A Practical Off-line Electronic Cash System on Smart Cards Achieving Untraceability, Divisibility, and Transferability

Ho Suk Chung
Dept. of Computer and Communication Engineering
Pohang University of Science and Technology, Pohang, 790-784, KOREA

Pil Joong Lee
Dept. of Electrical Engineering
Pohang University of Science and Technology, Pohang, 790-784, KOREA

October 21, 1994

Abstract

A divisible off-line electronic cash system based on cut-and-choose has first been proposed by [OO91] and recently more efficient single term divisible cash system was presented in [EO94] which is based on Brand's scheme [Bra93]. In this paper, we present a different type of single term divisible electronic cash system which is more efficient than previously proposed systems such as [OO91], [YLR93], and [EO94] in the standpoint of the amount of communication, the number of modular multiplications required in the payment transactions, and the storage requirement in the withdrawal protocol. Our scheme is a modified version of [LL93], where the major improvement has been made in its withdrawal transaction to introduce untraceability and multi-spendability. We have borrowed the idea of the withdrawal protocol of our scheme from [EO94] with minor modifications. Transferability in our scheme allows only a finite number of transfer. Our scheme satisfies all the desirable properties of an electronic cash system such as untraceability, divisibility and transferability. In addition, we present a n-spendable cash. The basic idea of extension to multi-spendability has been borrowed from [Bra93] with minor modifications.

Keywords : Smart cards, Digital signature, Electronic cash, Untraceability, Divisibility, Transferability

1 Introduction

According to T. Okamoto and K. Ohta, there are six criteria to be satisfied in order to be an ideal electronic cash system : (i) Independence (ii) Unreusability (Security) (iii) Untraceability (iv) Off-line payment (v) Transferability (vi) Divisibility. For the definition of six criteria for an ideal electronic cash system, the reader is refer to [OO91].

Several forms of an electronic cash have been proposed. Untraceable electronic payment system was first introduced in [Cha82] [Cha85] by D. Chaum. An off-line untraceable electronic cash system satisfying criteria (i), (ii), (iii), and (iv) was first presented by [CFN88], based on the cut-and-choose and a collision free one-way function. A more efficient version of [CFN88] was presented in [BCHMS89] in view of signature generation, multiplications, divisions, and the bit transmissions. Both [CFN88] and [BCHMS89] are based on cut-and-choose methodology. The difference between electronic coins in [CFN88], and electronic checks in [CFN88] and [BCHMS89] is that the electronic coins have a fixed value, whereas the electronic checks can be used for any amount up to a limit and returned for a refund of a unused part.

An electronic cash system satisfying criteria (i), (ii), (iii), and (iv), and (v) was then proposed by [OO89]. In [OO89], disposable zero-knowledge authentication scheme is used in place of the collision-free function technique in [CFN88]. Here, the authentication scheme is used to detect double-spending of the same coin. Electronic coupon ticket system was also introduced in [OO89], in which one piece of electronic cash can be subdivided into many pieces whose values are all equivalent. For example, a user with a piece of electronic cash worth $100 could subdivide it into 100 pieces of cash worth $1. The drawback of this coupon ticket system is that if a customer pays for an article with cents, the store receives an enormous amount of one-cent electronic coupon tickets from the customer which in turn causes storage management problem. In this sense, [OO89] does not satisfy criteria (vi).

The first version of an ideal electronic cash system that satisfies all the six criteria abovementioned was proposed by T. Okamoto and K. Ohta [OO91] where an electronic cash can be divided into a smaller portion and spent separately. This scheme uses the *cut-and-choose* methodology as all previous schemes. The key techniques are the application of the square root modulo N (N is the Williams integer), and the introduction of the hierarchical structure table corresponding to the structure of the cash system.

A new off-line electronic cash system which is by far more efficient and simpler has been proposed by [Fer93A] based on *single term* methodology. Here, efficiency has been achieved by avoiding the cut-and-choose which uses many terms, each for a single bit of the ahallenge. The main features of [Fer93A] are (i), (ii), (iii), and (iv).

At Eurocrypt '94, Eng and Okamoto proposed a scheme [EO94] which combines the desirable features of both [OO91] and [Bra93]. The scheme whose security is based on the discrete logarithm problem uses restricted blind signatures [Bra93], a binary tree structure for divisibility [OO91], and a three move protocol for disposable authentication. The communication and required memory sizes of [EO94] are less than 1/10 of those of the Okamoto-Ohta scheme [OO91]. However, [EO94] suffers from the problem that it demands tremendous load of modular multiplications from the customer in payment protocol, and the situation becomes worse if the node to be spent is located toward the

bottom of the tree. Furthermore, the amount of communication increases as the node to be spent is located toward the bottom of the tree.

In this paper, we present a different type of single term divisible off-line electronic cash system which is more efficient than previously proposed systems such as [OO91], [YLR93], and [EO94] in the standpoint of the amount of communication, and the number of modular multiplications required in the payment transactions, and the storage requirement in the withdrawal protocol. In addition, we have acquired the notion of untraceability and n-spendability. Our scheme is constructed on using restrictive blind digital signature scheme [Bra93], Schnorr's authentication scheme based on discrete logarithm problem [Sch91], and techniques of releasing the points on lines to catch double spenders.

The paper is organized as follows. In section 2, we present assumptions and several protocols. Section 3 examines the security and performance of the system. Finally, we conclude this paper with remarks in section 4.

2 Construction

In the next sections, we describe several protocols of the basic cash system.

2.1 Set-up of the system

- Let G_q denote a group of prime order q. For convenience, we use the subgroup G_q of Z_p^* with $q|p-1$ for a prime p.
- All the system parameters p and q are primes such that $p \geq 2^{512}$, $q|p-1$ and $q \geq 2^{160}$ and both are made public by the bank.
- All the generators g_1, g_2, g_z, g_k, d_z, $d_k \in G_q$ are made public by the bank B.
  1. The usage of g_1 and g_2 are related to the generation of customer's pseudonym, P.
  2. The usage of g_z is related *only* to coins having zero dollar.
  3. The usage of g_k is related *only* to coins having 2^k dollars.
  4. The usage of d_z is used to represent the value of the coin worth zero dollars.
  5. The usage of d_k is used to represent the value of the coin worth 2^k dollars.
- Let x be bank's secret key for signing 2^k dollar coin by the bank, and the corresponding public key is given as $h_k = g_k^x \bmod p$.
- Let x' be bank's secret key for signing zero dollar coin by the bank, and the corresponding public key is given as $h_z = g_z^{x'} \bmod p$.
- We assume the existence of a polynomial-time one way hash function $\mathcal{F} : \{0,1\}^* \rightarrow \{0,1\}^{3|q|}$.

- We assume the existence of a polynomial-time one way and uniform hash function $\mathcal{H}$ whose output lies in $[0, 2^t]$, where the security parameter $t \geq 80$.

2.2 Opening an account

When a customer U opens an account at the bank B, he generates random numbers $u_1, u_2 \in Z_q^*$, and computes $P = g_1^{u_1} g_2^{u_2} \bmod p$ that is his pseudonym connected to his account. The bank stores P with the user's real identity and account number A_U. Note that u_1 and u_2 must be kept secret by the customer.

2.3 Withdrawal protocol

Assume the customer U wishes to withdraw a divisible electronic coin $T_{U,k}$ of value 2^k dollars from his account A_U at bank B by conducting the following protocol. Recall that x is B's secret key for signing 2^k dollar coin, and the corresponding public key is given as $h_k = g_k^x \bmod p$.

step 1 The bank B deducts the proper amount of value, say the amount of value 2^k dollars, from U's account A_U and calculates $m = Pd_k \bmod p$. B then computes $z = m^x \bmod p$, $a = g^w \bmod p$, $b = m^w \bmod p$, and sends z, a, and b to U, where w is randomly chosen from Z_q.

step 2 The customer U generates random number $s, u, v \in Z_q^*$ and calculates $m' = m^s \bmod p = g_1^{u_1 s} g_2^{u_2 s} (d_k)^s \bmod p$. On the other hand, he selects a random value $\sigma \in Z_q$ for the coin worth 2^k dollars, and let $\mathcal{F}(\sigma) = (r_1 \| r_2 \| r_3)$. Note that each r_i, where $i = 1, 2, 3$ is of q bits long. U then calculates $T_{U,k} = g_1^{r_1} g_2^{r_2} (d_k)^{r_3} \bmod p$, $z' = z^s \bmod p$, $a' = a^u g^v \bmod p$, $b' = b^{su} (m')^v \bmod p$, $c' = \mathcal{H}(a', b', z', m', T_{U,k})$ and $c = c'/u \bmod q$. U then sends c to B.

step 3 B responds with $r = xc + w \bmod q$.

step 4 U checks the correctness of z, a, b, r, that is, U accepts if and only if $h^c a = g^r$ and $z^c b = m^r$. If the verification holds, U computes $r' = ru + v \bmod q$.

At the end of the withdrawal protocol, the customer ends up with an electronic coin $T_{U,k}$, and a valid signature $sign(T_{U,k}) = (z', a', b', r', m')$. This digital signature can be verified as follows:

$$g_k^{r'} \equiv (h_k)^{c'} a' \bmod p \quad (1)$$

$$(m')^{r'} \equiv (z')^{c'} b' \bmod p \quad (2)$$

$$c' = \mathcal{H}(a', b', z', m', T_{U,k}) \quad (3)$$

The withdrawal protocol is constructed on using the *restrictive* blind digital signature scheme. The term *restrictive* bears its name because the customer is restricted in his behaviour in the protocol such that he is not in a position to blind m at his disposal so that his identity P could no longer be determined if he double spends.

2.4 Payment protocol

Here, we describe two protocols for the payment transactions. First, we explain a basic payment protocol in which the customer U pays the exact amount of value of the coin without resorting to the dividing operation which subdivides the coin. Next we present a divisible payment protocol in that the customer U can pay any amount up to 2^k dollars with the divisible coin $T_{U,k}$ by subdividing it into two pieces of arbitrary values such that each subdivided piece is worth any desired value strictly less than 2^k dollars and the total value of all pieces is equivalent to 2^k. These protocols are based on the modified Schnorr identification scheme [Sch91], and the core idea of dividing operation of the coin is similar to that of [LL93] with minor modifications.

2.4.1 Basic payment protocol

Suppose the customer U initially has withdrawn an electronic coin, $T_{U,k} = g_1^{r_1} g_2^{r_2} (d_k)^{r_3} \bmod p$ which is worth 2^k dollars, from the bank. To pay a shop V an amount of money, say 2^k dollars, U and V proceed as follows :

step 1 The customer U sends $T_{U,k}$ and $sign(T_{U,k})$ to V.

step 2 The shop V verifies the validity of the coin $T_{U,k}$ after receiving $T_{U,k}$ and $sign(T_{U,k})$ [see (1), (2), and (3)]. V sends ρ_k, A_V and $Time$ to U, where ρ_k is a random number, A_V is V's account at bank B and $Time$ is the actual time and date the transaction occurred. V then computes $e_k = \mathcal{H}(T_{U,k} \| k \| \rho_k \| A_V \| Time)$.

step 3 U computes $e_k = \mathcal{H}(T_{U,k} \| k \| \rho_k \| A_V \| Time)$ and $Y_{1,k} = r_1 - u_1 s e_k \bmod q$, $Y_{2,k} = r_2 - u_2 s e_k \bmod q$, $Y_{3,k} = r_3 - s e_k \bmod q$. U then sends $Y_{1,k}, Y_{2,k}$, and $Y_{3,k}$ to V.

step 4 V verifies the following relations :

$$g_1^{Y_{1,k}} g_2^{Y_{2,k}} d_k^{Y_{3,k}} (m')^{e_k} \equiv T_{U,k} \bmod p \tag{4}$$

$$e_k = \mathcal{H}(T_{U,k} \| k \| \rho_k \| A_V \| Time) \tag{5}$$

If the customer U withdraws k electronic coins $T_{U,i}$, $i = 0, 1, \ldots, k-1$, of each 2^i dollars, then he is able to pay any amount up to $2^k - 1$ dollars. Notice that the double spending of the same coin $T_{U,k}$ in any way results in the discovery of perpetrator's identity [see section 3.1]

Next we describe a divisible payment protocol which gives more flexibility to the payer such that dividing operation of the coin enables the payer to spend any desirable amount of money strictly less than the maximum value. This is done by subdividing the coin of 2^k dollars into two pieces of arbitrary values such that each subdivided piece is worth any desired value strictly less than 2^k dollars and the total value of all pieces is equivalent to 2^k.

2.4.2 Divisible payment protocol

Assume the customer U wants to pay $J < 2^k$ dollars to the shop V with the electronic coin $T_{U,k} = g_1^{r_1} g_2^{r_2} (d_k)^{r_3} \bmod p$ which is worth 2^k dollars. U splits the coin $T_{U,k}$ into X_{1J} and X_{2L}, each of them being J (i.e.= 2^j) and L (i.e.= $2^k - J$) dollars, respectively. U then spends X_{1J} to the shop V by conducting the payment transaction described below.

step 1 U computes $X_{1J} = g_1^{r_1'} g_2^{r_2'} (d_j)^{r_3'} \bmod p$ and $X_{2L} = g_1^{r_1''} g_2^{r_2''} (d_l)^{r_3''} \bmod p$ and each of them represents the value of J and L dollars, respectively. Note that r_i' and r_i'', where $i = 1, 2$, and 3, are all random numbers of q bit size and are generated similarly as r_i of $T_{U,k}$ [see step 2 of section 2.3]. U also computes $e_k = \mathcal{H}(T_{U,k} \| k \| X_{1J} \| J \| X_{2L} \| L)$ and $Y_{1,k} = r_1 - u_1 s e_k \bmod q$, $Y_{2,k} = r_2 - u_2 s e_k \bmod q$, $Y_{3,k} = r_3 - s e_k \bmod q$. One can think of the computation as generating a signature for the message $(k, X_{1J}, J, X_{2L}, L)$ using Schnorr's scheme [Sch91]. We denote the transaction history as $H_k = (T_{U,k}, sign(T_{U,k}), Y_{1,k}, Y_{2,k}, Y_{3,k}, e_k, X_{2L})$, where $sign(T_{U,k}) = (z', a', b', r', T_{U,k})$. U then sends (X_{1J}, H_k) as an electronic coin of value J dollars to V.

step 2 The shop V first checks that $m' \neq 1$ (i.e. $s = 0$), and verifies the bank's signature by calculating c' in (3) and checks the relations (1) and (2) from the withdrawal protocol. The shop V then checks the validity of the coin as follows :

$$g_1^{Y_{1,k}} g_2^{Y_{2,k}} d_k^{Y_{3,k}} (m')^{e_k} \equiv T_{U,k} \bmod p \tag{6}$$

$$e_k = \mathcal{H}(T_{U,k} \| k \| X_{1J} \| J \| X_{2L} \| L) \tag{7}$$

step 3 From here on, the remaining steps are the same as the basic payment protocol. V sends ρ_{k1}, A_V and $Time$ to U, where $\rho_{k1} \in Z_q^*$ is a random number and $Time$ is the actual time and date the transaction occurred. V then computes $e_{k1} = \mathcal{H}(X_{1J} \| k \| \rho_{k1} \| A_V \| Time)$.

step 4 U computes $e_{k1} = \mathcal{H}(X_{1J} \| k \| \rho_{k1} \| A_V \| Time)$ and $Y_{1,k1} = r_1' - u_1 s e_{k1} \bmod q$, $Y_{2,k1} = r_2' - u_2 s e_{k1} \bmod q$, $Y_{3,k1} = r_3' - s e_{k1} \bmod q$. U then sends $Y_{1,k1}$, $Y_{2,k1}$, and $Y_{3,k1}$ to V.

step 5 V verifies the following relations :

$$g_1^{Y_{1,k1}} g_2^{Y_{2,k1}} d_k^{Y_{3,k1}} (m')^{e_{k1}} \equiv X_{1J} \bmod p \tag{8}$$

$$e_{k1} = \mathcal{H}(X_{1J} \| k \| \rho_{k1} \| J \| A_V \| Time) \tag{9}$$

In summary, in order for the customer U to pay $J < 2^k$ dollars to the shop, U follows the divisible payment protocol using (X_{1J}, H_k) as an electronic coin of value J dollars, where $H_k = (T_{U,k}, sign(T_{U,k}), Y_{1,k}, Y_{2,k}, Y_{3,k}, e_k, X_{2L})$. After the payment protocol has been completely executed, both the payer and payee must store the resulting transaction history, $H_{k1} = (H_k, e_{k1}, Y_{1,k1}, Y_{2,k1}, Y_{3,k1}, \rho_{k1}, A_V, J, Time)$, since the payer will need it for spending the remaining part of the electronic coin (i.e. X_{2L} of value L dollars) in later transaction, and the payee will need it either for his later payment or for deposit of that received coin.

2.5 Transfer protocol

This protocol allows the electronic coin of U_1 to be transferred to another customer U_2 by means of transferring the ownership of the original coin. U_2 then can later spend the transferred coin to a third person by taking the role of U in the payment protocol abovementioned. The number of possible transfers that can be made is determined by the number of *blank coins* (zero dollar coins) issued by the bank so that only a finite number of transferring is allowed. Suppose U_1 wishes to transfer the coin $T_{U_1,k} = g_1^{r_1} g_2^{r_2} d_k^{r_3} \bmod p$ to U_2. Then U_1 and U_2 proceeds as follows.

Assumptions :

- The customer U_1 and U_2 open their accounts A_{U_1} and A_{U_2}, respectively at bank B.
- The transferee U_2 must obtain a *blank coin* $T_{U_2,z}$ from the bank prior to his interaction with U_1 in the transfer protocol. Thus, U_2 conducts the withdrawal protocol as described in section 2.3, and ought to have $T_{U_2,z}$ and $sign(T_{U_2,z}) = (z'_z, a'_z, b'_z, r'_z, m'_z)$, where $T_{U_2,z} = g_1^{r'''_1} g_2^{r'''_2} (d_k)^{r'''_3} \bmod p$.
- A blank coin is defined as a coin having zero dollar bearing a valid bank's signature.
- Note that the number of blank coins in a smart card determines the number of possible transfers that can be made.

step 1 U_1 sends $T_{U_1,k}$ and $sign(T_{U_1,k})$ to U_2.

step 2 U_2 verifies the authenticity of the coin $T_{U_1,k}$ for $sign(T_{U_1,k})$ [see (1), (2), (3)].
U_2 then sends $T_{U_2,z}, sign(T_{U_2,z})$, and ρ_{k2} to U_1,
and calculates $e_k = \mathcal{H}(T_{U_1,k} \| k \| \rho_{k2} \| T_{U_2,z})$,
where $\rho_{k2} \in Z_q^*$ is a random number.

step 3 U_1 checks the validity of the coin $T_{U_2,z}$ for $sign(T_{U_2,z})$
then computes $e_k = \mathcal{H}(T_{U_1,k} \| k \| \rho_{k2} \| T_{U_2,z})$,
$Y_{1,k} = r_1 - u_1 s e_k \bmod q$, $Y_{2,k} = r_2 - u_2 s e_k \bmod q$,
$Y_{3,k} = r_3 - s e_k \bmod q$.
U_1 then sends $Y_{1,k}, Y_{2,k}, Y_{1,k}$ to U_2.

step 4 U_2 checks that

$$g_1^{Y_{1,k}} g_2^{Y_{2,k}} d_k^{Y_{3,k}} (m')^{e_k} \equiv T_{U_1,k} \bmod p \text{ and } e_k = \mathcal{H}(T_{U_1,k} \| k \| \rho_{k2} \| T_{U_2,z}).$$

If U_1 wishes to transfer X_{1J}, the subdivided coin of $T_{U_1,k}$, instead of transferring $T_{U_1,k}$, then, in step 1, U_1 sends $(T_{U_1,k}, sign(T_{U_1,k}), X_{1J}, H_k)$ to U_2, and in step 2 U_2 needs to verify the authenticity of X_{1J} for H_k, in addition to checking the validity of $T_{U_1,k}$ [see (8), (9) except that $e_k = \mathcal{H}(T_{U_1,k} \| k \| \rho_{k2} \| T_{U_2,z})$.

2.6 Payment of a transferred cash

If the customer U_2 wants to pay the shop V with his transferred coin $T_{U_2,z}$ which now is worth 2^k dollars, he engages in the following protocol.

Note that x is the bank's secret key for signing zero dollar coin, and the corresponding public key is $h_z = g_z^{x'} \bmod p$. We denote $TH_{12,k}$ as a transfer history containing all the information pertaining to the coin worth 2^k dollars that has been transferred from U_1 to U_2.

step 1 U_2 sends $(T_{U_2,z}, TH_{12,k})$ to V.

$TH_{12,k} = (T_{U_2,z}, sign(T_{U_2,z}), k, \rho_{k2}, T_{U_1,k}, sign(T_{U_1,k}), Y_{1,k}, Y_{2,k}, Y_{3,k})$
$sign(T_{U_2,z}) = (z'_z, a'_z, b'_z, r'_z, m'_z)$
$sign(T_{U_1,k}) = (z', a', b', r', m')$
$T_{U_2,z} = g_1^{r'''_1} g_2^{r'''_2} (d_k)^{r'''_3} \bmod p$
$T_{U_1,k} = g_1^{r_1} g_2^{r_2} (d_k)^{r_3} \bmod p$
$m'_z = m_z^{s'} \bmod p = g_1^{u_1 s'} g_2^{u_2 s'} (d_z)^{s'} \bmod p$
$m' = m^s \bmod p = g_1^{u_1 s} g_2^{u_2 s} (d_k)^s \bmod p$

step 2 V verifies the validity of the coin $T_{U_2,z}$ for $TH_{12,k}$. The verification is as follows:

Check list #1: Does $T_{U_2,z}$ bear the bank's digital signature?

$$g_z^{r'_z} \equiv (h_z)^{c'_z} a'_z \bmod p \tag{10}$$
$$(m'_z)^{r'_z} \equiv (z'_z)^{c'_z} b'_z \bmod p \tag{11}$$
$$c'_z = \mathcal{H}(a'_z, b'_z, z'_z, m'_z, T_{U_2,z}) \tag{12}$$

Check list #2: Does $T_{U_1,k}$ bear the bank's digital signature?

$$g_k^{r'} \equiv (h_k)^{c'} a' \bmod p \tag{13}$$
$$(m')^{r'} \equiv (z')^{c'} b' \bmod p \tag{14}$$
$$c' = \mathcal{H}(a', b', z', m', T_{U_1,k}) \tag{15}$$

Check list #3: Is $T_{U_2,z}$ a valid transferred coin from $T_{U_1,k}$?

$$g_1^{Y_{1,k}} g_2^{Y_{2,k}} d_k^{Y_{3,k}} (m')^{e_k} \equiv T_{U_1,k} \bmod p \tag{16}$$
$$e_k = \mathcal{H}(T_{U_1,k} \| k \| \rho_{k2} \| T_{U_2,z}) \tag{17}$$

step 3 If the corroboration of three check lists are successful, then V sends (rho_{kv}, A_V, $Time$) to U_2, and computes $e'_k = \mathcal{H}(T_{U_2,z} \| k \| \rho_{kv} \| A_V \| Time)$ and $Y_{1,k2} = r_1 - u_1 s' e_k \bmod q$, $Y_{2,k2} = r_2 - u_2 s' e_k \bmod q$, $Y_{3,k2} = r_3 - s' e_k \bmod q$. U then sends $Y_{1,k2}, Y_{2,k2}$, and $Y_{3,k2}$ to V.

step 4 V also computes $e'_k = \mathcal{H}(T_{U_2,z}\|k\|\rho_{kv}\|A_V\|Time)$ and verifies the following relations :

$$g_1^{Y_{1,k2}} g_2^{Y_{2,k2}} d_k^{Y_{3,k2}} (m'_z)^{e'_k} \equiv T_{U_2,z} \bmod p \tag{18}$$

$$e'_k = \mathcal{H}(T_{U_1,k}\|k\|\rho_{kv}\|A_V\|Time) \tag{19}$$

If the attestation of equation (10) through (19) are valid, then the shop V regards the coin $T_{U_2,z}$ as trustworthy. Notice that without a customer having obtained a valid blank coin from the bank, no customer is able to have a third person's coin be transferred. There are several possible inappropriate attempts that one might think of, but they all lead to either double-spending of the same coin or illegal usage of the coin. In order to spend the valid blank coin $T_{U_2,z}$, it requires any valid coin (i.e. $T_{U_1,k}$) with some value be transferred to the blank coin. Any form of double-spending the transferred coin leads to the detection of customer's identity as in usual basic payment protocol. Note that one can not spend only a blank coin (without another coin worth certain value being transferred to it) as if it were a coin having some amount since the bank has signed the coin with the secret key x' (corresponding public key is $h_z = g_z^{x'} \bmod p$) that is used only for the blank coin.

2.7 Deposit protocol

The shop V forwards his transcript of payment to the bank B. B then verifies all the signatures just as the shop did in the payment protocol and checks that the coins are not double spent. If all tests are successful, B credits V's account with proper value, and stores the transaction history to its database.

2.8 Multi-spendable electronic coin

In this section, we provide n-spendable coins, as opposed to 1-spendable coins, that is, coins that can be spent n times without his identity being revealed. However, spending $(n+1)$-th time (*or* more than n times) unfolds the violator's identity.

One of the foreseeable applications that might think of is in the area of n-trip subway ticket or n-spendable toll-gate ticket.

The n-spendable coins are more efficient than n 1-spendable coins in view of memory. In our scheme, n 1-spendable coin needs about $340n$ bytes, but n-spendable coin requires about $64n + 276$ bytes. However, the computational requirements for a customer in the payment are much higher for an n-spendable coin. For the sake of simplicity, we compare them in *basic* payment protocol. In our scheme, n-spendable coin requires $36n$ modular multiplications, whereas n 1-spendable coins need $5n$ modular multiplications for a customer in the payment protocol. On the verifier's side, he has to compute $840n$ number of modular multiplications for n 1-spendable coin, and $173n+720$ multiplications for n-spendable coin. So, using n-spendable coin lessens the verifier's computational load.

As all previously proposed systems, the usage of multi-spendable coin can be linked together by the bank. Thus, unlinkability is lost, but not untraceability.

The basic idea of extension to multi-spendability of our scheme has been borrowed from [Bra93] with minor modifications. We only give a brief descriptions.

At the withdrawal protocol, the bank first deducts n times the value of n-spendable coin from the customer's bank account. Then the customer U now randomly splits each of the exponents of m' as follows : $(m' = m^s \bmod p = g_1^{u_1 s} g_2^{u_2 s} (d_k)^s \bmod p)$

$$u_1 s = \sum_{i=1}^{n} \alpha_i \tag{20}$$

$$u_2 s = \sum_{i=1}^{n} \beta_i \tag{21}$$

$$s = \sum_{i=1}^{n} \gamma_i \tag{22}$$

Then U computes $B_1 = g_1^{\alpha_1} g_2^{\beta_1} (d_k)^{\gamma_1} \bmod p$, $B_2 = g_1^{\alpha_2} g_2^{\beta_2} (d_k)^{\gamma_2} \bmod p$, ..., and $B_n = g_1^{\alpha_n} g_2^{\beta_n} (d_k)^{\gamma_n} \bmod p$. U then sends $c = \mathcal{H}(z', a', b', r', B_1, B_2, \ldots, B_n, T_{U_1,k})/u \bmod q$ to the bank, where $T_{U_1,k} = g_1^{r_1} g_2^{r_2} (d_k)^{r_3} \bmod p$. B sends the response, $r = xc + w \bmod q$ to U, and U checks the correctness of response as in step 4 of section 2.3.

At payment time, U sends $(B_1, B_2, \ldots, B_n, z', a', b', r', T_{U_1,k})$ to the shop V. After receiving the data, V verifies the validity of the coin $T_{U_1,k}$ and sends a proper challenge e_k to U. U responses with

$$Y_{1,k} = r_1 + \sum_{i=1}^{n} \alpha_i e_k^i \bmod q \tag{23}$$

$$Y_{2,k} = r_2 + \sum_{i=1}^{n} \beta_i e_k^i \bmod q \tag{24}$$

$$Y_{3,k} = r_3 + \sum_{i=1}^{n} \gamma_i e_k^i \bmod q \tag{25}$$

V then verifies the following relationship before regarding the coin as valid.

$$g_1^{Y_{1,k}} g_2^{Y_{2,k}} d_k^{Y_{3,k}} \equiv (T_{U_1,k})(B_1)^{e_k} (B_2)^{e_k^2}, \ldots, (B_n)^{e_k^n} \bmod p \tag{26}$$

$$e_k = \mathcal{H}(T_{U_1,k} \| k \| \rho_k \| A_V \| Time) \tag{27}$$

Notice that spending the coin $T_{U_1,k}$ $(n+1)$-th time (*or* more than n times) unfolds the customer's identity.

3 Analysis of the system

Here, we will briefly explain that double spending of an electronic coin can be detected and discuss the performance of our proposed cash system with respect to its computation load, communication amount, and storage requirements. We assume the use of a square-and-multiply algorithm [Knu69] in performance analysis.

3.1 Security evaluation (Detection of multiple-spending)

We must guarantee that illegitimate transaction results in the revelation of a customer's identity, whereas legitimate spendings protect his privacy. Here, illegitimate transaction means that either spending the same coin more than once, or spending any of the child coins whose parent coins have already been spent. The term, *child coins*, is defined as all the subdivided coins of either the root coin or parent coins. The *root coin* is the original coin first issued by the bank to the customer with some value.

For example, when the same root coin $T_{U,k}$ is spent twice, we have a set of equations with the same r_i and s (i.e. with different challenge e_k in the response we have $Y_{1,k}, Y_{2,k}, Y_{3,k}, Y'_{1,k}, Y'_{2,k}, Y'_{3,k}$.) As a result, we can completely solve this system of equations for u_1 and u_2. Consequently, unvailing the violator's identity, $P = g_1^{u_1} g_2^{u_2} \bmod p$. Any form of double spending or inappropriate spending is detectable, and perpetrator's identity gets revealed.

3.2 Performance estimation

3.2.1 Computation time

The advantage of our scheme over [OO91], [YLR93] and [EO94] is that it requires much less computation time in the payment transaction.

[OO91] requires the payer to compute many of square-rooting operations, depending on the amount of payment. That is, the square-rooting operations increase as the coin to be spent is located toward the bottom of the hierarchical structure table. Therefore, either withdrawing a large amount of money initially from the bank, or paying the amount in small units (i.e. cents) burdens the customer in his payment when he tries to spend a node near the bottom of the hierarchical structure table. The number of square-rooting operations for single coin is equal to the level of coin to be spent.

In [YLR93], the number of modular multiplications in payment is about $770y + 2304$, where y is the number of coins to be spent.

In [EO94], many modular multiplications are required for the payer in the payment transaction. In particular, the situation becomes worse if the node (in a binary tree) to be spent is located toward the bottom of the tree. For example, for every spent node in the hierarchical structure tree it requires about $720(\log_2 \omega - \log_2 \omega')$ modular multiplications, where w and w' are the amount of money of root node and the amount of value of node to be spent, respectively. Notice that each argument of $\mathcal{H}'(\ )$ in [EO94] requires about $720(= 240 \cdot 3)$ modular multiplications. $\mathcal{H}'(\ )$ is a polynomial-time one-way hash function where its output is $3|q|$ bits long.

In our scheme, the total number of modular multiplications required by the customer in the divisible payment protocol is approximately 1450, and it is about three times more than that of [LL93]. Suppose U splits the coin $T_{U,k}$ into X_{1J} and X_{2L}, each of them being J and L $(= 2k - J)$ dollars, respectively, and wishes to spend X_{1J} to the shop V. The factor of three is mainly due to computing $X_{1J} = g_1^{r_1'} g_2^{r_2'} (d_j)^{r_3'} \bmod p$ and $X_{2L} = g_1^{r_1''} g_2^{r_2''} (d_j)^{r_3''} \bmod p$ where r_i' and r_i'' are each q bits long.

3.2.2 Communication amount

The communication amount in the payment transaction of our scheme is approximately equal to that of [LL93] and [EO94], and is more efficient than [OO91] and [YLR93].

Our scheme requires approximately $100t+440$ bytes, and [LL93] needs about $60t+370$ bytes, where t is the number of subdivision made for a coin.

In [EO94] the amount of communication for single coin in payment is about $60(\log_2 \omega - \log_2 \omega') + 530$ bytes, where w and w' are the amount of money of root node and the amount of value of current node, respectively. Thus, the communication load increases as the node to be spend tends to be located at the bottom of the tree and the original withdrawn coin has high value.

The communication amount in [OO91] is approximately $130y + 5200$ bytes, and that of [YLR93] is about $1470y + 180 \cdot 2^l + 5580$ bytes, where y is equal to the number of coins needed to pay desirable amount of payment to the shop and l is the number of levels in the binary tree.

3.2.3 Memory requirement

As mentioned earlier in [LL93], our scheme suffers from the fact that every subdividing operation of an electronic coin increases the data size of the resulting coin pieces and computational load for the shop for verifying them. Furthermore, the increased data which is primarily the transaction history, burdens the bank with its continuously growing database.

The data increase in our scheme is approximately 100 bytes per subdivision (e_{k1} : $10, Y_{1k,1} : 20, Y_{2k,1} : 20, Y_{3k,1} : 20, \rho_{k1} : 20, A_V : 4, J : 2, Time : 4$), whereas [LL93]'s are of 60 bytes per subdivision.

We consider the storage requirements in withdrawal protocol. For the sake of comparison, we give the results for withdrawing 16 electronic coins and for withdrawing single coin.

In [OO91], about 5300 bytes are required for electronic licence and its related data, and 64 bytes for a coin. Thus, about 5370 for single coin and about 6340 bytes for 16 coins are approximately needed. In [LL93], about 3180 and 790 bytes are required for withdrawing 16 electronic coins and for withdrawing single coin, respectively. In [YLR93], a customer needs about 4480 bytes for storing electronic licence and its related informations, and about $180 \cdot 2^l + 2440$ bytes for storing an electronic coin. Here, l denotes the number of levels in the binary tree. Thus, approximately $180 \cdot 2^l + 6930$ bytes are needed for a coin. In [EO94], about 5670 and 570 bytes are required for withdrawing 16 electronic coins and for withdrawing single coin, respectively. Our scheme requires

about 5770 and 670 bytes for withdrawing 16 electronic coins and for withdrawing single coin, respectively.

Comments In [LL93], Lim and Lee have introduced a trusted authority (i.e. License Issuing Authority) to trace monetary transactions under a legal permission. Theoretically, unconditional traceability in payment is attractive especially to the customers who are very concerned about their privacy, but in other situations traceability under a legal permission is desired in case investigation against criminal or terrorist groups is needed. Notice that, although the exact customer cannot be traced in [LL93], all transactions made with the same P_U (anonymous public key of the customer U) can be linked together! Consequently, the customer must obtain *several* electronic licenses simultaneously (of size 64 bytes each), and uses them at random in payment as mentioned in [LL93].

Any electronic cash system where its divisibility is based on fixed hierarchical structure tree not only suffers from its computational load as the coin to be spent is located at the bottom of the tree, but it also suffers from the standpoint of flexibility. Assume that a customer has \$512 ($= 2^9$) in his smart card, and wishes to spend \$250 at shop. He ought to compute for the coins worth \$128 ($= 2^7$), \$64 ($= 2^6$), \$32 ($= 2^5$), \$16 ($= 2^4$), \$8 ($= 2^3$), and \$2 ($= 2^1$) to meet the required amount of payment. However, the dividing operation with Schnorr's authentication scheme requires only one subdivision from \$512 with extra memory.

Table 1 and Table 2 summarize the characteristics of different types of electronic cash systems, and provides the evaluation according to each item, respectively. The number of asterisks (*) represents the degree of *efficiency* for each item. Each item may receive four asterisks for full credit, and none for the lack of property. The assessment has been made on the basis of the analysis in section 3.

4 Concluding Remarks

In this paper, we presented a practical off-line electronic cash system achieving untraceability, divisibility, transferability, and multi-spendability. The main advantage of our scheme is its *flexibility* and *efficiency* of dividing operation of an electronic coin than that of previously proposed schemes such as [OO91], [YLR93] and [EO94]. However, our scheme still inherits the problem of continuously growing data size as the coin gets subdivided, since the previous histories have to be accumulated for each subdivided coin pieces.

Table 1: Characteristics evaluation

	[OO91]	[YLR93]	[LL93]	[EO94]	**Our scheme**
Independence	****	****	****	****	****
Unreusability	****	****	****	****	****
Untraceability	****	****		****	****
Traceability[1]			****		
Off-line	****	****	****	****	****
Transferability	***	*	***		**
Divisibility	**	*	****	***	****
N-spendability[2]					****

1 : Traceability has been included as an evaluation criteria, since it is beneficial in the case where monetary transactions must be traced to investigate against criminal under a legal permission.

2 : For other N-spendable coin systems, the reader is refer to [Fer93B] and [Bra93].

Table 2: Storage, communication, and computation evaluation

	[OO91]	[YLR93]	[LL93]	[EO94]	Our scheme
MR	***	**	****	****	****
SH	***	**	***	****	***
CA	**	*	****	***	****
CL	**	**	****	**	***

MR : Initial storage requirement of a coin in withdrawal (including the storage of license, if any).

SH : Storage requirement of histories in deposit.

CA : Communication amount in payment.

CL : Computation load in payment by the customer.

References

[BCHMS89] Bert den Boer, David. Chaum, Eugene van Heyst, Stig Mjøsnes, and Adri Steenbeck, "Efficient offline electronic checks, " Advances in Cryptology-Eurocrypt '89, Lecture Notes in Computer Science, Springer-Verlag, 1990, pp.294-301.

[Bra93] S. Brands, "Untraceable Off-line Cash in Wallet with Observers," (Extended abstract) Advances in Cryptology-Crypto '93, Lecture Notes in Computer Science, Springer-Verlag, 1994, pp.302-318.

[Cha82] D. Chaum, "Blind signatures for untraceable payments," Advances in Cryptology-Crypto'82, Plenum Press, 1983, pp. 199-203

[Cha85] D. Chaum, "Security without Identification : Transaction Systems to Make Big Brother Obsolete," Comm. ACM, 28, 10, 1985, pp.1000-1044.

[CFN88] D.Chaum, A.Fiat, and M.Noar, "Untraceable Electronic Cash," Advances in Cryptology-Crypto '88, Lecture Notes in Computer Science, Springer-Verlag, 1990, pp.319-327.

[EO94] T. Eng, T. Okamoto, "Single-Term Divisible Electronic Coins," Pre-Proceedings of Eurocrypt '94, May 9-12, 1994, University of Perugia, Italy, pp313-323.

[Fer93A] Niels Ferguson, "Single term off-line coins," Advances in Cryptology-Eurocrypt '93, Lecture Notes in Computer Science, Springer-Verlag, 1990, pp.318-328.

[Fer93B] Niels Ferguson, "Extensions of Single-term Coins," Advances in Cryptology-Crypto '93, Lecture Notes in Computer Science, Springer-Verlag, 1994, pp.292-301.

[Knu69] Donald E. Knuth, The Art of Computer Programming, Vol 2: Seminumerical Algorithms, Addison-Wesley, Reading Mass., 1969, pp441-444.

[LL93] Chae Hoon Lim and Pil Joong Lee, "A Practical Electronic Cash System for Smart Cards," 1993 Korea-Japan Joint Workshop on Information Security and Cryptology '93 Proceedings, October 24-26, 1993, Seoul, Korea, pp34-47.

[OO89] Tatsuaki Okamoto and Kazuo Ohta, "Disposable Zero- Knowledge Authentications and Their Applications to Untraceable Electronic Cash," Advances in Cryptology-Crypto '89, Lecture Notes in Computer Science, Springer-Verlag, 1990, pp.482-496.

[OO91] Tatsuaki Okamoto and Kazuo Ohta, "Universal electronic cash," Advances in Cryptology-Crypto '91, Lecture Notes in Computer Science, Springer-Verlag, 1992, pp.324-337.

[Sch91] C.P. Schnorr, "Efficient Signature Generation by Smart Cards," Journal of Cryptology, Vol 4, No. 3, pp. 161-174 (1991)

[YLR93] HeungYoul Youm, SeoLae Lee, and ManYoung Rhee, "Practical Protocols for Electronic Cash," 1993 Korea-Japan Joint Workshop on Information Security and Cryptology '93 Proceedings, October 24-26, 1993, Seoul, Korea, pp10-22.