Design and Implementation of an Efficient Fair Off-line E-Cash System based on Elliptic Curve Discrete Logarithm Problem

Manho Lee, Gookwhan Ahn, Jinho Kim, Jaegwan Park,
Byoungcheon Lee, Kwangjo Kim, and Hyuckjae Lee

Abstract: **In this paper, we design and implement an efficient fair off-line electronic cash system based on Elliptic Curve Discrete Logarithm Problem (ECDLP), in which the anonymity of coins is revocable by a trustee in case of dispute. To achieve this, we employ the Petersen and Poupard's electronic cash system [1] and extend it by using an elliptic curve over the finite field $GF(2^n)$. This naturally reduces message size by 85% compared with the original scheme and makes a smart card to store coins easily. Furthermore, we use the Baek *et al.*'s provably secure public key encryption scheme [2] to improve the security of electronic cash system. As an extension, we propose a method to add atomicity into new electronic cash system. To the best of our knowledge, this is the first result to implement a fair off-line electronic cash system based on ECDLP with provable security.**

Index Terms: **Electronic cash, anonymity revocation, atomicity, elliptic curve discrete logarithm problem.**

I. INTRODUCTION

Recently, a variety of on-line businesses are rapidly emerging over the Internet, which is believed to be one of the most efficient and convenient ways to provide all electronic services. An efficient and secure electronic cash ("e-cash" in short) system plays an important role to support these businesses safely as a trustful payment over the Internet. Real money (or paper money) using traditional means of payment has potential security problems such as counterfeiting and forgeability. E-cash also exhibits similar drawbacks, but properly-designed e-cash system can provide more secure and flexible service for non face-to-face exchange of digital goods than real money.

After Chaum [3] introduced the anonymity of an e-cash using blind signature, numerous researches have been done in the field of e-cash system. Whether the bank is required to be on-line or not in the processing of an electronic transaction, Chaum [4] suggested an anonymous on-line e-cash system and Chaum *et al.* [5] proposed an anonymous off-line e-cash system, which satisfies double-spending prevention with the cut-and-choose method. The e-cash systems [6], [7] by Okamoto and Ohta satisfy the divisibility and transferability in addition. Their schemes overcome some limitations of previous e-cash systems and provide more efficient features than real money. Brands [8] proposed an efficient e-cash system with single-term method which is more efficient compared with the cut-and-choose method. Brands' scheme has been used as a basic model by other researchers.

However, Solms and Naccache [9] raised the issue of perfect crime by abusing the anonymity of the e-cash system. Other malicious actions such as blackmailing and money laundering were also presented in [1], [9], [10], and [11]. The revocable e-cash system [10], [12] (or fair payment system) in which anonymity can be revoked when needed, becomes one of the active research areas of preventing such misuses. In the revocable e-cash scheme, the identification of an illegal user can be traced by the cooperation of a trustee and a bank.

Along with countermeasures [10], [13] against the blackmailing and money laundering, many schemes in [1], [11], [12], [14], and [15] have been proposed to resist against the abuse of anonymity. Two schemes suggested by Camenish *et al.* [16] and Frankel *et al.* [17] require the trustee to take part in the initialization phase but do not provide a prevention against extortion and blindfolding attacks. Some schemes were suggested to prevent these attacks. Among them, Fujisaki and Okamoto's scheme [14] and Jakobsson and Moti's scheme [11] are said to be not efficient in a sense that users need to communicate with a trustee in every payment phase. On the other hand, Petersen and Poupard's scheme [1] (called as "PePo97") is considered to be an efficient off-line electronic payment system in which users need to communicate with the trustee in the registration phase only when registering a pseudonymous public key. This scheme is found to be secure against many kinds of attacks such as secret key extortion, framing against the user, trustee's blindfolding, blackmailing, and money laundering.

In addition to these security requirements, atomicity is also of another importance in serving a complete payment system. If the network crashes in the middle of the withdrawal or payment protocol, some disputes such as losing e-cash, repudiation, and no procurement of goods even after a full payment will happen. Under a network model where customers purchase electronic goods and receive its service, Tygar [18] introduced the problem of atomicity in electronic transactions and defined three classes of atomicity; money atomic, goods atomic, and certified delivery. Camp *et al.* [19] proposed a method to support atomicity using on-line TTP(Trusted Third Party). Moreover, Sirbu and Tygar [20] and Bellare *et al.* [21] suggested on-line account-

Manuscript received July 20, 2001: approved for publication by Cunsheng Ding, Division I Editor, December 6, 2001.

M. Lee is with KFTC, e-mail: manho@kftc.or.kr.
G. Ahn is with SECUi.COM, e-mail: tachyon@icu.ac.kr.
J. Kim is with KISA, e-mail: jhk026@kisa.or.kr.
J. Park is with LG CNS, e-mail: jgpark@icu.ac.kr.
B. Lee is Joongbu Univ. e-mail: sultan@joongbu.ac.kr.
K. Kim and H. Lee are with ICU e-mail: kkj@icu.ac.kr, hjlee@icu.ac.kr.
This work was done all the authors are with ICU.

based atomicity and token-based atomicity, respectively. Xu *et al.* [22] addressed money conservation via atomicity in fair off-line e-cash.

A. Security Requirements

In general, the security requirements of e-cash system can be classified into two parts [1], [11]. Hereafter, we use "coin" and "e-cash" interchangeably for the sake of simplicity.

Basic Requirements

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

Additional Requirements

- Revocability: Any coin which is legally obtained can be revealed.
- User-tracing: A bank can legally trace the user of a paid coin with trustee's help.
- Extortion-tracing: A bank can legally trace with the matching information of a paid or deposited coin with trustee's help.
- Blindfolded-freeness: No one can obtain a blinded coin without bank's knowledge that this particular coin has been blinded.
- Coin-tracing: A bank can legally link the information of a coin to be used with the information of deposited coin with trustee's help.
- Unlinkability: It is not possible to find any 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 with respect to the bank and the trustee must be guaranteed, but 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 identification of over-spending user.
- Transferability: Once withdrawn coins can be transferred to any other user.
- Divisibility: The denomination of a coin can be divided into lower unit.

B. Petersen and Poupard's Scheme

PePo97 which consists of initialization, account opening, registration, withdrawal, payment, deposit, and revocation protocols, 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 satisfied if existential forgery of a signature with respect to an adaptively chosen message attack is equivalent to a known computational hard problem (*e.g.*, factorization or discrete log.). Off-line property in PePo97 can be realized by the registration of user's pseudonymous public key to the trustee. Moreover, the prevention against extortion attack can be implemented by using the secure revocation lists and database. PePo97 has two types of protocols, i.e., the Internet payment and electronic purse protocols, which are quite different in terms of security and efficiency.

C. Our Goals and Approach

Our work aims at implementing an efficient Internet payment protocol with a smart card. This new e-cash system based on ECDLP (called as "Cashpia-v2") has targeted a small-amount-of-money[1] electronic transaction. The anonymity of transaction is of prime concern because it deals with a small-amount-of-money transaction. In general, when an honest user deals with a large-amount-of-money transaction(*e.g.*, purchasing a house, a car, *etc.*), it is usually not allowed to make a payment anonymously due to lawful taxation and registration. On the other hand, in a small-amount-of-money transaction (*e.g.*, purchasing private goods, *etc.*), there are many situations in which we want the purchasing details to be confidential. The tradeoff must be considered to design Cashpia-v2 because the complete anonymity will induce crimes when abused. Thus, anonymity must be guaranteed, but it can be revoked under a lawful order. In addition, Cashpia-v2 is also designed to provide many preventive functions against money laundering, blackmailing, and double spending. However Cashpia-v2 doesn't provide divisibility, transferability and overspent-tracing because these requirements make all protocols hard to implement. To achieve these goals, we choose PePo97 as an underlying frame since PePo97 is found to be a strong off-line e-cash scheme resistant against various attacks with good efficiency, and also has most features which we try to achieve. But PePo97 requires high cost in storage and communication due to processing lots of databases. Thus it is hard to use PePo97 in the storage-limited platform such as smart card. To overcome this problem we extend PePo97 into ECDLP-based scheme.

In order to enforce the high efficiency and provable security together, Cashpia-v2 employs Baek *et al.*'s public key encryption scheme [2]. This scheme called as PSLC-2 (Provably Secure Length-saving public-key encryption scheme based on Computational Diffie-Hellman Assumption) is optimally designed on the elliptic curve over the finite field and is length-efficient in a sense that the length of ciphertext is shorter than that of Pointcheval's scheme [23]. Under the CDH-A (Computational Diffie-Hellman Assumption), the security of PSLC-2 is suggested to be provably secure against an adaptive chosen ciphertext attack.

We implement Cashpia-v2 using the cryptographic library ICUCLIB-v2 (ICU Cryptographic LIBrary-version 2) developed and maintained by ICU. This library provides various cryptographic primitives to implement Cashpia-v2.

[1] It is very hard to define how much the small-amount-of-money absolutely is. It may depend on the local customs.

Organization:

In Section II, we overview cryptographic primitives such as PSLC-2 in brief. In Sections III and IV, we describe the details of functional design and the implementation of Cashpia-v2, respectively. In Section V, we suggest an intuitive method to provide atomicity in Cashpia-v2. In Section VI, we analyze Cashpia-v2 in terms of its security and efficiency. Finally, we summarize our results and suggest future work in Section VII.

II. CRYPTOGRAPHIC PRIMITIVES

A key exchange protocol is used for two entities to share a session key secretly. The shared session key is used to exchange authenticated messages between them. In Cashpia-v2 all transmitted messages are encrypted with the shared session key by using Diffie-Hellman key agreement protocol [24].

We also adapt PSLC-2 to encrypt coin information with trustee's public key. This operation is required to protect a withdrawn coin signed by the bank when the key is extorted. The following is a brief summary of PSLC-2 working on the elliptic curve over the finite field.

- Key generator $\mathcal{K}$
  - Choose a non-supersingular elliptic curve $E(GF(p))$ defined over Galois field $GF(p)$ and calculate the order of elliptic curve $\#E(GF(p))$. Let q be a large prime number dividing $\#E(GF(p))$ and let P be a point of order q on $E(GF(p))$.
  - For a given random number $u \in_R GF(q)$, compute $W = uP$. Then the public key is $pk = (E, P, q, W)$ and the corresponding private key is $sk = (E, P, q, u)$.
- Hash Function (two random oracles)
  - Let $k = k_0 + k_1 = |p|$ be a security parameter.
  - 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 bits of $[B \oplus G(x_{S'})]$.

Theorem 1: *PSLC-2 is a provably secure public key encryption scheme.*

Proof: See [2]. □

III. SYSTEM DESIGN

In this section, we describe the overall system architecture and each protocol step of Cashpia-v2 assuming that the bank and trustee may not falsely conspire together to frame an honest user, and the trustee is to be highly trustful.

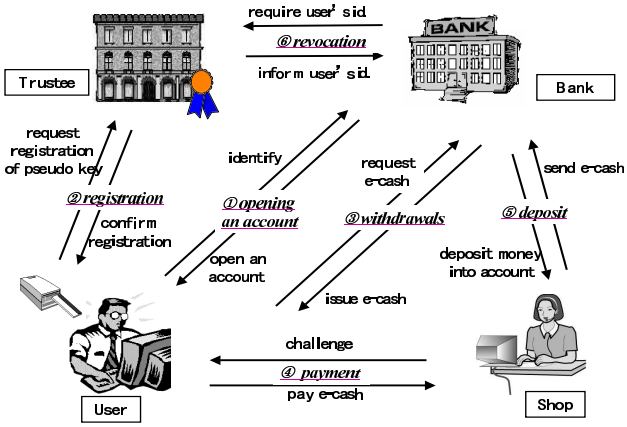

Fig. 1. Overall configuration of Cashpia-v2.

A. System Architecture

Fig. 1 shows the overall configuration of Cashpia-v2. In general, an e-cash system consists of three basic entities: user, bank, and shop. To support anonymity revocation, the trustee must participate in the e-cash system. Thus, Cashpia-v2 consists of four entities: user, bank, shop, and trustee. Bank can play the role of trustee to make Cashpia-v2 simple.

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 only when it is required. The major functions of each step are 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 in return for goods.
- Deposit: The shop transfers a coin to the bank and the bank deposits the corresponding coin to the shop's account.
- Anonymity revocation: The trustee reveals the owner of a coin from the transcripts of the withdrawal phase or user's identification from the transcripts of the payment phase.

Notation:

Throughout this paper, we use the following notation to describe Cashpia-v2.

- U, S, B, T: Identity of user, shop, bank, and trustee, respectively
- Z: an entity, $Z \in \{U, S, B, T\}$
- c: coin (randomly chosen 24-bit value)

- $||$: 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 h.
- 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 $x_{ps} \in_R [2, q-1]$ and pseudonymous public key $Q_{ps} = x_{ps}P$
- $K_{U,T}, K_{U,B}$: session key $K_{U,T}$ between U and T, and $K_{U,B}$ between U and B
- Z_{PK}, Z_{SK}: Z's public key and private key, respectively
- s_Z, s_c: Z's signature and U's signature for a coin
- σ_Z, σ_c: $\sigma_Z = s_Z||R_{Z_i}$ and $\sigma_c = s_c||R_c$
- 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 public key decryption scheme with a private key
- S_Z, V_Z: Z's signature scheme and verification scheme

B. Databases and Revocation Lists

Cashpia-v2 uses databases and revocation lists to store and manage the data, to cope with extortion attack, and to revoke the anonymity efficiently. In the following, classification, element items, access authority, and descriptions of databases and revocation lists are explained 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"];` Store user-related information. 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 of user's pseudonymous public key corresponding to the extorted pseudonymous private key.
- Bank-BL: $\{Q_{B_1}\}$, [`Trustee, Shop`]; List of bank's public key corresponding to the extorted private key.
- Trust-BL: $\{Q_{T_2}\}$, [`Trustee, Shop`]; List of trustee's public key corresponding to the extorted private key.
- Coin-WL: $\{Q_{ps}, c\}$, [`Trustee, Shop`]; List of withdrawn but unused legal coins signed by bank's extorted private key.
- PsdPub-WL: $\{Q_{ps}\}$, [`Trustee, Shop`]; List of honest user's pseudonymous public keys signed by trustee's extorted private key.
- Customer-BL: $\{id_U\}$, [`Bank`]; List of the identifications of double-spending users.

C. Sub-protocols

Initialization:

1. Each entity generates ephemeral key pairs under the assumption that each entity has its certificate.

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 Diffie-Hellman key agreement protocol to share a session key $K_{U,T}$. All transmitted messages in this phase are encrypted with $K_{U,T}$.
2. U randomly generates a pseudonymous private key x_{ps} in $[2, q-1]$. Corresponding pseudonymous public key Q_{ps} is obtained with scalar multiplication $x_{ps}P$. U also randomly generates an ephemeral private key k_{U_2} in $[2, q-1]$ and computes its 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 as concatenation of s_U and R_{U_2}, encrypts it with $K_{U,T}$ and sends it to T.
   Choose $x_{ps}, k_{U_2} \in_R [2, q-1]$
   $Q_{ps} = x_{ps}P$
   $R_{U_2} = k_{U_2}P$
   $s_U = x_{U_2}h(R_{U_2}||id_U||Q_{ps}) + k_{U_2}$
   $\sigma_U = (R_{U_2}||s_U)$ (1)
3. T verifies signature σ_U as shown in (2). T generates signature s_T for Q_{ps} and generates σ_T by concatenating s_T and R_{T_2} as shown in (3). T sends σ_T to U and stores id_U, Q_{ps}, σ_U, and $Ind(x_{T_2})$ in PsdPub-DB.
   if $(s_U P == h(R_{U_2}||id_U||Q_{ps})Q_{U_2} + R_{U_2})$, then accept. (2)
   Choose $k_{T_2} \in_R [2, q-1]$
   $R_{T_2} = k_{T_2}P$
   $s_T = x_{T_2}h(R_{T_2}||Q_{ps}) + k_{T_2}$
   $\sigma_T = (R_{T_2}||s_T)$ (3)
4. U verifies signature σ_T as (4). Then, U stores Q_{ps}, x_{ps}, R_{T_2}, and σ_T in PsdPrv-DB.
   if $(s_T P == h(R_{T_2}||Q_{ps})Q_{T_2} + R_{T_2})$, then accept. (4)

Withdrawal:

1. U and B perform Diffie-Hellman key agreement protocol and share the session key $K_{U,B}$. All 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}. (5) is the outcome of its encryption with PSLC-2. When extortion attack occurs, $E_{T_{PK}}(m,s)$ is transmitted to the trustee. U asks a withdrawal of coin to B.

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

3. B chooses a random number $k_{B_1} \in_R [2, q-1]$, computes $R'_{B_1} = k_{B_1}P$, and sends R'_{B_1} to U.
4. After receiving R'_{B_1} from B, U computes blinded value e' for c, Q_{ps} and R_{B_1} using secret random values u and v, and sends e' to B.

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

5. B generates blind Schnorr signature s'_B [31] for e' and sends s'_B to U.

$s'_B = x_{B_1}e' + k_{B_1}$ (7)

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

$s_B = s'_B u + v$
$\sigma_B = (R_{B_1}||s_B)$ (8)
if $(s_B P == h(R_{B_1}||c||Q_{ps})Q_{B_1} + R_{B_1})$, then accept. (9)

7. 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 (10).

$msg = h(id_S||time)$ (10)

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

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

3. S verifies signatures σ_T, σ_B, and σ_c as (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 its 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} is 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} is in PsdPub-WL. After checking, S stores transcripts $c, Q_{ps}, msg, \sigma_B, \sigma_T$, and σ_c in Pay-DB.

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

Deposit:

1. S sends c, Q_{ps}, σ_B, and σ_T obtained from U to B. B verifies the signatures as given in (12).
2. After the verification succeeds, B checks if c and Q_{ps} obtained from S exist in Dep-DB. If they do, 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 whether it is equal to σ_c. If σ'_c and σ_c are the same, S rejects performing protocol (double-deposit). Otherwise, S sends σ_c, id_S, acc_S, and msg to B. 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 the verification, B deposits the value of coin into the account of 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, double spending is found. B performs user tracing protocol with user's Q_{ps} and c and detects the identity of the double spender.

User Tracing:

1. B cooperates with T to detect the identity of the double spender. 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 to T.
2. T verifies all signatures by (12). After detecting double spender's Q_{ps}, id_U, and σ_U, T sends them to B.
3. B adds double spender'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, T recognizes this and records the corresponding public key in the revocation list. T distributes the modified revocation list to all participating shops. All shops use it to check the legality of coins 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. Also transparent blindfolding coins under trustee's secret key x_{T_2} is impossible since we use Schnorr signature scheme which is provably secure against chosen message attack [30].

IV. SYSTEM IMPLEMENTATION

A. Tools

To implement Cashpia-v2, we use the cryptographic library ICUCLIB-v2 in which various cryptographic primitives are provided. The primitive algorithms included in ICUCLIB-v2 are SEED [25], SHA-1, ElGamal public key encryption scheme [26], *etc.* It also includes cryptographic operations such as scalar multiplication on an elliptic curve, addition and doubling of points (Schroeppel *et al.*'s algorithm [27]) in optimal normal basis, and multiplication and multiplicative inverse operation over the finite field [28].

E-cash can be stored either in smart card or in hard disk. Smart card has limitation on memory capacity and processing

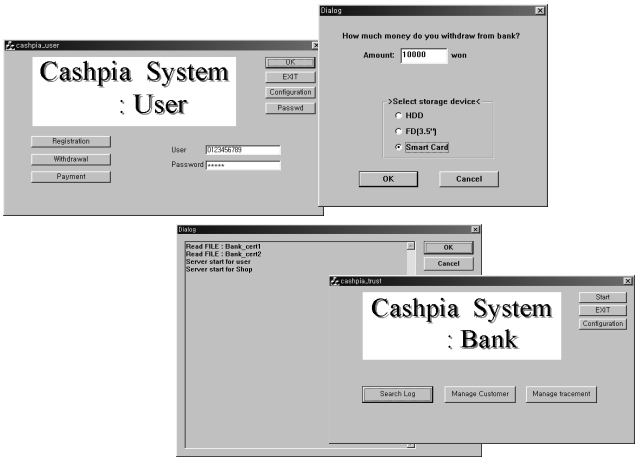

Fig. 2. Example of the user interface (withdrawal phase).

speed. On the other hand, an elliptic curve cryptographic algorithm can efficiently reduce data size and accelerate the computation. Cashpia-v2 can efficiently use smart card to store the coins.

B. Implementation Overview

We choose the elliptic curve over $GF(2^{131})$ as $y^2 + xy = x^3+1$ whose order is $4*6805647338418729269323201294934099985129$ [26]. The simulation prototype was implemented by using C++ on Windows NT 4.0 and Windows 95/98. Fig. 2 shows an example of GUI (Graphical User Interface) of the system at the withdrawal phase. Access DB is used for the database management and `cashpia.mdb` is the database table used in servers and clients. The system is assumed to work on the client/sever architecture. A client contacts a server in advance to make connections. The client side has the IP address and port configuration of the server. The login process requires client's ID and password. We assume that all the pertinent clients are registered in the server. In the server side, all the transactions with clients are monitored through a log screen. Participants' certificates are distributed each other in advance. Now, we describe the data structure and functions used to process the protocols at each 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 on transaction-enabled users and shops such as name, address, and phone number, *etc.* has been set to the database of the bank. Then bank issues account numbers to users and shops.

Registration:

In the registration phase, we apply `DiffieHellman_KeyAgreement` function of trustee and `DH_Key_Exchange_step1`, `DH_Key_Exchange_step2`, and `DH_Key_Exchange_step3` functions of user to produce the session key. User generates a pseudonymous public key by executing the function `PsdKey_Creat_Reg_Step1` and `PsdKey_Creat_Reg_Step2`. User encrypts it by using the session key and passes to trustee. Trustee generates a signature on user's pseudonymous public key and passes it to user in `Registering` function. And trustee's `Storing` function stores the transcripts of the registration phase in `PsdPub-tbl` table. User calls `CODBCset` function and stores the transcripts in `PsdPrv-tbl` table.

Withdrawal:

User produces a coin as a 24-bit random number by using `Generate_Regist_Coin_Step1` function configuring first 8-bit as the denomination of the coin and blinds the coin through `Generate_Regist_Coin_Step2` function. Bank signs the blinded coin by using `Withdrawal_ Blind_Sign` function and sends it to user. User who has received the signature unblinds it by `Generate_Regist_Coin_Step3` function. The transcripts of this phase are stored in each `WithBank-tbl` table and `Coin-tbl` table. If a smart card is used, the coin information to be stored in the `Coin-tbl` table will be saved in the card.

Payment:

`ReceiveData` functions of shop and user are the main function of this phase. When user is connected, shop produces challenge information through `Make_Mess` function, and sends it to user. The user generates signature after receiving challenge information through `ReceiveData` function and sends the signature to shop with the coin information withdrawn from `PsdPrv-tbl` table using `CuserPsdPrvDB` function. The shop verifies the signature received from `ReceiveData` function. After then, 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 user's pseudonymous public key are recorded in `CoinWL` table and `PsdPubWL` tables, respectively, shop stores all transcripts received from `CPay` function in `PayShop-tbl` table.

Deposit:

Shop reads the currency information from `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 `DepBank-tbl` table. If there exists the coin with the same ID, the bank must extract user's signature for the coin. If the signature to be sent by 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, shop must stop the process since it represents 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 represents double-spending.

Anonymity Revocation:

This phase traces the identification of double-spending user. The deposit phase can call this phase. Bank searches for double-spending information in `DepBank-tbl` table of `CReadDepBkTbl` function and passes the information to trustee. Then trustee verifies the received signatures in `SigVerify` function and searches the user's identification from the `PsdPub-tbl` table of `SearchDB` function, and passes the identification to the bank.

V. ATOMICITY

Xu *et al.* [22] addressed money conservation via atomicity in fair off-line e-cash. They assumed some possible scenarios in each phase of e-cash protocol, withdrawal, payment, and deposit protocol. For example, in the withdrawal protocol, the network can crash while a user is waiting for receiving the signature of e-cash. In this case, the e-cash that the user was supposed to receive would be lost because the user didn't receive the e-cash although the bank had already withdrawn the corresponding money. They solved such problems via withdrawal atomicity, payment-delivery atomicity, and deposit atomicity in fair off-line e-cash schemes [17], [11] by using *recovery, resolve, dispute handling protocols*, and *refund function*.

Cashpia-v2 can be extended to guarantee atomicity for purchasing electronic goods as follows:

1. After initialization, *U* and *S* negotiate through some interaction to agree on the price, goods, and *time-out* value, which means the predetermined time limit from receiving coin of *S* to deposit it in its own account.
2. *S* sends an encrypted goods $E_k(goods)$ and a transaction data *msg* to *U*.
3. When *U* claims that he has received goods different from what he had ordered, he computes $h(E_k(goods))$ and transmits it to *T* in order to get the validity of goods.
4. *U* sends $\sigma_c = (R_{U_4}||s_c||$ *time-out*) to *S* during the withdrawal phase, where σ_c includes the *time-out* value with the signature of coin.
5. After *S* verifies the signature of coin, *S* sends the encryption key *k* to *U*.

Withdrawal Atomicity:

If the network crashes during the withdrawal protocol, then recovery is processed as the following steps:

1. *U* sends (R'_{B_1}, e') to *B*, where R'_{B_1} is ephemeral public key of B and e' is blinded value computed by *U*.
2. If a withdrawal phase started before the network crashed, and hasn't completed yet, B sends the signature S'_B to U.
3. Otherwise, this protocol is cancelled.
4. To prevent a double withdrawal, *B* has to send a new R'_{B_1} to *U* and previously signed coin should be blacklisted.

Payment and Delivery Atomicity:

1. *U* sends the transcript of the broken payment phase transaction to *S*.
2. S verifies the transcript, in particular *time-out* value involved in the signature of coin. If a payment phase started before the network crashed, and still bounded within *time-out*, *S* sends the encryption key, k, for goods to *U*.
3. If the payment phase has been exceeded the *time-out*, *U* sends the transcript of the broken transaction to *T*.
4. *T* asks *B* whether this coin has ever been deposited.
5. If the deposit has been done, *T* asks *S* to send encryption key k to *U*. Otherwise, this payment is cancelled and the coin will be stored at the blacklist.

Deposit Atomicity:

1. *S* sends the transcript of the broken deposit phase transaction to *B*.
2. After B verifies the received transcripts, B processes the deposit phase.

VI. SECURITY AND EFFICIENCY

We will show that Cashpia-v2 satisfies most security requirements of e-cash system such as unforgeability, double-spending prevention, anonymity, untraceability, refundability, fairness, framing-freeness, coin-tracing, and blindfolding freeness.

Theorem 2: *Assuming (S_B, V_B) is a computationally secure elliptic curve blind signature scheme and the function h is a collision free hash function, the system supports the security against forgery of coin.*

Proof: Since the blind signature is secure against existential forgery, this allows only the legal bank to generate the signature for coin. As the hash function has the feature of collision free, the user cannot find a value $c' \neq c$ or $Q'_{ps} \neq Q_{ps}$ with $h(c'||Q_{ps}) = h(c||Q_{ps})$. Thus, the system satisfies unforgeability of coins. □

Theorem 3: *Assuming (S_B, V_B) is a computationally secure elliptic curve blind signature scheme, and $(E_{T_{PK}}, V_{T_{SK}})$ is a strong probabilistic encryption scheme, the system supports the security against tracing an honest user by the bank.*

Proof: Since the blind signature cannot give any information for the coin, the bank cannot link the blind coin with an encrypted coin by the probabilistic encryption scheme. □

Theorem 4: *Assuming (S_C, V_C) is a computationally secure elliptic curve signature scheme, the system supports the security against impersonation, framing attack by a bank, and tracing an honest user with the trustee's help.*

Proof: Since anyone (even the bank or the trustee) who doesn't know user's pseudonymous secret key cannot generate a signature on the coin, the impersonation by the bank or trustee is impossible. If the bank wants to frame a user, the bank must present at least two different signatures σ_c and σ'_c to claim double spending. Since (S_C, V_C) is a computationally secure elliptic curve signature scheme, the bank cannot forge any signature. Therefore, the framing attack is prevented. Given a coin, the

Table 1. Comparison of the message size.

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

knowledge of some signatures on the coin does not help to generate new signatures of the user on this coin. Thus, the system supports the security against tracing an honest user. □

Theorem 5: *Assuming (S_U, V_U) is a computationally secure elliptic curve signature scheme, the system is secure against framing an honest user by the trustee.*

Proof: Since duplication by anyone except the authorized user is computationally infeasible, framing an honest user by the trustee is prevented. □

Theorem 6: *Assuming (S_T, V_T) is a computationally secure elliptic curve signature scheme, the system is secure against money laundering and blindfolding by the trustee.*

Proof: Since the trustee who signs a pseudonymous public key knows the relation between user's identification and public key, money laundry is prevented. The user knows two valid signatures only after one interaction with the signer, which contradicts the existential unforgeability of signature. Therefore, the blindfolding is impossible. □

Theorem 7: *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.*

Proof: Since the relation between coin c and Q_{ps} is embedded in the structure of the coin and protected by σ_B, the extortion of coins from the user is impossible. Therefore, an honest user will not lose any unused coin. In case of extortion of the entity's secret keys, the corresponding public keys are blacklisted immediately, which prevents their further use. □

The overall efficiency is improved in Cashpia-v2 compared to PePo97 in terms of the size of message and storage space. We compare Cashpia-v2 which has a point P of 160 bits and q of 160 bits with PePo97 which has 1024 bit prime p and 160 bit prime q. A public key transmitted to generate the session key R_{U_1} in the registration phase is 1,024 bits in PePo97 and 160 bits in Cashpia-v2, and the messages in payment phase $c, Q_{ps}, \sigma_T, \sigma_B$, and σ_c is 4,600 bits in PePo97 and 1,144 bits in Cashpia-v2. Therefore, Cashpia-v2 has 76% to 85% reduction in the message size as shown in Table 1. For p of 512 bits and q of 160 bits, PePo97 has about 52% reduction in the message size compared with Internet payment with electronic purse payment, and Cashpia-v2 has about 44% to 68% reduction compared with Internet payment of PePo97.

VII. CONCLUSION

We have designed and implemented an e-cash system called Cashpia-v2 aiming at a small-amount-of-money transaction and an off-line Internet payment system. The security of Cashpia-v2 is based on ECDLP. Cashpia-v2 has the feature of revocability in order to provide security against blackmailing, money laundering, and double-spending attacks. It is also proved to be secure against secret key extortion, coin extortion, forgery, framing, and blindfolding attacks. Cashpia-v2 is designed under the limited-storage environment such as smart card. To achieve these goals, we choose PePo97 scheme and and extend to improve efficiency in terms of message size. The message size is reduced by 6.4 times compared to the original PePo97. Because of the shortened message size, handling and managing database is easy even under the smart card environment. Furthermore, we can achieve more secure and efficient electronic cash system by using PSLC-2. As an extension, we propose an intuitive method to add atomicity into an electronic cash system.

However, Cashpia-v2 doesn't satisfy divisibility and transferability which are required to build a versatile e-cash system. As a further work, we suggest to research an efficient versatile e-cash system on the elliptic curve and to apply it to the real implementation where the computing power is rather limited.

ACKNOWLEDEGMENT

The authors are very grateful to the anonymous reviewers for their valuable suggestion to improve our manuscript. Special thanks go to Mrs Heesun Kim for her powerful initiating this work when she was at ICU.

REFERENCES

[1] H. Petersen and G. Poupard, "Efficient scalable fair cash with off-line extortion prevention," (Technical Report, ENS, 33 pages, 1997), short version in *Proc. of Int. Conf. on Inform. and Commun. Security (ICICS'97)*, LNCS 1334, Springer-Verlag, 1997, pp.463–477.

[2] J. Baek, B. Lee, and K. Kim, "Provably secure length-saving public key encryption scheme under the computational Diffie-Hellman assumption," *ETRI J.*, vol.22, no.4, pp.25–32, 2000.

[3] D. Chaum, "Blind signatures for untraceable payments," *In Advances in Cryptology-Proc. of CRYPTO'82*, Plenum Press, 1983, pp.199–203.

[4] D. Chaum, "Privacy protected payments: Unconditional payer and/or payee anonymity," *Smart Card 2000: The future of IC Cards*, North-Holland, pp.69–93, 1989.

[5] D. Chaum, A. Fiat, and M. Noar, "Untraceable electronic cash," *In Advances in Cryptology-Proc. of CRYPTO'88*, LNCS 403, Springer-Verlag, 1988, pp. 319–327.

[6] T. Okamoto and K. Ohta, "Universal electronic cash," *In Advances in Cryptology-Proc. of CRYPTO'91*, LNCS 576, Springer-Verlag, 1991, pp. 324–337.

[7] T. Okamoto, "An efficient divisible electronic cash scheme," *In Advances in Cryptology-Proc. of CRYPTO'95*, LNCS 963, Springer-Verlag, 1995, pp. 438–451.

[8] S. Brands, "Untraceable off-line cash in wallets with observers," *In Advances in Cryptology-Proc. of CRYPTO'93*, LNCS 773, Springer-Verlag, 1994, pp. 302–318.

[9] S. von Solms and D. Naccache, "On blind signatures and perfect crimes," *Computers and Security*, pp. 581–583, 1992.

[10] E. Brickell, P. Gemmell, and D. Kravitz, "Trustee-based tracing extensions to anonymous cash and the making of anonymous exchange," in *Proc. of 6th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1995, pp. 457–466.

[11] M. Jakobsson and M. Yung, "Revokable and versatile e-money," in *Proc. of 3rd annual ACM Conf. on Computer and Commun. Security*, 1996, pp.76–87.

[12] J. Camenisch, J. M. Piveteau, and M. Stadler, "An efficient fair payment system," in *Proc. of 3rd ACM Conference on Computer and Commun. Security*, ACM Press, 1996, pp.88–94.

[13] M. Stadler, J. M. Piveteau, and J. Camenisch, "Fair-blind signatures," *In Advances in Cryptology-Proc. of EUROCRYPT'95*, LNCS 921, Springer-Verlag, 1995, pp. 209–219.

[14] E. Fujisaki and T. Okamoto, "Practical escrow cash system," in *Proc. of 1996 Cambridge Workshop on Security Protocols*, LNCS 1189, Springer-Verlag, 1997, pp. 33–48.

[15] D. M'Raïhi, "Cost effective payment schemes with privacy regulations," *In Advances in Cryptology-Proc. of ASIACRYPT'96*, LNCS 1163, Springer-Verlag, 1996, pp. 266–275.