TRANSFERABLE E-CASH REVISIT

Joseph K. Liu[1], Sandy H. Wong[2 *], Duncan S. Wong[3]

[1]*Department of Information Engineering*
The Chinese University of Hong Kong
Shatin, Hong Kong
ksliu@ie.cuhk.edu.hk

[2]*Wireless Technology*
Hong Kong Applied Science and Technology
Research Institute Company Limited,
Hong Kong
sandy@astri.org

[3]*Department of Computer Science*
City University of Hong Kong
Kowloon, Hong Kong
duncan@cityu.edu.hk

** The work described in this paper was done when this author was at the Department of Information Engineering, The Chinese University of Hong Kong*

Abstract: Incorporating the property of transferability into an offline electronic cash (e-cash) system has turned out to be no easy matter. Most of the e-cash systems proposed so far do not provide transferability. Those who support transferability are either inefficient or requiring to be online with the help of a trustee. In this paper we present an efficient offline transferable e-cash system. The computational complexity of our system is as light as a single term e-cash system [13]. Besides, no trustee is involved in the transfer protocol. In addition to it, we propose two e-check systems constructed using similar techniques to our e-cash system. One is as efficient as a single term e-cash system and supports partial unlinkability. The other one provides complete unlinkability with a more complex setting.

Key words: electronic payment systems, secure electronic commerce, transferable e-cash, e-check

1. INTRODUCTION

An electronic cash (e-cash) system provides a digital way to mint and transfer money, or so-called e-cash/e-coins. According to [17], an ideal e-cash system consists of six properties: independence, security, untraceability, offline payment, divisibility and transferability. Independence implies that the security of the e-cash system does not depend on any physical location, medium, time or users. Security means that an e-cash cannot be forged or double spent without being detected. Untraceability refers to the maintenance of the anonymity of any honest user. Offline payment does not need the bank to be involved during the payment process conducted by a customer and a merchant. Divisibility refers to the ability to divide an e-coin into smaller pieces provided that the total amount of those pieces equals the value of the original e-coin. Transferability allows a user to spend an e-coin received in a payment to another user immediately without having contact the bank.

Most of the e-cash systems [12,13,4,16,7,20] focus only on the first five properties but not on the last one: transferability. The first transferable e-cash system was proposed by Okamoto, et al. [17]. It is based on the costly cut-and-choose methodology. Chaum, et al. [11] outlined a method to extend [22] for transferability. Their idea is similar to ours. In this paper, we actually build a transferable one based on an entirely different and efficient scheme. In addition, we also illustrate how it can further be extended to construct e-check systems. Other transferable e-cash systems include the one proposed by Pagnia, et al. [18] and another one proposed by Anand, et al. [1]. However, this first one requires a trusted third party in the system to maintain users' anonymity while the second one requires the payment process to be online. Jeong, et al. also proposed a transferable cash system in [15] using group signatures [6,5,2]. However, the system needs an additional third party – the group manager. It can recover the identity of any group member and therefore it has to be trusted by all the users.

In addition to e-cash, a more convenient means of payment is the electronic check (e-check). An e-check can be used for only once but can be used for any amount up to a maximum value and then be returned to the bank for a refund of the unused part. Therefore an extra protocol, the refund protocol is needed. It is convenient because a customer can 'withdraw' some token with the bank and decides how much it wants to use later on. The leftover can be redeemed afterwards.

E-check was first proposed by Chaum, et al. in 1988 [8,10]. In [8], an online e-check system was proposed. Although [10,14] proposed offline e-check systems, they use the cut-and-choose methodology which appears to be quite inefficient in practice. The systems proposed in [3,21] avoid using

the cut-and-choose technique. However, [21] requires a trustee which knows the owner of each e-coin in the system even without double-spending. In this paper, we propose two e-check schemes by direct extension from a e-cash scheme [13]. The second scheme which provides complete linkability is as efficient as that in [3] in terms of computational complexity.

In this paper, we propose an efficient offline transferable e-cash system. It extends directly from Ferguson's single term e-cash scheme [13] which is described in Section 2. In our proposed system, only users and a bank is present. A *shop* is eliminated because a payment to a shop can be regarded as a transfer of an e-coin from a user to anther user. The scheme is untraceable but secure. Moreover, we also propose two offline e-check systems which are as efficient as the one in [3]. One is highly efficient and supports partial unlinkability. The other one is completely unlinkable with a more complex setting.

The rest of the paper is organized as follows. In Sec. 2, the Single Term offline e-cash scheme proposed by Ferguson [13] is reviewed. In Sec. 3, we describe our proposed transferable e-cash scheme. This is followed by the proposed two e-check schemes in Sec. 4 and conclude the paper in Sec. 5.

2. FERGUSON'S SINGLE TERM OFF-LINE COINS [13]

It is an offline untraceable e-cash system without providing transferability. The scheme is efficient and does not use the cut-and-choose methodology. Here we give a brief review of the scheme.

Preliminaries. Let $\{0,1\}^*$ denote the set of finite binary strings. To denote that an element a is chosen uniformly at random from a finite set A, we write $a \in_R A$. Let n be the public RSA modulus [19] of the bank and v be its public exponent. It is required that v is a reasonably large prime (say 128 bits). Let g_a, g_b, g_c, h_b, h_c be publicly known numbers such that $g_a, g_b, g_c \in Z_n^*$ have large order and h_b, h_c are of order n in $GF(p)$, where $p-1$ is a multiple of n. Let U be an identity which is the concatenation of the user's identity and a unique coin number so that U is distinct for each e-coin. Let $f_1 : \{0,1\}^* \to Z_v$ and $f_2 : \{0,1\}^* \to Z_n^*$ be cryptographic hash functions.

2.1. Withdrawal Protocol

The withdrawal protocol consists of three parallel runs of the randomized blinded RSA signature scheme [9,13]. It is proceeded as follows.

1. The user picks $c_1, a_1, b_1 \in_R Z_n^*$, $\sigma, r, \phi \in_R Z_v$, $\gamma, \alpha, \beta \in_R Z_n$, and computes

$$G_c = \gamma^v c_1 g_c{}^{\sigma} \bmod n,$$
$$G_a = \alpha^v a_1 g_a{}^{r} \bmod n,$$
$$G_b = \beta^v b_1 g_b{}^{\phi} \bmod n.$$

2. It sends $M_1 = (U, G_c, G_a, G_b)$ to the bank. For simplicity, we omit the notation of modular reduction in the rest of the paper when it gets clear from its context.

3. The bank picks $c_2, a_2, b_2 \in_R Z_n^*$ and sends $M_2 = (h_c{}^{c_2} \bmod p, a_2, h_b{}^{b_2} \bmod p)$ to the user.

4. The user picks $t_1 \in_R Z_v^*$ and computes

$$e_c = f_1(h_c{}^{c_1 c_2}) - \sigma \bmod v,$$
$$e_b = f_1(h_b{}^{b_1 b_2}) - \phi \bmod v,$$
$$a = (a_1 a_2 f_2(e_c, e_b))^{t_1} \bmod n,$$
$$e_a = \frac{1}{t_1} f_1(a) - r \bmod v.$$

5. It then sends $M_3 = (e_c, e_a, e_b)$ to the bank[10].

6. The user also signs (M_1, M_2, M_3) and sends the signature to the bank[11].

7. The bank computes

$$\overline{C} = G_c c_2 g_c{}^{e_c},$$
$$\overline{A} = G_a a_2 f_2(e_c, e_b) g_a{}^{e_a},$$
$$\overline{B} = G_b b_2 g_b{}^{e_b}$$

[10] Note that the exponents e_c, e_b and e_a are computed modulo v. Certain corrections in the final signature (S_a, S_b) are needed to make the blinding perfect. This is done by multiplying the final signature by a suitable powers of g_c, g_a and g_b [13]. Corrections are not shown in this paper.

[11] This corresponds to a signature of the user for all the data in the first three transmissions. It is used to protect the user against framing by the bank. We refer readers to [13] for detail.

and selects $t_2 \in_R Z_v^*$. It sends $(c_2, b_2, t_2, (\overline{C}^{t_2}\overline{A})^{1/v}, (\overline{C}^{U}\overline{B})^{1/v})$ to the user.

8. The user computes

$$c = c_1 c_2,$$
$$b = b_1 b_2,$$
$$t = t_1 t_2 \bmod v,$$
$$C = c g_c^{f_1(h_c{}^c)},$$
$$A = a g_a^{f_1(a)},$$
$$B = b g_b^{f_1(h_b{}^b)},$$
$$S_a = \left(\frac{(\overline{C}^{t_2}\overline{A})^{1/v}}{\gamma^{t_2}\alpha}\right)^{t_1},$$
$$S_b = \frac{(\overline{C}^{U}\overline{B})^{1/v}}{\gamma^{U}\beta}$$

and checks whether $S_a{}^v = C^t A$ and $S_b{}^v = C^U B$. If these two equalities hold, it accepts.

The user stores (a,b,c,t,S_a,S_b) as an e-coin. (a,b,c) are the *base numbers* of the coin.

2.2. Payment Protocol

To spend an e-coin (a,b,c,t,S_a,S_b), the user executes the following protocol with the shop.

9. The user sends (a,b,c) to the shop.

10. The shop randomly chooses a challenge x and sends it to the user.

11. The user computes and sends $r = tx + U$ and $S = (S_a)^x (S_b)$ to the shop.

12. The shop computes $C = c g_c^{f_1(h_c{}^c)}$, $A = a g_a^{f_1(a)}$, $B = b g_b^{f_1(h_b{}^b)}$ and checks if $S^v = C^r A^x B$. If the equality holds, the shop accepts the coin and stores (a,b,c,x,r,S). Otherwise, it rejects.

(x, r, S) is a proof of the user's ownership to the e-coin with base number (a, b, c). Obviously, the user can only provide one proof in order to prevent from revealing its identity.

2.3. Deposit Protocol

To deposit an e-coin, it sends (a, b, c, x, r, S) to the bank. The bank verifies the coin by following the steps below.

13. Compute $C = cg_c^{f_1(h_c^c)}$, $A = ag_a^{f_1(a)}$ and $B = bg_b^{f_1(h_b^b)}$.
14. Check if $S^v = C^r A^x B$. If it is false, the bank rejects the deposit.
15. Otherwise, it checks if (a, b, c) are already existed in its database. If yes, the bank rejects the deposit. Otherwise it accepts and stores (a, b, c) in its database and it credits the shop.

Double-spending is detected if the bank finds the same triple (a, b, c) are already in its database. If the corresponding (x, r, S) are the same as the ones stored in the database, the bank concludes that the shop is cheating. Otherwise, it concludes that the user double spends the coin. The identity of the user, U can be obtained easily by solving the two linear equations.

3. OUR PROPOSED TRANSFERABLE E-CASH

Let the bank issue e-coins with N different denominations. For the i-th denomination, the bank has a distinct and reasonably large prime v_i be the corresponding public exponent. Define a *zero-value* coin with the corresponding public exponent v_0 as a distinct large prime. A zero-value coin is an e-coin which is worth nothing. It preserves all the properties of a non-transferable e-coin. Essentially, if the zero-value coin is double spent, the identity of the user would be revealed. For distinction, we call other nonzero-value coins as positive-value coins. Note that coins with various denominations are sharing the same public RSA modulus n.

How It Works. Each user obtains a number of zero-value coins from the bank using the withdrawal protocol described below during the system setup. When an owner, Alice, of a positive-value coin transfers the coin to a user, Bob, she carries out the payment protocol described below which is similar to the original payment protocol reviewed in Sec. 2.2. That is, Bob obtains the coin's base numbers and a proof of Alice's ownership. When Bob wants to transfer this coin to another user, Carol, Bob has to send, through a transfer protocol, the coin's base numbers and the proof of Alice's ownership appended with the base numbers of one of his zero-value coins and a proof of his ownership to Carol. Now when Carol wants to transfer the coin to another user, Daniel, Carol sends, through the transfer protocol, the coin's base numbers, the proof of Alice's ownership, the base numbers of Bob's zero-value coin, the proof of Bob's ownership, appended with the base numbers of one of her zero-value coins and a proof of his ownership to Daniel. The procedure repeats until the final receiver of the coin decides to deposit it.

We refer to this transfer mechanism as a 'transfer-chain'. We will see shortly that this 'chain' is linked by a special relation between the proof of the sender's ownership of the coin and the base numbers of a receiver's zero-value coin. As long as a user provides only one proof of its ownership to a zero-value coin, the user's identity would not be compromised. This implies that each zero-value coin can only be appeared in at most one transfer-chain. Using twice or more will result in identity revocation.

When a user receives a transfer-chain, it can only transfer the chain to one user. Transferring to more than one user is equivalent to double-spending. On the other side, multiple transfer-chains can be combined into one transfer chain when they are transferring to one single user. That is, multiple coins can be transferred to a receiver in one run of the transfer protocol. This is accomplished by building up many-to-one relation of the proofs of multiple senders' ownerships of several coins and the base numbers of a receiver's zero-value coin.

In the system, a payment process is considered as a transfer of some e-coins from a user to a shop. It has no difference from a transfer process and a shop has no difference from a conventional user. Hence there are only users and a bank in the system and the payment protocol is replaced by a transfer protocol.

In the following, we describe the three protocols of our scheme, namely the withdrawal protocol, the transfer protocol and the deposit protocol.

3.1. Withdrawal Protocol

This protocol is executed when a user withdraws a coin from the bank, no matter the coin is a zero-valued or a positive-valued. The corresponding public exponent is used for each denomination of the coins. The protocol is similar to Ferguson's described in Sec. 2.1. For a user i, a zero-value coin and a positive-value coin are denoted as Z_i and P_i, respectively.

3.2. Transfer Protocol

As explained before, a transfer-chain is formed when a coin is transferred. Without loss of generality, let the transfer start from user 1, then to user 2, and so on. That is, user 1 withdraws a positive-value coin $P_1 = (a_1, b_1, c_1, t_1, S_{a_1}, S_{b_1})$ from the bank with corresponding identity U_1. It is later transferred to user 2 and then to user 3, and so on. Let the zero-value coin of user k, for $k > 1$, be $Z_k = (a_k, b_k, c_k, t_k, S_{a_k}, S_{b_k})$ with corresponding identities U_k. In this section, we will see that a transferred coin is derived directly from the concept of transfer-chain. Below is the structure of a transferred coin $Coin_k$ when it is transferred from user 1 all the way to user k, for $k > 1$.

Structure of a Transferred Coin. Suppose the value of P_1 is d-th denomination which corresponds to the public exponent v_d, where $1 \le d \le N$. After the coin has been transferred for $k-1$ times for $k > 1$, user k has the coin and the following components constitute the transferred coin $Coin_k$.

$$S_k = s_1 \,||\, s_2 \,||\cdots||\, s_{k-1}$$
$$A_k = a_1||a_2||\cdots||a_{k-1}||a_k$$
$$B_k = b_1||b_2||\cdots||b_{k-1}||b_k$$
$$C_k = c_1||c_2||\cdots||c_{k-1}||c_k$$
$$R_k = r_1||r_2||\cdots||r_{k-1}$$

where

- $s_i = (S_{a_i})^{x_i} S_{b_i}$, $1 \le i \le k-1$, $x_i = H(a_{i+1}, b_{i+1}, c_{i+1})$ and H is some appropriate cryptographic hash functions. Hence $(a_1, b_1, c_1, S_{a_1}, S_{b_1})$ are from P_1 and $(a_j, b_j, c_j, S_{a_j}, S_{b_j})$ are from Z_j for $j > 1$.
- $r_i = t_i x_i + U_i$, $1 \le i \le k-1$.

For the boundary case (when $k=1$), we define $Coin_1=(S_1,A_1,B_1,C_1,R_1)$, for $A_1=a_1$, $B_1=b_1$, $C_1=c_1$ and $S_1=R_1=\lambda$, where λ represents empty content.

Validation of a Transferred Coin. We describe the validation of a transferred coin $Coin_k$ as a function, $valid(Coin_k)$ which outputs accept if the coin is valid, otherwise, it outputs reject:

valid = "On input $Coin_k=(S_k,A_k,B_k,C_k,R_k)$, for any $k\geq 1$,

1. Compute $A_i=a_i g_a^{f_1(a_i)}$, $B_i=b_i g_b^{f_1(h_b^{b_i})}$, $C_i=c_i g_c^{f_1(h_c^{c_i})}$ and $x_i=H(a_{i+1},b_{i+1},c_{i+1})$, for $1\leq i\leq k-1$.
2. Check whether

$$S_1^{v_d}=C_1^{r_1}A_1^{x_1}B_1,$$

$$S_i^{v_0}=C_i^{r_i}A_i^{x_i}B_i, \text{for} 2\leq i\leq k-1$$

3. Output accept if all the equalities hold, otherwise output reject."

The Protocol. When user k (for any $k\geq 1$) transfers $Coin_k$ to user $k+1$, they execute the following transfer protocol. Here we assume that user $k+1$ has an unused (fresh) zero-value coin Z_{k+1} with base numbers $(a_{k+1},b_{k+1},c_{k+1})$.

1. User k sends $Coin_k=(S_k,A_k,B_k,C_k,R_k)$ to user $k+1$.
2. User $k+1$ executes $valid(Coin_k)$ to validate the coin. It continues if the function output accept. Otherwise, it halts with failure.
3. User $k+1$ computes and sends $x_k=H(a_{k+1},b_{k+1},c_{k+1})$ to user k.
4. User k computes $\begin{matrix} r_k=t_k x_k+U_k \\ s_k=(S_{a_k})^{x_k}S_{b_k} \end{matrix}$ and sends r_k,s_k to user $k+1$.
5. User $k+1$ computes

$$A_k=a_k g_a^{f_1(a_k)}$$

$$B_k=b_k g_b^{f_1(h_b^{b_k})}$$

$$C_k=c_k g_c^{f_1(h_c^{c_k})}$$

and checks whether

$$s_1^{v_d} = C_1^{r_1} A_1^{x_1} B_1, \text{if} k = 1$$
$$s_k^{v_0} = C_k^{r_k} A_k^{x_k} B_k, \text{if} k > 1.$$

6. It continues if the equality holds. Otherwise, it halts with failure.
7. User $k+1$ constructs

$$A_{k+1} = A_k \| a_{k+1}$$
$$B_{k+1} = B_k \| b_{k+1}$$
$$C_{k+1} = C_k \| c_{k+1}$$
$$R_{k+1} = R_k \| r_k$$
$$S_{k+1} = S_k \| s_k$$

and stores them as the new transferred coin $Coin_{k+1}$.

3.3. Deposit Protocol

The Deposit Protocol is straightforward. When user k deposits $Coin_k$ to the bank, the bank executes $valid(Coin_k)$ to validate the coin. Then it checks if (a_1, b_1, c_1) are already in its database. If not, the bank stores $Coin_k$ and credits user k. Otherwise, it means someone has double spent the coin.

Detection of Double-Spending. We use the following example to illustrate the detection mechanism of double-spending. Suppose user 1 withdraws a coin (U_1) from the bank and transfers to user 2 (U_2) and so on, until it reaches user 6 (U_6). User 6 deposits the coin. Also suppose that user 3 (U_3) and user 5 (U_5) double spend the coin. Their double-spent coins are finally transferred to user $6'$ and user $7''$, respectively, and then deposited to the bank. Hence the bank has three copies of the coin with the same initial base numbers (a_1, b_1, c_1). Let the transferred coin deposited by user 6, user $6'$ and user $7''$ be $Coin_6$, $Coin_{6'}$ and $Coin_{7''}$, respectively. The bank finds

- $\{(a_1, b_1, c_1, r_1), \cdots, (a_3, b_3, c_3, r_3), \cdots, (a_5, b_5, c_5, r_5), \cdots\}$ from $Coin_6$;
- $\{(a_1, b_1, c_1, r_1), \cdots, (a_3, b_3, c_3, r_{3'}), \cdots\}$ from $Coin_{6'}$; and
- $\{(a_1, b_1, c_1, r_1), \cdots, (a_5, b_5, c_5, r_{5''}), \cdots\}$ from $Coin_{7''}$.

From $Coin_6$ and $Coin_{6'}$, the bank finds the double spender to be user 3 and from $Coin_6$ and $Coin_{7''}$, the bank finds the double spender to be user 5. Their identities are easily obtained by solving the corresponding linear equations. For example, U_3 is obtained by computing $x_3 = H(a_4, b_4, c_4)$ and $x_{3'} = H(a_{4'}, b_{4'}, c_{4'})$ and solving the following equations:

$$r_3 = t_3 x_3 + U_3$$
$$r_{3'} = t_3 x_{3'} + U_3$$

Also, it is important to see that the identity of other honest users would not be revealed by the bank.

4. OUR PROPOSED E-CHECKS

In this section, we present two e-check systems. The first one is highly efficient and supports partial unlinkability. The second one support complete unlinkability with a more complex setting.

4.1. E-Check I

We base on Ferguson's e-cash system again and therefore use the same notations as before. In this e-check system, there is a list of reasonable large prime numbers $(v_1, \cdots, v_k)$ as public exponents of the bank with v_i corresponding the value of $\$2^{i-1}$. Define that multiplying any set of v_i, $1 \le i \le k$, represents to the sum of their corresponding values. v_d denotes the public exponent of the bank representing $\$d$ such that

$$v_d = \prod_{i=1}^{k} v_i^{<d>_i} \quad (1)$$

where $<d>_i$ denotes the value of the i-th least significant bit of d. For example, $<6>_1 = 0, <6>_2 = 1, <6>_3 = 1$. In this way, we can represent any amount up to $\$2^k - 1$.

4.1.1. Withdrawal Protocol

Without loss of generality, suppose a user wants to withdraw an e-check of $\$2^k - 1$ as its maximum value. The withdrawal protocol is the same as Ferguson's one (Sec. 2.1) by setting the public exponent to $v = v_1 \cdot v_2 \cdots v_k$. Note that the maximum value of the e-check must be in the form of $\$2^i - 1$, for any $i > 1$. That is, all the bits of the maximum value of the e-check should be 1 in its binary representation. This ensures that the devaluation of v_d (first step of the Payment Protocol below) is always computable. Let the e-check be denoted as $K = (a, b, c, t, S_a, S_b)$ where (a, b, c) are the base numbers of the check.

4.1.2. Payment Protocol

Suppose the user wants to spend $\$d$ to the shop, where $1 \le d \le 2^k - 1$. The corresponding public exponent of the bank is v_d which can be publicly computed using equation (1). In the first step of the protocol, the user 'devalues' the check from $\$2^k - 1$ to $\$d$. The protocol proceeds as follow.

1. The user computes $\overline{v_d} = v_1 \cdots v_k div v_d$, and
$$S'_a = (S_a)^{\overline{v_d}} \qquad S'_b = (S_b)^{\overline{v_d}}.$$
2. Note: div is normal division without taking modulo.
3. The user then sends the base numbers of the check (a,b,c) to the shop.
4. The shop randomly picks a challenge x and sends it to the user.
5. The user computes $r = tx + U$, $S = (S'_a)^x (S'_b)$ and sends r, S to the shop.
6. The shop computes $C = c g_c^{f_1(h_c{}^c)}$, $A = a g_a^{f_1(a)}$, $B = b g_b^{f_1(h_b{}^b)}$ and checks whether $S^{v_d} = C^r A^x B$. If the equality holds, the shop accepts and stores (a,b,c,x,r,S,d). Otherwise, it rejects.

4.1.3. Deposit and Refund Protocols

The deposit protocol of our e-check system is the same as Ferguson's deposit protocol reviewed in Sec. 2.3, with the public exponent $v = v_d$.

The user can refund the remaining $\$2^k - 1 - d$ from the bank by executing a refund protocol. The protocol is almost the same as the deposit protocol, except the checking of double spending. In the refund protocol, the user sends the used check-tuple (a,b,c,x,r,S,d) to the bank. The bank verifies user's ownership of the e-check by first carries out the steps similar to the payment protocol, namely it sends a challenge x' and obtains a response pair (r', S'). Then it checks if the base numbers (a,b,c) are already in its database. If it exists and the amount is d, the bank refunds the remaining $\$2^k - 1 - d$ to the user and updates its database to record that the e-check has already been refunded.

Note that this part is not anonymous. The bank knows the identity of the user who asks for refund. The bank can also link the e-check which has already spent by the user in earlier time.

4.2. E-Check II

The e-check system proposed in last section is linkable at the refund stage. In this section, we propose another scheme which is completely unlinkable. In this scheme, the bank has only one public exponent v. Instead, we use different elements $g_{a_i} \in Z_n^*, 1 \le i \le k$ of large order to represent different values of the e-check. Like the representation system in E-Check I, we use g_{a_i} to represent $\$2^{i-1}$. In this way, with k consecutive elements, the e-check has a maximum value of $\$2^k - 1$. We further use g_{a_0} to prevent a user from using the e-check twice or more. Thus g_{a_0} is included in the payment of an e-check regardless of the payment amount.

E-Check II is similar to Ferguson's e-cash system. However, there are k+1 signatures in each e-check if its maximum value is $\$2^k - 1$, one is for embedding the identity of the user to prevent double-spending while the others are for composing the value of the e-check.

4.2.1. Withdrawal Protocol

Without loss of generality, we assume a user wants to withdraw an e-check of $\$2^k - 1$. We follow the notations of Sec. 2.1. Let $g_{a_0}, g_{a_1}, \cdots, g_{a_k}, g_b, g_c$ be public where $g_{a_0}, g_{a_1}, \cdots, g_{a_k}, g_b, g_c$ are of large order in Z_n^*. The Withdrawal Protocol proceeds as follows.

1. The user picks $b_1, c_1, a_{1_0}, a_{1_1}, \cdots, a_{1_k} \in_R Z_n^*$, $\sigma, \phi, r_0, r_1, \cdots, r_k \in_R Z_v$ and $\gamma, \beta, \alpha_0, \alpha_1, \cdots, \alpha_k \in_R Z_n$. It then computes

$$G_b = \beta^v b_1 g_b^{\ \phi}$$

$$G_c = \gamma^v c_1 g_c^{\ \sigma}$$

$$G_{a_i} = \alpha_i^{\ v} a_{1_i} g_{a_i}^{\ r_i}, \text{for} i = 0, \cdots, k$$

2. and sends $M_1 = (U, G_b, G_c, G_{a_0}, \cdots, G_{a_k})$ to the bank.

3. The bank picks $b_2, c_2, a_{2_0}, a_{2_1}, \cdots, a_{2_k} \in_R Z_n^*$ and sends $M_2 = (h_b^{\ b_2}, h_c^{\ c_2}, a_{2_0}, a_{2_1}, \cdots, a_{2_k})$ to the user.

4. The user picks $t_{1_0}, t_{1_1}, \cdots, t_{1_k} \in_R Z_v^*$, computes

$$e_b = f_1(h_b{}^{b_1 b_2}) - \phi \bmod v$$
$$e_c = f_1(h_c{}^{c_1 c_2}) - \sigma \bmod v$$
$$a_i = (a_{1_i} a_{2_i} f_2(i, e_c, e_b))^{t_{1_i}}, \text{for} i = 0, \cdots, k$$
$$e_{a_i} = \frac{1}{t_{1_i}} f_1(a_i) - r_i \bmod v, \text{for} i = 0, \cdots, k$$

and sends $M_3 = (e_b, e_c, e_{a_0}, e_{a_1}, \cdots, e_{a_k})$ to the bank.

5. The user also signs (M_1, M_2, M_3) and sends the signature to the bank. (Note: refer to Sec. 2.1 for discussions).
6. The bank computes

$$\overline{C} = G_c c_2 g_c{}^{e_c}$$
$$\overline{B} = G_b b_2 g_b{}^{e_b}$$
$$\overline{A_i} = G_{a_i} a_{2_i} f_2(i, e_c, e_b) g_{a_i}{}^{e_{a_i}}, \text{for} i = 0, \cdots, k$$

7. The bank selects $t_{2_0}, t_{2_1}, \cdots, t_{2_k} \in_R Z_v^*$ and sends $c_2, b_2, \{t_{2_i}\}_{0 \le i \le k}, \{(\overline{C}^{t_{2_i}} \overline{A_i})^{1/v}\}_{0 \le i \le k}, (\overline{C}^U \overline{B})^{1/v}$ to the user.
8. The user computes

$$c = c_1 c_2$$
$$b = b_1 b_2$$
$$t_i = t_{1_i} t_{2_i} \bmod v, \text{for} i = 0, \cdots, k$$
$$B = b g_b{}^{f_1(h_b{}^b)}$$
$$C = c g_c{}^{f_1(h_c{}^c)}$$
$$A_i = a_i g_{a_i}{}^{f_1(a_i)}, \text{for} i = 0, \cdots, k$$
$$S_b = \frac{(\overline{C}^U \overline{B})^{1/v}}{\gamma^U \beta}$$
$$S_i = \left(\frac{\overline{C}^{t_{2_i}} \overline{A_i})^{1/v}}{\gamma^{t_{2_i}} \alpha_i}\right)^{t_{1_i}}, \text{for} i = 0, \cdots, k$$

and checks whether $S_b{}^v = C^U B$ and $S_i{}^v = C^{t_i} A_i$, for $i = 0, \cdots, k$. If all the equalities hold, he accepts.

The user stores $(a_0, \cdots, a_k, b, c, t_0, \cdots, t_k, S_b, S_0, S_1, \cdots, S_k)$ for the payment of the e-check.

4.2.2. Payment Protocol

Without loss of generality, suppose the user wants to spend $\$2^j - 1$, for some $1 \le j \le k$, to the shop. The payment protocol proceeds as follows.

1. The user sends $b, c, a_0, \cdots, a_j$ to the shop.
2. The shop selects a challenge number x and sends it to the user.
3. The user computes $r_i = t_i x + U$ and $S'_i = (S_b)(S_i)^x$, and sends (r_i, S'_i), $0 \le i \le j$, to the shop.
4. The shop computes

$$C = c\, g_c^{\,f_1(h_c{}^c)}$$

$$B = b\, g_b^{\,f_1(h_b{}^b)}$$

$$A_i = a_i\, g_{a_i}^{\,f_1(a_i)}, 0 \le i \le j,$$

and checks whether $S'_i{}^v = C^{r_i} A_i{}^x B$ for $0 \le i \le j$. If all the equalities hold, the shop accepts and stores $(a_0, \cdots, a_j, b, c, x, r_0, \cdots, r_j, S'_0, \cdots, S'_j)$. Otherwise, it rejects.

4.2.3. Deposit Protocol

The deposit protocol is constructed in its natural way. When the shop deposits the e-check, it sends the check-tuple

$$(a_0, \cdots, a_j, b, c, x, r_0, \cdots, r_j, S'_0, \cdots, S'_j)$$

to the bank. The bank verifies of the tuple as follows.

1. Compute $C = c\, g_c^{\,f(h_c{}^c)}$, $B = b\, g_b^{\,f(h_b{}^b)}$, $A_i = a_i\, g_{a_i}^{\,f(a_i)}$, for $i = 0, \cdots, j$.
2. Check whether $S'_i{}^v = C^{r_i} A_i{}^x B$, $0 \le i \le j$. If not all equal, the bank rejects the deposit.
3. Check whether the same values of (a_0, b, c) already exist in its database. If yes, the bank rejects the deposit and the double-spender can easily be found. Otherwise, it accepts and credits the shop.

4.2.4. Refund Protocol

If the user wants to refund the remaining amount of the e-check, that is, $\$2^k - 1 - (2^j - 1) = \$2^k - 2^j$, he has to inform the bank his account number and his identity U for the refund purpose and execute the following steps.

1. The user sends $U, a_{j+1}, \cdots, a_k$ and $t_{j+1}, \cdots, t_k$ to the bank.
2. The bank retrieves $\overline{B}, \overline{C}$ from the withdrawal record.
3. The bank checks if any of $a_{j+1}, \cdots, a_k$ are already in the database. If yes, it rejects. Otherwise, the bank selects a challenge number x and sends it to the user. The bank also computes $r_i = t_i x + U$, for $j+1 \le i \le k$.
4. The user computes $r_i = t_i x + U$ and $S''_i = (S_b)(S_i)^x \gamma^{r_i} \beta$, and sends S''_i, $j+1 \le i \le k$, to the bank.
5. The bank computes $A_i = a_i g_{a_i}^{f(a_i)}$ and checks whether $S''_i{}^v = \overline{C}^{r_i} A_i{}^x \overline{B}$ for $j+1 \le i \le k$. If not all of them are equal, the bank rejects. Otherwise, the bank records that the e-check has been refunded in its database and refunds $\$2^k - 2^j$ to the user.

Unlike E-Check I, in this e-check system, the bank is unable to link the refunded e-check with the e-check that the user has already spent to the shop.

5. CONCLUSION

We have proposed an off-line transferable e-cash system. Unlike [15], we do not require any group manager or trustee. Our scheme does not use cut-and-choose technique, thus more efficient than those using cut-and-choose such as [17]. In addition, we have also proposed two e-check systems. One is almost as efficient as a single term e-cash such as [13] with partial unlinkability only. The other one provides complete unlinkability with a more complex setting.

We do not address divisibility in our transferable e-cash system. We may consider divisibility to be less important in practice as this can be easily be solved as in the world of physical cash. That is, using various denominations and conducting changes in transactions as the coins are transferable. Hence we consider that with transferability, divisibility becomes a less important property of e-cash.

REFERENCE

[1] R. Sai Anand and C.E. Veni Madhavan. An Online, Transferable E-Cash Payment System. *INDOCRYPT 2000*, LNCS 1977, pp. 93-103. Springer-Verlag, 2000.

[2] G. Ateniese, J. Camenisch, M. Joye and G. Tsudik. A Practical and Provably Secure Coalition-Resistant Group Signature Scheme. *CRYPTO 2000*, LNCS 1880, pp. 255-270. Springer-Verlag, 2000.

[3] S. Brands. An Efficient Off-line Electronic Cash System based on the Representation Problem. *Technical Report CS-R9323*, CWI, April, 1993.

[4] S. Brands. Untraceable off-line cash in wallets with observers. *Proc. CRYPTO 93*, LNCS 0773, pp. 302-318. Springer-Verlag, 1993.

[5] J. Camenisch and M. Michels. A group signature schemes with improved efficiency. *Proc. ASIACRYPT 98*, LNCS 1514, pp. 160-174. Springer-Verlag, 1998.

[6] J. Camenisch and M. Stadler. Efficient group signature scheme for large groups. *Proc. CRYPTO 97*, LNCS 1296, pp. 410-424. Springer-Verlag, 1997.

[7] A. Chan, Y. Frankel and Y. Tsiounis. Easy Come - Easy Go Divisible Cash *EUROCRYPT 98*, LNCS 1403, pp. 561-575. Springer-Verlag, 1998.

[8] D. Chaum. Online Cash Checks. *Proc. EUROCRYPT 89*, LNCS 0403, pp. 288-293. Springer-Verlag, 1990.

[9] D. Chaum. Randomized Blind Signature. manuscript., 1992.

[10] D. Chaum, B. den Boer, E. van Heyst, S. Mjolsnes and A. Steenbeek. Efficient offline electronic checks. *Proc. EUROCRYPT 89*, LNCS 0403, pp. 294-301. Springer-Verlag, 1990.

[11] D. Chaum, T.P. Pedersen. Transferred Cash Grows in Size *Proc. EUROCRYPT 92*, LNCS 0658, pp. 390-407. Springer-Verlag, 1992.

[12] D. Chaum, A. Fiat and M. Naor. Untraceable electronic cash. *Proc. CRYPTO 88*, LNCS 0403, pp. 319-327. Springer-Verlag, 1990.

[13] N. Ferguson. Single Term Off-line Coins. *Proc. EUROCRYPT 93*, LNCS 0765, pp. 318-328. Springer-Verlag, 1993.

[14] R. Hirschfeld. Making Electronic Refunds Safer. *Proc. EUROCRYPT 92*, LNCS 740, pp. 106–112. Springer-Verlag, 1993.

[15] I.R. Jeong, D.H. Lee and J.I. Lim. Efficient Transferable Cash with Group Signatures. *ISC 2001*, LNCS 2200, pp. 462–474. Springer-Verlag, 2001.

[16] T. Okamoto. An efficient divisible electronic cash scheme. *Proc. CRYPTO 95*, LNCS 0963, pp. 438-451. Springer-Verlag, 1995.

[17] T. Okamoto and K. Ohta. Universal electronic cash. *Proc. EUROCRYPT 91*, LNCS 0547, pp. 324-337. Springer-Verlag, 1991.

[18] H. Pagnia and R. Jansen. Towards Multiple-payment Schemes for Digital Money. *Financial Cryptography 1997*, LNCS 1318, pp. 203-216. Springer-Verlag, 1997.

[19] R. Rivest, A. Shamir and L Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120-126, Feb 1978.

[20] T. Sander and A. Ta-Shma. Auditable, Anonymous Electronic Cash. *Proc. CRYPTO 1999*, LNCS 1666, pp. 555-572. Springer-Verlag, 1999.

[21] A. de Solages and J. Traore. An Efficient Fair Offline Electronic Cash System with Extensions to Checks and Wallets with Observers. *Financial Cryptography 1998*, LNCS 1465, pp. 275-295. Springer-Verlag, 1998.

[22] Hans van Antwerpen. Electronic Cash. *Master's Thesis*, CWI, 1990.