grow in size [8] each time it is spent because of the information it has to contain. This information is about every person who has spent the coin for the bank to maintain its ability to catch multiple spenders. This limits the maximum number of transfers allowed in the system by the allowable size of the coin.
- Money laundering and tax evasion are hard to detect since no records of the transactions are available.
- Each transfer delays detection of multiple spending or forged coins. Multiple spending will not be noticed until two copies of the same coin are deposited and it may be too late by then.
- Users can recognize their coin if they sees it later in another payment.

1.2.5 Divisibility

With divisibility we mean the ability to make change. So digital cash will come in cent or smaller denominations that can make high-volume, small-value transactions on the internet practical.

A solution for divisible coins is using coins that can be divided to coins whose total value is equal to the value of the original coin. This allows off-line payments to be made without the need to store a supply of coins of different denominations.(Observe that Okamoto believes that even normal paper cash can't satisfy this characteristic by being divisible [10]).

Three divisible off-line schemes have been proposed at a cost of transaction time and additional storage, Eng and Okamoto's scheme [17], Okamoto's scheme [23] and Okamoto and Ohta scheme [24].

1.2.6 Off-line Payment

Cash protocols can be implemented in either of two ways Off-line or On-line. An ideal cash system is the one which works off-line [24].

The discussion in this report addresses off-line payment systems.

Off-line

Off-line payment means that Bob submits Alice's electronic coin for verification and deposit sometime after the payment transaction is completed. It means that with an off-line system Alice can freely pass value to Bob at any time of the day without involving any third party like a bank.
Although off-line systems are preferable from a practical viewpoint, they are however susceptible to the multi-spending problem and therefore suitable for low value transactions.
Over the past years, some off-line cash systems have been designed that can not only guarantee security for the bank and shops, but also privacy for the users. Table 1. shows a list of these systems.

TABLE 1. a list of Off-line scheme which has been proposed [28].

Scheme	proposing year	Scheme	proposing year
Pfitzmnn and Waidner	1988	Cramer and Pedersen	1993
Chaum, Fiat and Naor	1988	Franklin and Yung	1993
Okamoto and Ohta	1989	Chan, Frankel and Tsiounis	1995
Damgård	1990	Chan, Frankel and Tsiounis	1996
Brands	1993		

On-line

On-line payment means that Bob calls the bank and verifies the validity of Alice’s token by a simple question like “have you already seen this coin” before accepting her payment and delivering his merchandise (This resembles many of today’s credit card transactions.).

On-line payment remains necessary for transactions that need a high value of security. With an on-line system, the payment and deposit are not separate steps. On-line systems require communication with the bank during each payment, which costs more money and time (communication costs, database-maintenance costs and turn-around time), however the protocols are just simplification of off-line protocols.

Since on-line systems have to be able to check the credibility of payers for shops, it is almost impossible to protect the anonymity of its users, besides as on-line systems require communication with a third party during the payment transaction, then we can not have transferable coin if the system is an on-line one.

2 Some Basic Encryption Algorithms

Since cryptography is such an important part of digital cash, this chapter presents some of encryption algorithms that are useful in constructing digital cash systems. These basic algorithms are the building blocks for the systems presented in this report.

2.1 Mathematical Background

This section is a collection of the mathematical fundamentals used in this report. Interested readers can refer to [22] for a more detailed discussion.

1. Z denotes the set of *integers*, $\{..., -2, -1, 0, 1, 2, ...\}$.
2. Q denotes the set of *rational numbers*, $\{a/b|(a,b \in Z,(b \neq 0))\}$.
3. R denotes the set of *real numbers*.
4. If A is a finite set, then $|A|$ denotes the number of elements in A, called *cardinality* of A, unless is $|A|$ the size of an integer in bits.
5. $a \in A$ means that element a is a member of the set A.
6. $\Sigma_{i=1}^{n} a_i$ denotes the sum $a_1 + a_2 + ... + a_n$.
7. $\Pi_{i=1}^{n} a_i$ denotes the product $a_1 \cdot a_2 \cdot ... \cdot a_n$.

Definition(1) A non-negative integer *d* is the greatest common divisor of integers *a* and *b*, denoted *d = gcd(a, b)*, if

(i) *d* is a common divisor of *a* and *b*, and

(ii) whenever $c|a$ and $c|b$, $c|d$.

Definition(2) Two integers *a* and *b* are said to be relatively prime if *gcd(a, b)* = 1.

Definition(3) An integer $p \geq 2$ is said to be *prime* if its only positive divisors are 1 and *p*.

Definition(4) If *a* and *b* are integers and *n* is a positive integer, then *a* is said to be *congruent to b modulo n*, written $a \equiv b \bmod n$, if *n* divides $(a - b)$. The integer *n* is called the *modulus* of the congruence.

Fact(1) (*properties of congruences*) For all $a, a_1, b, b_1, c \in Z$, the following are true.

(i) $a \equiv b \bmod n$ if and only if *a* and *b* leave the same remainder when divided by *n*.

(ii) (*reflexivity*) $a \equiv a \bmod n$.

(iii) (*symmetry*) If $a \equiv b \bmod n$ then $b \equiv a \bmod n$.

(iv) (*transitivity*) If $a \equiv b \bmod n$ and $b \equiv c \bmod n$, then $a \equiv c \bmod n$.

(v) If $a \equiv a_1 \bmod n$ and $b \equiv b_1 \bmod n$, then $a + b \equiv (a_1 + b_1) \bmod n$ and $a \cdot b \equiv (b_1 \cdot a_1) \bmod n$.

Definition(5) Z_n, the integer modulo *n*, is the set of $\{0, 1, \ldots, n-1\}$.

Definition(6) Z_n^* means the *multiplicative group* of Z_n, which is $Z_n^* = \{a \in Z_n | gcd(a,n) = 1\}$. In particular, if *n* is a prime, then $Z_n^* = \{a | (1 \le a \le n-1)\}$.

Definition(7) The order of Z_n^* is defined to be the numbers of elements in Z_n^*, namely $|Z_n^*|$.

Definition(8) Let $a \in Z_n^*$. The order of *a*, denoted *ord(a)*, is the least positive integer *t* such that $a^t \equiv 1 \bmod n$.

Definition(9) For $n \ge 1$, let $\phi(n)$ denote the number of integers in the interval $[1, n]$ which are relatively prime to *n*.

Definition(10) Let $g \in Z_n^*$. If the order of *g* is $\phi(n)$, then *g* is said to be a *generator* of Z_n^*. If Z_n^* has a generator, then Z_n^* is said to be *cyclic*.

Fact(2) (*properties of generators of* Z_n^*)

(i) Z_n^* has a generator if and only if $n = 2, 4, p^k \text{ or } 2p^k$, where *p* is an odd prime and $k \ge 1$. In particular, if *p* is prime, then Z_p^* has a generator.

(ii) If *g* is a generator of Z_n^*, then $Z_n^* = \{g^i \bmod n | (0 \le i \le \phi(n) - 1)\}$.

(iii) Suppose that g is a generator of Z_n^*. Then $b = g^i \bmod n$ is also a generator of Z_n^* if and only if $gcd(i, \phi(n)) = 1$. It follows that if Z_n^* is cyclic, then the number of generators is $\phi(\phi(n))$.

(iv) $g \in Z_n^*$ is a generator of Z_n^* if and only if $g^{(\phi(n))/p} \ne 1 \bmod n$ for each prime divisor *p* of $\phi(n)$.

Fact(3) Finding a generator *g* is easy if you know the factorization of *n-1*. You just need to calculate $g^{(n-1)/p} \bmod n$ for all values of *p*, the prime factors of *n*. If any of the results are 1, then *g* is not a generator.

Example say you want to see if *5* is a generator mod *2047*. The prime factors of $n - 1 = 2046$ are {2, 3, 11, 31}, so you calculate:

$5^{2046/2} \bmod 2047 = 1037$

$5^{2046/3} \bmod 2047 = 622$

$5^{2046/11} \bmod 2047 = 1435$

$5^{2046/31} \bmod 2047 = 622$

None of these turned out to equal 1, so 5 is a generator mod 2047.

Definition(10) $a \in Z_n^*$ is said to be a *quadratics residue* modulo *n*, if there exist an $x \in Z_n^*$ such that $x^2 \equiv a \bmod n$. The set of all quadratics residue modulo *n* is denoted by Q_n.

Example In the case of *n = 11*, the residues are *{1, 3, 4, 5, 9}* because:

$$1^2 = 1 \bmod 11,\ 2^2 = 4 \bmod 11,\ 3^2 = 9 \bmod 11,\ 4^2 = 5 \bmod 11$$

$$5^2 = 3 \bmod 11,\ 6^2 = 3 \bmod 11,\ 7^2 = 5 \bmod 11,\ 8^2 = 9 \bmod 11$$

Fact(4) Let *n* be the product of two distinct odd primes *p* and *q*, *n = pq*. Then $a \in Z_n^*$ is a *quadratics residue* modulo *n* if and only if $a \in Q_p$ and $a \in Q_q$. It follows that $|Q_n| = |Q_q||Q_p| = ((p-1)(q-1))/4$.

Definition(11) Let $a \in Q_n$. If $x \in Z_n^*$ satisfies $x^2 \equiv a \bmod n$, then *x* is called a *square root* of modulo *n*.

Definition(12) Let *p* be an odd prime and *a* an integer. The *Jacobi symbol (a/p)* is defined to be 1 if *a* is a quadratic residue of *p* and, -1 if it isn't.

Example (4/11) = 1 and (6/11) = -1.

Definition(13) If *n* is not a prime, then the *Jacobi symbol* (*a/n*) is defined to be

$$(a/n) = (a/p_1^{a_1} \dots p_n^{a_n}) = (a/p_1)^{a_1} \times \dots \times (a/p_n)^{a_n}.$$

Definition(14) If $p = 3 \bmod 4$ and $q = 3 \bmod 4$, where *p* and *q* are primes then *n = pq* is a Blum integer.

Definition(15) *N=pq* is called *Williams integer* if $p = 3 \bmod 8$ and $q = 3 \bmod 8$. A Williams integer is a specific type of the Blum integer and has all properties of the Blum integer.

Fact(5) (*number of square roots*)

(i) Each quadratic residue has two square roots if *n* is prime one of them is smaller than *(n/*2*)* and the other is larger.

(ii) If *n* is not prime then each of these quadratic residues has four square roots. This four roots can be divided into four different sets:

1. The values between 1 and *pq* which have *(a/p)* = 1 and *(a/q)* = 1,

$$x_1 = Z(1,1)^{pg}$$

2. The values between 1 and *pq* which have *(a/p)* = 1 and *(a/q)* = -1,

$$x_2 = Z(1,-1)^{pq}$$

3. The values between 1 and *pq* which have *(a/p)* = -1 and *(a/q)* = 1.

$$x_3 = Z(-1,1)^{pq}$$

4. The values between 1 and *pq* which have *(a/p)* = -1 and *(a/q)* = -1.

$$x_4 = Z(-1,-1)^{pq}$$

Fact(6) Let $x_1 \in Z(1,1)^{pq}$, $x_2 \in Z(1,-1)^{pq}$, $x_3 \in Z(-1,1)^{pq}$ and $x_4 \in Z(-1,-1)^{pq}$. Then $x_1 = -x_4$ and $x_2 = -x_3$, $(x_1/pq) = (x_4/pq) = 1$ and $(x_2/pq) = (x_3/pq) = -1$.

The important fact here is that knowing a pair of roots which are not negative of each other is enough to compute *pq*. That is a pair like x_1 and x_2 not a one like x_1 and x_4. For instance if

$$x_1^2 \bmod pq = x_3^2 \bmod pq,$$

then

$$x_1^2 - x_3^2 = 0 \mod pq$$

and

$(x_1^2 - x_2^2) = (x_1 - x_3)(x_1 + x_3) = kpq$, (for some integer *k*).

Definition(16) A *group* $(G, \bullet)$ consists of a set *G* with a binary operation $\bullet$ on *G* satisfying the following three axioms.

(i) The group operation is *associative*. That is, $a \bullet (b \bullet c) = (a \bullet b) \bullet c$, for all $a, b, c \in G$.

(ii) There is an element $1 \in G$, called the *identity element*, such that $a \bullet 1 = 1 \bullet a = a$ for all $a \in G$.

(iii) For each $a \in G$ there exists an element $a^{-1} \in G$, called the *inverse* of a, such that $a \bullet a^{-1} = a^{-1} \bullet a = 1$.

Definition(17) A non-empty subset *H* of a group *G* is a subgroup of *G* if *H* is itself a group with respect to the operation of *G*. If *H* is a subgroup of *G* and $H \neq G$, then *H* is called a proper subgroup of *G*.

Fact(7) If *G* is a group and $g \in G$, then the set of all powers of *g* forms a cyclic subgroup of *G*, called the subgroup generated by *g*, and denoted by $\langle g \rangle$.

Fact(8) Let *G* be a group.

(i) If the order of $a \in G$ is *t*, then the order of a^k is $t/(gcd(t,k))$.

(ii) If *G* is a cyclic group of order *n* and $d|n$, then *G* has exactly $\phi(d)$ elements of order *d*. In particular, *G* has $\phi(n)$ generators.

Example Consider the group $Z_{19}^{*} = \{1, 2, ..., 18\}$ of order *18*. The group is cyclic, and a generator is $g = 2$. The subgroups of Z_{19}^{*}, and their generators, are listed in the table.

TABLE 2. The subgroups of Z_{19}^{*}.

Subgroup	**Generators**	**Order**
{1}	1	1
{1,18}	18	2
{1,7,11}	7,11	3
{1,7,8,11,12,18}	8,12	6
{1,4,5,6,7,9,11,16,17}	4,5,6,9,16,17	9
{1,2,3,...,18}	2,3,10,13,14,15	18

Fact(9) Let *p* be a prime.

(i) (*Fermatas theorem*) If gcd(*a*, *p*) = 1, then $a^{p-1} = 1 \bmod p$.

(ii) If *r* mod (*p*-1) = *s* mod (*p*-1), then $a^r \bmod p = a^s \bmod p$ for all integers *a*.

(iii) In particular $a^p \bmod p = a \bmod p$.

Algorithm(1) *Decline algorithm* for computing the greatest common divisor of two integers.

INPUT: two non-negative integers *a* and *b* with $a \geq b$.

OUTPUT: the greatest common divisor of *a* and *b*.

```
1. While $b \neq 0$ do the following:
    1.1 Set $r \leftarrow a \bmod b$, $a \leftarrow b$, $b \leftarrow r$.
2. Return($a$).
```

2.2 RSA

In a public-key algorithm, the keys are formed in a pair of an encryption and a decryption key and it is infeasible to generate one key from the other. The algorithm keeps one key secret and sends the other to the partner over an open canal. To date many public-key cryptography algorithms have been proposed [26]. Many of them are impractical others insecure. Only a few algorithms are both secure and practical and of these algorithm just three [26] work for both encryption and digital signatures.

RSA is by far the easiest Public-Key algorithm to understand and implement. It was introduced in 1978 by Rivest, Shamir and Adleman and it works for encryption as well as for digital signatures.

Principles of RSA

RSA relies upon modulo arithmetic. Both encryption and decryption are completed by raising numbers to a power modulo, a number which is product of two large primes (at

least 100 to 200 digits).
To encode a message using RSA, a user needs a pair of keys.
To generate the two keys, take:

p, q: two large primes, for maximum security they should be of equal length.
n: the public key, the product of *p* and *q*.
e: the public key,
a randomly chosen number which is less than *n* and relatively prime to (*p* - 1)(*q* - 1), (*e* has no factors in common with (*p* - 1) and (*q* - 1),
d: the private key,
the inverse of (*e* mod (*p* - 1)(*q* - 1)) such that (*ed* = 1 mod (*p* - 1)(*q* - 1)).
(Euclidean's algorithm (see section 2.1) can be used to determine *d*)

The two large prime factors *p* and *q* must be kept secret or destroyed, since if anyone could factor *n* into *p* and *q*, the private key, *e*, could be obtained.
Encryption is simple. A message, *m*, is divided into numerical blocks, m_i, smaller than *n*. The encrypted message, *c*, will be made up of similarly sized message blocks, c_i, of about the same length. The encryption formula is

$$c_i = m_i^e \bmod n$$

and the decryption formula is

$$m_i = c_i^d \bmod n$$

The decryption process works because:
ed = 1 mod (*p* - 1)(*q* - 1),
then an integer *k* exist such that
ed = 1 + *k*(*p* - 1)(*q* - 1).
Now, if
gcd(*m*, *p*) = 1
then by Fermat's theorem (see section 2.1),

$$m^{p-1} = 1 \bmod p$$

Raising both sides of this term to the power *k*(*q* - 1) and then multiplying both sides by *m* gives

$$m^{1+k(p-1)(q-1)} = m \bmod p$$

On the other hand, if
gcd(*m*, *p*) = p
then this last term is again valid since each side is congruent to 0 modulo *p*. Hence, in all cases

$$m^{ed} = m \bmod p .$$

By the same argument,

$$m^{ed} = m \bmod q .$$

Finally, since *p* and *q* are distinct primes, it follows that

$$m^{ed} = m \bmod n ,$$

and, hence,

$$c^d \bmod n = (m^e)^d \bmod n = m \bmod n$$

Security of RSA

RSA gains its security from the difficulty of factoring large prime numbers. Recovering plaintext m from the corresponding ciphertext c, given the public-keys (n, e) is equivalent to problem of factoring n, depends on $n's$ length.

A 40 bit RSA key could probably be broken with no calculator. That is only 10^{12} digits and factoring such a small number is not difficult (tables of all primes up to million exist [27]).

The best size for an RSA key depends on one's security needs. The larger the key, the greater the security, but also the slower the RSA operations. One should choose a key length upon consideration, first, of one's security needs, such as the value of the protected data and how long it needs to be protected and second, of how powerful one's potential enemies are. It is believed that 512-bit keys no longer provide sufficient security with the advent of new factoring algorithms and distributed computing. Such keys should not be used after 1997 or 1998. The RSA Laboratories recommended key sizes are now 768 bits for personal use, 1024 bits for corporate use, and 2048 bits for extremely valuable keys [25].

Attacks against RSA

There are a few possible interpretations of "breaking RSA". The most damaging would be for an attacker to discover the private key corresponding to a given public key, this would enable the attacker both to read all messages encrypted with the public key and to forge signatures. The obvious way to do this attack is to factor the public key, *n*, into its two prime factors, *p* and *q*. From *p, q,* and *e,* the public exponent, the attacker can easily get *d*, the private exponent. The hard part is factoring *n*, the security of RSA depends on factoring being difficult. In fact, an easy factoring method would "break" RSA.

It is also possible to attack RSA by *guessing* the value of $(p-1)(q-1)$. This attack is not easier than factoring n.

Another possibility for an attacker would be to try every possible d until the correct one is found. This *brute-force* attack is even less efficient than trying to factor n.

Another way to break RSA is to find a technique to compute *e-the* roots mod *n*. Since

$$c = m^e \bmod n$$

the *e-the* root of *c* mod *n* is the message *m*. This attack would allow some one to recover encrypted messages and forge signatures even without knowing the private key. This attack is not known to be equivalent to factoring. No general methods are currently known that attempt to break RSA in this way.

The simplest single-message attack is the *guessed plaintext attack*. An attacker sees a ciphertext, guesses that the message might be "Attack at dawn" and encrypts this guess with the public key of the recipient, by comparison with the actual ciphertext, the attacker knows whether or not the guess was correct. This attack can be prevented by appending some random bits to the message.

Another single-message attack can occur if some one sends the same message *m* to three others, who each have public exponent *e = 3*. An attacker who knows this and sees the three messages will be able to recover the message *m*.

Let m_1 and m_2 be two plain text messages, and let c_1 and c_2 be their respective RSA encryptions. It then holds that

$$(m_1 m_2)^e \equiv m_1^e m_2^e \equiv c_1 c_2$$

and

$$c = c_1 c_2 \bmod n$$

$$m = m_1 m_2 \bmod n$$

This observation leads to the *adoptive chosen-ciphertext attack* as following,
Eve wants to decrypt ciphertext c, she chooses a random number x, and computes c' such that $c' = cx^e \bmod n$, and sends it for Alice. Alice computes m' for Eve. (under the assumption that Alice will decrypt some messages for Eve)
$m' \equiv (c')^d \equiv c^d (x^e)^d \equiv c^d x \equiv mx$ then Eve can compute $m = m'/x$.
A *cycling attack* would be for an attacker to find an integer k such that $c^{e^k} \equiv c$. If such a k' exist then must be the case that $c^{e^{k-1}} = m$ then by finding k, an attacker can recover the message. If some people have key pairs (e_i, d_i) with the same RSA modulus n, then it would be easy for Eve to recover a message by using *common modulus attack.* Let m be the plaintext that Eve wants to find and assume that Eve knows two ciphertext c_1 and c_2 encrypted by Alice and Bob, $c_1 = m^{e_1} \bmod n$ $c_2 = m^{e_2} \bmod n$ since e_1 and e_2 are relatively prime then Eve can find r and s, such that $re_1 + se_2 = 1$, then she can find m since $c_1^r c_2^s \equiv (m^{e_1})^r (m^{e_2})^s \equiv m^{re_1 + se_2} \equiv m$.

There is even the possibility of taking advantage of hardware faults to attack RSA scheme which is presented at [3].

RSA and Standards

RSA has become a part of many standards around the world. For instance, ISO 9796 [15], ITU (was called CCITT) X.509 [16] digital certification standard. It is included in France’s ETEBAC 5 standard and Australia key management standard, AS2805.6.3, and the ANSI X9.31 [1] draft standard for U.S. banking industry and RSA is a part of the Society for Worldwide Interbank Financial Telecommunications (SWIFT) standard.

RSA is a “de facto” standard in the financial community. The existence of a “de facto” standard is very important to the development of internet commerce. If one public-key system is available everywhere, then signed digital documents can be exchanged among users in many different nations, around the world, using different software on different platforms. If there is an accepted standard for digital signatures, it is possible to have passports, checks, wills, leases, etc. exits in electronic form and a paper version will be nothing more than a copy of the original electronic document.

RSA and Patents

RSA is patented under U.S. Patent 4405829, licensed by Public Key Partners(PKP), issued September 29, 1983 and held by RSA Data Security, Inc. of Redwood City, California, the patent expires 17 years after issue, on September 20, 2000. RSA Data Security usually allows free non-commercial use of RSA, with written permission, for academic or university research purposes.

2.3 Digital Signatures

Digital signatures were first proposed in 1976 by Whitfield Diffie. A digital signature is the electronic equivalent of a hand-written signature. The key aspects of both types of signatures are that only one person or service-provider is capable of producing the signature and all others are capable of verifying it.

A digital signature guarantees that anyone reading a digitally signed message can be certain of who sent it. Digital signatures employ a pair of keys, a private key, used to sign messages and a public key, used to decode them. Only a message signed by the private key can be used, decoded and verified using the public key. In this way one can create a digital signature, equivalent to a hand-written signature on a document.

Since a digital signature is not physically connected to the signed data or the originator, it depends on this data and on the secret key of the originators. Several signature schemes have been proposed. The RSA public-key crypto system can be used for both enciphering and digital signatures. Schemes which can only be used for digital signature purposes are the DSA [26] and the Fiat-Shamir scheme [26].

Assume Bob has received a digitally signed message from Alice. If Alice subsequently denies having sent the message, Bob can go to a third party (a judge), who will be able to obtain Alice’s public key. Subsequently he can verify the validity of the signature. In this way a digital signature can provide non-repudiation of origin. It is easy to see that the digital signature provides in additional data authentication, i.e., data integrity and data origin authentication.

One way to protect against recipients falsely claiming not to have received messages is similar to the way paper mail is certified, messages are only given to recipients once they provide digitally signed “receipts” of delivery. Another method holds people responsible for messages that are made a matter of public record, like legal notices in newspapers. Since, under the new approach, messages are broadcast, they can be certified in this way at little additional expense. When this method is used with messages encoded for confidentiality, either party can display the signed message and point to the corresponding doubly encoded transmission in the public record as evidence that the message was available for receipt, since decoding the signed message with the digital pseudonym of the sender yields the message content, and encoding it with the pseudonym of the recipient yields the transmission in the public record.

Digital signatures in mathematical terms based on RSA encryption

When Alice opens her account at the key-making bank, the bank generated two large prime numbers for her, *p* and *q*. The product of *p* and *q*, is the modulus for all of Alice's exponentiations.

The basis of the two-key system is that:

$$x^{(p-1)(q-1)} = 1 \bmod pq$$

then, the bank chooses values *e* and *d* such that:

$$ed = 1 \bmod (p-1)q-1)$$

Without looking at it, the bank assigns Alice *d*, which is her private key. It keeps a record of *e*, which is Alice's public key. Anything encrypted with *d* can be decrypted with *e*:

$$(x^d)^e = x \text{ (mod } pq)$$

Alice tells her friends her public key, *e* and her modulus *pq*. But she never tells anyone *p, q*, or her private key, *d*.

2.4 Blind Signatures

A blind digital signature is a special kind of digital signature. The difference does not lie in the signature itself, but in the document to which it is attached. When a person places a regular digital signature on a document, he is familiar with the contents of that document. A person placing a blind digital signature, on the other hand, has no or only partial knowledge of the document's contents.
Blind signatures provide the same authentication as digital signatures but do so in a non-identifiable manner. The recipient will be assured of the fact that the transmission is authentic and reliable, but will not know who sent it.

A blind signature works like this: A user brings a document to a notary. The user does not want anyone, including the notary, to know the contents of the document. The user seals the document in an envelope. A portion of the document is visible through the envelope. The notary places a wax seal on the visible portion. The seal is proof of the document's authenticity. When a blind digital signature is used, cryptography techniques replace the envelope and wax seal. The user enciphers the digital document, which is comparable to putting the document in an envelope. The notary places a digital signature on the document in the envelope. When the document is checked for authenticity, the signature is validated.

Using RSA, a blind signature can be implemented in the following way

1. Alice chooses a blinding factor *r* such that gcd(*r, n*) = 1, where (*n, e*) is the bank's public key and (*n,d*) is the private key and she presents her bank with

   $$m' = mr^e (mod \text{ n})$$

   where *m* is her original message.

2. Alice's bank signs it:

   $s' = (m')^d \bmod n = (mr^e)^d \bmod n$

3. Alice divides out the blinding factor:

   $s = s'/r \bmod n$

4. Alice uses

   $s = m^d$ for paying her bills.

Since *r* is random, Alice's bank cannot determine *m*. Therefore, it cannot connect the signing with Alice's payment.
This signature scheme is secure provided that the factoring and root extractions remain difficult. However, regardless of the status of these problems the signature scheme is unconditionally "blind" since *r* is random. The random *r* does not allow the signer to learn about the message. The problem here is if Alice can give an arbitrary message to be signed, then she is enable to mount a chosen message attack.

2.5 Cut-and-choose method

The cut-and-choose protocol was used by Michael Rabin for the first time in 1978. The reason for choosing the name cut-and-choose is its similarity to the classic protocol for dividing anything fairly [26].

1. Alice cuts the thing in half.
2. Bob chooses one of the halves for himself.
3. Alice takes the remaining half.

Alice has to divide fairly in step (1), because she doesn't know which half will Bob choose in step (2).

2.6 Secret Sharing

There are many times when it helps to split up a secret message among a number of different people. Only if all *m* parts are available can the message be reconstructed.
The simplest level of this scheme is to divide the message into *n* pieces, such that any *m* of them can be used to reconstruct the message. This scheme is called the (*m*, *n*)-threshold scheme. There are several other algorithms [26] in secret sharing which will not be discussed in this report.

2.7 Bit commitment protocol

The bit commitment protocol was developed to prevent people from changing answers. For instance you want to prove that you know which party will win the next election. You can write your answer in a file, encrypt the file and give it to your friend. When the election is over, you would give the key to your friend and he would decrypt the file and know if you were lying or not.

The problem is that you could cheat by having two different keys k_1 and k_2 such that each of them would give a different result. Then depending on what the election's result will be you can choose the write key and proving that you knew the result.

Bit commitment protocols are designed to prevent this deception from taking place.

There are three different kinds of bit commitment protocols:

1. *Bit commitment using symmetric cryptography*:
   (1) Bob generate a random bit string, *R*, and sends it to Alice.
   (2) Alice sends her encrypted message consisting Bob's random string back to him.
   $E_K(R,b)$
   (3) Alice sends Bob the key, *K*, when it's time to reveal the message.
   (4) Bob decrypts the message. He checks his random string to verify the bit's validity.

2. *Bit commitment using one-way functions:*
   (1) Alice generates two random bit strings, R_1 and R_2.
   (2) Alice creates a message consisting of R_1, R_2 and the bit she wishes to commit
   (R_1, R_2, b)
   (3) Alice computes the one-way function on the message and sends the results and one of the random strings to Bob.
   $H(R_1, R_2, b),R_1$

   (4) Alice sends Bob the original message and the random strings.see
   (R_1, R_2, b)

   (5) Bob computes the one way function on the message and compares it and R_1, R_2 with value and random string he received in step (3). If they match, the bit is valid.

3. *Bit commitment using Pseudo-Random-sequence Generators:*
   (1) Bob generates a random bit string and sends it to Alice.
   (2) Alice generates a random seed for a pseudo-random-bit generator. Then for every bit in Bob's random bit string, she sends Bob either:
   a- The output of the generator if Bob's bit is 0.
   b- The XOR of output of generator and her bit, if Bob's bit is 1.
   (3) Alice sends Bob her random seed.
   (4) Bob completes step (2) to confirm that Alice wasn't cheating.

2.8 Discrete logarithm problem

This problem is used as the basis for Brands digital cash system.

Let *p* and *q* be the primes and $q|(p-1)$. Let *g* be a generator (see section 2.1). Then given *p*, *q*, *g* and *y*, find [22] the unique integer *a*, $0 \le a \le q-1$, such that

$$g^a \equiv y \bmod p .$$

The discrete log assumption states that there is no polynomial-time algorithm which solves the Discrete Log problem with overwhelming probability of success [6].

2.8.1 Discrete logarithms signature scheme

The idea for this signature is [26] that if someone receives

$$g^a \bmod p$$

where *p* is a large prime number, *g* is a generator, and *a* is an integer, then determining *a* is very hard.

The private key is *a* and public key is

$$y = g^a \bmod p$$

1. Alice wants to sign a document, *m*, she just computes

$$m^a \bmod p .$$

Her signature is a list of

$m^a \bmod p$, *g*, *p* and $g^a \bmod p$.

2. Bob wants to verify her signature. Alice has to create a random number *w* to compute

$$g^w \bmod p$$

and

$$m^w \bmod p$$

and sends them to Bob.

3. Bob generates a random number *c*, as a challenge and sends it to Alice.
4. Alice sends back the response:

$r = ca + w$, (a new *c* and *w* for every verification)

5. Bob computes

$$g^r \ mod\ p$$

and compares it with

$$(g^a)^c \times (g^w) \ mod\ p \ .$$

They should be the same. So should be

$$m^r \ mod\ p \quad \text{and} \quad (m^a)^c \times (m^w) \ mod\ p \ .$$

Proof that signature verification works. If the signature is correct and no one has been cheating then

$$g^{a^c} \times g^w \equiv g^{ca+w} \equiv g^r$$

and

$$m^{a^c} \times m^w \equiv m^{ca+w} \equiv m^r$$

2.9 Schnorr signature scheme

Schnorr signature scheme is based on the difficulty of calculating discrete logarithms [26]. The private key is *a*, $0 \le a \le q-1$, if *p* and *q* are prime numbers such that $q|(p-1)$ *(q* is a prime factor of *(p-1)).* An element *g* is chosen, $1 \le g \le p-1$. The public key is

$$y = g^a \text{ mod } p \quad .$$

The signature protocol is:

Assume that we want to sign message *M*.

1. Alice chooses a random number, *w*, $1 \le w \le q-1$, and computes

$$x = a^w \text{ mod } p$$

She sends it to Bob.

2- Alice concatenates the message, *M*, and *x*, and hashes the result,

$$e = H(M,x) ,$$

H() is a one-way hash function.

3- Alice then computes

$$v = (r + ae) \ mod\ q \ .$$

She sends the signature, *(y, e)*, to Bob.

4- Bob computes

$$x' = a^v y^e \ mod\ p \ .$$

Then he controls that the concatenateion of *M* and x' hashes to *e*,

$$e = H(M,x')$$

If it does, the signature is valid.

Proof that signature verification works. If the signature was created by Alice then

$$x' \equiv a^v y^e \equiv a^v a^{-ae} \equiv a^r \equiv x \text{. And } \quad H(M,x') = e = H(M,x) .$$

2.10 Representation problem in groups of prime order

This problem is a basic concept i Brand's cash system.

The representation problem in group of prime order is given a prime group *G*, a generator tuple $(g_1, g_2, ..., g_k)$, $k \geq 2$, $g_i \in G$ and a fixed element $h \in G$, find a representation of *h* such that

$$h = \Pi_{i=1}^{k} g_i^{a_i}$$

where a_i is an integer.

Finding a representation of *h* is difficult, unless we first choose

$$(g_1, g_2, ..., g_k) \text{ and } (a_1, a_2, ..., a_k)$$

and then calculate *h*.

This assumption is used in Brand's cash system.

2.11 Hash functions

Hash functions play an important role in both Brand's and Okamoto's cash systems. The role of a hash function is to compress its output, in such a way that none can efficiently find two inputs that compress to the same string.

A hash function is a function *h* with at least following properties [22]:

1. *compression*- *h* maps an input *x*, to an output *h(x)*.
2. *ease of computation*- given *h* and an input *x*, it's easy to compute *h(x)*.

It is generally assumed that the algorithmic specification of a hash function is public knowledge.

There are some different classes of hash functions [22], here we just overview some of them which are used in digital cash protocols.

- A *one-way hash function* (OWHF) is a hash function *h* which is computationally infeasible to

  a- find an input *x*, given *y=h(x)*, and b- find any second input which has the same output as any specified input, also given x, find $x' \neq x$ such that $h(x) = h(x')$.

- A *collision resistant hash function* (CRHF) is a hash function *h* which is computationtiontionally infeasible to

  a- find any two distinct inputs x, x' which hash to the same output, such that $h(x) = h(x')$.

  b- find any second input which has the same output as any specified input, also given x, find $x' \neq x$ such that $h(x) = h(x')$.

2.12 Certificate Authority

A Certificate Authority (CA) is an entity that attests to the identity of a person or an organization. Certificate Authority is to bind a public key to the common name of the certificate, and thus assure third parties that some measure of care was taken to ensure that this binding is valid [26].
Certificates are a way of verifying someone's identity. Certificates are the digital equivalent of a driver's license or credit cards. A driver's license identifies someone who can legally drive in a particular country; a credit card identifies someone with a certain amount of credit as a particular bank. Digital certificate can be used to attest to someone's identity over a network.

You could have a ”Certification Authority”. Your Certification Authority (call it ”level one”) would certified by it's Certification Authority (call it ”level two”). The level two Certification Authority would issue all of the certificates for all of the level one Certification Authorities in a larger group. And so on. This creates a formal hierarchy of Certification Authorities (CA). Someone else trying to validate your certificate becomes a process of them trying to find some issuer in your path that they trust. Since there is a formal tree of CA's, it becomes much more likely that they can find a CA they trust. In fact, a CA in their hierarchy may be a CA in your hierarchy. If not, CA's can ”cross certify” each other. In this method, there are multiple hierarchies with no guarantee of a common point. The last case just guarantees a common point in the hierarchy, usually the top. There is one hierarchy (tree), and the top is called the root [22].

Attacks against CA

The certifying authority's key pair might be the target of an extensive cryptoanalytic attack. For this reason, CAs should use long keys, and should also change keys regularly. Top-level certifying authorities (root) need especially long keys, as it may not be practical for them to change keys frequently because the public key may be written into software used by a large number of verifiers.

Another attack can appear when Bob wishes to impersonate Alice. If Bob can convincingly sign messages as Alice, he can send a message to Alice's bank saying ”I wish to withdraw $10,000 from my account. Please send me the money.” To carry out this attack, Bob generates a key pair and sends the public key to a certifying authority saying ”I'm Alice. Here is my public key. Please send me a certificate.” If the CA is fooled and sends him such a certificate, he can then fool the bank, and his attack will succeed.

In order to prevent such an attack, the CA must verify that a certificate request did indeed come from its purported author. The CA may, for example, require Alice to appear in person and show a birth certificate. Some CAs may require very little identification, but the bank should not honour messages authenticated with such low-assurance certificates. Every CA must publicly state its identification requirements and policies [25]; others can then attach an appropriate level of confidence to the certificates.

In another attack, Bob bribes someone who works for the CA to issue to him a certificate in the name of Alice. Now Bob can send messages signed in Alice's name and anyone receiving such a message will believe it is authentic because a full and verifiable certificate chain will accompany the message. This attack can be hindered by requiring the cooperation of two (or more) employees to generate a certificate; the attacker now has to bribe two or more employees rather than one.

2.13 Elliptic Curve Cryptosystem

The use of elliptic curves in public-key systems was first proposed in 1985 by Victor Miller, and independently by Neal Koblitz. The basic idea is to use the group of points on an elliptic curve in existing discrete-log based systems.

Elliptic Curve Cryptosystem offers high efficiency and low overhead for encryption, digital signatures, and key management making it the ideal choice for digital cash transfered by smart cards.

The advantages of Elliptic Curve Cryptosystem include [11]:

- Highest cryptographic strength per bit of any known public-key system.
- Dramatic savings in computation, bandwidth, and storage over other public-key sytems.
- Bandwidth is reduced due to short signatures and certificates.
- Fast encryption and signature speed in both hardware and software.
- Ideal for very small hardware implementation, such as a smart card.
- Encryption and digital signatures stages can be separated for export simplicity.

Principles of Elliptic Curve

An elliptic curve is defined by an equation [20] and has the form:

$$y^2[+xy] = x^3 + ax^2 + b$$

where x and y are variables, a and b are constants.

Notation: The square brackets mean that the term is optional, sometimes it is there, sometimes it isn't.

The crucial property of an elliptic curve is that we can define a rule for “adding” two points which are on the curve, to obtain a 3rd point which is also on the curve. This addition rule satisfies the normal properties of addition.

The equations for the addition rule are:

- If the field is GF(***p***) where *p* is a large prime, the addition rule has the form

  $$(x_1, y_1) + (x_2, y_2) \Rightarrow (x_3, y_3)$$

  where

  $$x_3 = L^2 - x_1 - x_2$$
  $$y_3 = L(x_1 - x_3) - y_1$$
  $$L = (y_1 - y_2)/(x_2 - x_1)$$

  If $x_1 = x_2$ and $y_1 = y_2$ we must use instead,

  $$x_3 = L^2 - 2x_1$$
  $$y_3 = L(x_1 - x_3) - y_1$$
  $$L = (3x_1^2 + a)/(2y_1)$$

- If the field is GF(2^m), the addition rule has the form

  $$(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$$

  where

  $$x_3 = L^2 + L + x_1 + x_2 + a$$
  $$y_3 = L(x_1 + x_3) + x_3 + y_1$$
  $$L = (y_1 + y_2)/(x_1 + x_2)$$

  If $x_1 = x_2$ and $y_1 = y_2$ we must use instead

  $$x_3 = L^2 + L + a$$
  $$y_3 = {x_1}^2 + (L + 1)x_3$$
  $$L = x_1 + y_1/x_1$$

For addition to be well defined for any two points, we need to include an extra ‘zero’ point 0, which does not satisfy the elliptic curve equation. This ‘zero’ point is taken to be a fully paid up point of the curve. The order of the curve is the number of distinct points on the curve, including the zero point.

Having defined addition of two points, we can also define multiplication $k*P$ where *k* is a positive integer and *P* is a point as the sum of *k* copies of *P*.

The security of the elliptic curve is based on the assumption that it is difficult to compute *k* given *F* and $k*F$.

Use of Elliptic Curve in cryptography

Alice, Bob, Cathy, David... agree on a (non-secret) elliptic curve and a (non-secret) fixed curve point *F*. Alice chooses a secret random integer *Ak* which is her secret key, and publishes the curve point $Ap = Ak*F$ as her public key. Bob, Cathy and David do the same.

Now suppose Alice wishes to send a message to Bob. One method is for Alice to simply compute $Ak*Bp$ and use the result as the secret key for a conventional symmetric block cipher (say DES).

Bob can compute the the same number by calculating $Bk*Ap$, since

$$Bk*Ap = Bk*(Ak*F) = (Bk*Ak)^*F = Ak*(Bk*F) = Ak*Bp.$$

The *fixed point, c,* is a point on the curve such that

$$c*F = 0$$

where

$$c = b - a$$

$$a*F = b*F, b > a$$

The least *c* for which this is true is called the order of the point.

For good security, the curve and fixed point are chosen so that the order of the fixed point *F* is a large prime number.

With the above provisions, if the order of the fixed point *F* is an n-bit prime then computing *k* from $k*F$ and *F* takes roughly $2^{(n/2)}$ operations.

For example, if the order of *F* is a 240-bit prime, then an attack would be expected to need 2^{120} operations. This is what makes the use of elliptic curves attractive. It means that public keys and signatures can be much smaller than with RSA for the same predicted security.

3 Simple Anonymous Cash (Chaum-Fiat-Naor)

One of the most important features of electronic cash is the anonymity. A true anonymous cash system can be built by preventing the bank from discovering any identifying information about a bill. The anonymity is presented through the use of blinded signatures RSA-based (see section 2.4) and the cut-and-choose protocol (see section 2.5).

This protocol allows the user to create a bank certified check without letting the bank know what it is signing. If the bank doesn't know what was signed, then it can not recognize it when the check returns for payment, in other words the bank is not able to link a specific withdrawal with a specific deposit [10]. To prevent double-spending, seals the identity in a coin, then if the coin is spent twice, the information in both instances can be used to find the cheater.

The bank publishes a RSA key (e, n) and sets a security parameter k. The bank also publishes two collision-free functions, f and g, f acts as a random oracle and g is a one-to-one function.
Alice has a bank account number u and the bank keeps a counter v associated with it.

3.1 The withdrawal protocol

1. Alice creates *k* sample units, U_i. Each unit is given some random serial numbers, a_i, c_i, d_i, $1 \geq i \geq k$, chosen from a large enough pool to virtually ensure that no other unit will get the same value.

   $U_i = f(x_i, y_i)$, for $1 \geq i \geq k$,

   where

   $$x_i = g(a_i, c_i), \qquad y_i = g(a_i \oplus (u \wedge (v + i)), d_i)$$

   where

   $\oplus$ denotes exclusive or and $\wedge$ denotes concatenation.

2. Alice blinds this *k* units with random blinding factors $\{r_1 \ldots r_k\}$ and sends them to the bank. The blinding factors prevent the bank from checking the contents of the bills immediately.

   $$B_i = r_i^{b_i} U_i \bmod n$$

3. The bank chooses randomly *k/2* units to test.
4. Alice gives the bank r_i, a_i, c_i, d_i, for $1 \geq i \geq k/2$, (here we assumed that the bank chose *i* such that, $1 \geq i \geq k/2$).
5. The bank unblinds *k/2* units and checks to make sure that Alice has not tried to cheat. If there are no mistakes, the bank signs the units left with its private key and gives Alice,

   $$\prod_{k/2 \leq j \leq k} (B_j)^d \bmod n$$

and debits her account.

6. Alice unblinds the signed units by multiplying with inverse r_j^{-1}. After this step Alice has an electronic coin. She increments her copy of the counter *v* by *k*.

$$C = \prod_{k/2 \le j \le k} f(x_j, y_j)^d \bmod n$$

3.2 The payment protocol

1. Alice sends *C* to Bob.
2. Bob chooses a random binary string $z_1, z_2, ..., z_{k/2}$ and sends it to Alice.
3. Alice response as follows, for all $1 \le i \le k/2$

   a- if $z_i = 1$, then Alice sends Bob a_i, c_i and y_i.

   b- if $z_i = 0$ then Alice sends Bob $x_i, a_i \oplus (u \Lambda (v + i))$ and d_i.
4. Bob verifies that C is valid before he accepts Alice payment.

3.3 The deposit protocol

1. Bob sends the payment history to the bank.

2. The bank verifies the bank's digital signature.

3. The bank verifies that coin has not already been spent.

4. The bank enters the coin in spent-coin database and also stores binary string and corresponding response from Alice.(These will be later used to catch double-spender.)

5. The bank credits Bob's account.

3.4 How close to the e-cash features is this protocol?

Security: In this protocol the chance for fraud and the risk of getting caught are determined by k/2, even if k is small and equal to 2, then the odds are still 1 to 1. If the punishment is hard enough, fraud should be nonexistent.

One of the simplest fraudulent acts here is double-spending. To minimize this act, when the bank receives a coin for payment it can record the coin's number in a database, then if the same number appears twice, the bank knows that a fraudulent act has happened. If Alice is the cheater who spends the coin twice, then she is challenged the second time. Since each challenge is a random bit string the new challenge is bound to disagree with the old one in at least one bit. There will be two different sets of identity halves. The chance that one of the identity halves will match the other halve in the other coin is high and Alice's identity will be revealed. The chance that Alice will be asked to reveal the same identity halves in both transactions with the same coin is one out of $2^{k/2}$. The two halves will be revealed and Alice(cheater) will be caught, but if the identity halves don't match each other and don't reveal Alice's identity, then Bob is the cheater.

Privacy: It's true that Alice has to write her identity on every unit she creates, but it's just half of her identity and no one else can recognize her unless they could discover both of identity halves. On the other hand the coins can not be spend several times. They must be returned to the bank after each transaction because of identity revealing protocol. Then the bank knows who withdrew the money and who deposited it, but the bank does not know how they trade in between.

Portablity: It's easy to carry these coins around, they are nothing but digital numbers and some random numbers which Alice needs to use in the payment protocol.

Transferability: This is an untransferable system. These coins can not be spent several times. They must be returned to the bank after each transaction and be deposit by the payee(Bob).

Divisibility: This system doesn't give any alternative to a divisible coin.

3.5 Cost in both money and time

At the withdrawal protocol, Alice has to send K packets to the bank however the bank has to send back just one packet, depending on the size of K we have many packets which have to be transferred, and we have to notice that creating, blinding and again unblinding of K packets will increase computation, communication, storage costs and time which depends on the size of K, which has to be large enough, if we want to minimize the probability of cheating.

At the payment protocol after Alice sends Bob the coin, Bob sends a binary string to Alice as a challenge, then Alice has to send Bob $K/2$ different responses, each of these responses consists of many terms, which will result on increasing the time and computation, communication, storage costs.

3.6 Attacks

Since this system is RSA-based, then all the attacks again RSA (see section 2.2) could appear here too.

A possible attack against this system is a *cooperation attack* between Alice and Eve. If Alice after a payment transaction with Bob, sends her spent coin to Eve with the binary string chosen by Bob and the response to this string, then Eve will have an exact payment history as Bob and the bank will not be able to determine which one of them is cheating.

4 Traceable Anonymous Cash (Ferguson scheme)

This system uses secret sharing techniques (see section 2.6) to catch the cheater. The blind signature used here is called a randomized blind RSA-based signature. In this case the bank still doesn't know what is signing, but it knows that the data wasn't chosen maliciously.
This system needs 3 numbers, C, A and B. These will be of the form

$$C = cg_c^{f(h_c^c)}$$

$$A = ag_a^{f(a)}$$

$$B = bg_b^{f(h_b^b)}$$

Where the numbers g_c, g_a and g_b are publicly known and of the large order in group Z^*_n (see section 2.1). The number h_c and h_b are elements of order n and f is a one-way function. U is the Alice's identity, v is the bank's public key.

Randomized blind signature

For the construction of an efficient withdrawal protocol, a signature scheme with the following properties are required [18]:

- Alice receives an RSA-signature on a number in a special form, which she cannot create herself.
- The bank is sure that the number it signs was randomly chosen.
- The bank receives no information regarding which signature Alice gets.

The randomized blind RSA-based signature protocol

The bank publishes its public key, (v, n) a one-way function, f and a random number $g \in Z_n^*$

1. Alice chooses a random number $a_1 \in Z_n^*$, and two blinding factors, σ and Υ. she computes $\Upsilon^v a_1 g^\sigma$ and sends the result to the bank.
2. The bank chooses its own contribution a_2 and sends this back to Alice.
3. Alice replies with $f(a_1 a_2) - \sigma$.
4. The bank multiplies $\Upsilon^v a_1 g^\sigma$ by a_2 and $g^{f(a_1 a_2)^{-\sigma}}$ to get $\Upsilon^v a_1 a_2 g^{f(a_1 a_2)}$. The bank computes the v'th root of this number and sends it to Alice.
5. Alice divides out the blinding factor Υ to get the pair $(a, (ag^{f(a)})/v)$ The number $a = a_1 a_2$ is called the base number of the signature.

4.1 The withdrawal protocol

The withdrawal protocol consists of three parallel runs of the randomized blind signature scheme.

1. Alice chooses randomly a_1, c_1, b_1.(her contributions to the base numbers) $\sigma, \tau, \varnothing, \alpha, \beta$ and Υ (the blinding factors). Alice computes

   $\Upsilon^v c_1 g_c^{\alpha}$, $\alpha^v a_1 g_a^{\tau}$, $\beta^v b_1 g_b^{\varnothing}$ and sends these numbers to the bank.
2. The bank then chooses its three constrictions to the base numbers c_2, a_2, b_2. And sends $h_c^{a_2}$, $h_b^{b_2}$ and a_2 to Alice. Sending a_2 directly allows Alice to raise one of the resulting signatures to a power she chooses.
3. Alice chooses a random K_1, and computes

   $e_c = f(h_c^{c_1 c_2}) - \sigma$

   $e_b = f(h_b^{b_1 b_2}) - \varnothing$

   $a = (a_1 a_2 f_2(e_c, e_b))^{k_1}$, where f_2 is a one-way function.

   $e_a = (1/k_1) f(a) - \tau$. Alice sends e_a, e_b and e_c to the bank.
4. The bank compute the blinded versions of A, B and C.

   $\bar{C} = \Upsilon^v c_1 g_c^{\sigma} c_2 g_c^{e_c} = \Upsilon^v c g_c^{f(h_c^c)}$, for $c = c_1 c_2$.

   $\bar{\beta} = \beta^v b_1 g_b^{\varnothing} b_2 g_b^{e_b} = \beta^v b g_b^{f(h_b^b)}$, for $b = b_1 b_2$.

   $\bar{A} = \alpha^v a_1 g_a^{\tau} a_2 f_2(e_c, e_b) g_a^{e_a} = \alpha^v a g_a^{(1/k_1) f(a)}$.

   The following relations hold between the blinded numbers and their unblinded values:

   $$\bar{C} = \Upsilon^v C, \quad \bar{A} = \alpha^v A^{1/k_1}, \quad \bar{B} = \beta^v B.$$

   The bank then chooses a random k_2 and sends $c_2, b_2, k_2, (\bar{C}^{k_2}\bar{A})^{1/v}$ and $(\bar{C}^U \bar{B})^{1/v}$ to Alice.
5. Alice now constructs the numbers

   $A = a g_a^{f(a)}$, $B = b g_b^{f(h_b^b)}$, $C = c g_c^{f(h_c^c)}$

   Alice computes the first signature S_a,

   $S_a = ((\bar{C}^{k_2}\bar{A})^{1/v} / \Upsilon^{k_2}\alpha)^{k_1}$ and the second signature S_b,

   $S_b = (\bar{C}^U \bar{B})^{1/v} / (\Upsilon^U \beta)$

   Alice checks that the signatures she received are correct by verifying that

   $S_b^v = C^u B$ and $S_b^v = C^k A$ where $k = k_1 k_2$.

Alice ends up with the following numbers: c, a, b, (the base numbers) k, (the random parameter for the secret sharing line) S_a and S_b (the signatures). These 6 numbers plus the identity U are used as input to the payment protocol.

Alice Bank

$c_1, a_1, b_1 \in Z_n^*$

$\sigma, \tau, \varnothing \in Z_v^*$

$\Upsilon, \alpha, \beta = Z_n^*$

$$\xrightarrow{\Upsilon^v c_1 g_c^{\alpha},\ \alpha^v a_1 g_a^{\tau},\ \beta^v b_1 g_b^{\varnothing}}$$

$c_2, a_2, b_2 \in Z_n^*$

$$\xleftarrow{h_c^{a_2},\ a_2,\ h_b^{b_2}}$$

$k_1 \in Z_v^*$

$e_c = (f(h_c^{c_1 c_2}) - \sigma)$

$e_b = (f(h_b^{b_1 b_2}) - \varnothing)$

$a = (a_1 a_2 f_2(e_c, e_b))^{k_1}$

$e_a = (1/k_1) f(a) - \tau$

$$\xrightarrow{e_c,\ e_a,\ e_b}$$

$\bar{C} = \Upsilon^v c_1 g_c^{\sigma} c_2 g_c^{e_c}$

$\bar{A} = \alpha^v a_1 g_a^{\tau} a_2 f_2(e_c, e_b) g_a^{e_a}$

$\bar{B} = \beta^v b_1 g_b^{\varnothing} b_2 g_b^{e_b}$

$k_2 \in Z_v^*$

$$\xleftarrow{c_2,\ b_2,\ k_2,\ (\bar{C}^{k_2}\bar{A})^{1/v},\ (\bar{C}^{U}\bar{B})^{1/v}}$$

$c = c_1 c_2$

$b = b_1 b_2$

$k = k_1 k_2$

$C = c g_c^{f(h_c^c)}$

$A = a g_a^{f(a)}$

$B = b g_b^{f(h_b^b)}$

$S_a = ((\bar{C}^{k_2}\bar{A})^{1/v} / \Upsilon^{k_2}\alpha)^{k_1}$

$S_b = ((\bar{C}^{U}\bar{B})^{1/v} / (\Upsilon^{U}\beta))$

$S_b^v \overset{?}{=} C^k A$

$S_b^v \overset{?}{=} C^u B$

FIGURE 3. The withdrawal protocol

4.2 The payment protocol

1. Alice sends a, b and c to Bob.
2. Bob replies with a randomly chosen challenge x.
3. Alice responses with $r = kx + U$ and the signature $(C^r A^x B)^{1/v}$ to Bob.
4. Bob verifies the consistency of these two responses and then he accepts the payment.

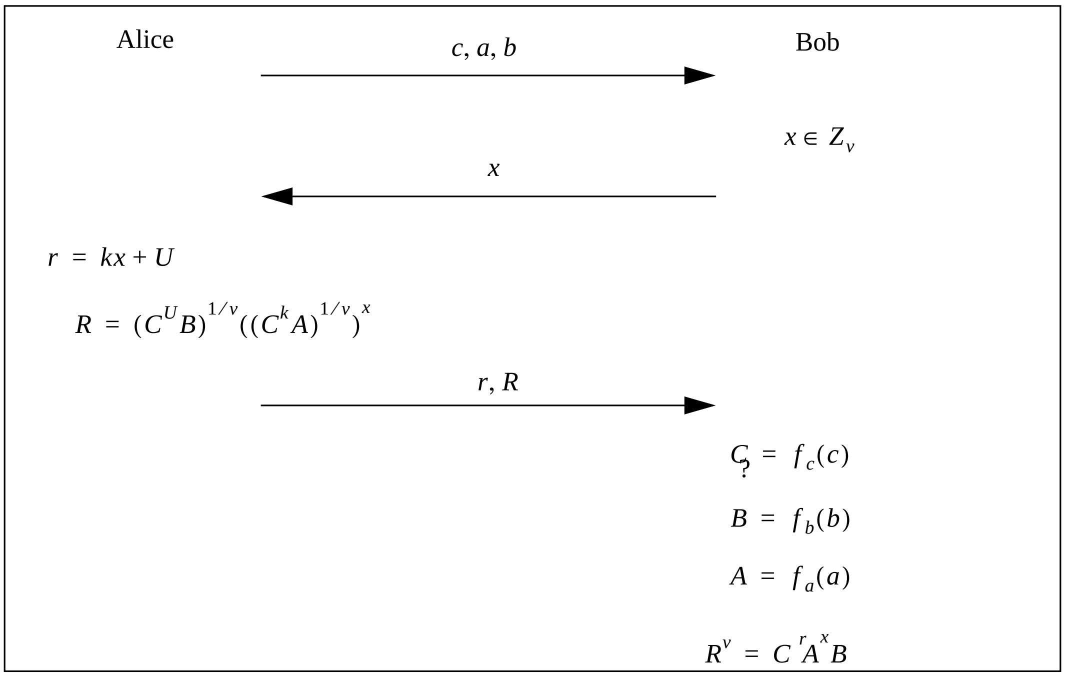


FIGURE 4. The payment protocol

4.3 The deposit protocol

1. Bob sends c, a, b, the challenge and responses to the bank.
2. The bank verifies the correctness of the coin and credits Bob with corresponding amount. (amount of the coin depends on the public key v)

4.4 How close to the e-cash features is this protocol?

Security: To prevent double-spending this system forces Alice to reveal a line based on her identity as the response part of the payment protocol. Then if Alice spends the same coin twice, she must reveal two different lines, $Ux + k$, which allows the bank to determine her identity *U*. This system uses randomized blind signature this allows the bank to control that the signatures are correctly chosen.

***Privacy*:** In this system the bank receives no information regarding which signature Alice gets, because of using randomized blind signature then the bank cannot link the coins withdrawaled by Alice to her identity, unless she spends them twice.

***Portablility*:** Each coin consists of 3 numbers plus two signatures and can be stored in about 250 bytes [18]. It appears that the possibility to have a portable coin exists in this system.

Transferability: The system is transferable referring to [8] but not a secret one because coins can float around many times, and every time a copy is made, there is a greater chance that fraud could appear. A transferable alternative for this cash system has the disadvantage that a coin grows in size with every transfer, since a transferred coin must contain the identity of all users that owned it so that double-spenders can be identified.

Divisibility: This system doesn’t give any alternative to a divisible coin.

4.5 Cost in both money and time

In this system the withdrawal protocol consists of three parallel runs of the randomized blind signature, each of them costs computation time. The communication cost for the withdrawal protocol is four move, three move for the payment protocol and one move for the deposit protocol.

4.6 Attacks

Since this system is RSA-based then all the attacks corresponding to an RSA scheme could occur here too.

A possible attack against this system is a *cooperation* between Alice and Eve. If Alice after a payment transaction with Bob, sends her spent coin to Eve and tells her the binary string chosen by Bob and the response to this string, then Eve will have an exact payment history as Bob and the bank will not be able to determine which one of them is cheating. To prevent this attack Ferguson suggest choosing the challenge as a hash value on the coin and the shop’s identity [18].

5 Brands scheme

This system is built on two concepts the Schnorr digital signature (see section 2.9) and the representation problem in groups of prime order (see section 2.10).

The bank publishes
p, *q*, two primes such that $q|(p-1)$,
(*g*, *h*), which is the bank's public key,
(g_1, g_2), a generator-tuple[6], (Note $g, g_1, g_2 \in G_q$)
d, a dummy generator, which is to ensure that the identity of honest users cannot be computed from an execution of the payment protocol.
Two hash functions.

The bank private key is $x \in Z_q$ such that:

$$x = \mathrm{Log}_g^{\,h}, \ (h = g^x)$$

5.1 Opening account protocol

1. Alice generates at random numbers $u_1, u_2 \in Z_q$ and computes $I = g_1^{u_1} g_2^{u_2}$. Then she sends *I* to the bank.
2. The bank stores *I* together with Alice's identity and account number.

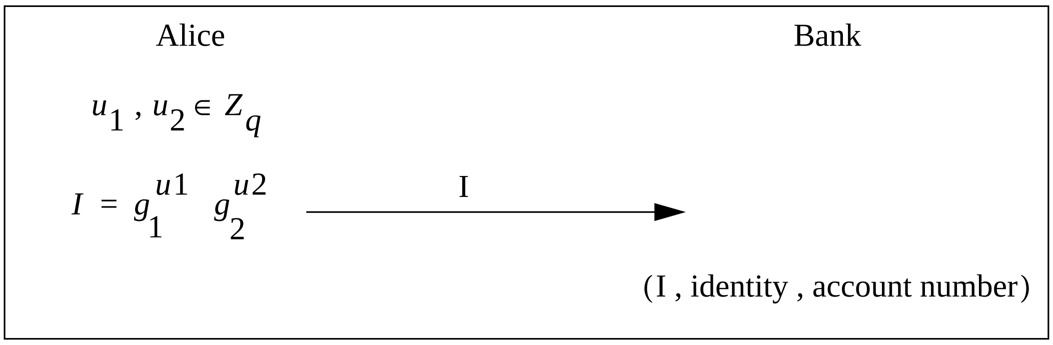


FIGURE 5. Opening account protocol

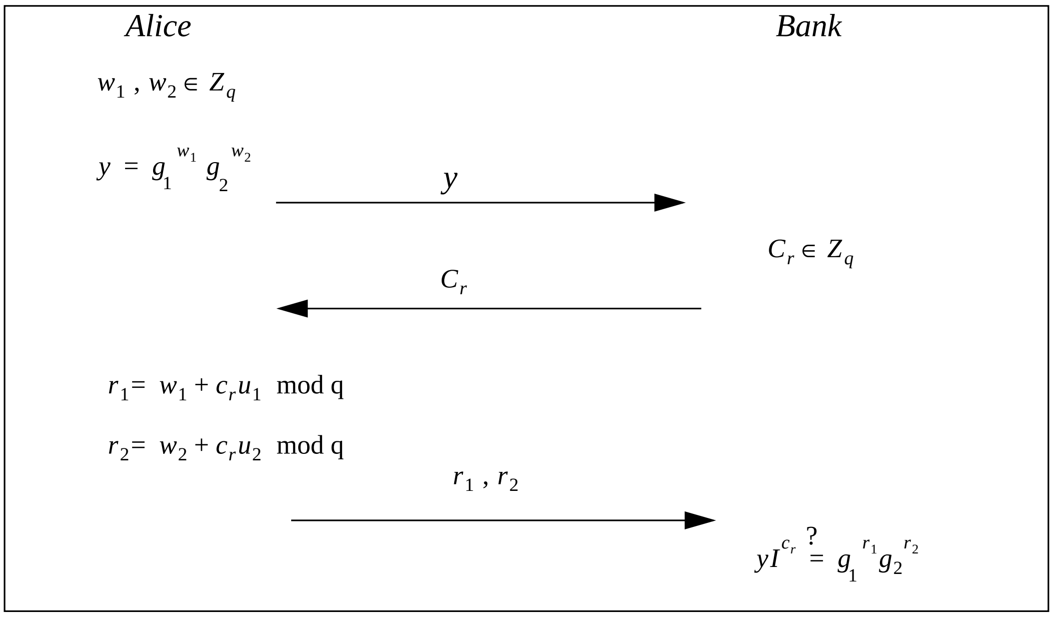
The representation problem

Alice wants to withdraw a coin from her account corresponding to *I*. She first must prove to the bank that she is the owner. This is done using the representation problem (see section 2.10) as following:

1. Alice generates at random two numbers $w_1, w_2 \in Z_q$, and sends $y = g_1^{w_1} g_2^{w_2}$ to the bank.
2. Bank generates at random a challenge $C_r \in Z_q$ and sends it to Alice.

3. Alice computes the responses $r_1 = w_1 + c_r u_1 \mod q$, $r_2 = w_2 + c_r u_2 \mod q$ and sends them to the bank.
4. The bank accepts it if and only if $yI^{c_r} = g_1^{r_1} g_2^{r_2}$. See figure 6.

Proof of correctness: If Alice is the owner, then she must know u_1, u_2 and if she knows them then r_1 and r_2 are correct which means $yI^{c_r} \equiv g_1^{w_1} g_2^{w_2} \times (g_1^{u_1} g_2^{u_2})^{c_r} \equiv g_1^{w_1 + u_1 c_r} g_2^{w_2 + u_2 c_r} = g_1^{r_1} g_2^{r_2}$ □



Alice
Bank
$w_1, w_2 \in Z_q$
$y = g_1^{w_1} g_2^{w_2}$
y
$C_r \in Z_q$
C_r
$r_1 = w_1 + c_r u_1 \mod q$
$r_2 = w_2 + c_r u_2 \mod q$
r_1, r_2
$yI^{c_r} \overset{?}{=} g_1^{r_1} g_2^{r_2}$

FIGURE 6. Alice proves that she owns I by proofing knowledge of (u_1, u_2)

5.2 The withdrawal protocol

If the bank accepts Alice's proof then the withdrawal protocol is executed:

1. Both Alice and bank forms $m = \mathrm{Id}$. The bank sends $z = m^x$, $a = g^w$ and $b = m^w$ (*w* is randomly chosen from Z_q) to Alice.
2. Alice chooses three random numbers $s \in Z_q^*$, $u, v \in Z_q$ to blinds *m*, *z*, *a* and *b*,

   $m' = m^s$, $z' = z^s$, $a' = a^u g^v$, $b' = b^{su} m^{sv}$.

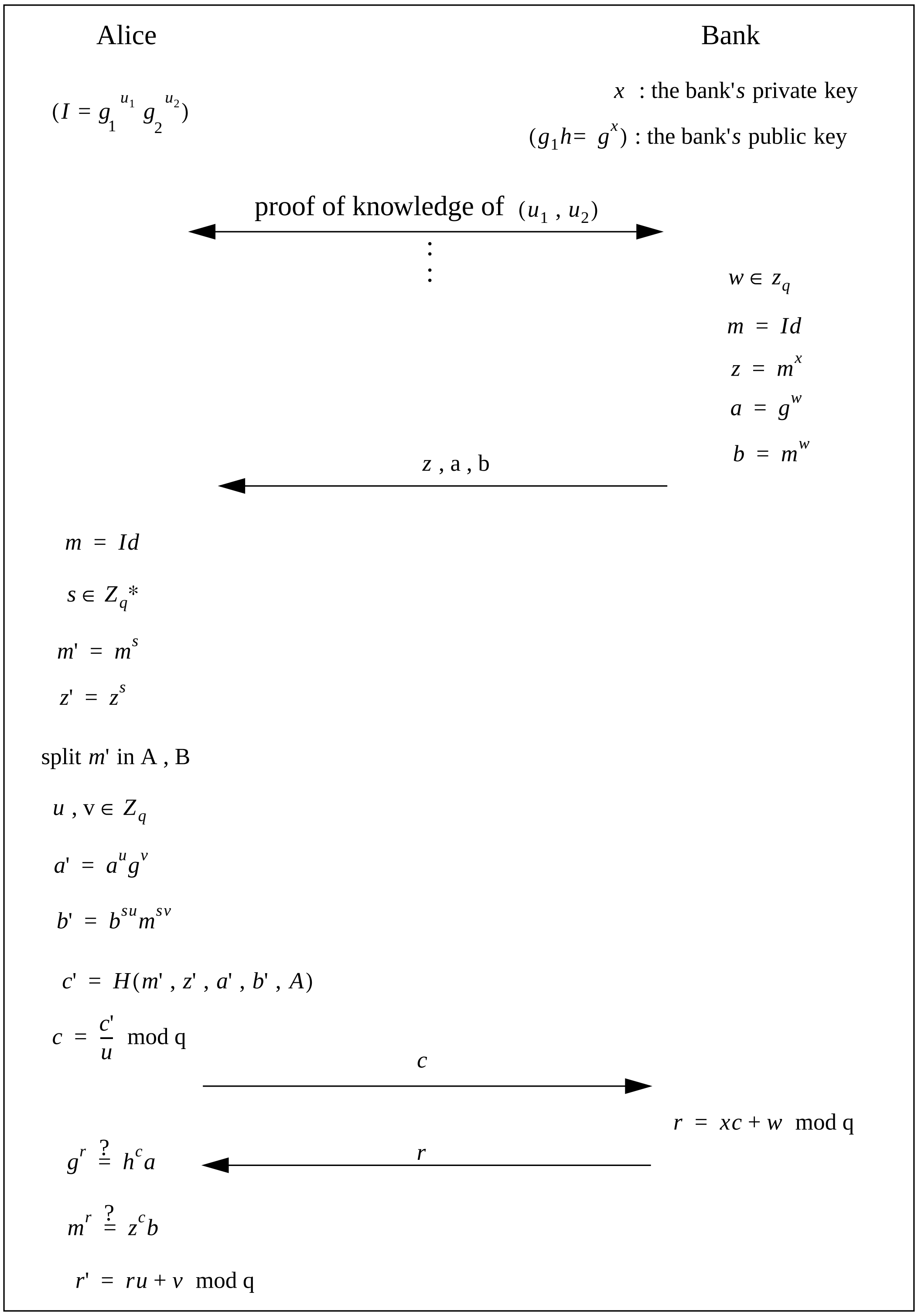
   Then she splits her blinded identity $u_1 s$ and $u_2 s$ in two parts,

   $u_1 s = x_1 + x_2 \mod q$ and $u_2 s = y_1 + y_2 \mod q$

   and even the random number, *s*, splits in part $s = z_1 + z_2 \mod q$ .Then she computes $A = g_1^{x_1} g_2^{y_1} d^{z_1}$ and $B = g_1^{x_2} g_2^{y_2} d^{z_2}$.

   Notice that here $m' = AB$.

3. Alice computes a challenge c', by using the hash function H. $c' = H(m', z', a', b', A)$ and blinds it $c = \frac{c'}{u} \bmod q$. She sends the challenge to the bank.



Alice
Bank
$(I = g_1^{u_1} g_2^{u_2})$
x : the bank's private key
$(g_1 h = g^x)$: the bank's public key
proof of knowledge of (u_1, u_2)
$w \in z_q$
$m = Id$
$z = m^x$
$a = g^w$
$b = m^w$
z , a , b
$m = Id$
$s \in Z_q^*$
$m' = m^s$
$z' = z^s$
split m' in A , B
u , v $\in Z_q$
$a' = a^u g^v$
$b' = b^{su} m^{sv}$
$c' = H(m', z', a', b', A)$
$c = \frac{c'}{u} \bmod q$
c
$r = xc + w \bmod q$
r
$g^r \overset{?}{=} h^c a$
$m^r \overset{?}{=} z^c b$
$r' = ru + v \bmod q$

FIGURE 7. Withdrawal protocol for coins

4. The bank sends back the response, *r* to Alice and debits the amount of money from her account.

   $r = xc + w \mod q$.

Alice accepts if and only if $g^r = h^c a$ and $m^r = z^c b$. Then she computes

$r' = ru + v \mod q$. (See figure 7)

Now Alice has her coin, a triple of A, B and sign (A, B):

$(A, B, \text{sign}(A, B))$ such that $(\text{sign}(A, B) = (z', a', b', r'))$.

But how do we know the value of a coin? There are two different ways to donate the coins value. The most obvious of these is to have the bank use a different public key for each denomination. That is, if there are to be *k* distinct coins, the bank publishes $(g_1, h_1) \ldots (g_k, h_k)$ as it's public key. (This is similar to the way paper cash is designed). An other way is to have *k* different dummy generators $d_1, \ldots, d_k$ published by the bank. Each of these generator is used to denote some fixed amount of money. We can have for example d_i denote for $\$2^{i-1}$.

5.3 The payment protocol

1. Alice wants to buy something from Bob. She sends her coin $(A, B, sign(A, B))$ to Bob.

2. Bob first verifies that $AB \neq 1$. (Note that if $AB = 1$, then *s* was zero, which is not allowed since u_1, u_2 will then not be revealed when the user double-spends.) Then he checks that the bank's signature, $sign(A, B)$ is valid. If it is, he challenges Alice with $c \in Z_q^*$.

   The challenge, *c*, doesn't need to be randomly chosen but it must be unique for each payment, therefore the bank can specifies to whom the payment is made.

   $c = H_0(A, B, I_b, \text{date/time})$, where *I* is Bob's identification and date/time is a time stamp for the transaction.

3. Alice responses with

   $r_1 = x_1 + cx_2 \mod q$, $r_2 = y_1 + cy_2 \mod q$, $r_3 = z_1 + cz_2 \mod q$

4. Bob verifies whether $g_1^{r_1} g_2^{r_2} d^{r_3} = AB^c$, and, if so he accepts the payment.

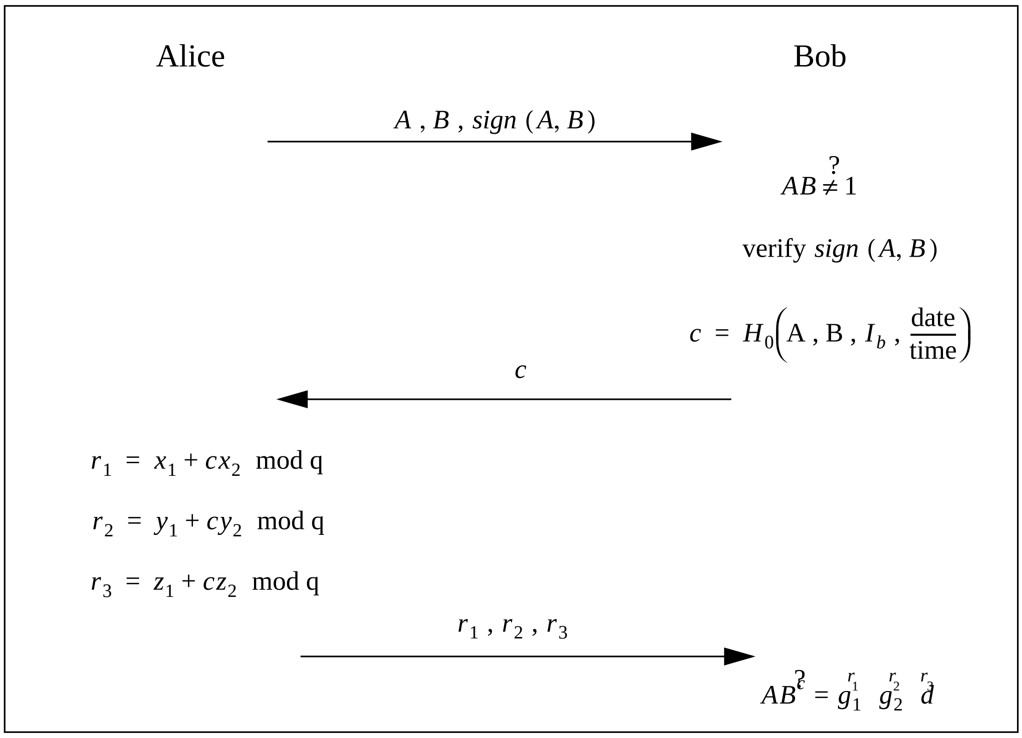

Alice
Bob
$A, B, sign(A, B)$
$AB \overset{?}{\neq} 1$
verify $sign(A, B)$
$c = H_0\left(A, B, I_b, \frac{\text{date}}{\text{time}}\right)$
c
$r_1 = x_1 + cx_2 \mod q$
$r_2 = y_1 + cy_2 \mod q$
$r_3 = z_1 + cz_2 \mod q$
r_1, r_2, r_3
$AB^c \overset{?}{=} g_1^{r_1} g_2^{r_2} d^{r_3}$

FIGURE 8. The payment protocol

5.4 The deposit protocol

1. Bob sends a transcript of the payment consisting of A, B, $sign(A, B)$, time stamp of transaction, c, r_1, r_2 and r_3 to the bank.
2. The bank checks that the signature is correct, the coin has not been spent before, and that Bob's challenge $c = H_0(A, B, I_b, \text{date}/time)$ and Alice's responses are r_1, r_2, r_3 valid. If they are, the bank will pay Bob.

5.5 How close to the e-cash features is this protocol?

Security: To prevent multispending, this algorithm forces the customer to reveal enough information without disclosing her identity. The identity is hidden by drawing lines through a point in space. Each time the coin is spent the payer has to reveal her identity by giving one line through the point representing the identity. If the coin is spent twice, the lines will cut each other at this point, uncovering the identity.

It means that if Alice wants to spend a coin twice, then she will be forced into the payment protocol (see figure 8) and be requested to provide a response to different challenges, c, c', which means she has to responds with both triples

$r_1 = x_1 + cx_2 \mod q$, $r_2 = y_1 + cy_2 \mod q$, $r_3 = z_1 + cz_2 \mod q$

and

$r'_1 = x_1 + c'x_2 \mod q$, $r'_2 = y_1 + c'y_2 \mod q$, $r'_3 = z_1 + c'z_2 \mod q$.

Two such triples will reveal her identity, at deposit time, by the bank. Hence,

$$AB^c = g_1^{r_1} g_2^{r_2} d^{r_3}, \; AB^{c'} = g_1^{r'_1} g_2^{r'_2} d^{r'_3}$$

$$A = g_1^{\frac{(c'r_1 - cr'_1)}{(c' - c)}} \quad A = g_2^{\frac{(c'r_2 - cr'_2)}{(c' - c)}} \quad A = d^{\frac{(c'r_3 - cr'_3)}{(c' - c)}}$$

$$B = g_1^{\frac{(r_1 - r'_1)}{(c - c')}} \quad B = g_2^{\frac{(r_2 - r'_2)}{(c - c')}} \quad B = d^{\frac{(r_3 - r'_3)}{(c - c')}} .$$

We know that

$$A = g_1^{x_1} g_2^{y_1} d^{z_1}$$

$$B = g_1^{x_2} g_2^{y_2} d^{z_2}$$

Since it must be one representation respect to (g_1, g_2, d) then it must be that:

$$\frac{(c'r_1 - cr'_1)}{(c' - c)} = x_1 \mod q$$

$$\frac{(c'r_2 - cr'_2)}{(c' - c)} = y_1 \mod q$$

$$\frac{(c'r_3 - cr'_3)}{(c' - c)} = z_1 \mod q$$

$$\frac{(r_1 - r'_1)}{(c - c')} = x_2 \mod q$$

$$\frac{(r_2 - r'_2)}{(c - c')} = y_2 \mod q$$

$$\frac{(r_3 - r'_3)}{(c - c')} = z_2 \mod q$$

Then the bank can compute

$$u_1 s = x_1 + x_2$$

$$u_2 s = y_1 + y_2$$

$$s = z_1 + z_2$$

and also u_1 and u_2. Then the bank can compute $I = g_1^{u_1} g_2^{u_1}$ and find the double spender.

To forge a coin is a difficult task, because Eve not only has to know the triple $A, B, sign(A, B)$ (which she could find out for example by applying a meet-in-the-middle attack), she must also know a representation of A and B.

Privacy: Alice identity is hidden all the time for both the bank and Bob unless she cheats, then her identity will be revealed to the bank. If Alice likes to keep her privacy, it’s better for her to not cheat.

Portablility: The format and existence of coins doesn’t depend on where they are or on what kind of storage device. Furthermore, Brand’s cash system can be used as a check [6] or in the wallet with observers [5].

Stefan Brand presents a variant of his cash system [18] which can be used for transfers by smart cards.

Transferability: We cannot use this kind of coins over and over again because the ability to catch multi-spending cheaters will be lost.

Divisibility*:* This system does not allow the subdivision of coins to be used in smaller pieces. However, Brand proposes an alternative to make the coins divisible. The idea is simple, a divisible coin can be viewed as a check, the unspent parts of which can be spent until the total amount of money is reached [6].

To achieve divisibility, instead of having one identity part and several denominations, we build a coin as a product of *k* denomination terms, each having its own identity part. In payment protocol, Alice releases two point and a line just for the denomination that she wishes to spend, if she releases the same denomination twice, her identity will be revealed. The side effect here is the ability to link all payments made with the same coin, which translates to Alice losing her privacy and one of the major properties of digital cash system.

5.6 Attacks

There is a kind of attack on Brand's scheme [7]. Alice can spend a coin many times without being identified by misbehaving in the opening account protocol.

We will explain this attack for $I = g_1^{u_1}$, the other cases will be the generalizing of this case.

When opening an account, Alice can instead of using $I = g_1^{u_1}$ choose $I = g_1^{u_1} g^{u_2}$, for random u_1, u_2 continue with the withdrawal protocol to end up with a signature on $A = (g_1^{u_1} g^{u_2 + 1})^s$ and $B = g_1^{x_1} g^{x_2}$. Alice is able to do this since she knows z_1 and the representation of *I* with respect to (g_1, g).

Sign (A, B) is

$$(Z' = (g_1^{u_1} g^{u_2 + 1})^{xs}, a' = g^{wu + v}, b' = (g_1^{u_1} g^{u_2 + 1})^{wsu + sv},$$
$$r' = H, (A, B, Z', a', b')x + wu + v).$$

At the payment protocol two different responses obtained by Alice will be:

$e_1 = de + x_1$ and $r_2 = df + x_2$,

where (e, f) is the representation of $A = g_1^e g^f$, $e = u_1 s$, $f = (u_2 + 1)s$.

The above leave the bank with two equations and three unknown variables u_1, u_2, s.

(Notice we can not extract s from the equation by using, $\gcd(e, f)$, because we are working with modulo q.)

5.7 Cost in both money and time

According to the communication complexity of the withdrawal and payment protocol each consisting of three moves, three moves for the representation problem, and the deposit protocol requires one move. The above shows a total communication complexity for the system of ten moves. (Note: each moves will cost not only money but also time.)

5.8 Disadvantages

Brand’s system is quite difficult to understand because of it’s complex use of mathematics, which maybe one of the reasons why it is not as popular as Chaum’s system. However, it is closer to Digital Cash features than Chaum’s.

The signatures used are larger than in the other Digital Cash protocols, because of the discrete logarithm signature scheme.

5.9 Advantages

This system has a major advantage since it does not use any cut-and-choose protocol or secret sharing therefore Alice isn’t required to create *k* different bill nor keep many copies of identity split into two. The bank is not required to verify the bill by taking a part (*k-1*) of them.

The security of Brand’s system is based on the difficulty in computing discrete logarithms rather than RSA-roots, therefore the ability of factoring prime numbers does not affect the security of this system.

Brand uses hash functions to counter the problem of message protection over an untrusted media like Internet. Messages transmitted between two users need to be authenticated and digital signatures can be used (see section 2.3), however it cost time and is not efficient if we want to use digital signatures for every message transmission. An other solution for message authenticity is to hash each message before sending to enable the receiver to recompute the hash and verify the messages.

5.10 Implementation aspects of Brands system

In this section, we will discuss implementation features and the basic problems accord ing to the Brand’s digital cash system.

Problems according to the generators

We know that the bank must publish generators $g_1, g_2 \in G$. These two generators will be used for representation of an account, it means if Alice owns an account say, *I*, then she should know a representation of *I* such that $I = g_1^{u_1}, g_2^{u_2}$, $u_1, u_2 \in Z$.

The numbers g_1 and g_2 should be huge enough to make it impossible for Eve to guess a representation for Alice’s account and at the same time, it must be easy both for the bank and Alice to compute the different steps of representation problem. On the other hand g_1 and g_2 must be chosen in a fashion that will make it possible to find enough pairs of (u_1, u_2) to represent all the customers who want to open an account at the bank.

Even when we choose g_1 and g_2 good enough to fill the mentioned conditions, another question will come up and that is how long can we keep the same generators.

We know that by using the same g_1 and g_2 over and over again, the chances for Eve to guess a representation of I will rise, we will increase the security of our system if we change the generators once in a period of n months (assume $n = 36$), but then we have to in some way mark the customers who opened their accounts by using say our first pair of generators, and we have to mark their coins, because the shop needs to know which pair of generators they used before he can accept the payment. The bank also needs to recognize the coin to be able to find Alice's identity in the case of double spending.

One solution to this problem could be to ask all of customers to reopen their account with the new pair of generators and to change their unspent coins. This solution will not only cost lots of money and time for the bank but also will not be an appreciated solution for the bank's customers. Then the problem with generators is still unsolved.

Problems according to the identofication numbers

Another question in the set up protocol is, how could Alice find two integers u_1 and u_2 to compute her identification number.

One solution is to let Alice to choose two random numbers or we could let a random function do this job for Alice. So far it looks like that every thing will work great, but note $u_1, u_2 \in Z_q$ and Brands suggest [6] that the length of q would be at least 140 bits, meaning that the length of u_1 and u_2 should be in average 70 bits. Now the problem is how Alice could remember u_1 and u_2. She must store these integers somewhere. Suppose that she stores them on her computer hard disk or just on a floppy disk, this is not a secure solution since Eve can find Alice's u_1 and u_2 and pretend to be her then she can spend all Alice's money. Another solution would be to store u_1 and u_2 on a smart card, but then we must solve the problems which appears with use of a smart card like how could Alice use u_1 and u_2 when she needs them? Should she own a card reader?

6 Divisible Electronic Cash

Digital cash systems described so far in this report do not have the ability to pass the coin along several people or be split into smaller sub parts. Okamoto offers the solutions to these problems [23]. His system is based on a bit commitment scheme (see section 2.7), the square root modulo *N*, and the binary representation tree. The security of this system comes from the difficulty of factorising (because the system is RSA-based) and hash functions.

6.1 The Binary Representation Tree

In this cash system, the binary representation tree plays an important role since it allows the electronic coin *C* to be subdivided into many pieces to allow each subdivided piece to be worth any desired value less than *C* and the total value of pieces equal to *C* [24].

The tree is arranged in a way that two nodes in the tree will reveal the secret identity if one of the nodes is directly related to the other. This hierarchy allows a coin to be split a minimum number of times so the change is kept small. Figure 9 shows a tree structure with a coin of $100 and how this system can be used to generate change.

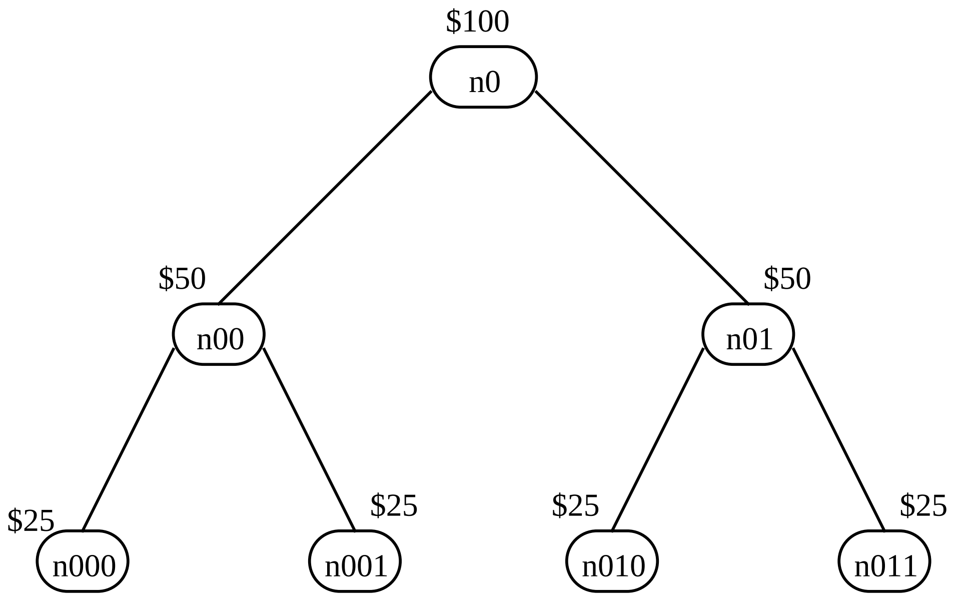


FIGURE 9. A table with a coin worth $100

Okamoto gives following rules to the usage of the coin related to the tree[23]:

1. **(Route node rule)** When a node is used, no one of his child and forefather can be used.
2. **(Same node rules)** No node can be used more than once.

The above rules guarantee that a user's identity will be revealed if she over-spends. (spends more than the total value of the coin.)

More generally, if Alice wants to use a coin worth \$1000 by the cent, she would need a tree of *17* levels because $\log_2 100000 \sim 16.5$.

6.2 The opening account protocol

1. The bank publishes its public's key (n_1, k), (n_2, k) and (a_1, a_2) and sends (alternative publishes) a prime ρ and a generator g, $(g \in z_p{}^*)$.
2. Alice selects two random prime numbers *p* and *q* such that:

   $$p = 3 \bmod 8\ ,\ |p| < k\ ,\ p > (1/4)\ \mathrm{n}_1^{\frac{1}{2}}$$

   $$q = 7 \bmod 8\ ,\ |q| < k\ ,\ q > (1/2)\ \mathrm{n}_1^{\frac{1}{2}}$$

   And sends *x* and *y* to the bank.

   $$x = g^p \bmod \mathrm{p}$$

   $$y = g^q \bmod \mathrm{p}$$

   she also sends s_1 and s_2 to the bank.

   $$S_i = (N + a_i) r_i^k \bmod n_i\ ,\ (i = 1/2)$$

   Where *N* is a Williams integer (see section 2.1).

   $$N = pq$$

3. Then after at Alice has been proved for the bank that $(s_1\ ,\ s_2\ ,\ \mathrm{x}\ ,\ \mathrm{y})$ is honestly generated, the bank sends δ_1 and δ_2 to Alice.

   $$\delta_i = s_i^{1/k} \bmod n_i\quad (i = 1/2)$$

4. Alice computes

   $$L_i = (N + a_i)^{1/k} \quad \bmod \quad n_i = \frac{\delta_i}{s_i} \bmod\ n_i\ \ (i = 1/2)$$

   Now Alice has her electronic license $= (\mathrm{N}, L_1, L_2)$, where

   $$N = pq\quad ,\ L_1 = (N + a_1)^{1/k} \bmod n_1\ ,\ L_2 = (N + a_2)^{1/k} \bmod n_2$$

6.3 The withdrawal protocol

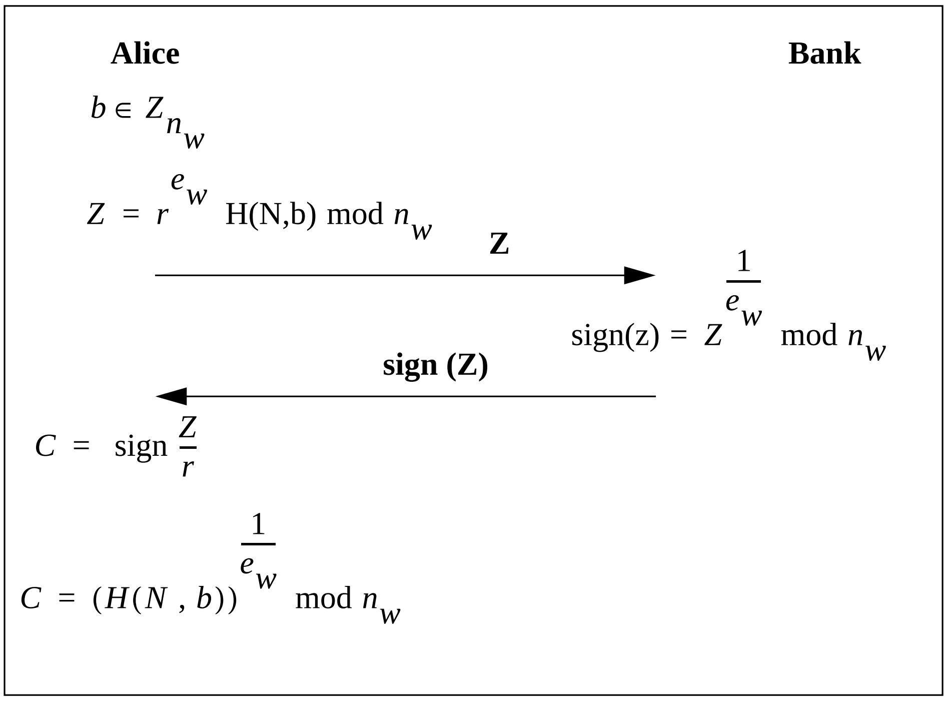
The withdrawal protocol is very simple, it's consists of an RSA blind signature on *N* (a part of Alice's license) concatenated with a random number, *b*, guaranteeing coin uniqueness.

Assume Alice wishes to withdraw a coin worth $w = 2^l$ dollars, (see figure 10). The bank has a public key of RSA signatures $(e_w\ ,\ n_w)$, which corresponds to $w = 2^l$ dollars (The bank has different public keys corresponded to different size of bill).

1. Alice chooses a random value, *b,* then forms and sends *z* to the bank.

$$z = r^{e_w}\ \mathrm{H(N,b)} \mod n_w,$$

where $r \in Z^{n_w}$ is a random integer, blinding factor and *H* is a one-way hash function acting like a random oracle.



Alice
Bank
$b \in Z_{n_w}$
$Z = r^{e_w}\ \mathrm{H(N,b)} \mod n_w$
Z
$\mathrm{sign(z)} = Z^{\frac{1}{e_w}} \mod n_w$
sign (Z)
$C = \mathrm{sign}\ \frac{Z}{r}$
$C = (H(N, b))^{\frac{1}{e_w}} \mod n_w$

FIGURE 10. The withdrawal protocol

2. The bank signs *Z* and charges Alice's account $*w*. The bank sends sign (Z) to Alice.

$$\mathrm{sign(Z)} = z^{1/e_w} \mod n_w$$

3. Alice now can unblinds sign(Z) and use it as her coin *C*.

$$C = (H(N, b))^{1/e_w} \mod n_w$$

6.4 The payment protocol

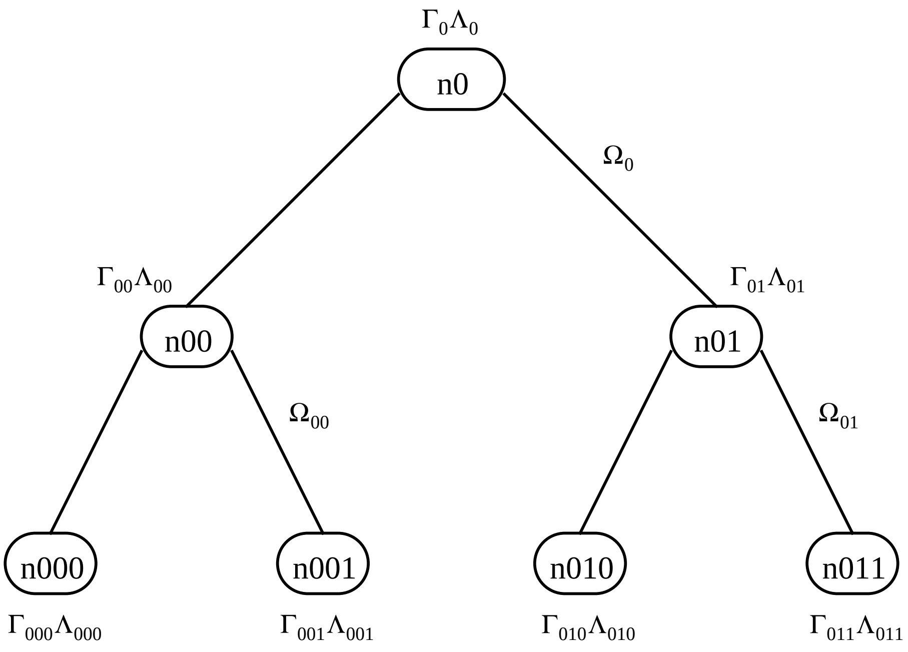
The payment protocol is presented in two parts: Coin authentication and denomination revelation.

During coin authentication, Bob verifies that the coin is legitimate.

1. Alice sends (L_1, L_2), N, (C,b) and *w* to Bob.
2. Bob controls the correctness of the electronic license by checking: $L_i^k \equiv N + a_i \mod n_i, (i = 1, 2)$
3. Bob verifies the bank's RSA signature on the coin: $C^{e_w} \equiv \mathrm{H(N,b)} \mod n_w$.
4. Bob checks that $\mathrm{J(-1, N)} = 1$ and $\mathrm{J(2, N)} = -1$

During **denomination revelation** Alice reveals some information specific to the nodes of the coin's tree that are being spent. We assume that f_Γ, f_Ω, f_Λ and H_0 are truly random functions presented by the bank and we assume that Alice wants to spend \$y of her \$$w$ $(= 2^l)$ coin. Let $[y_1, y_2, \ldots, y_{l+1}]$, $y_i \in \{0, 1\}$ for all $i \in \{1, \ldots, \mathrm{L+1}\}$, be the binary representation of *y*, then if $y_{L+2-t} = 1$ for some $1 \le t \le L+1$, Alice selects a node $n_{j, j_2 \ldots j_t}$ among the nodes in t-th level of the binary tree that does not violate the second rules (see 6.1). Here, Alice has memorized the nodes that have already been spent. The average number of nodes to be spend per payment are $(L+1)/2$, since, the binary representation of *y* contains $(L+1)/2$ bits on the average that are set to 1.

The revelation consists of two parts, one for the second rule and one for the third rule (see 6.1). Three types of values are introduced, Γ, Ω and Λ, these are used for the realization of the second rule (Γ, Ω) and third rule (Λ). (See figure 11)



$\Gamma_0\Lambda_0$
n0
Ω_0
$\Gamma_{00}\Lambda_{00}$
$\Gamma_{01}\Lambda_{01}$
n00
n01
Ω_{00}
Ω_{01}
n000
n001
n010
n011
$\Gamma_{000}\Lambda_{000}$
$\Gamma_{001}\Lambda_{001}$
$\Gamma_{010}\Lambda_{010}$
$\Gamma_{011}\Lambda_{011}$

FIGURE 11. Use of Γ, Ω and Λ values in the binary tree.

Notation: For each $a \in Z_N^*$, $<\mathrm{a}>_{QR}$ denotes the element in {a, -a, 2a, -2a}. $<\mathrm{a}>_1$ is the element in {a, 2a} which has the Jacobi symbol of 1. $[a^{1/2}]_{-1}$ is the square root of *a*, for which $(y/N) = -1$ and $0<\mathrm{y}<N_2$. $[a^{1/2}]_{-1}$ is similarly defined.

1. **(Realizing the second rule**) Alice computes $\Gamma_{j_1 \ldots j_t}$ and sends it to Bob
   $\Gamma_{j_1 \ldots j_t} = \left[\left(<\left(\Omega_{j_1 \ldots j_{t-1}}\right)^{2^{t-1} j_t}\left((\Omega_{j_1 \ldots j_{t-2}})^{2^{t-2} jt-1} \ \ldots (\Omega_{j1})^{2_{j2}} \times\right.\right.\right.$
   $\Gamma_{j_1 \ldots j_t} = [(<(\Omega_{j_1 \ldots j_{t-1}})^{2^{t-1} j_t}(\Omega_{j_1 \ldots j_{t-2}})^{2^{t-2} jt-1} \ \ldots (\Omega_{j1})^{2_{j2}} \ \times$
   $f_\Gamma(\mathrm{C}, 0, \mathrm{N}) > QR)^{1/2^t} \ \bmod \mathrm{N} \ ]_{-1}$, where
   $\Omega_{j_1 \ldots j_i} = \ < f_\Omega(C, j_1 \ldots j_i(, \mathrm{N})>_1$, (i = 1 , . . . , t-1)
2. Bob computes $\Omega_{j_1 \ldots j_i}$ when $j_{j+1} = 1$, for $i \in \{1, \ldots, \text{t-1}\}$.Then Bob verifies the validity of $\Gamma_{j_1 \ldots j_t}$, by checking that: $(\Gamma_{j_1 \ldots j_t} / N) = -1$ and $(\Gamma_{j_1 \ldots j_t})^{2t} \equiv d(\Omega_{j_1 \ldots j_{t-1}})^{2^{t-1 jt}} (\Omega_{j_1 \ldots j_{t-2}})^{2^{t-2 jt-1}} \ldots \ (\Omega_{j1})^{2_{j2}} f_\Gamma(\mathrm{C}, 0 \ \mathrm{N}) \ \bmod \mathrm{N}$; for some $d \in \{\pm 1, \pm 2\}$. If they are valid then,
3. **(Realizing the third rule)** Bob selects a random value $e' \in \{0, 1\}^k$ and sends his identity I_b, time T and e' to Alice, when $k = 0(p)$ is the security parameter, chosen by the bank. Both Alice and Bob compute $e = H_0(I_B, T, e')$, where $e \in \{0, 1\}^k$. Alice also computes $\Lambda_{j_1 \ldots j_t}$ such that
   $(\Lambda_{j_1 \ldots j_t})^{2^{4+1}} \equiv 2^{2e} < f_\Lambda(\mathrm{C}, j_1, \ \ldots \ , j_t, N) > QR \bmod \mathrm{N}$
4. Bob verifies the validity of $\Lambda_{j_1 \ldots j_t}$ by checking that:
   $(\Lambda_{j_1 \ldots j_t})^{2^{k+1}} \equiv d' \ 2^{2e} f_\Lambda(\mathrm{C}, j_1, \ \ldots \ , j_t, N) \bmod \mathrm{N}$,
   for some $d' \in \{\pm 1, \pm 2\}$. If verification succeeds, Bob accepts the payment.

6.5 The Deposit protocol

The deposit protocol is simple. Bob sends a transcript of payment to the bank.

6.6 How close to the e-cash features is this protocol?

Security: The security parameter for this system is *K* presented by the bank and it determines the keys bit. For example if $k = 2^8 + 1$, then |p| = |q| = 256 bits, |N| = 512 bits. $|n_i| = \mathit{514}$ bits (i = 1, 2). Then $|\rho| \geq \mathit{1030}$ bits, |b| = 64 bits, and $|n_w| = \mathit{512}$ bits.

Okamoto's cash system's security is based on RSA signatures security, we can trust this system as long as we don't know any polynomial attack against RSA signatures. Okamoto's system will reveal Alice identity if she is a double-spender (in a divisible system even if she is an over-spender). For Alice to overspend a coin she must violate one of the two rules of the binary tree which will cause her identity to be revealed.

First assume that she violates the Route node rule. Let $n_{n_{0 j_1 j_2 \ldots j_i}}$ and $n_{n_{0 j_1 j_2 \ldots j_i \ldots j_t}}$, be the used nodes. Then Alice sends $\Gamma_{j_1 \ldots j_i}$ and $\Gamma_{j_1 \ldots j_t}$ to Bob, and these values are finally sends to the bank. The bank can first detect the violation of the rule, by checking the coin values *(C, b)* along with (L_1, L_2, N) and consumed nodes $n_{n_{0j_1j_2\cdots j_i}}$ and $n_{n_{0j_1j_2\cdots j_i\cdots j_t}}$, in the bank's data base. Then the bank knows that:

$$\Gamma_{j_1\cdots j_i} =$$

$$\left[\left(\left(\left(<(\Omega_{j_1\cdots j_{i-1}})^{2^{i-1}j_i} \ldots (\Omega_{j_1})^{2^{j_2}} f_\Gamma(C, 0, N)>_{QR}\right)^{1/2^i} \bmod \right) N\right]_{-1}$$

On the other hand from $\Gamma_{j_1\cdots j_t}$, the bank can computes:

$$(\Gamma_{j_1\cdots j_i})^{2^{t-i}} \equiv \left((\Omega_{j_1\cdots j_{t-1}})^{2^{t-i-1}j_i} \ldots (\Omega_{j_1\cdots j_i})^{2^0 j_{i+1}}\right)\times$$

$$\left[\left(<(\Omega_{j_1\cdots j_{i-1}})^{2^{i-1}j_i} \ldots (\Omega_{j_1})^{2^{j_2}} f_\Gamma(C, 0, N)>_{QR}\right)^{1/2^i} \bmod N\right]_1$$

Since $(\Gamma_{j_1\cdots j_t})^{2^{t-i}} \bmod N$ is the quadratic residue (see section 2.1) and $(\Omega_{j_1\cdots j_i}/N) = 1$ (the Jacobi symbol (see section 2.1)). Therefore, the bank can compute

$$\left[\left(<(\Omega_{j_1\cdots j_{i-1}})^{2^{i-1}j_i} \ldots (\Omega_{j_1})^{2^{j_2}} f_\Gamma(C, 0, N)>_{QR}\right)^{1/2^i} \bmod N\right]_1$$

Using this value and $\Gamma_{j_1\cdots j_i}$, then the bank can factor *N* (see section 2.1) and obtains *p* and *q*, from which the bank can find Alice's identity (see the opening protocol for Okamoto's cash system) by computing *x* and *y*,

$$x = g^p \bmod p, \quad y = g^q \bmod p.$$

Note by violating the "root route rule", the bank obtains two square roots (see section 6.1) of $(f_\Gamma(C, 0, N))$, and can factor *N*.

Assume that Alice tries to spend the same coin twice, to do so she must violate the "same node rule" of the binary tree (see section 6.1). Therefore if Alice is uses a node twice then she will challenge Bob twice (say e_1 and e_2), e_1 and e_2 should be different because of the hash function H_0 (unless Bob is cheating not Alice). The bank can factor *N* and find *q* and *p*, revealing Alice's identity (therefore *N* is a Williams integer (see section 2.1)).

Privacy: Each customer receives an electronic license = (N, L_1, L_2) to send to the shop during the payment phase, therefore binding each payment to a license and the coins used linked to each other, making it easy to follow each payment and guessing who the customer is, unless the customer opens a new account for each coin, which is not practical (specially with such a complex opening account protocol). Given this protocol, Okamoto does not offer an anonymous cash system.

Portablility: When the divisible coin is ready it may look easy to take it around but we must not forget that every time Alice wants to spend $x of her coin she has to compute the nodes representation on a binary tree (see the payment protocol) and answer to the

Bob’s challenge before she can spend her coin, besides she has to memorize the spent nodes. The coins developed by this system are not easy to carry around, however Okamoto says that corresponding to a security parameter, $k = 257$, with a coin worth \$1000 and spending 1 cent at a time, the total amount of stored data required (the coin + electronic license) is 264 bytes, for the withdrawal protocol 64 bytes, for the payment protocol 2880 bytes on average, totalling 3208 bytes [23]. It therefore would be possible to carry around our coins if we could store approximately 3.2 KB of data on a smart card. Most present day smart cards stores up to 3 KB of data [2] (Some of them can store up to 8 KB), therefore unabling us to store coins on a smart card and rendering the coins of this system not portable.

Transferability: Okamoto doesn’t provide the possibility of a transferable system but by looking at [8, 24], we can suggest a solution for such a system.

Using the opening account protocol and withdrawal protocol steps as stated, suppose Alice wants to pay Bob \$25 of her \$100 worth coin, and Bob wants to pay this \$25 coin later to Eve. Alice and Bob must follow the coin authentication part of the payment protocol as before. At the denomination revelation part of the payment protocol, Alice sends a certification T that denotes the transfer of the coin C to Bob. For example T could be, $T = H(\mathrm{C}, 011, \text{ public key of Bob})$, and both Alice and Bob follows the payment protocol as before. When Bob wants to pay Eve, he sends all of the payment history from the payment between him and Alice to Eve. Eve checks the validity of this history and then they follow the payment protocol as before, but in this case Bob sends messages corresponding to nodes Γ_{011} and Λ_{011} to Eve. The challenge here is to construct a certification method T, an unique and easy way to find the cheater, in the case of double-spending even if Alice and Bob cooperate.

Divisibility: The idea with this cash system from beginning has been to develop a divisible cash system using the binary tree representation (see section 6.1) for a coin, making a divisible coin possible.

6.7 Attacks

Since this system is RSA-based, then all the attacks corresponding to an RSA scheme could occur here. However, here Okamoto uses a RSA signature with one-way hash functions to minimize the risk of the chosen message attacks on his system.

It exit an attack on opening account protocol [28]. Alice can cheat during the opening of an account so that she can later overspend the coins without being revealed.

Identity of an over-spender (or double spender) can be uncovered if N (= pq) is a Williams integer and p, q are primes committed to x and y by

$$x = g^p \bmod \mathrm{p},\ y = g^q \bmod \mathrm{p}.$$

established at opening account protocol. But it exit a possibility that

$$N = p^i q^j$$

where i and j are odd integers and

$$p \equiv 3 \bmod 8\,,\; q \equiv 7 \bmod 8\,.$$

Then Alice can attack the system by choosing

$$p_1 \equiv 3 \bmod 8 \quad \text{and} \quad q_1 \equiv 7 \bmod 8$$

Where

$$i' < i,\; j' < j$$

$$p = p_1^{i'} q_1^{j'}$$

$$q = p_1^{i-i'} q_1^{j-j'}$$

Then the bank will give Alice an electronic license on $N = pq (= p_1^i q_1^j)$.

At withdrawal, Alice uses $N = p_1^i q_1^j$.

And in the case of over-spending, the bank will obtain

$$p_2 = p_1^i \neq p \text{ and } q_2 = q_1^j \neq q\,.$$

Then Alice identity will still be hidden from the bank and they can not find the cheater.

6.8 Cost (both money and time)

There isn't any known implementation of Okamoto's digital cash system, therefore it is hard to compute the costs of this system. Okamoto believes [23] that by choosing a length of 512 bits for a key, we will just need 64 bytes for withdrawal protocol, a user will need 1152 bytes to store both her electronic license and a coin worth $1000. He believes that the only part of his system which is ineffective is the opening account protocol.

We have to notice that Okamoto gives us, the number of bytes which would be needed in the case of 512 bits for a key length, however we know that his system is RSA-based and the recommended length for an RSA key is more than 512 bits (see section 2.2).

We should also take a look at communication complexity in his system. The opening account and the withdrawal protocol both consists of two moves, the payment protocol requires three moves and one move for the deposit protocol. Then the total communication complexity for the system is eight moves. (Note: each move will cost money and time.)

6.9 Disadvantages

The major disadvantages of Okamoto's system is that the coins made by the system are linkable, leading to the identification of the user, therefore this cash system doesn't protects the user's anonymity.

Okamoto's cash system offers a divisible coin but this divisibility is based on the use of complex mathematic functions, which will lead to difficulty in implementing the system. The system's protocol looks difficult for the average user to understand which will lead to the mistrust in the system security.

7 A Product Overview

Presently many companies are in the process of implementing and developing digital cash systems, DigiCash, Open Market, Cybercash, First Virtual, NetBill and Mondex are some of them. To date none have been able to develop an ideal cash system. This chapter takes a closer look at three of the above systems.

7.1 DigiCash

DigiCash was founded and created by David Chaum in 1990 and is located in Amsterdam. It develops and license payment technology products [13].

One of the company's products is ecash, a prototype for digital cash which has been in progress since 1995. This system is designed for secure payments from any personal computer to any other work stations, over email or Internet. Ecash is implemented almost around Chaum's digital cash system [10], however the main difference is that ecash is an on-line system.

How ecash works inside

Each person using ecash has an ecash account at a digital bank on the Internet, for example EU Bank in Finland. Using that account people can withdraw and deposit ecash. Ecash is a coin based system, which means that digital money is implemented by digital signatures representing a certain fixed amount of money.

Ecash security is based on the RSA principles (see section 2.2). When the ecash is executed on your computer for the first time, the ecash software automatically generates a pair of RSA encryption keys. Every person or entity using ecash has a unique pair of keys. One key is kept secret (the secret key) and the other key made public (the public key). Alice who wants to authenticate a message encrypts it with her own secret key, everyone can verify that Alice signed this message by decoding it with the Alice's public key. If Alice wants to send a confidential message, encrypts the message with the public key of the receiver, the receiver is the only one who will be able to decode the message.

When an ecash withdrawal is made, the PC of the user calculates how many digital coins and of what denominations are needed to withdraw the requested amount. The PC generates random serial numbers for those coins, which will act as the coin's serial number and a blinding factor is included. The blinding factor is required so the bank is not required to maintain a list of the coin's serial numbers and discover where the money is spent. The result of these calculations will be sent to the digital bank.

The bank will first verify the message and then encode the blinded numbers with its secret key (digital signature), at the same time debiting the account of the client for the same amount. The authenticated coins are sent back to the user and finally the user will take out the blinding factor that he introduced earlier. The serial numbers plus their signatures are now digital coins, their value is guaranteed by the bank.

The coins can be stored locally on the PC of the user. As soon as he wants to make a payment, his PC collects the coins needed to reach the requested total value. These coins are sent to the receiver, then the receiver sends them directly to the digital bank before he accepts your payment. The bank verifies the validity of these coins and that they have not been spent before. The account of the receiver is credited. Every coin is used only once. Another withdrawal is needed if the receiver wishes to have new coins to spend.

Ecash works like travel checks if you lose it, you have to report this to your bank and supply them with the serial number of the lost coins, then the bank can check if the coin are really lost and refund them. A user can cancel a payment if she is not satisfied with her shopping by revealing the coins serial number to the bank and asking them for payment cancellation.

The systems advantages are privacy for payer, low transaction cost and the ability of person to person payments. This system provides anonymity for payer since the coins are created with blind signatures on the withdrawal protocol. (However if users use their own servers for the payment protocol, then the payment is not completely anonymous for them since the payee will have knowledge of their Internet address. The payers can keep their anonymity by using a forwarding agent.) The payment is not private for payees, he must turn the coins immediately to the bank to determine if the coins are valid, before delivering the sold item to payer. DigiCash argues that this form of payment protocol minimizes the risks of double spending.

A customer can only lose as much money as they are carrying. People may be more willing to deal with electronic cash and only risk the $20 in their electronic wallet than to send their $5,000 gold card number across the net.

The disadvantages of ecash is that Alice has to open an account in a bank accepting ecash and Bob has to deposit the money at the same bank. And the bank has to have a large database for storing the coins series number.

If you PC hard drive were to crash, your e-bank goes under, or hackers were to decode your numbers, there would be no way to retrieve your lost cash (just as if you dropped a $20 bill on the street and lost it). Since the bank does not link the money to your name they would have no way to reimburse you unless (just in the case of hard disk crash) you accept to give up your anonymity to get the ecash.

Administration Problems

Administration problems are that kind of problems which must be solved before a digital cash product could take off.

Today exist some of these problems by some of Digicash actors. The most important one of these problems is to keeping an on-line service. We remember that one of major advantages with digital cash is that people can pay their payment fast and effective (see section 1.2.6). This problem appears when some of the actors like the EUnet of finland can not have an on-line service for the customers. The customers should wait approximately a week in some cases even longer to get their accounts open or get answer for an usual question like how they could open an account.

Another important problem is security problem. These actors send the customers account number and password without using any encryption algorithm with an E-mail. Then with help of a man-in-the-middle attack anyone could access to customer’s account and pretend to be her.

7.2 CyberCash

The CyberCash company is located in U.S.A and was founded in 1994 by William Melton and Daniel Lynch [29].

The company offers a product called Wallet [12] to its customers (so far Wallet has been implemented just for Windows and Macintosh).The users who have a Wallet have access to credit card and Cyber coin payment. Cyber coin act like cash and are used on the Internet for small transactions, less than $10 [12].

The customers open an Wallet by down loading the software from CyberCash and filling a form. The customers include their credit card number on the form, since CyberCash is not a bank then they take the withdrawal money from user’s credit card. Then each user will receive a 768-bit RSA key and she locks her RSA key by using a password.

How does the payment protocol work?

1. Alice wants to buy something from Bob
2. Bob’s Wallet sends an invoice to Alice.
3. Alice’s Wallet hashes Alice’s credit card number, her identification data (like her name) and the price and asks Alice to sign. The Wallet then encrypts the signed message with a public key of CyberCash server and sends it (m) to Bob.
4. Bob adds his identification data and the price to the m and signs the result, then sending it to CyberCash server.
5. CyberCash unblinds this message and checks if Bob and Alice wrote same price. It then sends this information to the bank and the bank reveals Alice’s credit card number to Bob. Bob gets paid and sends the item to Alice [29].

CyberCash system offers a higher security than credit cards.

Neither Alice nor Bob’s privacy is protected in the above system. This system is Online and Bob can not send the item to Alice before he gets the bank’s response.

7.3 First Virtual

The First Virtual company is one of the first companies to offer a digital money transfer system created for the Internet [19].

How does the payment protocol work?

1. The customer opens an account. This can be done if the customer has an email address and is reading to provide a credit card number because all bills will be sent to the buyer as a charge against this card.
2. When the customer wants to buy something, she and the seller negotiates the price and then she gives a copy of her account id to the seller.
3. The seller sends a transfer request including his and the customer's account id and a description of the transaction to First Virtual through an email message.
4. First Virtual sends a request for a conformation to the customer's e-mail account.
5. The customer can answer it with three different responses:

   <u>YES</u> All is well. The customer authorizes First Virtual to bill the credit card on file for the amount.

   <u>NO</u> The customer is refusing to pay. This is a significant event and First Virtual keeps records of this. If a customer does this too often, the First Virtual may terminate her account. First Virtual will make this determination because it doesn't want people to take advantage of sellers by refusing payment.

   <u>FRAUD</u> The customer never authorized the transaction and First Virtual should investigate.
6. When the customer paid the credit card company, First Virtual deposits the correct amount in the checking account of the seller.

First Virtual uses no encryption. They replace the security of using encryption with a centralized transaction machine that confirms all transactions. Their argument is that if a buyer's electronic mail is secure then anyone committing fraud will be stopped when the customer refuses to confirm the transaction. Sensitive information like credit card number never has to travel over the Internet. Transactions are all handled with the customer's unique First Virtual account identity. There are several major advantages of living without encryption. The company doesn't need to ask the U.S. government for export permission. First Virtual doesn't need to get complicated purchasing software available on many machines. A customer only needs to get its account name to the seller. This could be done by speaking it over a telephone, sending it via e-mail or by faxing it.

This argument is fair. It is correct that the credit card number never travels over the net, they have only replaced it with account identity. This account identity does travel over the net and may be stolen. Confirming the transaction with help of e-mail account is not enough secure since there is no guaranty that this e-mail account is not compromised.

The system uses no cryptography so a seller must have an on-line connection to verify that a particular account identity offered by a customer is valid.

First Virtual offers no anonymity. The customer, the merchant and the bank know everything.

The fact that First Virtual does not use any encryption and all there partners (customer, seller and the bank) know everything, makes it possible to attack the system. This attack may look as follows:

1. Eve gains access to the someone’s electronic mail.
2. She opens a bank account with a fake name.
3. Eve creates a seller’s account and registers it with the bank account opened with this fake name.
4. Eve gets the customer’s account identity by reading the mail.
5. She starts a transaction between the customer and her seller account.
6. Eve intercepts the confirmation request and responds with a ”YES”.
7. When the money is deposited in the checking account. Eve withdraws it in cash and disappears.

8 Digital Cash in practice

In the previous chapters we discuss what digital cash is and some different schemes for digital cash systems. However an idea without implementation is worth nothing, therefore the following chapter will explore the implementation aspects, problems and the user's views of digital cash.

8.1 Comparison

Comparing the four digital cash systems presented in this report results the following conclusion:

With respect to the security: The Chaum-Fiat-Naor scheme (see chapter 3), the Ferguson scheme (see chapter 4) and the Okamoto scheme (see chapter 6) are RSA-based systems which means finding a fast way to factoring large numbers will result in damaging these systems security since these systems security is based on the difficulty to factoring large numbers. Even if it will take years until a new factoring algorithm will be find, we have to increase the key length of these systems by time since factoring of large numbers becomes an easier task by time, hence the computers capacity will rise.

The security of Brands scheme is based on the difficulty to computing discrete logarithms rather than RSA-roots, therefore the ability of factoring large numbers does not affect the security of this system. Then Brand's cash system is offering a better security.

With respect to communication costs: The Chaum-Fiat-Naor digital cash system consists of eight moves; four moves for the withdrawal protocol, three moves for the payment protocol and one move for the deposit protocol. The Ferguson digital cash system contents of eight moves; four moves for the withdrawal protocol, three moves for the payment protocol and one move for the deposit protocol. Brand's digital cash system contents of ten moves; six moves for the withdrawal protocol, three moves for the payment protocol and one move for deposit protocol. The Okamoto divisible digital cash system contents of six moves; two moves for the withdrawal protocol, three moves for the payment protocol and one move for the deposit protocol.

Each of these moves will cost time and needs security measure depending on which kind of communication medium is used. It means that Brands cash system's communication needs some more moves than the others. However, it doesn't means that it cost more becouse the Brands system is single-term since it does not use the cut-and-choose protocol. Ferguson scheme is single term too but both the Okamoto scheme and Chaum-Fiat-Naor scheme are using cut-and-choose protocol which make them ineffectiv and couse high communication costs.

With respect to the value of the coin: All of the presented digital cash systems in this report use a common way to denate the coins value. The way is to have the bank to use a different public key for each denomination. That is, if there are to be *k* distinct coins, the bank publishes $(n_1, e_1)(n_2, e_2) \ldots (n_k, e_k)$ as it's public key. However, Brand's cash system can use both the common way and another way which is to have *k* different dummy generators $d_1, \dots, d_k$ (see chapter 5) published by the bank. Each of these generator is used to denote some fixed amount of money. We can have for example d_i denote for $\$2^{i-1}$. These two different way to donate the coins value give Brands cash system the possibility to generate both different denominations and currency.

With respect to implementation complexity: Chaum-Fiat-Naor schema is the easiest of these four scheme presented here to understand and implement, because of its use of well known algorithms like RSA which is available in both hardware and software. Okamoto's schema is the most complex one, since Okamoto uses a combination of binary tree (see section 6.1), William integer (see section 2.1) and square root (see section 2.1) which results in a complex algorithm to understand and implement.

We have to recognize that Okamoto's system is the only one which use a complex opening account protocol, the other system's opening account protocol is easy to implement.

With respect to the double-spending problem: All of the presented systems have the weakness that they are just able to find the double-spender after the fact, which may be to late by then.

With respect to an ideal cash system features: The Okamoto's schema is the only one presented here, that offers a divisible cash system at the cost of lost anonymity.

All of the above systems can be transferable at the cost of lost anonymity and security and security, since a transferable coin can be recognized by it's old users and catching the double-spenders will be possible after a longer period of time. A transferable alternative for these systems has the disadvantage that a coin grows in size with every transfer, since a transferred coin must contain the identity of all users that owned it so that double-spenders can be identified.

None of these systems has the ability to be portable by using today's computation and storage units.

All of these systems offers a anonym cash system except Okamoto's system which is linkable and reveals the identity of its users after a time.

8.2 Implementation aspects

In this section, we will discuss implementation features and the basic problems according to the digital cash systems.

Storage problems will appear often after the withdrawal protocol. A customer needs to store the withdrawald coin and other information corresponding to the withdrawal process of the coin which she needs to reuse again at the payment protocol.

An important problem in implementing the system is the security of communication channels. We must be sure that our communication channels are secure, if they are not we should use encryption algorithms which make it secure to send the packets through the untrusted channels.

It will cost the payee if he sends the coin to the bank after each payment. However, if he store the coins and then send them to the bank he will lose the interest. If he chooses to store the coins, he needs to have access to a data base in which he can store the coins together with the transcript of the payment corresponding to that coin. After a period of time (a month), he will send the data base to the bank for deposit. The data base should be kept encrypted and secure during this time.

8.3 User's acceptance

There are some open questions which must be solved before people could accept digital cash and use it. Some of these questions are:

- Liability, if a coin is lost or stolen who suffers the loss?
- What kind of value should coins have? U.S. dollars, German marks or should we have a new cyber cash value? How should we change the money?
- How should the fees be charged? On a transaction or on the creation of a coin?

8.4 Smart Cards

Smart cards play an important role in creation of digital cash systems because:

- One of the important properties of digital cash is portablility (see section 1.3). It means that digital cash system's problem is to find a storage medium with computer power which is easy to carry around and has the ability to be connected to a computer network for transferring cash over Internet. Smart cards are the solution to this problem. They are easy to carry around, have a computer architecture and can be connected to a network thanks to card readers.
- One of major problems with digital cash is double-spending. Previous chapters, presented a solution to this problem (see 3.1, 4.1, 5.1 and 6.2) which is to reveal cheater's identity after the fact which maybe is too late to find the double-spender by then. Another solution to double-spending problem is to create a special smart card containing a tamper proof chip called *observer* [6]. The observer chip keeps a mini database of all the pieces of digital cash spent by that smart card. If Alice attempts to copy some digital cash and spend them twice, the observer chip would detect the attempt and would not allow the transaction.

8.5 What is a smart card?

A smart card is a plastic card embedded with a $25\ \text{mm}^2$ (ISO standard) integrated circuit(IC) chip. The chip stores information while protecting it from unauthorized. It is also possible to build in high levels of computing power and security, since smart card's IC has a computer architecture.

There are some different types of smart cards.

- *Simple memory cards.* These cards act as a storage medium. They carry an application code and a simple mechanism to specify the issuer of the card.
- *Hard wired logic cards.* These cards contain memory and processor and have data processing capabilities which permits the dynamic storage management. The data processing power is often used to encrypt/decrypt data.

8.6 Using digital signatures in smart card systems

The digital signature given out by the smart card in the payment process is the basis for settlement between the customer and shop, since it authenticates both the customer and amount of money. This section describes the various ways[14] in which smart card systems use digital signatures.

- *Shared-key systems.* Secret key in the chip lets the card authenticates its communication with any device sharing the same key. These systems are often based on DES. Their security is depend on the security of the master key, because this key must be distributed to many users across the whole system. All the payments a user makes by this kind of card is linkable, since cards are given unique keys to increase security.
- *Public-key signature-creating systems.* In these system, for each card, the bank creates a specific pair of secret and public keys and stores them in the card. Because the public key signature function requires a lot processing power, the smart card will need a co-processor[9] capable of making digital signatures. All the payments a user makes are linked together by the card identity.
- *Public-key signature-transporting system.* The bank creates pre-signatures for a specific card before transaction by using the system's secret key and stores them on the card's chip. During a transaction, the smart card transforms one pre-signature into a full signature. This transformation requires only limited processing power. The shop's terminal verifies the signature using the system's public key (p). The payment made by signature-transporting cards are not linkable, since instead of a single key per card, cards use a different signature per payment.

8.7 To transfer digital cash by smart cards

There are two different techniques for transferring digital cash by smart cards.

- *Coin-transporting technique.* This technique is based on coins made during the withdrawal protocol.The value on a smart card is represented by the stored coins. Coin-transporting systems are more secure because balances are not stored in the smart card. Drawback of this system is that storing coins requires considerable amount of non-volatile memory (many kilobytes of EEPROM) which becomes less significant as smart card memory becomes cheaper[14].

- *Balance-counter technique.* There is a balance-counter on the smart card. The bank adds value to a card by increasing the balance. When the smart card gives out a signature representing a certain amount, this amount is deducted from the balance in the card [4].

9 Conclusion

Discussing and comparing the four digital cash systems presented in this report show us that Brands system is the best of them. It offers a high level of security because of its use of the discrete log scheme instead of RSA, and its use of the presentation problem for authenticity. Brands system does not use cut-and-choose protocol which make it a single-term and effective system. This system offers the possibility to generate both different denominations and currency, hence, the system offers two different ways to denote the coins value (see section 8.1). The coins generated by this system are secure and provides their user's anonymity. Brand presents the alternatives of his system to generate portable coins [4]. The only drawback of Brands system is the high communication cost (see section 8.1), which can however be lowered by asking the user to withdraw several coins at the same time.

However, even Brands cash system is neither divisible nor transferable (Notice: he offers an alternative to creating a divisible system on the cost of anonymity [6]) and implementation of this system cause some problems which are not easy to solve. So even this system does not fulfil all the requirements of an ideal digital cash system.

Different companies like Digicash (see chapter 7) are known to have tried for generating a digital cash system but none of them have been successful because of security problems, implementation problems and administrative problems. Even if cryptographs can suggest an ideal digital cash scheme, user acceptance problems (see section 8.3) like "if a coin is lost who suffers the loss?" must be solved before digital cash could be as common as paper cash is today.

10 REFERENCES

[1] Ankney R.,Introduction to Cryptographic Standards, http://www.itd.nrl.navy.miIT D/5540/ieee/cipher/cipher-crypto-stds.html.

[2] A Frequently Asked Questions list (FAQ) for alt.technology.smartcards, http:// ps.superb.net/FAQ/

[3] Boneh D. & DeMillo R. & Lipton R., On the Importance of checking computations, Math and Cryptography Research Group, Bellcore.

[4] Brands S. (1994), Off-line cash transfer by Smart Cards, Centrum voor Wiskunde en Informatica, Computer science/department of algorithmics and architecture, CS R9455, http://www.cwi.nl/~brands/cash.html

[5] Brands S. (1993), Untraceable Off-Line Cash in Wallets with observers, Advances in Cryptology CRYPT '93, Springer-Verlag, pp. 302-318.

[6] Brands S. (1993), An efficient Off-line electronic cash system based on the representation problem. Technical report, Centrum voor Wiskunde en Informatica, CS-R9323, http://www.cwi.nl/~brands/cash.html.

[7] Chan A. & Frankel Y. & MacKenzie P. & Tsiounis Y. , Misrepresentation of identities in E-cash schemes and how to prevent it, Technical report, Centrum voor Wiskunde en Informatica.

[8] Chaum D. and Pedersen T., Transferred Cash Grows in size.

[9] Chaum D., Prepaid smart card techniques, A brief introduction and comparison.

[10] Chaum D. & Fiat A. & Naor M. (1988), Untraceable Electronic Cash, Advances in CRYPTO '88, Springer-Verlag, pp 319-327.

[11] Certicom's elliptic curve cryptosystem & standards page, http://www.certicom.ca html/ecc.htm.

[12] CyberCash home page , http://www.cybercash.com/.

[13] DigiCash homepage, http://www.digicash.nl/.

[14] Digital signatures and smart cards (1996), Delivered at the third internationa smart card conference, Amsterdam.

[15] Digital Signature Scheme ISO 9796 , http://www.ewos.be/sec/gdss.htm.

[16] Directory Authentication Framework (X.509), http://www.securityserver.com/cat egory/@x509.htm.

[17] Eng T. and Okamoto T. (1994), Single-Term Divisible Electronic Coins, Advances in Cryptology EUROCRYPT '94, Springer-Verlag, pp. 311-323.

[18] Ferguson N. (1993), Single Term Off-Line Coins, Centrum voor Wiskunde en Informatica Computer science/department of algorithmics and architecture, CSR9318.

[19] First Virtual homepage, http://www.fv.com/.

[20] George Barwood's elliptic curve cryptography FAQ, http://ds.dial.pipex.co george.barwood/ec_faq.htm.

[21] Matonis J. (1995), Digital Cash and Monetary freedom, presented at INET '95, Internet Society Annual conference, Honolulu, Hawaii.

[22] Menezes A. & Oorschot P. & Vanstone S. (1996), Handbook of Applied Cryptora phy, CRC.

[23] Okamoto T. (1995), An Efficient Divisible Electronic Cash Scheme, Advances in Cryptology CRYPTO ‘95, Springer-Verlag, pp. 438-451.

[24] Okamoto T. and Ohta K. (1991), Universal electronic Cash, Advances in Cryptology CRYPTO ‘91, Springer-Verlag, pp. 324-337.

[25] RSA Laboratories Cryptography FAQ, http://www.rsa.com/rsalabs/newfaq.

[26] Schneier B. (1996), Applied cryptography, second ed. Wiley.

[27] sci.math FAQ, Prime Numbers, http://www.utm.edu/research/primes/lists/.

[28] Tsiaunis S. (1997), Efficient electronic cash, new Notions and techniques, Ph.t Thesis, http://www.ccs.neu.edu/home/yiannis/pubs.html

[29] Wayner P. (1995), Digital Cash, first ed. AP Professional.

11 Appendices

11.1 Appendix A: RSA in Hardware and practice

RSA in Hardware

The RSA algorithm is available in a number of hardware implementation, listed in Table 1.1.

Tabell 3: Partial List of RSA Hardware Available

	Company Speed	Baud Rate Cycles per 512 Bits	Clock to Encrypt 512 Bits	Bits Per Technology	Chip	Transistors
Alpha Tech	25Mhz	13K	.98M	2 micron	1,024	180,000
AT&T	15Mhz	19K	.4M	1.5 micron	298	100,000
British Telecom	10Mhz	5.1K	1M	2.5 micron	256	NA
Business Sim., Ltd.	5Mhz	3.8K	.67M	Gate array	32	NA
Calmos Syn.,Inc.	20Mhz	28K	.36M	2 micron	593	95,000
CNET	25Mhz	5.3K	2.3M	1 micron	1.024	100,000
Cryptech	14Mhz	17K	.4M	Gate array	120	33,000
Cylink	16Mhz	6.8K	1.2M	1.5 micron	1.024	150,000
GEC Mar-coni	25Mhz	10.2K	.67M	1.4 micron	512	160,000
Pijnenburg	25Mhz	50K	.256M	1 micron	1.024	400,000
Plessy Crypto.	NA	10.2K	NA	NA	512	NA
Sandia	8Mhz	10K	.4M	2 micron	272	86,000
Siemens	5Mhz	8.5K	.03M	1 micron	512	60,000

RSA in Practice

It is known that the DES algorithm is much faster than RSA. When implemented using software, DES is generally at least 100 times faster as RSA, while the hardware implementation can increases the DES routine advantage between 1,000 and 10,000 times faster. In practice RSA is often combined with DES to take advantage of the high speed of DES with the key-management convenience of RSA.

11.2 Appendix B: Implementation of Brand's digital cash system