Proposal of a Fast Public Key Cryptosystem

Kouichi Itoh[†], Eiji Okamoto, Masahiro Mambo

School of Information Science
Japan Advanced Institute of Science and Technology
E-mail : k-itou@jaist.ac.jp, okamoto@jaist.ac.jp, mambo@jaist.ac.jp

Abstract. Public key cryptosystem is a means to establish a secure channel, and has been used recently. For the implementation on simple hardware in case of mobile communication, for example, a scheme with small amount of computation is necessary. RSA is one of the famous public key cryptosystems. RSA and its derivative schemes have high level of security, but require large amount of computation.

A linear cryptosystem whose complexity depends on the knapsack problem requires small amount of computation, but there is a general algorithm to cryptanalyze it, and almost all linear cryptosystems have been broken.

This paper proposes a cryptosystem whose encryption and decryption look like a linear cryptosystem, yet its security is high.

Index Terms — Public keys, random key choice, key trace, combination of key choice.

1 Introduction

This paper proposes a public key cryptosystem which runs fast and has high level of security. In the proposed scheme, Alice chooses some keys from many public keys and encrypts a plaintext according to the chosen keys. An outsider Oscar cannot detect which keys are used for the encryption of the plaintext. But Bob can trace Alice's random key choices with secret information and decrypts the ciphertext. Number of combinations of Alice's random key choices is exponentially large, and this works as a trapdoor against Oscar's attack.

In encryption, the proposed scheme requires small number of operations, though the size of public keys is large. Computational complexity for encryption is given as $o(x^3)$, where x is the bit length of r of ciphertexts (m, r).

The process of decryption looks like linear cryptosystem, so that decryption is fast. Computational complexity for decryption is given as $o(x^2)$. It is smaller than that of some other public key cryptosystems because RSA and its derivative require $o(x^3)$ for decryption.

2 The Proposed Public Key Cryptosystem

1.Proposed scheme

public information : $(e_i \pmod{M}, k_i \pmod{N}), (1 \le i \le n), M, N, n, l$

secret information : $P, Q, t \pmod{N}, q_i, q_i'$

Where,

[1]Present affiliation is Fujitsu Labratories LTD. S Project Group. E-mail:kito@flab.fujitsu.co.jp

- $(e_i, k_i)(1 \leq i \leq n)$ are pairs of public encryption keys.
- e_i is random.
- k_i satisfies

$$k_i \equiv tq_i \pmod{P}, (1 \leq i \leq n). \tag{1}$$

  for t, q_i and P.
- M is size of plaintext and requires no particular conditions.
- N is a product of two large primes P, Q.
- q_i satisfies

$$(q_{max})^l < P \tag{2}$$

  for l, and $o(q_i) = o(P^{\frac{1}{l}})$, where q_{max} is a maximum value of q_i, and

$$q_i = u_i q_i'$$

  where q_i' are primes and different each other. For any u_i and q_j', $gcd(u_i, q_j') = 1$ is satisfied.

2.Encryption

Alice chooses l pairs of keys from n pairs of public keys(permitted to choose same keys). From a plaintext $s \pmod{M}$, Alice composes a pair of ciphertext (m, r) as

$$\begin{cases} m \equiv e_a + e_b + \ldots + e_l + s \pmod{M} \\ r \equiv k_a k_b \ldots k_l \pmod{N} \end{cases} \tag{3}$$

where $e_a, e_b, \ldots, e_l$ and $k_a, k_b, \ldots, k_l$ are chosen keys.

3.Decryption

Bob decrypts (m, r) with secret information. At first, Bob figures following r':

$$\begin{aligned} r' &\equiv (t^l)^{-1} r \\ &\equiv (t^l)^{-1} t^l q_a q_b \cdots q_l \\ &\equiv q_a q_b \cdots q_l \pmod{P}. \end{aligned}$$

From (2), r' and $q_a q_b \ldots q_l$ are equated.

$$\begin{aligned} r' &= q_a q_b \cdots q_l \\ &= u_a u_b \ldots u_l \cdot q_a' q_b' \cdots q_l'. \end{aligned}$$

With r', Bob decrypts ciphertexts as follows.

1. Let $i = 1$.
2. Figure out $r' \bmod q_i'$.
3. If $r' \equiv 0 \pmod{q_i'}$, Let $m := m - e_i \pmod{M}$ and $r' := r'/q_i$, otherwise Let $i := i + 1$.
4. repeat 2., 3. until $r' = 1$, and Bob gets m as a plaintext.

$(e_1, k_1), (e_2, k_2), (e_3, k_3), (e_4, k_4)$ are public keys, and let $l = 3$. Alice chooses $l(= 3)$ keys from $n(= 4)$ keys such as key1, key1 and key3. In this case, ciphertexts are

$$\begin{cases} m \equiv e_1 + e_1 + e_3 + s \pmod{M} \\ r \equiv k_1 k_1 k_3 = t^3 q_1 q_1 q_3 \pmod{N} \end{cases}$$

and these are sent to the Bob.

At first, Bob figures

$$r' = (t^3)^{-1} \cdot r \pmod{P} = u_1 u_1 u_3 q_1' q_1' q_3'$$

and decrypts ciphertexts as follows.

- $r' = u_1 u_1 u_3 \cdot q_1' q_1' q_3' \equiv 0 \pmod{q_1'}$, so that $r' := r'/q_1$,
  $m := m - e_1 \pmod{M}$... key1 is detected.
- $r' = u_1 u_3 \cdot q_1' q_3' \equiv 0 \pmod{q_1'}$, so that $r' := r'/q_1$,
  $m := m - e_1 \pmod{M}$... key1 is detected.
- ($r' \equiv 0 \pmod{q_1', q_2'}$ are not satisfied.)
- $r' = u_3 \cdot q_3' \equiv 0 \pmod{q_3'}$, so that $r' := r'/q_3$,
  $m := m - e_3 \pmod{M}$... key3 is detected.
- From $r' = 1$, trace finished.

With these procedures, Bob detects that Alice had chosen key1, key1 and key3 in encryption, and gets a plaintext $s \equiv m - e_1 - e_1 - e_3 \pmod{M}$.

3 The Trapdoor in the Proposed Scheme

In the proposed scheme, number of all possible combinations of Alice's random key choices works as a trapdoor. In this section, we describe the number of all possible combinations of the key choices.

Suppose that Oscar intends to detect Alice's random key choices. From ciphertext r and public information k_i, Oscar tries to solve

$$r \equiv k_1^{x_1} k_2^{x_2} \cdots k_n^{x_n} \pmod{N} \tag{4}$$

for x_i (with "all-search"), without secret information. Hence, number of solutions x_i equals to number of Alice's random key choices and we consider the number.

From

$$r \equiv \overbrace{k_a k_b \ldots k_l}^{l} \pmod{N}$$

number of solutions of (4) equals to number of integer solutions of

$$x_1 + x_2 + \ldots + x_n = l \tag{5}$$

where $0 \leq x_i \leq l$. And number of solutions for (5) corresponds to number of possible permutations such that l of $\bigcirc$s are divided into n sections with $n - 1$ of | s.(see below example case in

$n = 4$ and $l = 3$.)

$$\overbrace{\underbrace{\bigcirc\bigcirc}_{x_1=2}\,|\,\underbrace{\quad}_{x_2=0}\,|\,\underbrace{\bigcirc}_{x_3=1}\,|\,\underbrace{\quad}_{x_4=0}}^{l}$$
$$(\bigcirc = l, | = n-1)$$

Number of $\bigcirc$s contained in ith section equals x_i. Hence, all solution for (5) corresponds to all number of permutations and number of permutations is

$$_{n+l-1}C_l. \tag{6}$$

Note that (6) increases exponentially according to the increment of n, l. Hence, Bob decrypts ciphertexts with dividing r' by q_i for some finite steps, but the Oscar must analyze r which takes exponentially number of possible cases.

4 Argument over the Security of the Proposed Scheme

In this section, we suppose various analysis for the proposed scheme and describe conditions of parameters for the security of the proposed scheme.

1.Analysis for m.

We suppose an analysis for m of ciphertexts (m, r). From (3), m is written as

$$m \equiv e_a + e_b + \ldots + e_l + s \pmod{M}.$$

With x_i, this equation can be transformed into

$$m \equiv e_1x_1 + e_2x_2 + \ldots + e_nx_n + s \pmod{M}. \tag{7}$$

(7) can be solved for x_i and s with L^3 algorithm and plaintext s seems to be revealed from ciphertext m.

If we consider x_i is a number of times that ith key is chosen, another condition for x_i is

$$x_1 + x_2 + \ldots + x_n = l. \tag{8}$$

Hence, x_i and s are solution of (7)(8).

From previous section, number of solutions of (8) is $_{l+n-1}C_l$. And for arbitrary solution x_i of (8), a corresponding solution s of (7) exists, so that number of solutions s, x_i of (7)(8) is $_{l+n-1}C_l$ which is exponentially number. To decide one plaintext s from $_{l+n-1}C_l$ of candidates, r is required. Therefore, an analysis for only m is invalid, and security of ciphertexts (m, r) depends on the security of r.

2.Necessity of t.

We suppose that k_i is not multiplied by t in (1).

$$k_i \equiv q_i \pmod{P} \tag{9}$$

And we consider the security of the proposed scheme in this case. (9) is rewritten as

$$k_i \equiv p_iP + q_i \pmod{N}$$

and if q_i is moved to the left term, this equation becomes

$$k_i - q_i \equiv p_i P \pmod{N}.$$

Note that $o(q_i) = o(P^{1/l})$ is very small compared with P, so that P is revealed from $gcd(k_i - q_i, N) = P$ by "all-search" for q_i.

In case k_i is multiplied by t, above equation becomes

$$k_i - tq_i \equiv p_i P \pmod{N}.$$

Oscar can limit $t \pmod{N}$ as $0 < t < P$, and try $gcd(k_i - tq_i, N) = P$ *or* 1 (Oscar must not know P, but he can tell $gcd(x, N) = P$ if it is some very large number.), but its steps is as large as P. Therefore, t is necessary for the security of the proposed scheme.

3. Analyze k_i with L^3 algorithm

We suppose that Oscar tries to solve

$$k_i \equiv P_i + q_i t \pmod{N} \tag{10}$$

for (P_i, t). Where q_i is correct(which is occasionally correct by all-search for q_i) but Oscar does not know whether q_i is correct. After (10) is solved, Oscar can solve (1) by solving

$$P_i \equiv p_i P \pmod{N} \tag{11}$$

where the solution does not exist unless

$$P_i \equiv 0 \pmod{P} \tag{12}$$

is satisfied.

Oscar can solve (10) for (P_i, t) with L^3 algorithm. Next, Oscar tries to solve (11). If

$$P_i \equiv k_i - tq_i \pmod{N} \tag{13}$$

is considered, there is $N - 1$ of P_i which corresponds to $N - 1$ of t. Therefore, number of pairs of (P_i, t) (whether it equals secret information or not) satisfies (10) is $N - 1$. But, solution of (11) does not always exist for all $N - 1$ pairs of (P_i, t), because P_i varies q_i if t varies one (see equation (13)) and from $gcd(P, q_i) = 1$, very few pairs of (P_i, t) satisfy (12).

Therefore, this analysis is invalid, and analyzing k_i with L^3 algorithm is not efficient to reveal the secret information from the public information.

4. Differential attack

We consider equations

$$\begin{cases} k_i \equiv p_i P + tq_i \pmod{N} \\ k_j \equiv p_j P + tq_j \pmod{N} \end{cases}$$

and consider a differential

$$\begin{aligned} q_j k_i - q_i k_j &\equiv (p_i q_j - p_j q_i)P + t(q_i q_j - q_j q_i) \pmod{N} \\ &\equiv (p_i q_j - p_j q_i)P \pmod{N}. \end{aligned}$$

Hence, with correct value of q_i and q_j, P is revealed by trying $gcd(q_j k_i - q_i k_j, N) = P$ or 1. Therefore, P is revealed by all-search analysis for q_i and q_j which requires

$$q_{max}^2 \tag{14}$$

of steps maximum. And this shows the security against the analysis for the proposed scheme.

Considering the security described in this section, an example of optimal parameters in the proposed scheme are as follows:

- $N = 1024bit, P = 768bit, Q = 256bit$.
- Pairs of the public key, $n = 180$.
- Number of keys Alice must choose from n keys, $l = 17$.
- Security of the trapdoor, ${}_{180+17-1}C_{17} = 2^{80}$.
- $q_{max} = 2^{45}(= 6byte)$, and from (14), security against analysis $= 2^{90}$.
- $q'_{max} = 2^{32}(= 4byte)$.
- Size of the public keys, $180 \times 1024bit \times 2 = 45Kbyte$.
- Size of the secret keys, $180 \times (6 + 4)byte = 1.8Kbyte$.

And these parameters are used in the computer simulation shown in the next section, adding to set the size of the plaintext $M = 1024bit$.

5 Result of a Computer Simulation.

In this section, we show a result of computer simulation. We repeated encryption and decryption 100 times, and measured a running time. Result is an average running time of 100 times encryption and decryption, where parameters of the proposed scheme is same as that of previous section. In RSA scheme, we set size of two large primes $p = q = 512$ bit, so that size of the plaintext is same as the proposed scheme. And we set size of the encryption key $e = 65537$, which is used well in practice, such as SET. We used Chinese remainder theorem in decryption. And simulation is done on the computer whose CPU is Pentium 200MHz, under OS Linux2.0.0. with lrgnum calculation package which can used as C++ library.

	Encryption	Decryption
Proposed ($time_A$)	0.0211 sec.	0.018 sec.
RSA ($time_B$)	0.0278 sec.	0.87 sec.
$time_B/time_A$	1.32	48.3

Average running time of 100 times encryption and decryption.

Simulation under above settings showed result that running speed of the proposed scheme is 1.32 times as fast as RSA scheme in encryption, and 48.3 times as fast as RSA scheme in decryption.

6 Summary and Conclusion

We proposed a new type of public key cryptosystem and result of computer simulation showed we have achieved a fast public key cryptosystem.

Encryption of the proposed scheme requires $l - 1$ times multiplication. Let x be bit length of N, computational complexity of the proposed scheme in encryption is

$$o(l \cdot x^2)$$

and we consider how l increases when x increases. From (2),

$$(q_{max})^l < P$$

and if q_{max} is fixed(in fact, q_{max} affects the security of the proposed scheme and must increase at the increment of N), l is $o(\log P)$. Therefore, computational complexity of the proposed scheme in encryption is

$$o(x^3)$$

whose coefficient is small.

Decryption of the proposed scheme looks like linear scheme, so that its computational complexity is

$$o(x^2).$$

The specific of the proposed scheme is as follows.

- Encryption and decryption are fast.
- Alice's key choices are required for the security.
- No particular condition for the size of plaintext M, hence extension of the band length does not occur in encryption and decryption.
- Size of keys are large.

References

[1] A.K.Lenstra, H.W.Lenstra,Jr, and L. Lovasz, "Factoring Polynomials with Rational Coefficients" *Mathematische Annalen* No.261 pp.515-534, Springer-Verlag, 1982.

[2] H.W.Lenstra, "Integer Programming with a Fixed Number of Variables" *Mathmatics of Operations Research* vol.8, No.4, pp.538-548, November 1983.

[3] R.Kannan, "Improved algorithms for integet programming and related lattice problems" *In Proc.15th ACM Symp. Theory of Computing*, pp.193-206, 1983.

[4] C.P.Schnorr and H.H.Horner, "Attacking the Chor-Rivest Cryptosystem by Improved Lattice Reduction" *Advances in Cryptology,* Proc. EUROCRYPT 95, pp.1-12, Springer-Verlag LNCS 921, 1995.

Efficient Convertible Undeniable Signature Schemes

(Extended Abstract)

Markus Michels Markus Stadler

Ubilab, UBS
Bahnhofstr. 45
8021 Zurich, Switzerland

Markus.Michels@ubs.com, Markus.Stadler@ubs.com

Abstract

Undeniable signatures are a digital signatures which are not universally verifiable but can only be checked with the signer's help. However, the signer cannot deny the validity of a correct signature. An extended concept, convertible undeniable signatures, allows the signer to convert single undeniable signatures or even the whole scheme into universally verifiable signatures or into an ordinary digital signature scheme, respectively.

In this paper we propose a new convertible undeniable signature scheme and provide proofs for all relevant security properties. The scheme is based on Schnorr's signature scheme and it is efficient.

Unlike previous efficient solutions, this new scheme can be used as a basis for an efficient extension to threshold signatures, where the capability of signing (and of verifying signatures) is shared among n parties using a t out of n threshold scheme.

1 Introduction

The two most important properties of ordinary digital signatures are non-repudiation and universal verifiability. Non-repudiation guarantees that a signer cannot deny his or her commitment to a message or a contract at a later time, and the property of universal verifiability allows everybody to check the correctness of a signature. However, for certain applications, universal verifiability is not required or even not desired. Therefore, the concept of undeniable signatures was introduced by Chaum and van Antwerpen [5]. Undeniable signatures are like ordinary digital signatures, with the only difference that they are not universally verifiable. Instead, there exist (often interactive) protocols which allow the signer to convince a verifier about the validity or invalidity of a signature. Non-repudiation is still guaranteed, since the signer cannot convince the verifier that a correct signature is invalid or that an incorrect signature is valid. Various realizations of undeniable signature schemes have been proposed (see [5, 3]). Some concerns about the security of [5] have been discussed in [9, 4, 17]. Moreover, a scheme based on fail stop signatures was suggested [6] and non-interactive undeniable signatures have been introduced [18]. Harn and Yang extended the concept of undeniable signature schemes to a threshold model [14]: the capability of the signer is shared among n parties such that a coalition of at least t parties has to co-operate to sign messages and to verify signatures. They presented schemes for the cases $t = 1$ and $t = n$, however, the latter was successfully attacked by Landau [19]. Recently, Lin, Wang and Chang presented a solution that works with any t, $1 \leq t \leq n$ [20], but it is flawed as well, if signers are not assumed to be honest.

An extended concept of undeniable signatures, called convertible undeniable signatures, was suggested in [2]. With a convertible scheme, the signer can convert undeniable signatures into ordinary, i.e. universally verifiable signatures. This can be done either selectively for single signatures or totally for the whole scheme. Several realizations have been proposed: In [2], a secure but inefficient solution was presented. Practical schemes based on ElGamal signatures [10], which have been proposed in [2] and [23], were shown to be insecure [22], and the solution of [22] lacks detailed security proofs. A scheme proposed by van Heyst and Pedersen [16] can be converted to fail stop signatures, but the key length is linear in the number of signatures that can be signed. Two convertible undeniable signature schemes that are secure w.r.t. forgery have been proposed by Damgård and Pedersen in [8]. These schemes are also based on the ElGamal signature scheme and on techniques for proving that an encrypted signature is valid. One of them uses Rabin-encryption [25] and the interactive verification protocols can be done efficiently. The drawback is that the extension to a threshold scheme is hard to obtain, as a suitable composite modulus must be computed jointly. The second scheme uses ElGamal-encryption, which is somewhat inefficient, as the verification protocol requires several rounds to become secure.

In this paper we present an efficient convertible undeniable signature scheme, which can be proved secure under reasonable cryptographic assumptions. In this scheme, the signer cannot only selectively convert valid signatures into digital signatures, but he or she can also convert any invalid signature into an universally verifiable statement about this fact. The scheme is based on Schnorr's signature scheme [26] and on an efficient zero-knowledge protocol for proving the equality or inequality of discrete logarithms. Furthermore, we show how to extend this scheme to a threshold undeniable signature scheme. Very recently, Gennaro, Krawczyk and Rabin suggested a scheme [13] that is as secure as RSA with respect to forgery. It provides efficient verification protocols but it's less suitable to be a basis of a threshold scheme as a trusted dealer is usually involved to generate the composite RSA-modulus (see [12] for a threshold RSA-scheme) [1].

Our paper is structured as follows: In Section 2 we describe the model of a convertible undeniable signature scheme, then we present a building block which

[1]A way to compute an RSA modulus jointly is suggested in [1], but neither security against an active attacker nor the use of strong primes is guaranteed.

will be used in our protocol. We present our solution in Section 4 and analyze its security. Based on this solution we suggest a threshold scheme in Section 5. Further extensions are discussed in Section 6.

2 Model

A convertible undeniable signature scheme consists of the following procedures.

- A probabilistic set-up algorithm *Setup* which returns the system parameters P.
- A probabilistic key generation algorithm KeyGen_P which, on input the system parameters, returns a key pair (x, y), where x denotes the secret key and y the public key.
- A (possibly probabilistic) signature generation algorithm $\mathit{SigGen}_P(m, x)$ which, on input the secret key x and a message m, returns an undeniable signature s on m.
- A (possibly interactive) verification protocol $\mathit{Ver}_P(m, s, y, x)$ between the signer and the verifier. The signer's input is the secret key x, the message m and the 'alleged' signature s, the verifier's input is m, s and the public key y. The protocol convinces the verifier whether s is a valid signature on m or not.
- A (possibly probabilistic) individual receipt generation algorithm $\mathit{RecInd}_P(m, s, x)$ which, on input a message m, an 'alleged' signature s, and the secret key x, returns an individual receipt r which makes it possible to universally verify whether s is valid or not. A signature can selectively be converted by issuing r.
- An individual verification algorithm $\mathit{VerInd}_P(m, s, y, r)$ which, on input a message m, an 'alleged' signature s, the public key y, and a correct individual receipt r with respect to s, outputs that the receipt r is invalid with respect to s or that r is valid w.r.t. s. If the latter is true, it also outputs whether s is a valid signature on m or not.
- A (possibly probabilistic) universal receipt generation algorithm $\mathit{RecUni}_P(x)$ which, on input the secret key x, returns a universal receipt R which makes it possible to universally verify all signatures. The scheme can be totally converted by releasing R.
- A universal verification algorithm $\mathit{VerUni}_P(m, s, y, R)$ which, on input of a message m, an 'alleged' signature s, the public key y, and a universal receipt R, outputs the the receipt R is invalid or not. If the latter is true is also outputs, whether s is either a valid or a invalid signature on s.

The following statements must hold for a secure undeniable signature scheme:

- *Unforgeability:* The signature scheme is existentially unforgeable under an adaptive attack, i.e., there is no efficient algorithm that returns a valid signature s on an arbitrary message m with non-negligible probability, even if a polynomial number of valid signatures on chosen messages are given.
- *Invisibility:* There exists no efficient algorithm which, on input the public key y, a message m, and an 'alleged' signature s, can decide with non-negligible probability better than guessing whether s is either valid or not.
- *Completeness and soundness of verification:* The verification algorithms *Ver*, *VerInd* and *VerUni* are complete and sound, where completeness means that valid (invalid) signatures can always be proved valid (invalid), and soundness means that no valid (invalid) signature can be proved invalid (valid). Indirectly, this must also hold for the algorithms *RecInd*, *RecUni*.
- *Non-transferability:* A verifier participating in an execution of the interactive verification *Ver* of a signature does not obtain information that could be used to convince a third party about the correctness of a signature (although this verifier knows whether the given signature is valid or not).

3 Preliminaries and Building Blocks

In this section we first describe briefly the notation we use. Then we present an efficient interactive zero-knowledge proof for showing that two discrete logarithms are either equal or not. This protocol will be used later in our scheme, but is of independent interest. Such a proof is also called a *biproof* [11], because it proves that the input word belongs to one of two languages. A less efficient bit-wise proof for this problem has been suggested in [11].

3.1 Notations

$\mathbf{Z}_q$ denotes the ring of integers modulo q and $\mathbf{Z}_p^*$ denotes the multiplicative group modulo p. We write $a \in_R \mathcal{A}$ to indicate that the value a is chosen randomly from the set $\mathcal{A}$ according to the uniform distribution.

In the sequel, we will make use of a cyclic group $G = \langle \alpha \rangle$ of prime order q, in which computing discrete logarithms is infeasible. For instance, G could be constructed as a subgroup of the group $\mathbf{Z}_p^*$ for a suitable prime with $q|(p-1)$, or G could be an elliptic curve.

We also assume collision resistant hash functions $\mathcal{H}_\ell : \{0,1\}^* \times G \rightarrow \{0,1\}^\ell$ (with $\ell = O(\log_2 q)$, in a practical realization e.g. $\ell \approx 160$), the hash function family $\mathcal{H}_t : G^t \rightarrow \{0,1\}^\ell$ and $\mathcal{H}_G : \{0,1\}^* \rightarrow G$. If $G \subset \mathbf{Z}_p^*$, the latter could for instance be constructed by first hashing to a string of length $\log_2 p$ and then computing the $((p-1)/q)$-th power of this value.

3.2 Proving the equality or inequality of two discrete logarithms

An important component of our realization of a convertible undeniable signature scheme is an efficient protocol that allows a prover to convince a verifier about the equality or inequality of two discrete logarithms, such that no additional information about these logarithms is leaked. More precisely, assume the prover knows the discrete logarithm x of $y = \alpha^x$ and wants to allow the verifier to decide whether $\log_\beta z = \log_\alpha y$ for given group elements β and z. Therefore, the prover and the verifier execute the following protocol.

1. The verifier chooses random values $u, v \in \mathbf{Z}_q$, computes $a := \alpha^u y^v$, and sends a to the prover.
2. The prover chooses random values $k, \tilde{k}, w \in \mathbf{Z}_q$, computes $r_\alpha := \alpha^k$, $r_\beta := \beta^k$, $\tilde{r}_\alpha := \alpha^{\tilde{k}}$, and $\tilde{r}_\beta := \beta^{\tilde{k}}$, and sends r_α, r_β, $\tilde{r}_\alpha$, $\tilde{r}_\beta$, and w to the verifier.
3. The verifier opens his commitment a by sending u and v to the prover.
4. If $a \neq \alpha^u y^v$ the prover halts, otherwise he computes $s := k - (v+w)x \pmod q$, $\tilde{s} := \tilde{k} - (v+w)k \pmod q$ and sends s and $\tilde{s}$ to the verifier.
5. The verifier first checks whether $\alpha^s y^{v+w} = r_\alpha$, $\alpha^{\tilde{s}} r_\alpha^{v+w} = \tilde{r}_\alpha$, and $\beta^{\tilde{s}} r_\beta^{v+w} = \tilde{r}_\beta$ and then concludes:

$$\begin{cases} \text{if } \beta^s z^{v+w} = r_\beta & \text{then } \log_\beta z = \log_\alpha y \\ \text{if } \beta^s z^{v+w} \neq r_\beta & \text{then } \log_\beta z \neq \log_\alpha y \end{cases}$$

The following theorem can be proved.

Theorem 1 *The above protocol is complete and sound. It is zero-knowledge under the assumption that there exists no algorithm running in expected polynomial time which decides with non-negligible probability better than guessing whether two discrete logarithms are equal.*

Proof (Sketch): The completeness of the protocol is obvious because an honest prover can always convince an honest verifier of the equality or inequality of the two discrete logarithms. To prove the soundness property, it is important to note that the commitment a sent by the verifier in the first message does not reveal any information (in an information-theoretic sense) about the value v and that therefore the "challenge" $v + w$ is truly random for the prover. Based on this observation it can easily be shown that successful cheating is only possible with negligible probability.
To prove the zero-knowledge property of the protocol, we show how to construct a simulator that returns a protocol transcript with a probability distribution indistinguishable from the distribution of a verifier's protocol view. The simulator uses the verifier as a black-box, i.e., it works independently from the verifier's strategy.

1. The verifier's algorithm is used to compute the commitment a.

2. The simulator randomly chooses s, $\tilde{s}$, w, and $c \in \mathbf{Z}_q$ and computes $r_\alpha := \alpha^s y^c$, $\tilde{r}_\alpha := \alpha^{\tilde{s}} r_\alpha^c$, and $\tilde{r}_\beta := \beta^{\tilde{s}} r_\beta^c$. The value r_β is computed as $\beta^s z^c$ or chosen randomly from $G \backslash \{\beta^s z^c\}$, depending on which protocol outcome should be simulated.

3. The verifier's algorithm is used to compute u and v on input the values r_α, r_β, $\tilde{r}_\alpha$, $\tilde{r}_\beta$, and w.

4. If $a \neq \alpha^u y^v$ the simulator returns as protocol transcript the values r_α, r_β, $\tilde{r}_\alpha$, $\tilde{r}_\beta$, w, and `halt`. Otherwise, the simulator repeats steps 2 and 3 until the commitment a is correctly opened with values u' and v'; step 2 is modified such that c is not chosen randomly but set to $c = v + w$.

5. If the simulator finally stops and if $u = u'$, it returns as transcript the values r_α, r_β, $\tilde{r}_\alpha$, $\tilde{r}_\beta$, w, s and $\tilde{s}$. If $u \neq u'$ then the discrete logarithm x of y to base α can be extracted and the simulator returns as transcript the value of x.

It can easily be seen that the expected number of repetitions of steps 2 and 3 is constant and that therefore the simulator runs in expected polynomial time. Let us now explain briefly why the output of the simulator is computationally indistinguishable from a protocol view:

- the probability that a `halt` occurs is the same as in a protocol execution.
- the probability that the simulator returns x, the discrete logarithm of y to the base α, is negligible because of the assumption (otherwise, the simulator could be used to test the equality of discrete logarithms with non-negligible probability).
- all values, except $\tilde{r}_\beta$, are distributed according to the same distribution in the simulator's output and the protocol view, and the two distributions of $\tilde{r}_\beta$ can distinguished only by deciding whether $\log_\alpha \tilde{r}_\alpha$ equals $\log_\beta \tilde{r}_\beta$, which is not possible according to the assumption.

□

To obtain a designated verifier proof [18], the commitment a must be computed as $a := \alpha^u y_V^v$, where y_V is the verifier's public key and $w = \mathcal{H}_{10}(\alpha, y, \beta, z, r_\alpha, r_\beta, \tilde{r}_\alpha, \tilde{r}_\beta, a, y_V)$. This proof is non-interactive, but as the verifier can generate this proof as well, it's not a receipt.

The protocol can also easily be turned into a non-interactive argument by omitting the commitment a, setting w to 0 and computing v as $v = \mathcal{H}_8(\alpha, y, \beta, z, r_\alpha, r_\beta, \tilde{r}_\alpha, \tilde{r}_\beta)$.

4 New convertible signature scheme

We describe our scheme and analyze its security.

4.1 Basic scheme

The protocol can be described as follows:

1. *Set up:* The system parameters are G, α, q, $\mathcal{H}_\ell$, and $\mathcal{H}_G$.
2. *Key generation:* Each user picks at random two numbers x_1 and x_2 from $\mathbf{Z}_q$ as secret keys and computes the public keys $y_1 := \alpha^{x_1}$ and $y_2 := \alpha^{x_2}$.
3. *Signature generation:* A message m is signed in the following way:

   (a) $k \in_R \mathbf{Z}_q$, $r := \alpha^k$, $\tilde{r} := \mathcal{H}_G(r)^{x_2}$

   (b) $c := \mathcal{H}_\ell(m, \tilde{r})$

   (c) $s := k - cx_1 \pmod q$

   The resulting signature on m is the pair $(\tilde{r}, s)$.
4. *Interactive verification:* The signature can be verified or denied by interactively proving the equality or inequality of the discrete logarithms of $\tilde{r}$ and y_2 to the bases $\mathcal{H}_G(\alpha^s y_1^{\mathcal{H}_\ell(m,\tilde{r})})$ and α, respectively, using the interactive protocol described in section 3.2. Alternatively, the non-interactive designated verifier proof outlined in section 3.2 can be used.
5. *Individual receipt generation:* To selectively convert a signature, this proof is turned into an non-interactive argument using the non-interactive argument protocol described in section 3.2. This argument is the individual receipt. Note that this also allows to make it publicly verifiable that a given signature is *invalid.*
6. *Individual verification:* By checking the validity of the individual receipt, a verifier can see whether the related signature is either valid or invalid.
7. *Universal receipt generation:* In order to totally convert all undeniable signatures into digital signatures, the secret key x_2 is published as universal receipt.
8. *Universal verification:* The verifier checks whether

$$\mathcal{H}_G(\alpha^s y_1^{\mathcal{H}_\ell(m,\tilde{r})})^{x_2} \equiv \tilde{r}$$

   holds.

4.2 Efficiency

We analyse the efficiency of our scheme, where G is chosen as the multiplicative subgroup of order q of Z_p^* and q is small. Thus in general we have short exponents. Only for the computation of $\mathcal{H}_G$ we need a full exponentiation, as we exponentiate an output of a hash function $h : \{0,1\}^* \rightarrow Z_p$ with $(p-1)/q$ to get an element in G. Let $M_l(i)$ denote the number of $|p|$-bit multiplications that are required to compute i cascaded exponentiations of the form $\prod_{j=1}^{i} r_j^{d_j}$.

Operations	Signer	Verifier
Signature generation	$2 \cdot M_Q(1) + M_{P-Q}(1)$	–
Interactive verification	$5 \cdot M_Q(1) + M_{P-Q}(1)$	$4 \cdot M_Q(2) + M_{P-Q}(1)$
Selective conversion	$5 \cdot M_Q(1) + M_{P-Q}(1)$	–
Individual verification	–	$4 \cdot M_Q(2) + M_{P-Q}(1)$
Total conversion	–	–
Universal verification	–	$M_Q(1) + M_Q(2) + M_{P-Q}(1)$

Figure 1: Costs for operations

where l is the length of the exponents. Let $P = |p|$ and $Q = |q|$. Figure 1 shows the costs for the different operations.

Let us illustrate the costs in an example with $P = |p| = 1024$ and $Q = |q| = 256$. Using methods of [29], we have $M_P(1) = 308, M_Q(2) = 373$ multiplications for an exponentiation with one and two 256-bit exponents, respectively and $M_{P-Q}(1) = 902$ multiplications for an exponentiation with one 768-bit exponent. The costs for the different operations in this example are described in Figure 2.

Operations	Signer	Verifier
Signature generation	1518	–
Interactive verification	2442	2394
Selective conversion	2442	–
Individual verification	–	2394
Total conversion	0	–
Universal verification	–	1583

Figure 2: Costs in number of 1024-bit multiplications

Even better results can be achieved if G is chosen to be an elliptic curve over a finite field.

4.3 Security analysis

We distinguish the analysis in the parts unforgeability, invisibility, untransferability and completeness & soundness of verification.

Unforgeability

Theorem 2 *Under the assumption that the hash functions $\mathcal{H}_\ell$ and $\mathcal{H}_G$ are truly random functions, forging valid signatures is equivalent to forging Schnorr signatures.*

Proof (Sketch): In the converted scheme, with known x_2, let the function $\mathcal{H}_N : \{0,1\}^* \times G \to \{0,1\}^\ell$ be defined as

$$\mathcal{H}_N(m,r) := \mathcal{H}_\ell(m, \mathcal{H}_G(r)^{x_2}).$$

The converted undeniable signature scheme is equivalent to a Schnorr signature scheme using the hash function $\mathcal{H}_N$. But because $\mathcal{H}_N$ is indistinguishable from a truly random function (the only differences are possible collisions of $\mathcal{H}_G$), the converted signature scheme is secure. As a consequence, also the non-converted scheme has to be secure. □

Invisibility

To prove the invisibility of the scheme prior to conversion, we need an additional assumption, the Decision Diffie-Hellman assumption. We first define the two sets

$$\mathcal{DH} := \{(\alpha, y, \beta, z) \in G^4 \mid \log_\alpha y = \log_\beta z\}$$

$$\mathcal{NDH} := \{(\alpha, y, \beta, z) \in G^4 \mid \log_\alpha y \neq \log_\beta z\}$$

of Diffie-Hellman and non-Diffie-Hellman 4-tuples.

Assumption For all probabilistic polynomial time algorithms $\mathcal{A} : G^4 \to \{0, 1\}$, the two probability distributions

$$\mathrm{Prob}_{T \in_R \mathcal{DH}}\,[\mathcal{A}(T) = 1] \quad \text{and} \quad \mathrm{Prob}_{T \in_R \mathcal{NDH}}\,[\mathcal{A}(T) = 1]$$

are computationally indistinguishable (the probabilities are taken over the random coin tosses of $\mathcal{A}$ and over the random choices from $\mathcal{DH}$ and $\mathcal{NDH}$, respectively).

Theorem 3 Provided that the assumption holds and that the the hash function $\mathcal{H}_G$ can't be distinguished from a random function, verifying a given signature without the assistance of the signer can be done only with negligible probability, even if a polynomial number of valid signatures is known.

Proof (Sketch): The basic idea is to transform an instance (α, y, β, z) of the above problem into an instance of the signature scheme (we assume that the instance (α, y, β, z) is already randomized, e.g. see [28]). Concretely, let α be the generator, let y be y_2, and let $y_1 = \alpha^{x_1}$ for a randomly chosen x_1. Furthermore, we simulate the hash function $\mathcal{H}_G$ to be able to generate signatures. To generate a correct signature, we proceed as described, except that we set $\mathcal{H}_G(r) = \alpha^t$ and $\tilde{r} = y_2^t$ for a randomly chosen t (this guarantees that the signature is correct, even if we don't know x_2). For the signature whose verification is to be equivalent to solving the above problem, let $\mathcal{H}_G(r) = \beta$ and $\tilde{r} = z$: this signature is only valid if the above 4-tuple is a Diffie-Hellman tuple. Therefore, if there was any efficient algorithm which can decide (better than guessing) whether this signature is valid, such an algorithm could also be used to solve the Decision-Diffie-Hellman problem, but this would lead to a contradiction.

Non-transferability

Non-transferability follows directly from the zero-knowledge property of the interactive protocol for proving the equality/inequality of discrete logarithm.

Soundness & completeness of verification

Before conversion, these properties follow from the soundness and the completeness property of the used zero-knowledge protocols. In the selectively converted version, these properties are inherited because of the impossibility to issue wrong receipts provided the used hash function is collision resistant.

5 Robust convertible undeniable threshold signature scheme

Using standard techniques for verifiable sharing of discrete logarithms [23] and methods from secure multi-party computations, our basic convertible undeniable signature scheme presented in Section 4 can be adapted for the threshold scenario.

5.1 Model

In a convertible undeniable threshold signature scheme there is a group of n signers such that any coalition of at least t signers can jointly sign a message.

The communication model is as follows: During the signature generation, it is assumed that the signers can broadcast messages to each other and the signers check proofs of other signers. During the interactive verification each signer has a channel to communicate to the verifier.

The threshold scheme consists of the same procedures as listed in section 2, however, the key generation algorithm must output shares of the secret key for each signer such that only at least t signer are able to sign a document on behalf of the group, and the other algorithms should be adopted accordingly.

With regard to security against dishonest signers, we have to distinguish between passive and active cheaters. Passive cheaters follow the protocols honestly but try to gain additional knowledge by pooling their information, while active cheaters can even deny service or send wrong values.

5.2 New scheme

We outline the robust convertible undeniable threshold signature scheme, based on the scheme given in the previous section.

It is assumed that x_1 and x_2 are shared among n provers using a verifiable t out of n threshold secret sharing scheme (for details see [27, 23]). A share of signer i of a variable or value a is denoted $Share_i(a)$. Given at least t distinct (correct) shares, the value of a can be reconstructed using Lagrange interpolation (see [23]). We also assume that shadows of the form $\alpha^{Share_i(x_1)}$ are publicly known.

A message m is signed in the following way by d signers ($t \leq d \leq n$):

1. The signers jointly compute $r := \alpha^k$ in a distributed manner. Each signer i gets a verifiable share $Share_i(k)$. A shadow $\alpha^{Share_i(k)}$ is revealed and publicly known.

2. Each signer computes $\tilde{r}_i := \mathcal{H}_G(r)^{Share_i(x_2)}$ and proves interactively and in zero-knowledge to all other signers that this is correctly done. This requires only a simple zero knowledge proofs of equality of discrete logarithms. If at least t signers are honest, each of them can compute $\tilde{r} = \mathcal{H}_G(r)^{x_2}$ by combining the values $\tilde{r}_i$ of the honest signers.

3. Each signer computes $c := \mathcal{H}_\ell(m, \tilde{r})$.

4. Signer i computes $Share_i(s) := Share_i(k) - c \cdot Share_i(x_1) \pmod q$ and broadcasts this value to all other signers. These shares are checked using the revealed shadows. If at least t signers are honest, the value s can be reconstructed, which is then sent to the verifier.

The verification protocols, as well as the procedures for generating receipts, can be adapted from the basic protocols described in Section 4 in an analogous way.

5.3 Security analysis

For an outsider attacker the security analysis does not differ from the analysis given in the previous section.

As d signers $(t \le d \le n)$ sign a message, we have to assume that there are at most $t-1$ dishonest signers and among those, there must not be more than $min(d-t, t-1)$ actively cheating signers. We further have to assume that the verifier has no unconditional trust to any signer.

- *Key Generation:* It was already shown that any group of $t-1$ members does not obtain any knowledge concerning the secret keys and it's impossible to cheat for signers during the key generation protocol [23].

- *Unforgeability:* The dishonest signer could pick a document m and try to get a threshold signature on it, although the honest signers are not aware of this document (e.g. they might think to sign another document m'). Such an attack was successful in some multi party signature schemes as pointed out in [21]. Here, however, such an attack is not successful. It is impossible to transform a partial signature of a honest signer on m' to a partial signature on m, as r can't be fully determined by the dishonest signer and r and m' are both input of the hash function.

- *Invisibility:* Invisibility still holds even if $t-1$ signers send the knowledge they gained during signature generation to the verifier provided the verifier does not trust any signer unconditionally. In fact, r is known by the verifier anyway, but some $\mathcal{H}_G(r)^{Share_i(x_2)}$ and the partial signatures can be send him as extra knowledge. However, the relation $\tilde{r} = \mathcal{H}_G(r)^{x_2}$ can't be proved by $t-1$ signers. The partial signatures are useless as well, as they can be simulated by one signer.

- *Non-transferability:* As the interactive verification is zero-knowledge and the information from selective conversion for given signatures does not help to verify another alleged signature, non-transferability holds.
- *Robustness:* As only up to $min(d-t, t-1)$ signers are totally controlled by the attacker, the signature can always be generated by the t remaining signers, that are either honest or only passively cheating.

6 Further extensions

Clearly, the security model can be somewhat strengthend by updating the individual shared parameters from time to time without changing the public key [15]. Furthermore, the threshold schemes can be transformed into shared signature schemes with arbitrary access structure just by substituting the used secret sharing scheme. It's also possible to share the verifier by using ideas proposed in [23, 18] or to use publicly verifiable secret sharing [28] instead of the non-publicly verifiable sharing scheme.

References

[1] D.Boneh, M.Franklin, "Efficient generation of shared RSA keys", LNCS, Proc. Crypto'97, Springer Verlag, (1997).

[2] J.Boyar, D.Chaum, I.Damgård, T.Pedersen, "Convertible undeniable signatures", LNCS 537, Proc. Crypto '90, Springer Verlag, (1991), pp. 189–205.

[3] D.Chaum, "Zero-knowledge undeniable signatures", LNCS 473, Proc. Eurocrypt '90, Springer Verlag, (1991), pp. 458–464.

[4] D.Chaum, "Some weakness of "Weaknesses of Undeniable Signatures"", LNCS 547, Proc. Eurocrypt '91, Springer Verlag, (1992), pp. 554–556.

[5] D.Chaum, H. van Antwerpen, "Undeniable Signatures", LNCS 435, Proc. Crypto '89, Springer Verlag, (1990), pp. 212–216.

[6] D.Chaum, E. van Heyst, B.Pfitzmann, "Cryptographically strong undeniable signatures, unconditionally secure for the signer", LNCS 576, Proc. Crypto'91, (1992), pp. 470–484.

[7] D.Chaum, T.Pedersen, "Wallet databases with observers", LNCS 740, Proc. Crypto'92, (1993), pp. 89 – 105.

[8] I.Damgård, T.Pedersen, "New convertible undeniable signature schemes", LNCS 1070, Proc. Eurocrypt'96, Springer Verlag, (1996), pp. 372–386.

[9] Y.Desmedt, M.Yung, "Weaknesses of undeniable signature schemes", LNCS 547, Proc. Eurocrypt '91, Springer Verlag, (1992), pp. 205–220.

[10] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms", LNCS 196, Proc. Crypto'84, (1985), pp. 10–18.

[11] A.Fujioka, T.Okamoto, K.Ohta, "Interactive Bi-Proof Systems and undeniable signature schemes", LNCS 547, Proc. Eurocrypt '91, Springer Verlag, (1992), pp. 243–256.

[12] R.Gennaro, S.Jarecki, H.Krawczyk, T.Rabin, "Robust and efficient sharing of RSA functions", LNCS 1109, Proc. Crypto'96, Springer Verlag, (1996), pp. 157–172.

[13] R.Gennaro, H.Krawczyk, T.Rabin, "RSA-based undeniable signatures", LNCS, Proc. Crypto'97, Springer Verlag, (1997).

[14] L.Harn, S.Yang "Group-oriented undeniable signature schemes without the assistance of a mutually trusted party", LNCS 718, Proc. Asiacrypt'92, Springer Verlag, (1993), pp. 133–142.

[15] A.Herzberg, S.Jarecki, H.Krawczyk, M.Yung, "Proactive secret sharing or: How to cope with perpetual leakage", LNCS 963, Proc. Crypto'95, Springer Verlag, (1995), pp. 339–352.

[16] E.van Heyst, T.Pedersen, "How to make efficient fail stop signatures", LNCS 658, Proc. Eurocrypt'92, Springer Verlag, (1993), pp. 366–377.

[17] M.Jakobsson, "Blackmailing using undeniable signatures", LNCS 950, Proc. Eurocrypt'94, Springer Verlag, (1995), pp. 425–427.

[18] M.Jakobsson, K.Sako, R.Impagliazzo, "Designated verifier proofs and their applications", LNCS 1070, Proc. Eurocrypt'96, Springer Verlag, (1996), pp. 143–154.

[19] S.Landau, "Weaknesses in some threshold cryptosystems", LNCS 1109, Proc. Crypto'96, Springer Verlag, (1996), pp. 74–83.

[20] C.-H.Lin, C.-T.Wang, C-C.Chang, "A group oriented (t,n) undeniable signature scheme without trusted center", LNCS 1172, Information Security and Privacy, ACISP'96, Springer Verlag, (1996), pp. 266–274.

[21] M.Michels, P.Horster, "On the risk of disruption in several multiparty signature schemes", LNCS 1163, Proc. Asiacrypt '96, Springer Verlag, (1996), pp. 334–345.

[22] M.Michels, H.Petersen, P.Horster, "Breaking and repairing a convertible undeniable signature scheme", Proc. 3rd ACM Conference on Computer and Communications Security, ACM Press, (1996), pp. 148–152.

[23] T.P.Pedersen, "Distributed provers with applications to undeniable signatures", LNCS 547, Proc. Eurocrypt '91, Springer Verlag, (1992), pp. 221–242.

[24] D.Pointcheval, J.Stern, "Security proofs for signature schemes", LNCS 1070, Proc. Eurocrypt'96, Springer Verlag, (1996), pp. 387–398.

[25] M.O.Rabin, "Digitalized signatures and public-key functions as intractable as factorization", MIT/LCS/TR-212, MIT Lab. for Computer Science, Cambridge, Mass., 1979.

[26] C.P.Schnorr, "Efficient signature generation for smart cards", Journal of Cryptology, Vol. 4, (1991), pp. 161–174.

[27] A.Shamir, "How to share a secret", Communications of the ACM, Vol.22, (1979), pp. 612–613.

[28] M.Stadler, "Publicly verifiable secret sharing", LNCS 1070, Proc. Eurocrypt'96, Springer Verlag, (1996), pp. 190–199.

[29] S.-M.Yen, C.-S.Laih, "The fast cascade exponentiation algorithm and its applications on cryptography", LNCS 718, Proc. Asiacrypt'92, Springer Verlag, (1993), pp. 447–456.

One-Response Off-Line Digital Coins

Khanh Quoc Nguyen, Yi Mu, and Vijay Varadharajan
Distributed System and Network Security Research Unit
Department of Computing University of Western Sydney, Nepean,
Kingswood, NSW 2747, Australia
Email: {qnguyen,yimu,vijay}@st.nepena.uws.edu.au

Abstract

Current off-line electronic cash systems are often too complex to implement. In this paper, we propose an efficient off-line digital cash payment system that still maintains basic security features such as double-spending detection and client anonymity. Using a one-way hash function chain, our method verifies the clients to perform one single online computation (response) for the whole payment.

Keywords: Secure Electronic Commerce, Cryptography

1 Introduction

Off-line digital cash systems are more preferable than on-line cash systems, since in off-line digital cash systems banks do not need to be involved in payment processes. There have always been two major concerns for off-line systems: double-spending and customer's privacy. In particular, double-spending is a serious threat for off-line schemes. Customer's privacy protection, including untraceability of transactions and anonymity of customer, is the other very important requirement for any electronic payment scheme.

Current off-line electronic cash systems [3, 1, 4] tend to provide double-spending detection, client anonymity, and transaction untraceability. However, as there is always a trade-off between double-spending detection and transaction untraceability, it is often complex to be realized at a low cost. Even in the most efficient systems[1, 4], many discrete exponential computations are required for each digital monetary unit in order to achieve the untraceability. To design an electronic payment system that allows small amount of payment, heavy use of discrete exponential computation must be avoided. In fact, this requirement makes all current off-line cash systems economically infeasible.

In this paper, we propose an efficient approach to off-line digital cash schemes that makes small payment amounts possible. Our scheme provides client anonymity and double-spending detection. In our proposal, we use one-way hashing functions to create some links between coins spent in the same transaction. This method requires the client to perform only **one** major computation and there are no discrete exponential computations in the payment phase. This feature leads to a significant improvement in computational efficiency in contrast to all previously proposed schemes.

The remainder of this paper is organised as follows. Section 2 gives the cryptographic background and introduces Schnorr's signature scheme. The one-time feature of Schnorr's scheme forms the main basis of our cash protocol. In section 3, we describe the working principles of our cash scheme. Section 4 discusses various security and efficiency features of the our protocol; we also includes some comparisons between our scheme and previous works. Finally Section 5 concludes our paper.

2 Schnorr's one-time signature scheme

The security of Schnorr's signature scheme[5] depends on the difficulty of calculating discrete logarithms. Users in the system can share a random number g and two prime numbers, p and q such that q is a prime factor of $p-1$, $g \neq 1$ and $g^q \equiv 1 \bmod p$.

To generate a particular pair of private/public key, a customer (say, Alice) chooses a random number s as her private key, $0 < s < q$. Alice then computes her public key v as:

$$v := g^{-s} \bmod p$$

To sign a message m, Alice picks a random number $r \in \mathcal{Z}_q^*$ and does the following computations:

$$\begin{aligned} x &:= g^r \bmod p \\ c &:= h(m\|x) \\ y &:= (r + sc) \bmod q \end{aligned}$$

where $h(.)$ is a suitable collision-free one-way hash function. The signature on the message m is the pair (c, y). To verify the signature, Bob checks:

$$x \stackrel{?}{=} g^y v^c \bmod p$$

and tests if c is equal to $h(m\|x)$. If the test is OK, the signature is valid.

The value r must be treated as one-time number. It must not be used more than once to generate different signatures. If Alice has used r to sign two different messages m and m', then one has two signatures (c, y) and (c', y'). With these two signatures, one can compute Alice's private key s as follows:

$$s = (\frac{y - y'}{c - c'}) \equiv (\frac{(r + sc) - (r + sc')}{c - c'}) \bmod q$$

Schnorr's scheme allows most of the computation for signature generation to be completed independent of the message being signed. This forms an important feature in the efficiency of our scheme.

3 Our off-line Payment Scheme

In our system, there are three main players, the bank, clients and vendors. We denote the bank by $\mathcal{B}$, a generic vendor by $\mathcal{V}$ and a generic client by $\mathcal{C}$. We assume that each *coin* in this scheme represents a monetary unit. The face value of each coin is decided by the bank. We denote that a coin with a face value c_i as C_i. We also assume that the bank has a RSA public/secret key (e, d) with the composite modulus n is a product of two large prime numbers, q_1, q_2, a number g such that $g^q \equiv 1 \bmod p$ and $gcd(g-1, n) = 1$. The values of g, p, q, n, and e are public information.

Also to simplify our presentation, in this section only, we use " $=$ " to represent a computation; " $==$ " to represent a comparison and " $\equiv$ " to denote a mathematical transformation. Transformations are only used to give the readers a clearer picture of our protocol, they are not carried out by any respective party.

3.1 Account's Opening Phase

When C wishes to open an account at $\mathcal{B}$, after identifying himself to $\mathcal{B}$, C uses a zero-knowledge process to obtain a blind-signature from $\mathcal{B}$ on $h(g^U \bmod p)$ as $(h(g^U))^d \bmod n$. U is constructed as $U = I\|k$ $(0 < U < q)$, where $\|$ denotes a concatenation of bits, k is a random number, and I is the client identity registered with the bank (also referred as the client's bank account number). The bank should not have any knowledge about the value of k and consequently the value of U. There have been several such zero-knowledge processes described in literatures; see for instance[8, 6].

The length of I and k should be fixed, at least 80 bits each, so that given U, it is feasible to obtain I. The zero knowledge proof also ensures that the order of I and k cannot change. After the account's opening phase, the client has an anonymous bank certificate $Cert$ as $(h(g^U))^d$. This certificate would remain anonymous as long as nobody is able to compute U. Extracting U from $Cert$ is infeasible unless the client double-spends under the discrete logarithms assumption (see section 4 for further discussions). After the account's opening process, C stores $Cert$ and $g_C = g^U \bmod p$.

3.2 Withdrawal Phase

Before withdrawing any money from the bank, the client C proves his ownership of I to $\mathcal{B}$. If the client wishes to withdraw n coins, he chooses a random number c_n and computes $c_i = h(c_{i+1})$ for $\forall i \in \{1, \ldots, n-1\}$. For each c_i, C uses a blind signature technique [2] to withdraw an anonymous coin from $\mathcal{B}$ using the following protocol (see Figure 1):

1. C generates a random number x_i, computing: $x'_i = g^{x_i} \bmod p$, $m_i = h(c_i\|x'_i)$.
2. C then uses blind signature technique [2] to obtain a bank signature on m_i by choosing a blind factor r_i and sending $t_i = r_i^e \cdot m_i \bmod n$ to $\mathcal{B}$. $\mathcal{B}$ signs the value of t_i and return the signature as t'_i. The client then removes the blind factor r_i to obtain the bank blind signature $m'_i = t'_i/r_i = m_i^d$.

For each signature, the bank deducts the client's account by an equivalent value of a coin. After the withdrawal, C has each coin C_i with a face value of c_i in the form of $[h(c_i\|x'_i)]^d \bmod n$. It is unforgeable unless the factorisation of n is known[3]. For each coin C_i, C stores $[c_i, x_i, x'_i, m'_i]$.

$$\begin{array}{ccc}
\mathcal{C} & & \mathcal{B} \\
x_i, r_i \in \mathcal{Z} & & \\
x'_i = g^{x_i} \bmod p & & \\
m_i = h(c_i \| x'_i) & & \\
t_i = r_i^e \cdot m_i \bmod n & & \\
 & \xrightarrow{t_i} & \\
 & & t'_i = t_i^d \bmod n \\
 & \xleftarrow{t'_i} & \\
m'_i = t'_i/(r_i) \equiv m_i^d \bmod n & & \\
m'^e_i == m_i \bmod n & &
\end{array}$$

Figure 1: Illustration of the withdrawal protocol.

3.3 Payment Phase

When the client wants to spend the coin chain $C_1, C_2, \ldots, C_n$ to $\mathcal{V}$, he must spend them in the order $C_1, C_2, \ldots, C_n$.

Without the loss of generality, we assume that $\mathcal{C}$ has already spent all the coins $C_0, C_1, \ldots, C_{i-1}$ in some previous payments. Now if $\mathcal{C}$ wishes to pay some coins to $\mathcal{V}$, $\mathcal{C}$ must send the coins to $\mathcal{V}$ in the exact sequence $C_i, C_{i+1}, \cdots, C_j, \cdots$ according to the following process:

- For the first coin C_i (see also Figure 2):

$$\begin{array}{ccc}
\mathcal{C} & & \mathcal{V} \\
 & & a \in \mathcal{Z} \\
 & \xleftarrow{\quad a \quad} & \\
b = (x_i - Ua) \bmod q & & \\
 & \xrightarrow{Cert, g_C, b, c_i, x'_i, m'_i} & \\
 & & h(g_C) == Cert^e \bmod n \\
 & & h(c_i \| x'_i) == m'^e_i \bmod n \\
 & & g_C^a g^b == x'_i \bmod p
\end{array}$$

Figure 2: Illustration of the payment protocol for the first coin.

1. $\mathcal{V}$ generates a random challenge a and sends it to $\mathcal{C}$. This challenge should be unique for each transaction. For example, it can be computed as $a = h(\mathcal{V} \| Date \| Time)$.
2. $\mathcal{C}$ computes the response $b = x_i - Ua \bmod q$ for the challenge a and sends it along with $(Cert,\ g_C,\ b, c_i, x'_i, m'_i)$ to $\mathcal{V}$. The response b is also considered as Schnorr's one-time signature on the message a, where x_i is one-time material.

$\mathcal{V}$ accepts the coin if and only if $Cert$ and m'_i are valid bank signatures on g_C, $h(c_i \| x'_i)$ respectively and $g_C^a . g^b == x'_i \bmod p$

- For every coin C_j, thereafter (see also Figure 3):

$$\begin{tikzcd} \mathcal{C} \arrow[r, "c_j{,}x_j{,}m'_j"] & \mathcal{V} \\ & m'^e_j == h(c_j \| g^{x_j}) \\ & h(c_j) == c_{j-1} \end{tikzcd}$$

Figure 3: Illustration of the payment protocol.

$\mathcal{C}$ sends $[x_j, c_j, m'_j]$ to $\mathcal{V}$. $\mathcal{V}$ accepts the coin C_j if and only if $h(c_j) = c_{j-1}$ (where c_{j-1} obtained from the previous coin) and m'_j is a valid bank signature on $h(c_j \| g^{x_j})$.

For the sake of convenience, let us name the first coin C_i as **signed coin** and all the other coins C_j as **normal coins.**

3.4 Deposit Phase

In deposit phase, $\mathcal{V}$ deposits all the received coins at $\mathcal{B}$ by sending $[Cert, g_C, a, b, c_i, x'_i, m'_i]$ for each signed coin and $[c_j, x_j, m'_j]$ for each normal coin. $\mathcal{B}$ goes through exactly the same verification process as $\mathcal{V}$ did in the payment phase. If everything is OK, $\mathcal{B}$ pays $\mathcal{V}$ an equivalent amount of money and stores $[a, b, c_i]$ for the first coin, $[c_j, x_j]$ for each other coin in its coin database.

4 Discussion

In this section, we will closely examine security and efficiency features of the system, including double-spending detection, client anonymity, and efficiency.

4.1 Double Spending

Double-spending occurs when $\mathcal{C}$ double spends some coins in the hope that $\mathcal{B}$ cannot detect the identity. In our protocol, double-spending is detected as follows:

Proposition 1 *If the coin C_i is spent in a transaction $\mathcal{T}$ as a normal coin, then the coin C_{i-1} is spent in $\mathcal{T}$.*

Proof: Due to the nature of our scheme, when C_i is spent in $\mathcal{T}$ and C_{i-1} is **not** spent in $\mathcal{T}$, C_i must be the first coin spent in the transaction, i.e. C_i is a signed coin, but not a normal coin. Therefore, if C_i is spent as a normal coin in a transaction, then C_{i-1} is also spent in the same transaction.

Definition 1 *The coin C_i is called the first double-spent coin if C_i is the coin with the the smallest subscript i among double-spent coins.*

Proposition 2 *The first double-spent coin C_i must be spent as a signed coin in at least one transaction.*

Proof: If C_i was spent as a normal coin in two transactions, $\mathcal{T}_1$ and $\mathcal{T}_2$, according to Proposition 1, C_{i-1} was also spent in both transactions, i.e. C_{i-1} was double spent. This contradicts the assumption, in that C_i is the double-spent coin with the smallest subscript. Hence, C_i cannot be spent as a normal coin in two transactions. In the other words, C_i is spent as a signed coin at least once.

When $\mathcal{C}$ double spent some coins, for the first double-spent coin C_i, it must be a signed coin in at least one transaction. So there are only two possibilities: C_i is spent as either a signed coin in the both transactions or as a signed coin in one transaction and as a normal coin in another transaction.

- **Signed-Signed coin:** $\mathcal{C}$ spends C_i as a signed coin twice, i.e. for two different challenges a and a', $\mathcal{B}$ therefore has $b = x_i - Ua \pmod q$ and $b' = x_i - Ua' \pmod q$. $\mathcal{B}$ can easily find U by computing:

$$U = \frac{b - b'}{a' - a} \bmod q$$

- **Signed-Normal coin:** $\mathcal{C}$ spends C_i twice, once as a normal coin and the other as a signed coin. $\mathcal{B}$ therefore has a and $x_i - Ua$ from the signed coin and x_i from the normal coin. These information are sufficient to compute U.

So in either case, the value U can be computed. After obtaining U, $\mathcal{B}$ extracts I and matches it with the client's ID stored in its database. Once, a match is found, $\mathcal{B}$ asks $\mathcal{C}$ to reveal the value U incorporated in his $Cert$. If this value matches the value U obtained by $\mathcal{B}$ from the first double-spent coin, $\mathcal{C}$ must have double spent the same coin. The evidence is *undeniable* because U is client's secret information, which is infeasible for anyone else to compute unless the client had double spent a coin.

4.2 Anonymity

Client anonymity is protected unconditionally in our protocol. The zero-knowledge process used in the account's opening phase completely hides the identity of the client. The bank will not be able to link $Cert$ to $\mathcal{C}$'s ID, once $Cert$ is issued. On the other hands, our coins are blindly signed by the bank so the bank cannot trace any particular coin to any particular client.

During the payment process, the client only has to show $Cert$, which is an anonymous certificate and reveals $x_i - Ua \pmod q$ for each signed coin and x_j for each normal coin. For two different coins, as their corresponding (x_i, x_j) are chosen at random, they are different and unlinkable. Having only a, $\mathcal{B}$ cannot obtain U from $x_i - Ua \pmod q$ (since x_i is chosen at random).

4.3 Efficiency

The account's opening phase is a one-off process, so eventhough the zero-knowledge process is inefficient, it will not affect the efficiency of our system for any transaction later on.

Our withdrawal phase is very efficient. To withdraw a coin, ignoring the number of hash operations, $\mathcal{C}$ only has to compute *two* exponentiations. This is computationally more efficient than all current off-line electronic cash schemes. The number of discrete exponentiations required in Chaum's[3], Ferguson's[4], Brands'[1] protocols, are forty, seventeen, and ten respectively. In contrast to these schemes, our protocol needs only two multiplication operations.

In our payment protocol, for the whole transaction, the client only has to compute a single response i.e. $b = x_i - Ua \bmod q$. This is far more efficient than all known off-line electronic cash schemes to-date, especially as the response message does not involve any discrete exponential computation. Moreover, the vendor, in the payment phase, does not need to perform any complicated verification. In fact, the vendor only has to verify one RSA signature per coin plus a certification *Cert* and a Schnorr's one-time signature for each transaction.

Hence our protocol is much more efficient than other existing electronic cash schemes such as those in [3, 1, 4, 8].

4.4 Other features

Coin Forgery Forging our coins is equivalent to creating $(x, h(x)^d \bmod n)$. This is proved infeasible unless the factorization of n is known.[3] As the factorization of n is only known to $\mathcal{B}$, forging our electronic coins is infeasible for any other party.

Framing To frame a particular client $\mathcal{C}$, $\mathcal{B}$ needs U which is (provably) unobtainable unless $\mathcal{C}$ double spends.

Consider the case where clients and vendors collude to double-spend some coins without being detected by the bank. It is not possible in our scheme as the vendor must show the anonymous certificate of the client (*Cert*) to the bank in the deposit phase.

5 Conclusion

In this paper, we have proposed an efficient off-line digital coin system. Our method uses one-way hash chain to reduce client's computations to a simple calculation per transaction. This make our scheme a practical proposition.

We would like to distinguish our work from the scheme proposed in [7] that used a similar approach which requires only one response from the client for each payment. It is noted that the scheme[7] requires the vendor to sign next unspent sub-coin-chain; this allows the second vendor to know who is the first vendor. Moreover, in that system, forging a coin is only equivalent to breaking a one-way hash function; whereas in our system, it is equivalent to breaking the RSA scheme.

Acknowledgement: We would like to thanks anonymous referees for their comments to improve the representation of this paper and the necessary bit-length of U.

References

[1] S.Brands, " Untraceable off-line cash in wallet with observers", *Advances of Cryptology - CRYPTO '93 Proceedings*, Springer-Verlag,1994, pp.302-318.

[2] D.Chaum, " Security without Identification: Transaction systems to make Big Brother obsolete," *Communications of ACM, vol.28, no.10, pp.1030-1044, Oct.85.*

[3] D.Chaum, A.Fiat and M.Naor, "Untraceable electronic cash", in *Advances in Cryptology - CRYPTO '88 Proceedings*, pp.319-327,1990.

[4] N.T.Ferguson, " Single Term Off-Line Coins",*Advances in Cryptology - EUROCRYPT '93 Proceedings*, Springer-Verlag, 1994, pp.318-328

[5] C.Schnorr, "Efficient signature generation for smart cards", *Journal of Cryptology*, 4(3):161-174,1991.

[6] W. Mao, " Blind Certification of Public Keys and Off-Line Electronic Cash", HP Laboratories Technical Report, HPL-96-71, May 1996.

[7] W.Mao, "Light-weight Micro-Cash for the Internet", *ESORICS'96 Proceedings*, Springer-Verlag, 1996.

[8] Y.Yacobi, "An efficient off-line cash", *Advances of Cryptology - Asiacrypt '94 Proceedings* , Springer-Verlag, 1994.