Design and Implementation of Revocable Electronic Cash System based on Elliptic Curve Discrete Logarithm Problem

Heesun Kim[1], Joonsang Baek[2], Gookwhan Ahn[1], Jinho Kim[1], Hyuncheol Park[1], Boyeon Song[1], Manho Lee[1], Jaegwan Park[1], Jaeseung Go[1], Byoungcheon Lee[1] and Kwangjo Kim[1]

[1] Information and Communications University(ICU), 58-4 Hwaam-dong, Yusong-gu, Taejon, 305-732, Korea
[2] SECUi.COM, 647-9, Yeoksam-dong, Kangnam-gu Seoul, 135-080, Korea

Abstract. We have designed and implemented a revocable electronic cash system whose main security is based on ECDLP (Elliptic Curve Discrete Logarithm Problem). To achieve this, we employed a known secure electronic cash system based on DLP (Discrete Logarithm Problem) suggested by Petersen and Poupard [19] and extended it on an elliptic curve over the finite field $GF(2^n)$. This naturally reduces the message size to 85% compared with the original scheme and makes it possible to handle a smart card. Furthermore, we have achieved more secure and efficient electronic cash system by using Song and Kim's key agreement protocol [21] and Baek *et al.*'s provably secure public key encryption scheme [1]. To the best of our knowledge, this is the first trial to implement revocable electronic cash system based on ECDLP having provable security.

1 Introduction

A variety new services using cryptographic primitives becomes popular over the existing network. Among these, the electronic cash system is one of enabling and attractive technological application to electronic commerce now.

The research in the field of the electronic cash (e-cash) has progressed rapidly since the concept of the anonymous e-cash with blind signature [8] was initially suggested by Chaum. Later, he suggested anonymous on-line payment system [6] and he also proposed anonymous off-line e-cash system [7], which motivated the research of the e-cash system. They achieved double-spending prevention of an e-cash with cut-and-choose method. The e-cash system proposed by Okamoto and Ohta satisfies the requirements of divisibility and transferability [17, 18]. Their scheme overcomes some limitations of the previous e-cash system and provides more efficient features than the physical currency. Brands proposed the efficient e-cash system with single-term method which is more efficient compared with cut-and-choose method [2]. Brands' scheme has been used as a basic model by many other researches.

Some problems, however, regarding the abuse of perfect anonymity in the e-cash system are raised in [25]. Related to the problem such as blackmailing and money laundering are also presented in [3, 12, 19, 25]. The revocable e-cash system (fair payment system) in which anonymity can be revoked by a trustee [3, 5] becomes one of the research areas in order to address these issues. In the revocable e-cash scheme the user's identification can be traced by the cooperation of a trustee and a bank.

Along with measures [3, 24] against the blackmailing and money laundering, many schemes [5, 10, 12, 16, 19] intended to protect the attack based on the abuse of anonymity have been proposed. Two schemes [4, 9] require the trustee to take part in the initialization phase and other systems to participate in the account opening or withdrawal phase. However, these systems cannot prevent extortion and blindfolding attacks. The systems to prevent these attacks are suggested in [10, 12, 19].

However, the protocols [10, 12] are not efficient owing to the communication with the trustee in the payment phase. Therefore, if an attack is reported in these system, it is required to review the payment protocol among user, shop and trustee to prevent the illegal usage of the e-cash. On the other hand, Petersen and Poupard suggested more efficient off-line electronic payment system [19]. Their scheme makes it possible by letting the user register the pseudonymous public key to the trustee in the registration phase. It is also secure against many kinds of attacks such as secret key extortion, framing against the user, trustee's blindfolding, blackmailing, and money laundering.

1.1 Security Requirements

In general, we can classify the security requirements [12, 13, 19] of e-cash system into the two part as :

Basic Requirements

- Unforgeability : Only authorized bank can issue coins.
- Double-spending prevention : One coin cannot be used more than once.
- Anonymity : A bank cannot link a coin to the honest owner of a coin without the trustee's help.
- Untraceability : A bank cannot trace the relationship between a coin and the user of a coin without a lawful order.
- Efficiency : The system is efficient in terms of storage, communication and computation.

Additional Requirements

- Revocability : Any coin which bank robber obtains must be revealed.
- User-tracing : A bank can legally trace the user of a paid coin with the trustee's help.
- Extortion-tracing : A bank can legally compute the matching information of the paid or deposited coin with the trustee's help.

- Blindfolded-freeness : Anyone cannot obtain a blinded coin without the bank's knowledge that this particular coin has been blinded.
- Coin-tracing : A bank can legally link the information of the coin to be used with the information of deposited coin with the trustee's help.
- Unlinkability : It is not possible to find relation between different coins used by the same user.
- Refundability : It is possible to refund a coin if a legally withdrawn coin cannot be accepted by a bank or a trustee.
- Fairness : User's anonymity w.r.t. the bank and the trustee must be guaranteed and the anonymity must be revocable with the cooperation of the bank and the trustee.
- Framing-freeness : Any user or shop cannot be falsely incriminated by a bank or a trustee.
- Overspent-tracing : It is possible to trace the over-spending user's identification.
- Transferability : Once withdrawn coins can be transferred to any other user.
- Divisibility : The denomination of a coin can be divided into lower unit.

1.2 Petersen and Poupard's Scheme

Petersen and Poupard [19] suggested an off-line, revocable e-cash system (abbreviated as "PePo97") which consists of initialization, account opening, registration, withdrawal, payment, deposit, and revocation protocols. PePo97 is shown to be secure against the attacks of secret key extortion, coin extortion, blackmailing, forgery, framing, and blindfolding. It also satisfies the requirements of double-spending prevention, k-spendability, divisibility, anonymity revocation, coin-tracing, and refundability. These requirements were provided if existential forgery of a signature w.r.t. an adaptively chosen message attack is equivalent to a known computational hard problem (e.g. factorization or discrete log.) [19]. Off-line protocol is achieved by the registration of the user's pseudonymous public key to the trustee and the prevention against extortion attack by the concrete usage of secure revocation lists and database. PePo97 has two type of protocols, i.e., internet payment and electronic purse protocols. Two protocols are quite different in terms of the security and efficiency.

1.3 Our Goals and approach

On the other hand, we have aimed to implement the internet payment protocol handling the smart card. A new e-cash system based on ECDLP (called as "cashpia-v2") has targeted small-amount-of-money transaction for anyone who wants to use an e-cash without a credit card. The anonymity is of prime concern because it deals with small-amount-of-money transaction. Generally, when an honest one pays large-amount-of-money transaction(e.g. purchasing a house, purchasing or renting a car, *etc.*), he may not want to pay it anonymously, because most people may want to make sure they paid the money themselves.

While, in the small-amount-of-money transaction (e.g. using adult service, purchasing private goods, *etc.*), most people want purchasing details to be unknown. The anonymity has to be applied to cashpia-v2 because the complete anonymity induces crimes when abused. Cashpia-v2 satisfies the anonymity but, if necessary, it is designed to revoke the anonymity and also provide preventive functionalities against money laundering, blackmailing, and double spending. Cashpia-v2 has aimed to satisfy basic and most additional requirements of e-cash system. As stated before, however, cashpia-v2 dosen't meet divisibility, transferability and overspent-tracing.

We have adapted PePo97 because it has most of the functionalities our system is trying to achieve. Furthermore, it is also easy to implement due to the usage of simple and concrete DB. However, their scheme requires high cost w.r.t. the storage and communication to handle continuously increasing DB. Therefore, it is hard to implement and manage the system over the storage limited environment such as a smart card.

To enforce the efficiency and security, we employed Song and Kim's new key agreement protocol (abbreviated as "SK's protocol") [21] and Baek *et al.*'s scheme as public key encryption scheme [1]. Both schemes are optimally designed on the elliptic curve over the finite field. SK's protocol satisfies K-KS (Known-Key Security), FS (Forward Secrecy), K-CI (Key-Compromise Impersonation) and UK-S (Unknown Key-Share). And Baek *et al.*'s scheme is length-efficient in a sense of a shorter ciphertext than other scheme [20] based on CDH-A (Computation Diffie-Hellman Assumption) and provably secure against the chosen-ciphertext attack. This scheme is called PSLC-2 (Provably Secure Length-saving public-key encryption scheme based on Computational D-H assumption).

We have implemented cashpia-v2 using the cryptographic library ICUCLIB-v2 (ICU Cryptographic LIBrary-version 2) [11] developed by ICU. ICUCLIB-v2 provides cashpia-v2 with various cryptographic primitives and makes the implementation easy.

Organization :

In Section 2, we describe the cryptographic primitives such as key agreement protocol and public key encryption scheme in brief. In Section 3, we describe the details of cashpia-v2 and show each protocol. And the implementation of cashpia-v2 is described in Section 4. In Section 5, we analyze cashpia-v2 in terms of the security and efficiency. Finally, we conclude this paper with summary and suggest future work in Section 6.

2 Cryptographic primitives

In this section we describe basic cryptographic primitives used to design cashpia-v2. We perform key exchange protocol to generate a session key, which is required to identify between two entities and to exchange authenticated messages. Cashpia-v2 encrypts all messages to be transferred using the session key. We use SK's key agreement protocol which is strong against known attacks. We also adapt PSLC-2 used for the encryption of the information of a coin with trustee's public key. This operation is required to protect a withdrawn coin signed by bank's secret key when the key is extorted. We summarize SK's protocol and PSLC-2 working on the elliptic curve over the finite field in brief.

2.1 SK's Key Agreement Protocol

Cashpia-v2 generates a session key in the registration and withdrawal protocols. The session key is required to authenticate communication messages and shared by SK's protocol on the elliptic curve over the finite field.

Notation :

- A, B : honest entities
- Y : entities, $Y \in \{A, B\}$
- $E(GF(p))$: elliptic curve
- $\#E(GF(p))$: order of $E(GF(p))$
- P : base point
- q : large prime, $q|\#E(GF(p))$
- (Q_Y, x_Y) : Y's static key pairs. Y's public key $Q_Y = x_Y P$ for Y's private key x_Y
- (R_Y, k_Y) : Y's ephemeral key pairs. Y's public key $R_Y = k_Y P$ for Y's private key k_Y
- w : cofactor $w = \#E(GF(p))/q$

Public Key Validation :
A purported public key $Q = (x_Q, y_Q)$ can be validated by verifying that :

V1. Q is not equal to $\mathcal{O}$ which is a point at infinity.
V2. x_Q and y_Q are elements in the $GF(p)$.
V3. Q satisfies the defining equation of E.
V4. $nQ = \mathcal{O}$.

To reduce operation cost of the scalar multiplication, V4 can be omitted during public key validation. It is called embedded public key validation.

Protocol :
Entities' certificates have to be previously distributed to each other. If not, certificates must be sent to each other with ephemeral public key.

P1. Entity A generates a random integer $k_A, 1 \leq k_A \leq n-1$, computes the point $R_A = k_A P$, and sends it to entity B.
P2. Entity B generates a random integer $k_B, 1 \leq k_B \leq n-1$, computes the point $R_B = k_B P$, and sends it to entity A.

P3. A performs an embedded public key validation of R_B. If the validation fails, then A terminates performing the protocol with failure. Otherwise, A computes $Z = k_A Q_B + (x_A + k_A) R_B$ and $K = wZ$. If $K = \mathcal{O}$, then A terminates performing the protocol with failure.

P4. B performs an embedded public key validation of R_A. If the validation fails, then B terminates performing the protocol with failure. Otherwise, B computes $Z = k_B Q_A + (x_B + k_B) R_A$ and $K = wZ$. If $K = \mathcal{O}$, then B terminates performing the protocol with failure.

P5. The shared secret key is the point K.

2.2 PSLC-2 Public Key Encryption Scheme

We describe PSLC-2 as follows :

- Key generator $\mathcal{K}$
  - Choose a non-supersingular elliptic curve defined on Galois field $GF(p)$, $E(GF(p))$, and calculate the order of $E(GF(p))$, $\#E(GF(p))$. Let q be a large prime number dividing $\#E(GF(p))$ and let P be a point of order q on $E(GF(p))$.
  - $pk = (E, P, q, W(= uP))$ and $sk = (E, P, q, u)$ where $u \in_R GF(q)$ and $|p| = k = k_0 + k_1$.
- Hash Function (two random oracles)
  - Choose $H : \{0,1\}^k \rightarrow GF(q)$, and $G : GF(p) \rightarrow \{0,1\}^k$.
- Encryption $\mathcal{E}$
  - Compute $R = tP$ and $S = tW$ where $t = H(m||s)$, message $m \in \{0,1\}^{k_0}$, and $s \leftarrow_R \{0,1\}^{k_1}$.
  - $\mathcal{E}_{pk}(m, s) = (A, B) = (R, G(x_S) \oplus (m||s))$ where x_S is the x-coordinate of S.
- Decryption $\mathcal{D}$
  - Compute $S' = uA$ and $t' = H(B \oplus G(x_{S'}))$.
  - If $A = t'P$, output $\mathcal{D}_{sk}(A, B) = [B \oplus G(x_{S'})]^{k_0}$. Otherwise, output "null". Here, $x_{S'}$ denotes the x-coordinate of S' and $[B \oplus G(x_{S'})]^{k_0}$ denotes the first k_0 bit of $[B \oplus G(x_{S'})]$.

3 System Design

In this section, we describe overall architecture and each protocol step of cashpia-v2. We assume that bank and trustee cannot falsely conspire together to frame an honest user. It is also assumed the bank and the user trust the trustee completely.

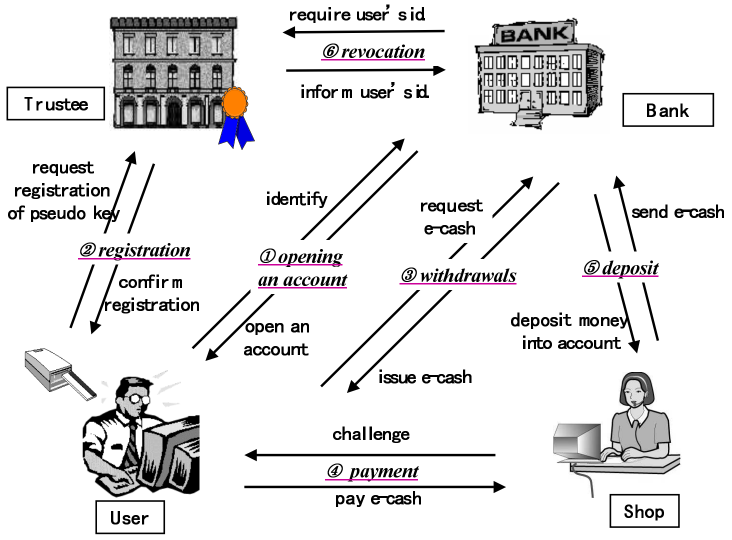

Fig. 1. Configuration of Electronic Cash System

3.1 System Architecture

Fig. 1 shows the overall architecture of cashpia-v2. In general, an e-cash system consists of three basic components, user, bank and shop. To support anonymity revocation, the trustee must participate in the e-cash system. Thus, cashpia-v2 is assumed to be composed of four participants: user, bank, shop, and trustee.

As shown in Fig. 1, cashpia-v2 performs five payment protocols which consist of opening an account, registration, withdrawal, payment, and deposit protocols. Before beginning these protocols, the initialization protocol must be executed. The anonymity revocation protocol may be performed if necessary. Each step works as follows:

- Initialization : System parameters and all participants' key pairs are generated.
- Opening an account : The bank opens user's account and registers user's private information.
- Registration : The user generates and registers a pseudonymous public key to the trustee.
- Withdrawal : The user withdraws a coin from his own account into his device (PC or a smart card).
- Payment : The user pays the withdrawn coin to the shop for the goods.
- Deposit : The shop transfers a coin to the bank and the bank deposits it to the shop's account.

- Anonymity revocation : The trustee computes a coin from the transcripts of the withdrawal phase or user's identification from the transcripts of the payment phase.

Notation :
The following notations are used to describe cashpia-v2 afterwards.

- U, S, B, T: user, shop, bank, trustee
- Z: entity, $Z \in \{U, S, B, T\}$
- c: randomly chosen 24 bit coin
- $||$: concatenation of messages
- msg: shop's challenge information at payment phase
- id_Z: Z's identification
- acc_Z: Z's account
- $Ind(j)$: index of stored key j
- $h(A||B||..)$: result of collision free hash function
- x_{Z_i}: Z's static private key, $x_{Z_i} \in_R [2, q-1]$, $i \in \{1, 2, ..\}$
- Q_{Z_i}: Z's static public key, $Q_{Z_i} = x_{Z_i} P$, $i \in \{1, 2, ..\}$
- k_{Z_i}: Z's ephemeral private key, $k_{Z_i} \in_R [2, q-1]$, $i \in \{1, 2, ..\}$
- R_{Z_i}: Z's ephemeral public key, $R_{Z_i} = k_{Z_i} P$, $i \in \{1, 2, ..\}$
- x_{ps}, Q_{ps}: U's pseudonymous private key $x_{ps} \in_R [2, q-1]. Pseudonymous public key Q_{ps} = x_{ps} P$
- $K_{U,T}, K_{U,B}$: session key $K_{U,T}$ between U and T, session key $K_{U,B}$ between U and B
- Z_{PK}, Z_{SK}: Z's public key, private key
- s_Z, s_c: s_Z is Z's signature, s_c is U's signature for a coin
- σ_Z, σ_c: $\sigma_Z = s_Z || R_{Z_i}$, $\sigma_c = s_c || R_c$
- E_K, D_K: symmetric encryption scheme and decryption scheme with a session key
- $E_{Z_{PK}}, D_{Z_{SK}}$: public key encryption scheme with a public key and decryption scheme with a private key
- S_Z, V_Z: Z's signature scheme and verification scheme

3.2 Database and Revocation Lists

Cashpia-v2 uses the database and revocation lists to store and manage the data, cope with extortion attack, and efficiently revoke the anonymity. The followings describe classification, element items, access authority and descriptions of database and revocation list in the form of classification : $\{element\ items\}$ / [`access authority`] / ; Description.

– Coin-DB : $\{c, Ind(Q_{ps}), \sigma_B\}$, [`User`]. ; Store coin and signature obtained from the bank in the withdrawal phase.
– User-DB : $\{id_Z, Name, acc_Z, Address, e-mail\}$, [`Bank`]. ; It is set up in account opening phase. Here, acc includes acc_U and acc_S.
– PsdPub-DB : $\{id_U, Q_{ps}, \sigma_U, Ind(x_{T_2})\}$, [`Trustee`]. ; Store registered user's pseudonymous public key and other transcripts of the registration phase.

- Pay-DB : $\{c, Q_{ps}, msg, \sigma_B, \sigma_T, \sigma_c\}$, [Shop]. ; Store coin information obtained from a user in the payment phase.
- PsdPrv-DB : $\{x_{ps}, Q_{ps}, R_{T_2}, \sigma_T\}$, [User]. ; Store pseudonymous key pairs and other transcripts of the registration phase.
- With-DB : $\{id_U, Ind(x_{B_1}), e', E_{T_{PK}}(m, s)\}$, [Bank]. ; Store transcripts of the withdrawal phase.
- Dep-DB : $\{c, Q_{ps}, id_S, \sigma_B, \sigma_T, \sigma_c\}$, [Bank]. ; Store coin information obtained from the shop in the deposit phase.
- User-BL : $\{Q_{ps}\}$, [Trustee, Shop]. ; List user's pseudonymous public key corresponding with pseudonymous private key extorted.
- Bank-BL : $\{Q_{B_1}\}$, [Trustee, Shop]. ; List bank's public key corresponding with private key extorted.
- Trust-BL : $\{Q_{T_2}\}$, [Trustee, Shop]. ; List trustee's public key corresponding with private key extorted.
- Coin-WL : $\{Q_{ps}, c\}$, [Trustee, Shop]. ; List withdrawn but unused legal coins signed by extorted bank's private key.
- PsdPub-WL : $\{Q_{ps}\}$, [Trustee, Shop]. ; List honest user's pseudonymous public keys signed by extorted trustee's private key.
- Customer-BL : $\{id_U\}$, [Bank]. ; List the identification of double-spending user.

3.3 Cashpia-v2

The following describes each phase protocol of cashpia-v2.

Initialization :

Each entity generates ephemeral key pairs. We assume that each certification can be easily accessible.

Opening an Account :

1) U is identified by B.
2) B opens user's account acc_U and sends it to U.

Registration :

1) U and T perform SK's protocol to share the session key $K_{U,T}$. Transmitted messages in this phase are encrypted with $K_{U,T}$.
2) U randomly generates pseudonymous private key x_{ps} in $[2, q-1]$. Corresponding pseudonymous public key Q_{ps} is computed with scalar multiplication $x_{ps}P$. U also randomly generates ephemeral private key k_{U_2} in $[2, q-1]$ and computes corresponding public key R_{U_2}. And U generates signature s_U for id_U, Q_{ps}, and R_{U_2}. U generates σ_U with concatenation of s_U and R_{U_2} and sends it to T after encrypting with $K_{U,T}$.

   Choose $x_{ps}, k_{U_2} \in_R [2, q-1]$
   $Q_{ps} = x_{ps}P$
   $R_{U_2} = k_{U_2}P$

$$s_U = x_{U_2} h(R_{U_2} || id_U || Q_{ps}) + k_{U_2} \quad (1)$$
$$\sigma_U = (R_{U_2} || s_U)$$

3) T verifies signature σ_U as the equation (2). T generates signature s_T for Q_{ps} and generates σ_T by concatenating s_T and R_{T_2} as the equation (3). T sends σ_T to U and stores id_U, Q_{ps}, σ_U, and $Ind(x_{T_2})$ in the PsdPub-DB.

$$\text{if } ((s_U P! = h(R_{U_2} || id_U || Q_{ps}) Q_{U_2} + R_{U_2})) \text{ reject} \quad (2)$$
$$\text{else} \quad \text{accept}$$

$$\text{Choose } k_{T_2} \in_R [2, q-1]$$
$$R_{T_2} = k_{T_2} P \quad (3)$$
$$s_T = x_{T_2} h(R_{T_2} || Q_{ps}) + k_{T_2}$$
$$\sigma_T = (R_{T_2} || s_T)$$

4) U verifies signature σ_T as the equation (4). Then, U stores Q_{ps}, x_{ps}, R_{T_2}, and σ_T in PsdPrv-DB.

$$D_{K_{U,T}}(E_{K_{U,T}}(\sigma_T)) = \sigma_T$$
$$\text{if } (s_T P! = h(R_{T_2} || Q_{ps}) Q_{T_2} + R_{T_2}) \text{ reject} \quad (4)$$
$$\text{else accept}$$

Withdrawal :

1) U and B perform SK's protocol and share the session key $K_{U,B}$. Transmitted messages in this phase are encrypted with $K_{U,B}$.
2) U randomly generates a coin c with 24 bits. U generates encryption $E_{T_{PK}}(m, s)$ for c and Q_{ps} using the trustee's public key Q_{T_2}. The equation (5) is the encryption result with PSLC-2. When extortion attack occurs, $E_{T_{PK}}(m, s)$ is transmitted to the trustee.

$$\text{Choose } s \leftarrow_R \{0,1\}^{k_1}$$
$$m = [h(Q_{ps} || c)]^{k_0}$$
$$t = H(m || s)$$
$$R_{U_3} = tP \quad (5)$$
$$S = tQ_{T_2}$$
$$E_{T_{PK}}(m, s) = (A, B) = (R_{U_3}, G(x_s) \oplus (m || s))$$

3) After receiving R'_{B_1} from B, U generates blind value e' for c, Q_{ps}, and R'_{B_1} using private key u and v and sends it to B. Here, R'_{B_1} is the public key corresponding with Bank's ephermeral private key k_{B_1} in $[2, q-1]$.

$$\text{Choose } u, v \in_R [2, q-1]$$
$$\text{Obtain } R'_{B_1} \text{ from the Bank}$$
$$R_{B_1} = uR'_{B_1} + vP \quad (6)$$
$$e = h(R_{B_1} || c || Q_{ps})$$
$$e' = e - u$$

4) B generates blind Schnorr signature s'_B [22] for e' and sends s'_B to U.

$$\text{Choose } k_{B_1} \in_R [2, q-1] \quad (7)$$
$$s'_B = x_{B_1} e' + k_{B_1}$$

5) U computes B's signature s_B for s'_B using u and v. U generates σ_B by concatenating s_B and R_{B_1} as the equation (8). U verifies σ_B as the equation (9) and stores σ_B, c, and $Ind(Q_{ps})$ in Coin-DB.

$$s_B = s'_B u + v \tag{8}$$
$$\sigma_B = (R_{B_1} || s_B)$$

$$\text{if } ((s_B P == h(r_{B_1}||c||Q_{ps})Q_{B_1} + R_{B_1})) \tag{9}$$
$$\text{then accept}$$

6) B stores $id_U, Ind(x_{B_1}), e'$, and $E_{T_{PK}}(m, s)$ in With-DB. And B withdraws the coin value from acc_U of User-DB.

Payment :

1) S sends challenge msg to U as the equation (10).

$$msg = h(id_S||time) \tag{10}$$

2) U generates signature σ_c as the equation (11). U sends $c, Q_{ps}, \sigma_T, \sigma_B$, and σ_c to S. At this time, to prevent framing attack for the shop S, σ_c may be encrypted with public key of the shop S.

$$\text{Choose } k_{U_4} \in_R [2, q-1]$$
$$R_{U_4} = k_{U_4} P \tag{11}$$
$$s_c = x_{ps} h(R_{U_4}||c||id_S||msg||Q_{ps}) + k_{U_4}$$
$$\sigma_c = (R_{U_4}||s_c)$$

3) S verifies signatures σ_T, σ_B, and σ_c as the equation (12) and stores transcripts in Pay-DB. However, if extortion attack was reported, S must examine black/white lists. If U's private key x_{ps} was extorted, S must check that corresponding public key Q_{ps} is in User-BL. If B's x_{B_1} was extorted, S must check that Q_{B_1} is in Bank-BL and c signed by x_{B_1} in Coin-WL. If T's x_{T_2} was extorted, S must check that Q_{T_2} is in Trust-BL and Q_{ps} signed by x_{T_2} in PsdPub-WL. After checking, S stores transcripts $c, Q_{ps}, msg, \sigma_B, \sigma_T$, and σ_c in Pay-DB.

$$\text{Obtain } c, Q_{ps}, \sigma_B, \sigma_T, \sigma_c \text{ from the user}$$
$$\text{if } ((s_T P == h(R_{T_2}||Q_{ps})Q_{T_2} + R_{T_2}) \&\&$$
$$(s_B P == h(R_{B_1}||c||Q_{ps})Q_{B_1} + R_{B_1}) \&\& \tag{12}$$
$$(s_c P == h(R_{U_4}||c||id_S||msg||Q_{ps})Q_{ps} + R_{U_4}))$$
$$\text{then accept}$$

Deposit :

1) S sends c, Q_{ps}, σ_B, and σ_T obtained from U to B. B verifies the signatures as the equation (12).
2) After verification, B checks if c and Q_{ps} obtained from S already exist in Dep-DB. If the values exist, B finds σ'_c for the deposited coin in Dep-DB and sends it to S (detection of double-deposit or double-spending).
3) If S receives σ'_c from B, S checks if it equals with sending value σ_c. If equals, S rejects performing protocol (double-deposit). Otherwise, S sends σ_c, id_S, acc_S, and msg to B (double-spending). If S does not receive any signature from B, S sends σ_c, id_S, acc_S, and msg to B.

4) If B receives σ_c, id_S, acc_S, and msg from S, B verifies signature σ_c. After verification, B deposits the value of e-cash into the account of shop S. B stores $c, Q_{ps}, id_S, \sigma_T, \sigma_c$, and σ_B in Dep-DB.
5) If the same coin was already deposited, this is called double spending. B performs user tracing protocol with user's Q_{ps} and c and detects double spending user.

User Tracing :

1) B cooperates with T to detect double spender's identification. B finds double-spent information $(c, Q_{ps}, \sigma_B, \sigma_T, \sigma_c)$ and $(c, Q_{ps}, \sigma_B, \sigma_T, \sigma_c')$ from Dep-DB and sends them all to T.
2) T verifies all signatures as the equation (12). T detects double spender's Q_{ps}, id_U, and σ_U and sends them to B.
3) B appends user's identification in Customer-BL.

Extortion Tracing :

The process of this phase is similar to that of PePo97. If the extortion of the secret key occurs, it is reported to the trustee T. T records corresponding public key in the revocation list as mentioned in **Database and Revocation Lists** above. T distributes the modified lists to all shops. The shops use the lists to check legal coin in the payment phase.

Since key agreement protocol is performed after identifying U and B, transparent blindfolding coins under B's secret key x_{B_1} is impossible. And transparent blindfolding coins under trustee's secret key x_{T_2} is impossible since unforgeable signature scheme is used.

4 System Implementation

4.1 Tools for the Implementation

We use ICUCLIB-v2 which provides cryptographic algorithm to cashpia-v2. The algorithms are SEED [27], SHA-1, ElGamal encryption scheme [15], scalar multiplication on an elliptic curve, addition of points for optimal normal basis and doubling of Schroeppel's algorithm [23] for optimal normal basis, and multiplication and reverse operation over the finite field [26].

We can choose a smart card or a personal computer as a storage device for e-cash. The smart card has limitation on the memory capacity of data and computing speed. Therefore, the cryptographic algorithm over the finite field limits the usage of the smart card because the length of the key is very long and the computational speed is slow. The elliptic curve cryptographic algorithm, however, can efficiently save data in a limited storage and accelerate the computational speed by the shortened ciphertext. To use the smart card as the storage device of the e-cash, cashpia-v2 overcomes the limitation w.r.t. the storage capacity as implementing the system over the elliptic curve. We store coin information in the smart card by the note-based type.

4.2 Implementation Overview

We choose the elliptic curve over $GF(2^{131})$ as $y^2 + xy = x^3 + 1$ and its order as $4 * 680564733841872926932320129493409985129$ [15] using C++ on Windows NT 4.0 and Windows 95/98. Fig. 2 shows the graphical user interface of the system. Access DB is used for the database management and `cashpia.mdb` for manipulating tables on servers and clients. The system is assumed to be working on the client/sever architecture. A client contacts a server in advance to make connections. The client environment is composed of the IP address and ports configuration of the server and the login process of inputting clients' ID and password. We assume that all the intending clients are registered in the server. In the server environment, all the transactions with clients are monitored through a log-in screen. Participants' certificates are distributed to one another in advance. Now, we describe the data structure and the functions used to process the protocols at each phase.

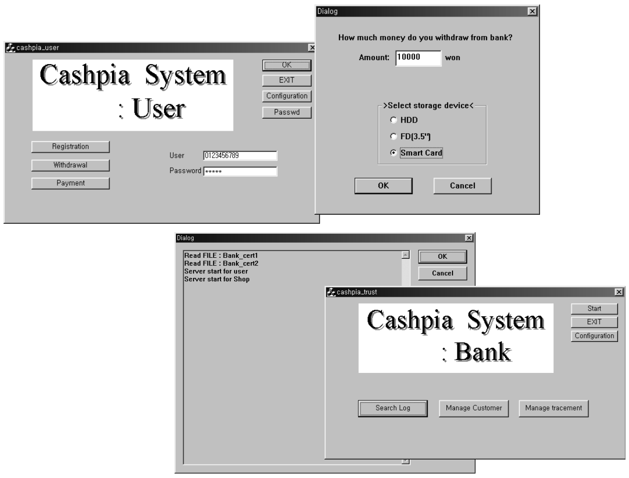

Fig. 2. Example of the User Interface (Withdrawal Phase)

Data Structure :

`Structure Name` { MEMBER VARIABLE TYPE *member variable name* // description }

`ECI_CURVE` {	INDEX	*form*	// type of the elliptic curve
	GF2N	*pnt_order*	// order of the base point
	GF2N	*cofactor*	// cofactor = $\#E/pnt\_order$
	GF2N	a_2	// $y^2 + xy = x^3 + a_2 * x^2 + a_6$
	GF2N	a_6	// a_2, a_6 are constants over the elliptic curve }

`ECI_POINT` {	GF2N	x	// x coordinate value of the point
	GF2N	y	// y coordinate value of the point}
`EC_PARAMETER` {	CURVE	crv	// elliptic curve
	POINT	pnt	// base point of the elliptic curve }
`EC_KEYPAIR` {	GF2N	$prvt\_key$	// private key
	POINT	$pblc\_key$	// public key }

Opening an Account :
The information of transaction-enabled users and shops, such as name, address, phone number, etc. has set to the database of the bank. And the bank has issued their account numbers.

Registration :
In the registration phase, the application of `DiffieHellman_KeyAgreement` function of the trustee and `DH_Key_Exchange_step1`, `DH_Key_Exchange_step2`, and `DH_Key_Exchange _step3` functions of the users produce the session key. The user generates a pseudonymous public key by executing the function `PsdKey_Creat_Reg_Step1` and `PsdKey_Creat_Reg_Step2`. The user encrypts it using the session key and then passes it to the trustee. The trustee generates a signature for the user's pseudonymous public key and passes it to the user in `Registering` function. And the trustee's `Storing` function stores the transcripts of the registration phase in `PsdPub-tbl` table. The user calls the `CODBCset` function and stores the transcripts in `PsdPrv-tbl` table.

Withdrawal :
The user produces a coin as 24 bit random number by using the `Generate_Regist_Coin _Step1` function, configures first 8 bit as the denomination of the coin, and blinds the coin through the `Generate_Regist_Coin_Step2` function. The bank signs on the blind coin by using `WithdrawalBlindSign` function and sends it to the user. The user who have received the signature unblinds it through the `Generate_Regist_Coin_Step3` function. The transcripts of this phase are stored in each `WithBank-tbl` table and `Coin-tbl` table. In case the smart card is used in the system, the coin information to be stored in the `Coin-tbl` table is saved in the card.

Payment :
`ReceiveData` functions of the shop and the user are the main functions of this phase. When the user is connected, the shop produces challenge information through the `Make_Mess` function and sends it to the user. The user generates signature after receiving challenge information through `ReceiveData` function, and sends the signature to the shop with the coin information withdrawn from `PsdPrv-tbl` table using `CuserPsdPrvDB` function. The shop verifies the signature received from the `ReceiveData` function. After then, the shop reads table `TrustBL`, `UserBL`, and `BankBL` from `CTrustBL`, `CUserBL`, and `CBankBL` functions respectively. If the public key for each DB is not found and the coin and the

user's pseudonymous public key is recorded in `CoinWL` table and `PsdPubWL` table respectively, the shop stores all transcripts received from `CPay` function in `PayShop-tbl` table.

Deposit :
The shop reads the currency information received from the `Pay-tbl` table of `Pay-tbl` function in the payment phase and sends the data to the bank. The bank verifies the signature by analyzing the data received from `ReceiveData` function. Then, the bank calls `CDepBank_DB` function and checks if it is already recorded in the `DepBank-tbl` table. If there exists the coin with the same ID, the bank must extract the user's signature for the coin. If the signature to be sent by the shop is not equal to the signature received from the bank or if the shop does not receive the signature from the bank, the shop passes the rest of the coin information to the bank through the `ReceiveData` function. If the signatures have the same value, the shop must stop the protocol since it is the double-deposit. If the same coin already exists in DB but the signatures are not identical, the bank verifies all the signatures in the `ReceiveData` function and calls the revocation protocol since it is the double-spending.

Anonymity Revocation :
This phase traces the double-spending of a user's identification. The deposit step calls for this phase. The bank searches double-spending information in `DepBank-tbl` table of the `CReadDepBkTbl` function and passes the information to the trustee. The trustee verifies the received signatures in the `SigVerify` function. After then, the trustee detects the user's identification accessing to `PsdPub-tbl` table of the `SearchDB` function and passes the identification to the bank.

5 Security

Cashpia-v2 satisfies the security requirements of unforgeability, double-spending prevention, anonymity, untraceability, refundability, fairness, framing-freeness, coin-tracing, and blindfolding freeness as follows :

- Cashpia-v2 supports the security against forgery of a coin by using the provably computational secure elliptic curve blind signature scheme (S_B, V_B) and collision free hash function. It is because our blind signature is secure against existential forgery. This allows only bank to generate the signature for a coin. As the hash function has the feature of collision free, the user cannot forge the coin.
- Cashpia-v2 supports the security against tracing an honest user by the bank by using provably computational secure elliptic curve blind signature scheme (S_B, V_B) and strong probabilistic encryption scheme $(E_{T_{PK}}, V_{T_{SK}})$. It is because blind signature cannot give any information for the coin and the bank cannot link the blind coin with an encrypted coin by the public key encryption scheme.

- Cashpia-v2 supports the security against forgery attack, framing attack by a bank and tracing an honest user with the trustee's help by using provably computational secure elliptic curve signature scheme (S_c, V_c) by the user. It is because anyone (even the bank or the trustee) who doesn't know user's pseudonymous secret key cannot generate the signature for the coin.
- Cashpia-v2 supports the security against framing for an honest user by the trustee by using provably computational secure elliptic curve signature scheme (S_U, V_U) by the user. It is because anyone except the authorized user who generates valid signature is computationally infeasible.
- It is possible to trace a user, thus it is secure against money laundering and blindfolding by the trustee by using provably computational secure elliptic curve signature scheme (S_T, V_T) by the trustee. It is because only trustee who signs on a pseudonymous public key know the relation between the user's identification and public key.
- Cashpia-v2 is secure against eavesdropping of coins, pseudonymous keys and framing for a shop by the bank because key agreement protocol satisfies K-KS, FS, K-CI, UK-S and breaking symmetric encryption scheme (E_K, D_K) without the knowledge of the session key K is impossible. It is because our session key obtained from reliable key agreement protocol is reliable and the communication in the registration and withdrawal phase is secure under the session key to be strong against eavesdropping and man-in-the-middle attack. The bank cannot know the signature and frame a shop because the user passes the signature for a coin encrypted by a shop's public key to the shop.
- Assuming that all revocation lists are properly used in the system, the system is said to be secure against coin-extortion attack and secret key-extortion attack. Therefore, an honest user cannot lose any unused coin.

Cashpia-v2 increases an overall efficiency compared with PePo97 in terms of the size of the message and the storage space. Assuming a prime modulus p and q in PePo97 to be 1,024 bit and 160 bit respectively, we compare PePo97 with cashpia-v2 which has a point P of 160 bit and q of 160 bit. A public key transmitted to generate the session key, e.g. R_{U_1} of the registration phase, is 1,024 bit in PePo97, while 160 bit in cashpia-v2. And the messages of payment phase $c, Q_{ps}, \sigma_T, \sigma_B$, and σ_c is 4,600 bit in PePo97 and 1,144 bit in cashpia-v2. Therefore, cashpia-v2 has 76% to 85% reduction in the message size. For p of 512 bit and q of 160 bit, PePo97 has about 52% reduction in the message size comparing internet payment with electronic purse payment and cashpia-v2 has about 44% to 68% reduction comparing with internet payment of PePo97.

Table 1 shows the these comparison for p of 1,024 bit and q of 160 bit of PePo97.

message	PePo97 (A)	cashpia-v2 (B)	improvement(A/B)
R_{U_1}	1,024 bit	160 bit	6.4 (85%)
$c\|\|Q_{ps}\|\|\sigma_B\|\|\sigma_T\|\|\sigma_c$	4,600 bit	1,144 bit	4.0 (76%)

Table 1. Comparison of the Message Size

6 Conclusion

We have designed and implemented cashpia-v2, the e-cash system whose security is based on ECDLP targeting small-amount-of-money transaction and an off-line internet payment system. Cashpia-v2 has the revocability to be secure against the attacks such as blackmailing, money laundering, and double-spending. It is also secure against the attacks such as secret key extortion, coin extortion, forgery, framing, and blindfolding. We have constructed it considering the environment of the limited storage hardware such as a smart card. To achieve our goal, we employed PePo97 and improved it to be more efficiently w.r.t. the message size. The efficiency has been enhanced by 6.4 times w.r.t. the message size compared with PePo97. Owing to the shortened message size, cashpia-v2 is easy to handle and manage a great deal of database even on the smart card environment.

However, cashpia-v2 doesn't satisfy divisibility and transferability which are required to build versatile e-cash system. It also needs to study high speed operation over the elliptic curve and to apply it to the real implementation.

References

1. J. Baek, B. Lee, and K. Kim, "Secure Length-saving ElGamal Encryption under the Computational Diffie-Hellman Assumption", To appear *ETRI J.* Vol.22, No.12, Dec. 2000.
2. S. Brands, "Untraceable Off-line Cash in Wallets with Observers", *In Advances in Cryptology-Proc. of CRYPTO'93*, LNCS 773, Springer-Verlag, pp.302-318, Aug. 1994.
3. E. Brickell, P. Gemmell and D. Kravitz, "Trustee-based Tracing Extensions to Anonymous Cash and the Making of Anonymous Exchange", *Proc. of 6th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp.457-466, 1995.
4. J. Camenisch, U. Maurer and M. Stadler, "Digital payment Systems with Passive Anonymity-Revoking Trustees", *Proc. of ESORICS'96*, LNCS 1146, Springer-Verlag, pp.31-43, 1996.
5. J. Camenisch, J. M. Piveteau and M.Stadler, "An efficient Fair Payment System", *Proc. of 3rd ACM Conference on Computer and Communications Security*, ACM Press, pp.88-94, 1996.
6. D. Chaum, "Privacy Protected Payments: Unconditional Payer and/or Payee Anonymity", *Smart Card 2000: The future of IC Cards*, North-Holland, pp.69-92, 1989.
7. D. Chaum, A. Fiat and M. Noar, "Untraceable Electronic Cash", *In Advances in Cryptology-Proc. of CRYPTO'88*, LNCS 403, Springer-Verlag, pp.319-327,
8. D. Chaum, "Blind Signatures for Untraceable Payments", *In Advances in Cryptology-Proc. of CRYPTO'82*, Plenum Press, pp.199-203, 1983.
9. Y. Frankel, Y. Tsiounis and M. Yung, ""Indrect discourse Proofs": Achieving Efficient Fair Off-Line E-Cash", *In Advances in Cryptology-Proc. of ASIACRYPT'96*, LNCS 1163, Springer-Verlag, pp.286-300, Nov. 1996.
10. E. Fujisaki and T. Okamoto, "Practical Escrow Cash System", *Proc. of 1996 Cambridge Workshop on Security Protocols*, LNCS 1189, Springer-Verlag, pp.33-48, 1997.

11. ICU, "A Study on Information Security and Authentication Technique in Distributed Environments", Cryptology & Information Security Laboratory, Information and Communications University, Sep. 1999.
12. M. Jakobsson and M. Yung, "Revokable and Versatile E-money", *Proc. of 3rd annual ACM Conference on Computer and Communication Security*, pp.76-87, March 1996.
13. H. Kim, M. Seo, J. Baek, and K. Kim, "Requirements and Comparison of the Existing Electronic Cash Protocols", *Proc. of WISC99*, 1999.
14. L. Law, A. Menezes, M. Qu, J. Solinas and S. Vanstone, "An Efficient protocol for Authenticated Key Agreement Protocol", *Technical Report CORR 98-5*, University of Waterloo, Canada, March 1998.
15. R. Michael, "Implementing Elliptic Curve Cryptography", Manning, 1998.
16. D. M'Raihi, "Cost Effective Payment Schemes with Privacy Regulations", *In Advances in Cryptology-Proc. of ASIACRYPT'96*, LNCS 1163, Springer-Verlag, pp.266-275, Nov. 1996.
17. T. Okamoto and K. Ohta, "Universal Electronic Cash", *In Advances in Cryptology-Proc. of CRYPTO'91*, LNCS 576, Springer-Verlag, pp.324-337, Aug. 1991.
18. T. Okamoto, "An Efficient Divisible Electronic Cash Scheme", *In Advances in Cryptology-Proc. of CRYPTO'95*, LNCS 963, Springer-Verlag, pp.438-451, Aug. 1995.
19. H. Petersen and G. Poupard, "Efficient Scalable Fair Cash with Off-line Extortion Prevention", (Technical Report, ENS, 33 pages, 1997), short version *Proc. of Int. Conference on Information and Communications Security (ICICS'97)*, LNCS 1334, Springer-Verlag, pp.463-477, Nov. 1997. Aug. 1989.
20. D. Pointcheval, "Chosen-Ciphertext Security for any One-Way Cryptosystem", In Proc. of PKC2000, LNCS 1751, pp.223-238, Springer-Verlag, 2000.
21. B. Song and K. Kim, "Two-Pass Authenticated Key Agreement Protocol with Key Confirmation", *To appear in Proc. of INDOCRYPT2000*, Dec. 2000.
22. C. P. Schnorr, "Efficient Identification and Signatures for Smart Cards", *In Advances in Cryptology-Proc. of CRYPTO'89*, LNCS 435, Springer-Verlag, pp.239-251, 1990.
23. R. Schroeppel, H. Orman, S. O'Malley and O. Spatscheck, "Fast Key Exchange with Elliptic Curve System", *In Advances in Cryptology Proc. of CRYPTO'96*, LNCS 963, Springer-Verlag, pp.43-56, 1995.
24. M. Stadler, J. M. Piveteau and J. Camenisch, "Fair-Blind Signatures", *In Advances in Cryptology-Proc. of EUROCRYPT'95*, LNCS 921, Springer-Verlag, pp.209-219, 1995.
25. S. von Solms and D. Naccache, "On Blind Signatures and Perfect Crimes", *Computers and Security*, pp.581-583, Nov. 1992.
26. IEEE P1363 Draft Version 9. "Standard Specifications For Public Key Cryptography", 1999, `http://grouper.ieee.org/groups/1363/`
27. Standard Specifications (the latest draft : June 29, 1999) : 128-bit symmetric block cipher SEED (in Korean)