A Novel ID-based Electronic Cash System from Pairings

Jue-Sam Chou*[1], Yalin Chen[2] , Ming-Hsun Cho[3] , Hung-Min Sun[4]

[1] Department of Information Management, Nanhua University, Taiwan R.O.C
*: corresponding author: jschou@mail.nhu.edu.tw
Tel: 886+ (0)5+272-1001 ext.56536

[2] Institute of information systems and applications, National Tsing Hua University

d949702@oz.nthu.edu.tw

[3]Department of Information Management, Nanhua University, 32, Chung Keng Li, Dalin Chiayi 622 Taiwan, ROC

dsshing@gmail.com

[4] Institute of information systems and applications, National Tsing Hua University

hmsun @ cs.nthu.edu.tw

Abstract

Recently, Chen et al. and Juang et al. each proposed one and two e-cash payment systems respectively. They claimed that their schemes are secure. However, in this paper, we will present the shortcomings of their schemes and then propose a novel one from pairings. After security analysis and comparison, we conclude that our scheme not only is more secure but also possesses more functions that a secure electronic cash system should encompass than all of the proposed protocols.

Keywords: Electronic cash, e-cash, Off-line, Bilinear pairing, Blind signature, ID-based

1. Introduction

Nowadays, electronic commerce is becoming more and more popular on the Internet. Many kinds of internet services are therefore developed. Among them, electronic payment system is one of the most important technology.

In 1983, Chaum [3] first proposed an untraceable electronic cash system based on blind signature which allows the requester to obtain a message signature from a signer without revealing the message content and makes the signer cannot link any signed message to its signature. In 1984, Shamir [1] introduced the concept of ID-based cryptography to simplify key management procedures for public key infrastructures. It requires no key exchange. In it, a user's public key can be derived from his identity, and his private key is generated by a trusted third party called Private Key Generator (PKG). Boneh and Franklin [2] proposed a practical ID-based encryption scheme in 2001 and Zhang and Kim [4] proposed the first ID-based blind signature scheme in 2003, respectively. In the systems, data encryption and signature verification processes require only using user's identity along with some public system parameters.

Subsequently, many blind signature schemes [15-20] and many electronic cash systems [5-14] employing blind signature technology based on either traditional DLP or ID-based cryptosystem have been proposed. In 2001, H.Wang et al. [8] proposed an untraceable off-line electronic cash scheme. They claimed that in their scheme, the user remains anonymous from the withdrawal phase, payment phase to the deposit phase. However, we found their scheme not only can't satisfy the anonymous property but also can't prevent an adversary from using the coin for shopping. We will demonstrate the weaknesses in Appendix A.(3) and (4) respectively. In 2004, Juang et al. [5] proposed a practical anonymous multi-authority e-cash scheme to achieve the goal of anonymity, security and verifiability. However, their scheme does not satisfy the unforgeable property because an adversary can fake an e-coin for shopping over the internet. In 2005, Chen et al. [7] proposed a RSA based deposit delegation scheme. They claimed that their scheme is simple and secure. Yet, we found it is insecure for that an adversary can easily make the system confused by masquerading as other merchant. Moreover, their scheme isn't a complete solution for an electronic cash system since it only concerns about the deposit function. In 2007, Juang [6] proposed a D-cash system. They claimed that their scheme is practical and flexible. However, after analysis, we found Juang's scheme does not satisfy the unforgeable property because an adversary can easily fake a signature for the customer during the withdrawal phase (Later, we will demonstrate the weaknesses in schemes [5, 6, 7] in this article.). Also in 2007, Popescu et al. [9] proposed an off-line electronic cash based on pairings. They claimed their scheme is secure. However, it has a serious drawback that the e-coin in the withdrawal phase is different from the one in the payment phase. This violates the basic rule of an e-cash system. In 2008, Shangping Wang et al. [10] proposed a certificateless electronic cash scheme with multiple banks based on group signatures. They claimed that their scheme can satisfy various security requirements. But we found it has a shortcoming that an adversary can successfully pass the identity verification to withdraw an e-coin from the withdrawal phase. We will demonstrate the weaknesses in Appendix A. (4). Also in 2008, Alfredo De Santis et al. [13] showed an attack on Wang et al.'s flexible payment scheme [14]. Their attack uses only public information to construct a faked proof for the ownership of a coin. In 2009, Fan et al. [22] proposed a fair anonymous rewarding based on electronic cash and Ashrafi et al. [23] proposed a privacy-preserving e-payments using one-time payment. However, the former emphasizes on fair rewarding, not on the e-cash mechanism itself and the latter just focuses on the payment phase. That is, there still lacks a secure complete system nowadays. Hence, in this paper, we propose such a system. We will first review the shortcomings of three e-cash schemes [5-7] and then propose a secure ID-based electronic cash system from pairings.

The remainder of this paper is organized as follows. In Section 2, we describe the roles and functions in a secure electronic cash system, the background concepts of bilinear pairings, and some related mathematical problems. In Section 3, we analyze the schemes of Juang et al.'s [5], Juang's [6], and Chen et al.'s [7], respectively. After that, we present our ID-based bilinear pairing electronic cash system in Section 4 and analyze its security in Section 5. Then, we make comparisons with other schemes in Section 6. Finally, a conclusion is given in Section 7.

2. Background

In this section, we briefly review the roles and functions in a secure complete electronic cash system, the basic knowledge of bilinear pairings, and some related mathematical problems on which the security of our scheme is based.

2.1. Roles and functions in a secure complete electronic cash system

A simple electronic cash system [8] consists of three parties (a customer C, a bank B, and a merchant M) and three main procedures (withdrawal, payment and deposit). In an e-coin's life-cycle, the customer C first opens an account in a bank B. To obtain an e-coin, C performs a withdrawal protocol with B. C then performs a payment protocol for shopping in a merchant by using the withdrawn e-coin. After receiving the e-coin, M sends it to the bank B and B transfers the corresponding money to M's account. Moreover, From [6, 8, 9, 10, 24, 25], we summarize five important requirements for a secure complete electronic cash system. They are: (1)mutual authentication, (2)verifiability, (3)anonymity, (4)unforgeability, and (5)traceability. "mutual authentication" means that two parties can authenticate each other correctly. "verifiability" points out that one can ensure the correctness and integrity of messages transmitted by the other designated party. "anonymity" signifies that no e-coin can be linked by the bank to its corresponding withdrawal. "unforgeability" indicates that no adversary can create a valid e-coin without having to withdraw from a bank. And "traceability" means that we can reveal the identity of customer if the same e-coin is spent twice.

2.2. Bilinear Pairings

Let G_1 be a cyclic additive group of order q generated by a base point P, G_2 be a cyclic multiplicative group of the same order. It is assumed that solving the discrete logarithm problem (DLP) in both G_1 and G_2 is difficult [2]. A cryptographic bilinear map e is defined as $e: G_1 \times G_1 \to G_2$ which has the following properties [21]:

(1) Bilinear: $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$,where $P, Q \in G_1$ and all $a, b \in Z_q^*$.

(2) Non-degeneracy: There exists $P, Q \in G_1$ such that $\hat{e}(P, Q) \neq 1$, in other words, the map does not send all pairs in $G_1 \times G_1$ to the identity in G_2.

(3) Computable: There is an efficient algorithm to compute $\hat{e}(P, Q)$ for all $P, Q \in G_1$.

2.3. Computational Problems on Elliptic curve

Here, we introduce some mathematical problems which form the security basis of our scheme.

(1) Elliptic curve Discrete logarithm problem (ECDLP): Given two elements $P, Q \in G_1$, find an integer $a \in Z_q^*$, such that $Q = aP$.

(2) Computational Diffie-Hellman problem (CDHP): For any $a, b \in Z_q^*$, given $< P, aP, bP >$, compute abP.

(3) Decisional Diffie-Hellman problem (DDHP): For any $a, b, c \in Z_q^*$, given $< P, aP, bP, cP >$, decide whether $c \equiv ab \bmod q$.

(4) Bilinear computational Diffie-Hellman problem (BCDHP): For any $a, b, c \in Z_q^*$, given $< P, aP, bP, cP >$, compute $e(P, P)^{abc}$.

3. Reviews and attacks on three e-cash schemes

In this section, we review and show our attacks on Juang et al.'s e-cash scheme [5] in Section 3.1, Juang's D-cash scheme [6] in Section 3.2, and Chen et al.'s deposit delegation scheme [7] in Section 3.3, respectively.

3.1. Review and attack on Juang et al.'s scheme

In 2004, Juang et al. proposed an anonymous multi-authority e-cash scheme [5] concerning only the withdrawing phase and paying phase. They claimed that their scheme is secure. However, we found that it does not satisfy the unforgeable property. Because that an adversary can fake a valid e-coin for shopping over the internet. In the following, we will demonstrate this by first review their scheme in part (A), then show the weakness in part (B).

(A) Review of Juang et al.'s scheme

In the following, we first list the definitions of used notations then show the four phases in their scheme.

(a) Definitions of used notations

ξ : a public one-way permutation function

$\mathcal{H}$: a public one-way hash function

n' : the number of money issuers before the preparation phase

QUAL: the set of non-disqualified money issuers after the preparation phase

n : the number of non-disqualified money issuers in QUAL

$\mathcal{I}_i$, $1 \le i \le n'$: the identity of money issuer i before the preparation phase

I_i ,$1 \le i \le n$: the identity of non-disqualified money issuer i after the preparation phase

d_c : the secret key chosen by customer C

$d_{\mathcal{I}_i}$: the secret key chosen by $\mathcal{I}_i$

e_c : customer C's corresponding public key

$e_{\mathcal{I}_i}$: $\mathcal{I}_i$'s corresponding public key

p , q : two large strong prime numbers satisfying q divides $(p-1)$

ρ : a generators of Z_p^*

ζ : a random value generated by a generic distributed coin flipping protocol, also a generators of Z_p^*

(b) The four phases

Juang et al.'s scheme consists of four phases: (1)initialization phase, (2)preparation phase, (3)withdrawal phase, and (4)paying phase. We describe them as follows.

(1) ***Initialization phase***

The bank publishes all public parameters *n*, *t*, *p*, *q*, *g*, and *h*, all identifications of e-coins issuers $\mathcal{I}_i$, $1 \le i \le n'$, the public one-way permutation ξ, and the public one-way hash function $\mathcal{H}$.

(2) ***Preparation phase*** *(as shown in Fig. 1.)*

All $\mathcal{I}_i$, $1 \le i \le n'$, must cooperate to distribute their secret shadows to each other. They carry out the following steps:

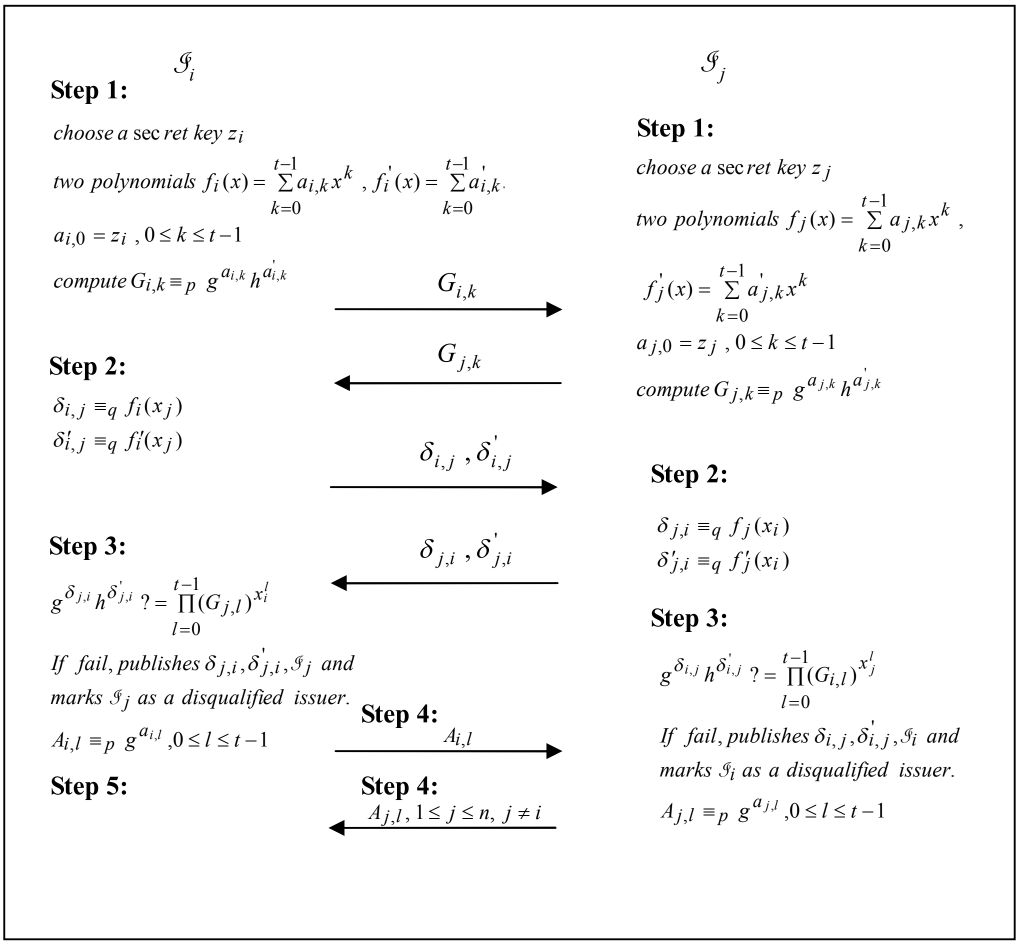
Step1: $\mathcal{I}_i$ chooses a secret key $z_i \in Z_q$ and two secret polynomials $f_i(x) = \sum_{k=0}^{t-1} a_{i,k} x^k$, $f_i'(x) = \sum_{k=0}^{t-1} a'_{i,k} x^k$ such that $a_{i,0} = z_i$. He computes and sends $G_{i,k} \equiv_p g^{a_{i,k}} h^{a'_{i,k}}$, $0 \le k \le t-1$ to $\mathcal{I}_j$, $1 \le j \le n'$, $j \ne i$.

Step2: Upon receiving $G_{j,k}$, $1 \le j \le n'$, $0 \le k \le t-1$, $j \ne i$ from all other issuers, $\mathcal{I}_i$ secretly sends $\delta_{i,j} \equiv_q f_i(x_j)$ and $\delta'_{i,j} \equiv_q f_i'(x_j)$ to every other $\mathcal{I}_j$, where x_j is a unique public number for $\mathcal{I}_j$.

Step3: When $\mathcal{I}_i$ receives all $\delta_{i,j}$ and $\delta'_{i,j}$, $1 \le j \le n'$, $j \ne i$ from other issuers, he verifies if the shares, $\delta_{i,j}$ and $\delta'_{i,j}$, received from $\mathcal{I}_j$ is consistent with the certified values $G_{j,l}$, $0 \le l \le t-1$, by checking whether or not $g^{\delta_{j,i}} h^{\delta'_{j,i}} \equiv_p \prod_{l=0}^{t-1} (G_{j,l})^{x_i^l}$. If it fails, $\mathcal{I}_i$ broadcasts that an error has occurred. He publishes $\delta_{i,j}$, $\delta'_{i,j}$ and the authentication information for $\mathcal{I}_j$. Each issuer, except for the dishonest issuer $\mathscr{I}_j$, then marks $\mathscr{I}_j$ as a disqualified issuer and excludes $\mathscr{I}_j$ from the set of non-disqualified issuers QUAL.

Step4: Every issuer $\mathscr{I}_i$, $i \in QUAL$, broadcasts $A_{i,l} \equiv_p g^{a_{i,l}}, 0 \le l \le t-1$.

Step5: When $\mathscr{I}_i$, $i \in QUAL$, receives all $A_{j,l}$, $1 \le j \le n$, $j \ne i$, from other issuers in QUAL, he verifies whether $g^{\delta_{j,i}} \equiv_p \prod_{l=0}^{t-1}(A_{j,l})^{x_i^l}$. If this check fails for an index j, $\mathscr{I}_i$ broadcasts that an error has been found. He publishes $\delta_{i,j}$, $\delta'_{i,j}$ and the authentication information for $\delta_{j,i}, \delta'_{j,i}$. $\mathscr{I}_i$ and any t issuers in QUAL can cooperate to compute z_j, $f_j(x)$, and $A_{j,k}$, $0 \le k \le t-1$. Anyone then can compute the public shadows $\Phi_{j,i} \equiv_p g^{\delta_{j,i}} \equiv_p \prod_{l=0}^{t-1}(A_{j,l})^{x_i^l}$ and the group public key $y \equiv_p \prod_{j \in QUAL} y_i \equiv_p \prod_{j \in QUAL} A_{j,0}$, where $i, j \in$ QUAL. The group public key *y*, all public shadows $\Phi_{j,i}$, and the personal public key $y_i \equiv_p A_{i,0} \equiv_p g^{z_i}$, where *i* and $j \in$ QUAL, can then be published by each issuer $\mathscr{I}_i$. They assume that the n non-disqualified issuers in QUAL are I_i, $1 \le i \le n$.



$\mathscr{I}_i$
$\mathscr{I}_j$
Step 1:
$choose\ a\ \sec ret\ key\ z_i$
$two\ polynomials\ f_i(x) = \sum_{k=0}^{t-1} a_{i,k}x^k\ ,\ f'_i(x) = \sum_{k=0}^{t-1} a'_{i,k}$
$a_{i,0} = z_i\ , 0 \le k \le t-1$
$compute\ G_{i,k} \equiv_p g^{a_{i,k}} h^{a'_{i,k}}$
$G_{i,k}$
$G_{j,k}$
Step 2:
$\delta_{i,j} \equiv_q f_i(x_j)$
$\delta'_{i,j} \equiv_q f'_i(x_j)$
$\delta_{i,j}\ , \delta'_{i,j}$
Step 3:
$g^{\delta_{j,i}} h^{\delta'_{j,i}}\ ? = \prod_{l=0}^{t-1}(G_{j,l})^{x_i^l}$
$If\ fail, publishes\ \delta_{j,i}, \delta'_{j,i}, \mathscr{I}_j\ and\ marks\ \mathscr{I}_j\ as\ a\ disqualified\ issuer.$
$A_{i,l} \equiv_p g^{a_{i,l}}, 0 \le l \le t-1$
Step 5:
$\delta_{j,i}\ , \delta'_{j,i}$
Step 4:
$A_{i,l}$
Step 4:
$A_{j,l}, 1 \le j \le n,\ j \ne i$
Step 1:
$choose\ a\ \sec ret\ key\ z_j$
$two\ polynomials\ f_j(x) = \sum_{k=0}^{t-1} a_{j,k}x^k\ ,$
$f'_j(x) = \sum_{k=0}^{t-1} a'_{j,k}x^k$
$a_{j,0} = z_j\ , 0 \le k \le t-1$
$compute\ G_{j,k} \equiv_p g^{a_{j,k}} h^{a'_{j,k}}$
Step 2:
$\delta_{j,i} \equiv_q f_j(x_i)$
$\delta'_{j,i} \equiv_q f'_j(x_i)$
Step 3:
$g^{\delta_{i,j}} h^{\delta'_{i,j}}\ ? = \prod_{l=0}^{t-1}(G_{i,l})^{x_j^l}$
$If\ fail, publishes\ \delta_{i,j}, \delta'_{i,j}, \mathscr{I}_i\ and\ marks\ \mathscr{I}_i\ as\ a\ disqualified\ issuer.$
$A_{j,l} \equiv_p g^{a_{j,l}}, 0 \le l \le t-1$

Fig. 1. The preparation phase of multi-authority e-cash scheme.

(3) ***Withdrawal phase*** *(as shown in Fig. 2.)*

In this phase, customer ID_c employs a threshold blind signature scheme to get a blind e-coin from *t* honest issuers. Without loss of generality, they assume that the t out of n issuers requested by ID_c are $I_j, 1 \le j \le t$. Customer ID_c and I_j, $1 \le j \le t$, then together perform the following protocol.

Step1: Each I_j randomly chooses a number $k_j \in Z_q$, computes $\hat{r}_j \equiv_p g^{k_j}$, and sends $\hat{r}_j$ to the customer.

Step2: After receiving all $\hat{r}_j$, the customer does the following:

① Computes the value $m = H_{ID} \| RD \| H_x(\sigma)$, where $H_0(\sigma) = \sigma$, $H_i(\sigma) = H(H_{i-1}(\sigma))$, $1 \le i \le x$, RD is the redundancy information for verification, σ and λ are two random numbers, and $H_{ID}\,(=\xi(ID_c \| \lambda))$ is an unique header.

② Chooses two random numbers $\alpha, \beta \in Z_q^*$ and computes $r_j \equiv_p g^{\alpha}\hat{r}_j^{\beta}$, $r \equiv_p m\prod_{k=1}^{t} r_k$, and $\hat{m} \equiv_q \beta^{-1} r$.

③ Checks to see if $\hat{m} \ne 0$. If so, he sends the string $\hat{m}$ to all I_j, $1 \le j \le t$; otherwise, goes back to step ②.

Step3: Upon receiving message $\hat{m}$, I_j, ($1 \le j \le t$), checks to see if the customer has enough money in the bank. If so, he informs the bank to deduct x dollars from the customer's account, computes $\hat{s}_j \equiv_q \hat{m}(z_j + \sum_{l=t+1}^{n}(f_l(x_j)(\prod_{k=1,k\ne j}^{t}(-x_k/(x_j - x_k))))) + k_j$, and sends $\hat{s}_j$ back to the customer. Else, I_j rejects the money withdrawing of the customer.

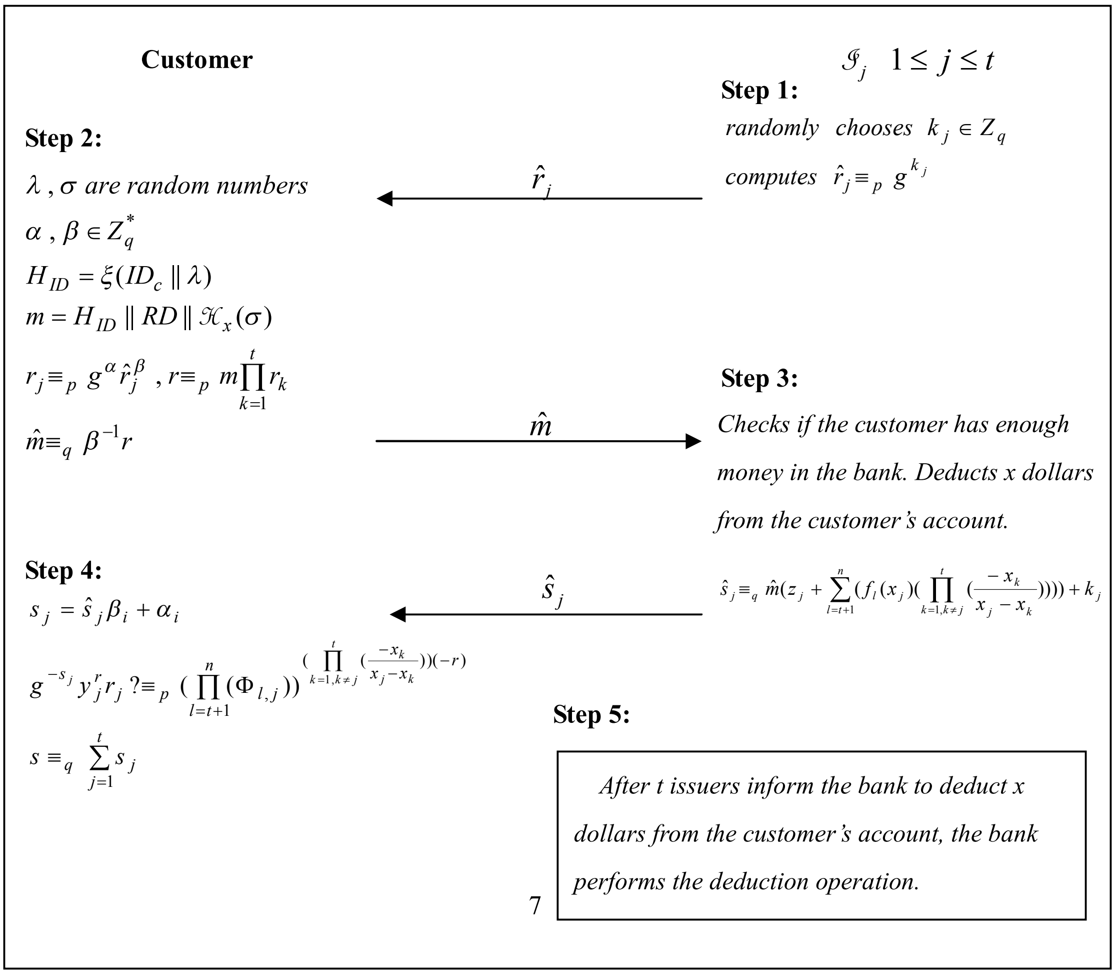

Customer
$\mathcal{I}_j \quad 1 \le j \le t$
Step 1:
randomly chooses $k_j \in Z_q$
computes $\hat{r}_j \equiv_p g^{k_j}$
Step 2:
λ, σ *are random numbers*
$\alpha, \beta \in Z_q^*$
$H_{ID} = \xi(ID_c \| \lambda)$
$m = H_{ID} \| RD \| \mathcal{H}_x(\sigma)$
$r_j \equiv_p g^{\alpha}\hat{r}_j^{\beta}$, $r \equiv_p m\prod_{k=1}^{t} r_k$
$\hat{m} \equiv_q \beta^{-1} r$
$\hat{r}_j$
$\hat{m}$
Step 3:
Checks if the customer has enough money in the bank. Deducts x dollars from the customer's account.
Step 4:
$s_j = \hat{s}_j \beta_i + \alpha_i$
$g^{-s_j} y_j^{r} r_j \,?\!\equiv_p (\prod_{l=t+1}^{n}(\Phi_{l,j}))^{(\prod_{k=1,k\ne j}^{t}(\frac{-x_k}{x_j - x_k}))(-r)}$
$s \equiv_q \sum_{j=1}^{t} s_j$
$\hat{s}_j$
$\hat{s}_j \equiv_q \hat{m}(z_j + \sum_{l=t+1}^{n}(f_l(x_j)(\prod_{k=1,k\ne j}^{t}(\frac{-x_k}{x_j - x_k})))) + k_j$
Step 5:
After t issuers inform the bank to deduct x dollars from the customer's account, the bank performs the deduction operation.

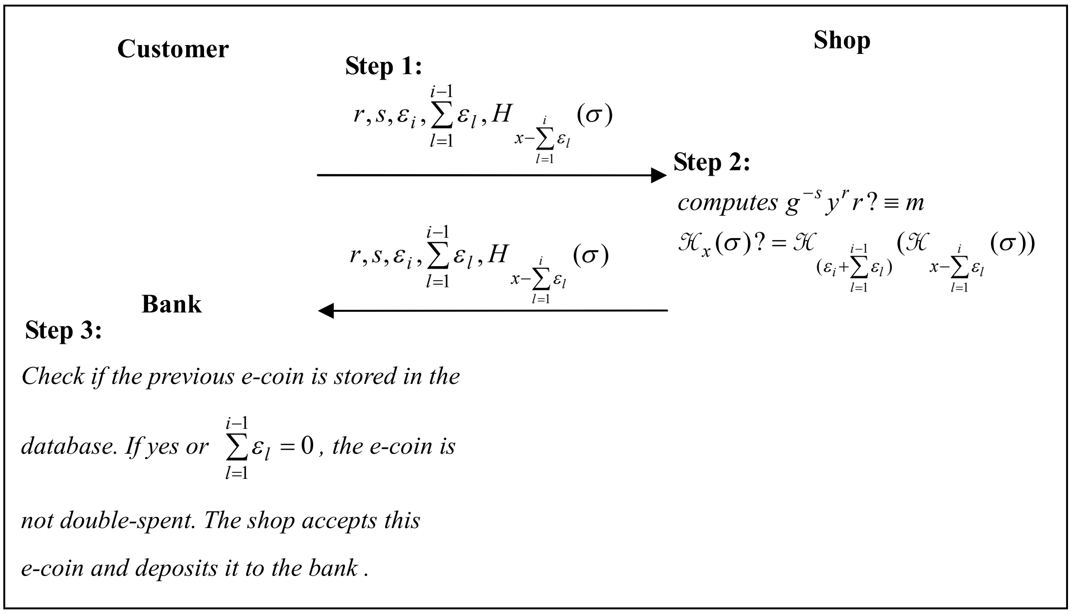
Fig. 2. The withdrawal phase of multi-authority e-cash scheme.

Step4: After receiving all $\hat{s}_j$, $(1 \le j \le t)$, the customer computes $s_j \equiv_q \hat{s}_j \beta_i + \alpha_i$, and checks to see if $g^{-s_j} y_j^r r_j \equiv_p (\prod_{l=t+1}^{n}(\Phi_{l,j}))^{(\prod_{k=1,k\neq j}^{t}(\frac{-x_k}{x_j-x_k}))(-r)}$, $(1 \le j \le t)$, holds. If it doesn't hold, $\hat{s}_j$ is not valid. The customer then has to ask the corresponding issuer to send it again. Otherwise, he computes $s \equiv_q \sum_{j=1}^{t} s_j$.

Step5: After exactly t issuers inform the bank to deduct x dollars from the customer's account, the bank performs the deduction operation.

(4) ***Paying phase*** *(as shown in Fig. 3.)*

Assume that a customer has accumulatively spent $\sum_{l=1}^{i-1}\varepsilon_l$ dollars in some shops for an e-coin $(r, s, \sum_{l=1}^{i-1}\varepsilon_l, H_{x-\sum_{l=1}^{i}\varepsilon_l}(\sigma))$, where ε_l denotes the amount of dollars that are spent at the l^{th} transaction and $\sum_{l=1}^{i-1}\varepsilon_l < x$. In this phase, if the customer wants to pay the shop ε_i dollars, then the customer and the shop should cooperate to do the following steps:


Customer
Shop
Step 1:
$r, s, \varepsilon_i, \sum_{l=1}^{i-1}\varepsilon_l, H_{x-\sum_{l=1}^{i}\varepsilon_l}(\sigma)$
Step 2:
computes $g^{-s} y^r r ? \equiv m$
$\mathcal{H}_x(\sigma) ? = \mathcal{H}_{(\varepsilon_i+\sum_{l=1}^{i-1}\varepsilon_l)}(\mathcal{H}_{x-\sum_{l=1}^{i}\varepsilon_l}(\sigma))$
$r, s, \varepsilon_i, \sum_{l=1}^{i-1}\varepsilon_l, H_{x-\sum_{l=1}^{i}\varepsilon_l}(\sigma)$
Bank
Step 3:

Check if the previous e-coin is stored in the database. If yes or $\sum_{l=1}^{i-1}\varepsilon_l = 0$, *the e-coin is not double-spent. The shop accepts this e-coin and deposits it to the bank .*

Fig. 3. The paying phase of multi-authority e-cash scheme.

Step1: If $\sum_{l=1}^{i}\varepsilon_l \le x$, then the customer sends the e-coin ($r, s,$ ε_i , $\sum_{l=1}^{i-1}\varepsilon_l$, $\mathcal{H}_{x-\sum_{l=1}^{i}\varepsilon_l}(\sigma)$) representing ε_i dollars to the shop. Otherwise, he stops.

Step2: After receiving the e-coin, the shop verifies whether the e-coin is valid by computing $g^{-s} y^r r \equiv_p m$ and checks to see if both $\mathcal{H}_x(\sigma) = \mathcal{H}_{(\varepsilon_i+\sum_{l=1}^{i-1}\varepsilon_l)}(\mathcal{H}_{x-\sum_{l=1}^{i}\varepsilon_l}(\sigma))$ and *RD* contains some redundancy information hold. If both hold, the shop calls the bank to check if the e-coin is double-spent by sending him the e-coin

$(r, s, \varepsilon_i, \sum_{l=1}^{i-1}\varepsilon_l, \mathcal{H}_{x-\sum_{l=1}^{i}\varepsilon_l}(\sigma))$.

Step3: The bank checks to see if the previous spent e-coin $(r, s, \sum_{l=1}^{i-1}\varepsilon_l, \mathcal{H}_{x-\sum_{l=1}^{i-1}\varepsilon_l}(\sigma))$ was stored in the database. If so or $\sum_{l=1}^{i-1}\varepsilon_l = 0$, the bank confirms that the e-coin is not double-spent. Then the shop accepts the e-coin and deposits ε_i dollars into the bank. The bank increases the shop's account by ε_i dollars and stores the e-coin $(r, s, \sum_{l=1}^{i}\varepsilon_l, \mathcal{H}_{x-\sum_{l=1}^{i}\varepsilon_l}(\sigma))$ in his database for subsequent double-spent checking.

(B) Attack on Juang et al.'s scheme

It is obvious that Juang et al.'s e-cash scheme suffers from man-in-the-middle attack in the paying phase. For example, if the customer has one hundred dollars and wants to spend 30 dollars for a commodity, he sends $r, s, 30, 10, H_{60}(\sigma)$ to the shop, the adversary can simply intercept and change this e-coin to $r, s, 20, 10, H_{70}(\sigma)$. He then can send this faked e-coin to the shop or another for purchasing his wanted commodity which costs 20 dollars. The shop will verify it as legal unconsciously.

3.2. Review and attack on Juang's D-cash scheme

In 2007, Juang proposed a flexible pre-paid e-cash scheme for date-attachment [6]. His scheme mainly concerns with withdrawing phase and date-attaching phase. He claimed that his scheme is secure. However, we found that the scheme does not satisfy the unforgeable property since an adversary can fake a signature for the customer during the withdrawal phase. In the following, we will do: (A) briefly review his scheme, and (B) show our attack.

(A) Review of Juang's D-cash scheme

Juang's scheme consists of four phases: (1) initializing phase, (2) withdrawing phase, (3) date-attaching phase, and (4) depositing phase. In the following, we will do: (a) show the definitions of used notations, and (b) describe the four phases.

(a) Definitions of used notations

h_1, h_2: two secure one-way hash functions

p, q: two large strong prime numbers satisfying q divides $(p-1)$

ρ: a generator of Z_p^*

c_1, c_2: the withdrawing date and the effective date, respectively

g: a generator of a subgroup of Z_p^* with order q

$\varpi(x, y, c)$: a public polynomial defined as $h_1(c)x + h_2(c)y \bmod q$

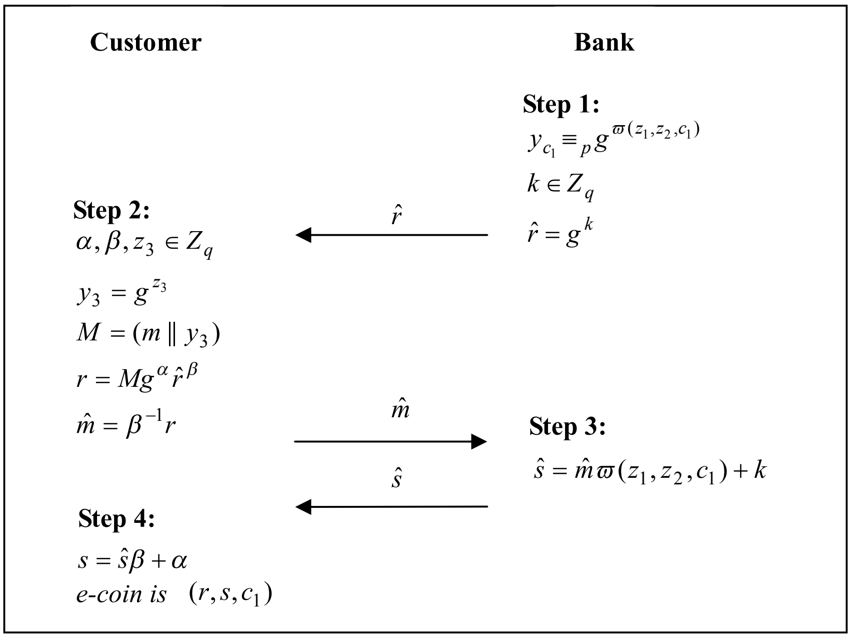
(b) The four phases

(1) Initializing phase

The bank generates his private keys $z_1, z_2 \in Z_q$ and sets $y_1 \equiv_p g^{z_1}$, $y_2 \equiv_p g^{z_2}$ to be his corresponding public keys.

(2) Withdrawing phase *(as shown in Fig. 4.)*

When a customer withdraws an e-coin from a bank, he and the bank together perform the following steps. Here, the bank's public key is $y_{c_1} (\equiv_p g^{\varpi(z_1,z_2,c_1)} \equiv_p g^{h_1(c_1)z_1+h_2(c_2)z_2} \equiv_p y_1^{(h_1(c_1))} y_2^{(h_2(c_1))})$ which contains the withdrawing date c_1, and it's corresponding private key is $\varpi(z_1,z_2,c_1)$ $(\equiv_q h_1(c_1)z_1 + h_2(c_1)z_2)$.



Customer
Bank
Step 1:
$y_{c_1} \equiv_p g^{\varpi(z_1,z_2,c_1)}$
$k \in Z_q$
$\hat{r} = g^k$
$\hat{r}$
Step 2:
$\alpha, \beta, z_3 \in Z_q$
$y_3 = g^{z_3}$
$M = (m \,\|\, y_3)$
$r = Mg^{\alpha}\hat{r}^{\beta}$
$\hat{m} = \beta^{-1} r$
$\hat{m}$
Step 3:
$\hat{s} = \hat{m}\varpi(z_1,z_2,c_1) + k$
$\hat{s}$
Step 4:
$s = \hat{s}\beta + \alpha$
e-coin is (r, s, c_1)

Fig. 4. The withdrawing phase of D-Cash.

Step1: The bank randomly chooses a number $k \in Z_q$, computes and sends $\hat{r} \equiv_q g^k$ to the customer.

Step2: After receiving $\hat{r}$, the customer does the following.

① Randomly chooses a private key $z_3 \in Z_q$ as his pseudonym and sets $y_3 \equiv_p g^{z_3}$ to be the corresponding public key for the pseudonym. He then computes $M = (m \,\|\, y_3)$, where m is the blind message containing a predefined message pattern for the e-coin.

② Chooses two random numbers $\alpha, \beta \in Z_q^*$ and computes $r \equiv_p Mg^{\alpha}\hat{r}^{\beta}$, $\hat{m} \equiv_q \beta^{-1} r$.

③ Checks to see if $\hat{m} \neq 0$. If so, sends $\hat{m}$ to the bank; otherwise, goes back to ①.

Step3: Upon receiving $\hat{m}$, the bank computes and sends $\hat{s} \equiv_q \hat{m}\varpi(z_1,z_2,c_1) + k$ back to the customer. It then deducts w dollars from the customer's bank account.

Step4: After receiving $\hat{s}$, the customer computes $s \equiv_q \hat{s}\beta + \alpha$ and obtains the e-coin (r, s, c_1).

***(3) Date-attaching phase** (as shown in Fig. 5.)*

When the customer wants to spend the e-coin in a merchant, he and the merchant together perform the following steps.

Step1: The customer signs the effective date c_2 on the e-coin (r, s, c_1) by computing $r' \equiv_q (g^{k'} \bmod p)$ and $s' \equiv_q k'^{-1}(h_1(r \| s \| c_1 \| c_2) - z_3 r')$, where $k' \in Z_q$ is randomly chosen by the customer. He then sends the result (r, s, r', s', c_1, c_2) to the merchant.

Step2: After receiving (r, s, r', s', c_1, c_2), the merchant will verify the validity of this e-coin by computing $M \equiv_p g^{-s} y_{c_1}^{r} r = (m \| y_3)$, and checking if m contains the predefined message pattern. If so, he computes $u_1 = h_1(r \| s \| c_1 \| c_2)\ s'^{-1} \bmod q$, $u_2 = r's'^{-1} \bmod q$ and checks to see whether both $g^{u_1} y_3^{u_2} \bmod q = r'$ and $c_2 \geq c_1$ hold. If both hold, the merchant calls the bank to check the freshness of the e-coin. If it is fresh, the merchant will pay the customer the interest generated during the withdrawal date c_1 and the effective date c_2.

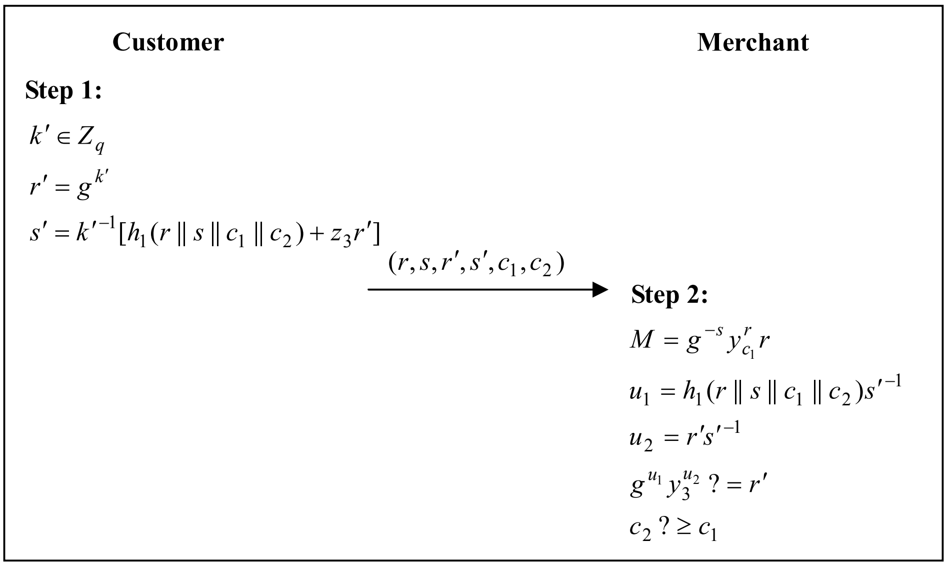

Customer
Merchant
Step 1:
$k' \in Z_q$
$r' = g^{k'}$
$s' = k'^{-1}[h_1(r \| s \| c_1 \| c_2) + z_3 r']$
(r, s, r', s', c_1, c_2)
Step 2:
$M = g^{-s} y_{c_1}^{r} r$
$u_1 = h_1(r \| s \| c_1 \| c_2) s'^{-1}$
$u_2 = r's'^{-1}$
$g^{u_1} y_3^{u_2} ?= r'$
$c_2 ? \geq c_1$

Fig. 5. The date-attaching phase of D-Cash.

(4) Depositing phase

After the effective date c_2 of the e-coin, the merchant can deposit it to his bank account. The bank will add w dollars to the merchant's account, pay the merchant interest generated during the withdrawal date c_1 and the deposit date of this e-coin, and store this transaction in his database.

(B) Attack on Juang's D-cash scheme

Although Juang claimed that his scheme is secure, we found that it has a serious weakness. We describe it as follows.

In the withdrawing phase, when the bank sends $\hat{r}$ to the customer, an adversary can intercept $\hat{r}$, compute $\hat{r}' = \hat{r}g^{\widetilde{k}}$ $(= g^{k+\widetilde{k}})$, where $\widetilde{k}$ is a randomly chosen number by the adversary. The adversary then sends $\hat{r}'$ to the customer. Upon receiving $\hat{r}'$, the customer computes $\widetilde{r} = Mg^{\alpha}\hat{r}'^{\beta}$, $\hat{m}' = \beta^{-1}\widetilde{r}$ and sends $\hat{m}'$ back to the bank. Upon receiving $\hat{m}'$, the bank computes and sends $\hat{s}' = \hat{m}'\varpi(z_1, z_2, c_1) + k$ to the customer. The adversary can intercept $\hat{s}'$, replace it with $\hat{s}'' = \hat{s}' + \widetilde{k}$, and send it to the customer. Upon receiving $\hat{s}''$, the customer computes $\widetilde{s} = \hat{s}''\beta + \alpha$. Then the faked e-coin for the message m is $(\widetilde{r}, \widetilde{s}, c_1)$. When the customer pay the faked e-coin $(\widetilde{r}, \widetilde{s}, r'', s'', c_1, c_2)$ to the merchant in the date-attaching phase, where $r'' = g^{k''}$, $s'' = k''^{-1}[h_1(\widetilde{r} \parallel \widetilde{s} \parallel c_1 \parallel c_2) + z_3 r'']$, and $k'' \in Z_q$ is the random number chosen by the customer, the merchant first computes

$$g^{-\widetilde{s}} y_{c_1}^{\widetilde{r}} \widetilde{r}$$
$$= g^{-(\hat{s}''\beta+\alpha)} g^{\varpi(z_1,z_2,c_1)\widetilde{r}} \widetilde{r}$$
$$= g^{-[(\hat{s}'+\widetilde{k})\beta+\alpha]} \quad g^{\varpi(z_1,z_2,c_1)\widetilde{r}} \quad \widetilde{r}$$
$$= g^{-\{[\hat{m}'\varpi(z_1,z_2,c_1)+k+\widetilde{k}]\beta+\alpha\}} \; g^{\varpi(z_1,z_2,c_1)\widetilde{r}} \; \widetilde{r}$$
$$= g^{-\{[\beta^{-1}\widetilde{r}\varpi(z_1,z_2,c_1)+k+\widetilde{k}]\beta+\alpha\}} \; g^{\varpi(z_1,z_2,c_1)\widetilde{r}} \; \widetilde{r}$$
$$= g^{-(k+\widetilde{k})\beta-\alpha} \; Mg^{\alpha}\hat{r}'^{\beta}$$
$$= g^{-(k+\widetilde{k})\beta-\alpha} \; Mg^{\alpha}g^{(k+\widetilde{k})\beta}$$

$= M$, where $M = (m \parallel y_3)$, to obtain y_3. He then computes

$$u_1' = h_1(\widetilde{r} \parallel \widetilde{s} \parallel c_1 \parallel c_2)\; s''^{-1} \bmod q,$$
$$u_2' = r''s''^{-1} \bmod q$$

to verify the validity of the e-coin by checking whether or not $g^{u_1'} y_3^{u_2'} = r''$ holds. The merchant performs the verification process as follows:

$$g^{u_1'} y_3^{u_2'}$$
$$= g^{h_1(\widetilde{r}\parallel\widetilde{s}\parallel c_1\parallel c_2)s''^{-1}} \quad g^{z_3 r'' s''^{-1}}$$
$$= g^{s''^{-1}[h_1(\widetilde{r}\parallel\widetilde{s}\parallel c_1\parallel c_2)+z_3 r'']}$$
$$= g^{(s'')^{-1}s''k''}$$
$$= g^{k''}$$
$$= r''$$

Hence, the faked e-coin $(\widetilde{r}, \widetilde{s}, c_1)$ can be verified successfully by the merchant in the date-attaching phase. Besides, Juang claimed that his scheme also satisfies the anonymous property. However, we found that in the withdrawing phase, although the values of *r* and *s* in the e-coin (r, s, c_1) are only known to the customer, the bank can know the e-coin owner's identity by the withdrawing date c_1 with non-negligible probability since the bank needs to deduct *w* dollars from the customer's bank account in step 3 of the withdrawing phase (as shown in Fig. 4.).

3.3. Review and attack on Chen et al.'s RSA based deposit delegation scheme

Chen et al. proposed a RSA based deposit delegation scheme in [7]. They claimed that their scheme is secure. However, we found that it has a security weakness. In the following, we will first describe their RSA based deposit delegation scheme (as shown in Fig. 6.) in part (A), then show the weakness found in part (B).

(A) Review of Chen et al.'s scheme (as shown in Fig.6)

Chen et al.'s scheme contains five steps. In the following, we will do: (a) list the definitions of used notations, and (b) show the four steps.

(a) Definitions of used notations

(p_x, q_x) : a pair of large prime numbers
N_x : a large number, where $N_x = p_x \cdot q_x$
$\phi(N_x)$: $(p_x - 1) \cdot (q_x - 1)$
C_{R1}, C_{R2} : two ciphertexts
$Sig_{Acquirer}$: the acquirer's signature

(b) The four steps

Step1: Initially, each bank X chooses a pair of prime numbers (p_x, q_x) and computes the product as N_x. He then generates the public key PK_x and the corresponding secret key SK_x as the RSA encryption/decryption key such that $PK_x \cdot SK_x = 1 \ (\text{mod}\ \phi(N_x))$. Before the merchant delegates its received electronic cash to the acquirer, the merchant randomly selects a blinding factor α and computes the following parameters:

$Mcash = (cash \,\|\, ID_{Merchant}) \oplus \alpha$,

$C_{R1} = (Mcash)^{PK_{Acquirer}} \bmod N_{Acquirer}$,

$C_{R2} = (ID_{Acquirer} \,\|\, Mcash \,\|\, \alpha)^{PK_{Issuer}} \bmod N_{Issuer}$.

The merchant then sends (C_{R1}, C_{R2}) to the acquirer.

Step2: After receiving the above message, the acquirer computes

$(C_{R1})^{SK_{Acquirer}} \bmod N_{Acquirer} = Mcash \bmod N_{Acquirer}$,

$Sig_{Acquirer} = (Mcash)^{SK_{Acquirer}} \bmod N_{Acquirer}$.

He then sends his signature $Sig_{Acquirer}$ back to the merchant as the non-repudiation proof and forwards C_{R2} to the issuer for the issuer to deposit money to his account.

Step3: The issuer uses its secret key SK_{Issuer} to decrypt C_{R2}, obtaining α , $ID_{Acquirer}$, $cash$, and $ID_{Merchant}$. He then computes $(C_{R2})^{SK_{Issuer}} \bmod N_{Issuer} = ID_{Acquirer} \,\|\, Mcash \,\|\, \alpha$ and $Mcash \oplus \alpha = (cash \,\|\, ID_{Merchant})$. The issuer then verifies the *cash* to see if it is valid. If it is, he records these parameters *cash* , $ID_{Acquirer}$, and $ID_{Merchant}$ for the necessity of double deposit checking.

Step4: Finally, the issuer transfers the corresponding funds to the designated acquirer $ID_{Acquirer}$ for $ID_{Acquirer}$ to transfer it to $ID_{Merchant}$'s account.

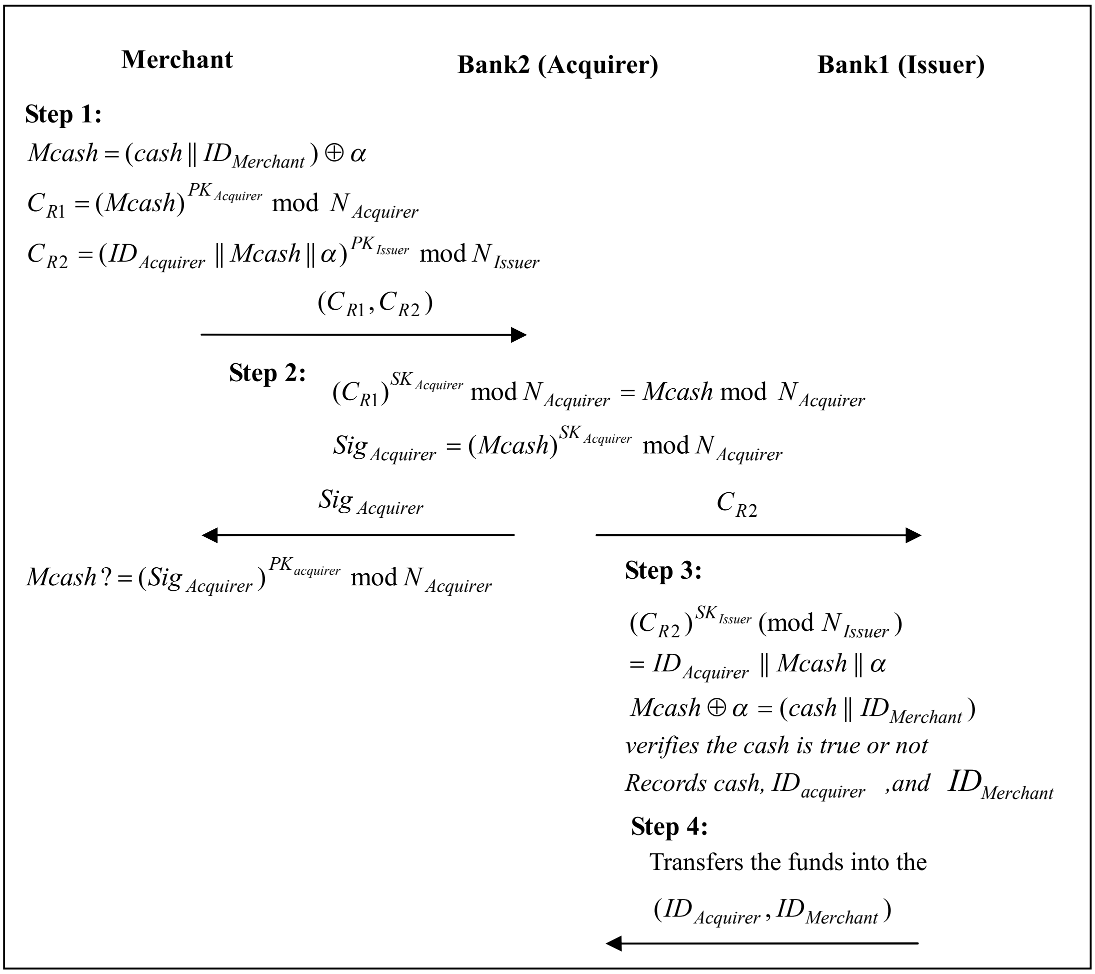

Merchant
Bank2 (Acquirer)
Bank1 (Issuer)
Step 1:
$Mcash = (cash \| ID_{Merchant}) \oplus \alpha$
$C_{R1} = (Mcash)^{PK_{Acquirer}} \bmod N_{Acquirer}$
$C_{R2} = (ID_{Acquirer} \| Mcash \| \alpha)^{PK_{Issuer}} \bmod N_{Issuer}$
(C_{R1}, C_{R2})
Step 2:
$(C_{R1})^{SK_{Acquirer}} \bmod N_{Acquirer} = Mcash \bmod N_{Acquirer}$
$Sig_{Acquirer} = (Mcash)^{SK_{Acquirer}} \bmod N_{Acquirer}$
$Sig_{Acquirer}$
C_{R2}
$Mcash ?= (Sig_{Acquirer})^{PK_{acquirer}} \bmod N_{Acquirer}$
Step 3:
$(C_{R2})^{SK_{Issuer}} (\bmod N_{Issuer})$
$= ID_{Acquirer} \| Mcash \| \alpha$
$Mcash \oplus \alpha = (cash \| ID_{Merchant})$
verifies the cash is true or not
Records cash, $ID_{acquirer}$,and $ID_{Merchant}$
Step 4:
Transfers the funds into the
$(ID_{Acquirer}, ID_{Merchant})$

Fig. 6. Chen et al.'s RSA based deposit delegation scheme.

(B) Attack on Chen et al.'s scheme

We found that, in Chen et al.'s scheme, if an adversary intercepts *cash* from the withdrawal or the payment phase, he can masquerade as a merchant, $Merchant'$, to make the system confused. We illustrate our attack as follows (also shown in Fig.7):

Step 1: Assume that an adversary intercepts *cash* from the withdrawal or the payment phase. He can compute

$Mcash' = (cash \| ID_{Merchant'}) \oplus \alpha'$,

$C'_{R1} = (Mcash')^{PK_{Acquirer}} \bmod N_{Acquirer}$, and

$C'_{R2} = (ID_{Acquirer} \| Mcash' \| \alpha')^{PK_{Issuer}} \bmod N_{Issuer}$, where α' is a randomly blinding factor chosen by $Merchant'$. He then sends (C'_{R1}, C'_{R2}) to the acquirer.

Step 2: After receiving (C'_{R1}, C'_{R2}), the acquirer computes

$(C'_{R1})^{SK\,Acquirer} \bmod N_{Acquirer} = Mcash' \bmod N_{Acquirer}$ and

$Sig'_{Acquirer} = (Mcash')^{SK\,Acquirer} \bmod N_{Acquirer}$.

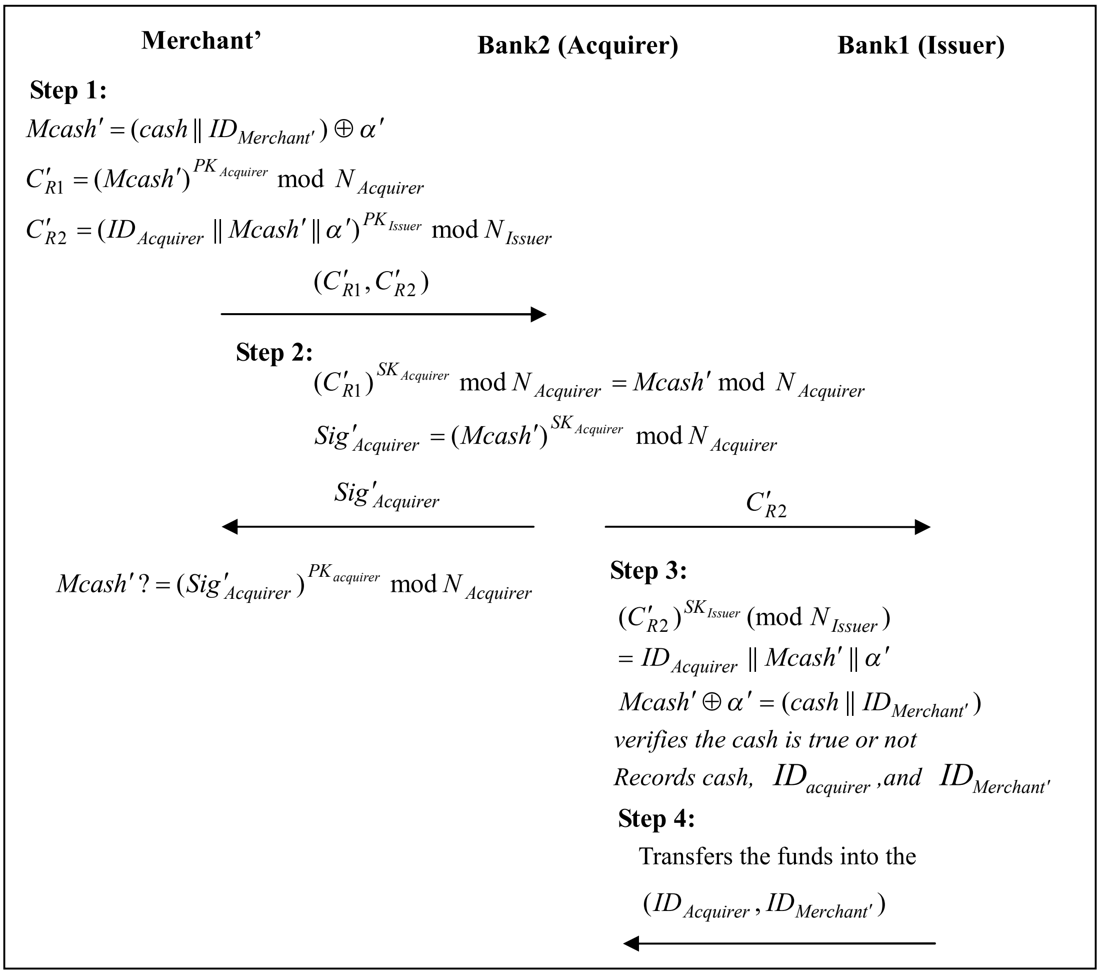
He then sends the signature $Sig'_{Acquirer}$ back to the merchant as the non-repudiation proof and sends C'_{R2} to the issuer for depositing money to his account.

Step 3: After receiving C'_{R2}, the issuer can obtain *cash*, $ID_{Merchant'}$, and $ID_{Acquirer}$ by computing

$(C'_{R2})^{SK\,Issuer} (\bmod N_{Issuer}) = ID_{Acquirer} \parallel Mcash' \parallel \alpha'$ and

$Mcash' \oplus \alpha' = (cash \parallel ID_{Merchant'})$.

Step4: Finally, the issuer transfers the corresponding funds to the designated acquirer $ID_{Acquirer}$ for $ID_{Acquirer}$ to transfer it to the adversary $ID_{Merchant'}$'s account.



Merchant'
Bank2 (Acquirer)
Bank1 (Issuer)
Step 1:
$Mcash' = (cash \parallel ID_{Merchant'}) \oplus \alpha'$
$C'_{R1} = (Mcash')^{PK_{Acquirer}} \bmod N_{Acquirer}$
$C'_{R2} = (ID_{Acquirer} \parallel Mcash' \parallel \alpha')^{PK_{Issuer}} \bmod N_{Issuer}$
(C'_{R1}, C'_{R2})
Step 2:
$(C'_{R1})^{SK_{Acquirer}} \bmod N_{Acquirer} = Mcash' \bmod N_{Acquirer}$
$Sig'_{Acquirer} = (Mcash')^{SK_{Acquirer}} \bmod N_{Acquirer}$
$Sig'_{Acquirer}$
C'_{R2}
$Mcash' ?= (Sig'_{Acquirer})^{PK_{acquirer}} \bmod N_{Acquirer}$
Step 3:
$(C'_{R2})^{SK_{Issuer}} (\bmod N_{Issuer})$
$= ID_{Acquirer} \parallel Mcash' \parallel \alpha'$
$Mcash' \oplus \alpha' = (cash \parallel ID_{Merchant'})$
verifies the cash is true or not
Records cash, $ID_{acquirer}$, and $ID_{Merchant'}$
Step 4:
Transfers the funds into the
$(ID_{Acquirer}, ID_{Merchant'})$

Fig. 7. The attack on Chen et al.'s RSA based deposit delegation scheme.

It is obvious that the issuer can verify the cash successfully as indicated in step 3 of Fig.7. The issuer then transfers the money from the customer account to the bank account of *Merchant'* in the acquirer. But when the true merchant, the real cash owner, wants to send the cash to the acquirer, the issuer will find that the cash is double-spending. However, it's difficult for the issuer to find out which merchant should be the real owner of the cash. Because $Merchant'$ can also provide the $Sig'_{Acquirer}$ to prove that he is the legal owner of the cash. So, an adversary can easily make the system confused by masquerading as any other merchant.

4. Our proposed scheme

Due to that most of the proposed e-cash protocols are either insecure or incomplete, in this section, we present a novel secure and complete e-cash scheme. Our protocol consists of six phases. They are: (1)setup phase, (2)register phase, (3)withdrawal phase, (4)payment phase, (5)deposit phase, and (6)tracing phase.

4.1 Setup phase

Let G_1 be a cyclic additive group, generated by the generator P , whose order is a prime q , G_2 be a cyclic multiplicative group of the same order. A bilinear pairing is a map $e: G_1 \times G_1 \to G_2$. We define three hash functions, H_1, H_2, and H_3, to be $H_1: \{0,1\}^* \to G_1$, $H_2: G_2 \to \{0,1\}^n$, and $H_3: Z_q \times G_1 \to Z_q$ respectively. The public key generator (PKG) chooses a random number $s \in Z_q^*$ and sets $P_{pub} = sP$. PKG keeps s as his private key and publishes the system parameters set $\{G_1, G_2, e, q, P, P_{pub}, H_1, H_2, H_3\}$.

Customer C submits his identity ID_C to PKG over a secure channel. PKG computes C's public key as $Q_C = H_1(ID_C)$ and the matching private key as $S_C = s \cdot Q_C$. Similarly, PKG also generates the public/private key pairs (Q_T, S_T), (Q_B, S_B), and (Q_M, S_M) for the Trusted Third Party (TTP), bank, and merchant respectively.

4.2 Register phase

After the customer obtains a key pair (Q_C, S_C) from PKG, he has to register to TTP to obtain a certificate *u*. Our scheme permits a customer to register over the internet without the necessity of face-to-face or through secure channel registration. This can greatly reduce our system's cost either in traffic overhead or in setting the secure channel. We describe the details as follows and also illustrate it in Fig.8.

Step1: The customer C chooses a random number $a \in Z_q^*$ and computes the session key shared with TTP as $K_{CT} = H_2(e(S_C, aQ_T))$. He then sends $E_{K_{CT}}(Auth_{CT})$, aQ_C to TTP, where $Auth_{CT} = \{ID_C, ID_T, T_s, a\}$, T_s is a timestamp, and $E_{K_{CT}}(Auth_{CT})$ is the encryption of $Auth_{CT}$ by session key K_{CT}.

Step2: On receiving the messages from C, TTP will run the following steps:

(1) computes $K_{CT} = H_2(e(S_T, aQ_C))$ and decrypts $E_{K_{CT}}(Auth_{CT})$ to obtain $Auth_{CT} = \{ID_C, ID_T, T_s, a\}$.

(2) checks to see if $T' - T_s$ is less than ΔT (Assume that TTP's current system timestamp is T'), where ΔT is the tolerant time for transmission delay. If it isn't, the request is rejected.

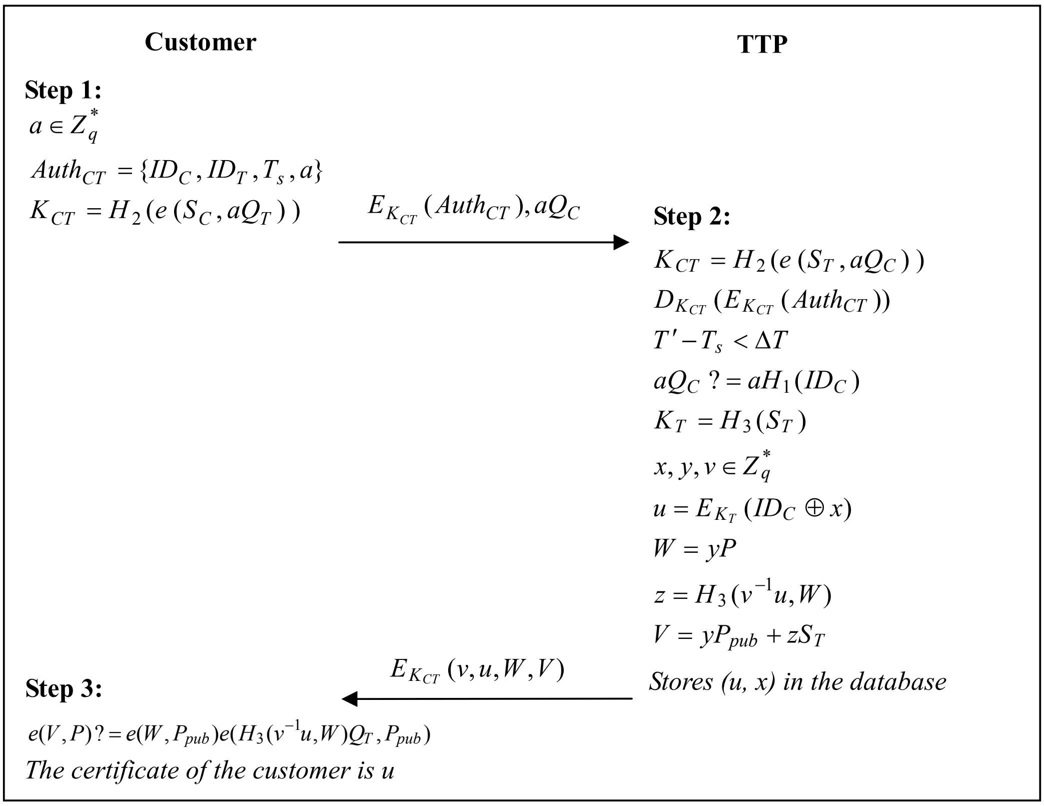
(3) uses a and ID_C in $Auth_{CT}$ to compute $aH_1(ID_C)$ and checks to see if this computed value is equal to the received aQ_C. If it isn't, the request is rejected.

(4) chooses three random numbers $x, y, v \in Z_q^*$ and computes $u = E_{K_T}(ID_C \oplus x)$, where $K_T = H_3(S_T)$ is only known by TTP.

(5) computes $W = yP$, $z = H_3(v^{-1}u, W)$, $V = yP_{pub} + zS_T$, and sends $E_{K_{CT}}(v, u, W, V)$ to C.

(6) stores the entry (u, x) in the database.

Step3: After receiving the message from TTP, C verifies if $e(V, P)$ is equal to $e(W, P_{pub})e(H_3(v^{-1}u, W)Q_T, P_{pub})$. If it is, the certificate of C is u. Otherwise, C rejects the message.



Customer **TTP**

Step 1:

$a \in Z_q^*$

$Auth_{CT} = \{ID_C, ID_T, T_s, a\}$

$K_{CT} = H_2(e(S_C, aQ_T))$

$\xrightarrow{E_{K_{CT}}(Auth_{CT}), aQ_C}$

Step 2:

$K_{CT} = H_2(e(S_T, aQ_C))$

$D_{K_{CT}}(E_{K_{CT}}(Auth_{CT}))$

$T' - T_s < \Delta T$

$aQ_C \ ?= aH_1(ID_C)$

$K_T = H_3(S_T)$

$x, y, v \in Z_q^*$

$u = E_{K_T}(ID_C \oplus x)$

$W = yP$

$z = H_3(v^{-1}u, W)$

$V = yP_{pub} + zS_T$

Stores (u, x) in the database

$\xleftarrow{E_{K_{CT}}(v, u, W, V)}$

Step 3:

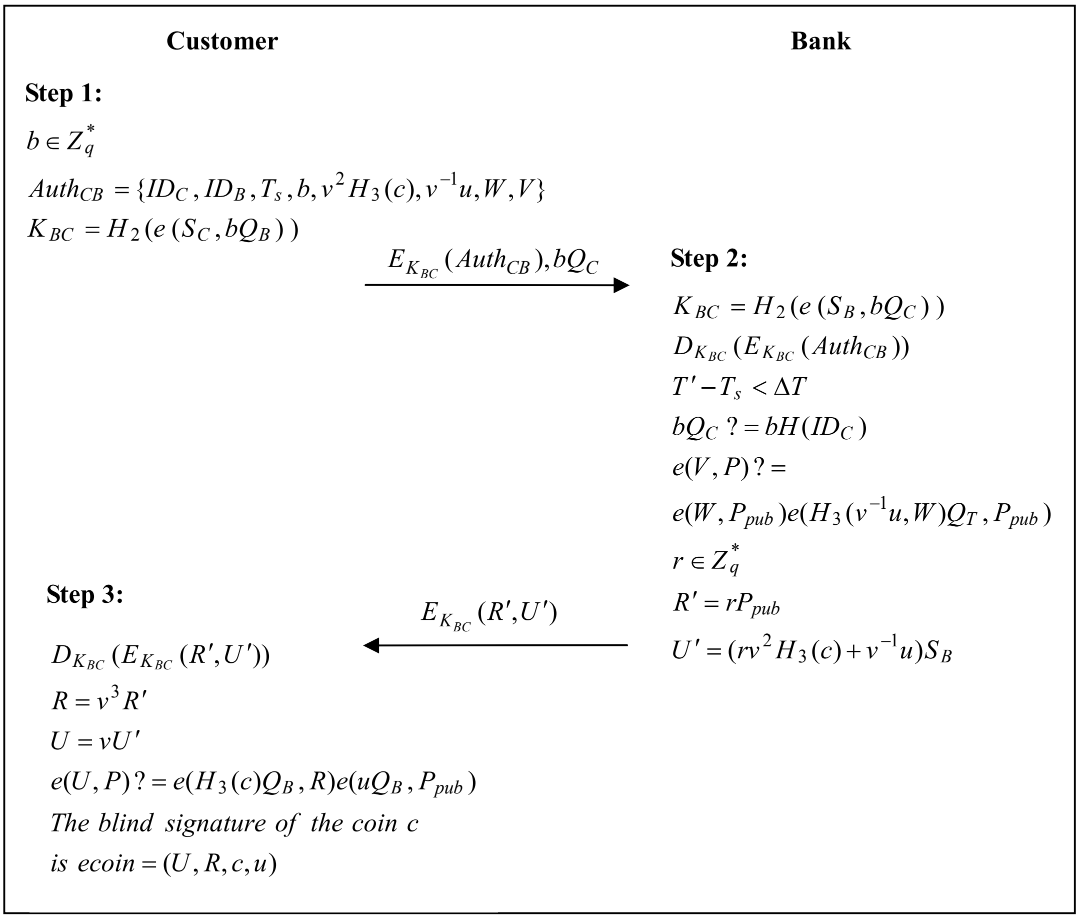
$e(V, P) ?= e(W, P_{pub})e(H_3(v^{-1}u, W)Q_T, P_{pub})$

The certificate of the customer is u

Fig. 8. The register phase of our proposed scheme.

4.3 Withdrawal phase

In this phase, customer C employs a blind signature technique to withdraw an e-coin from bank B. We describe the details as follows and also illustrate it in Fig.9.



Customer **Bank**

Step 1:

$b \in Z_q^*$

$Auth_{CB} = \{ID_C, ID_B, T_s, b, v^2 H_3(c), v^{-1}u, W, V\}$

$K_{BC} = H_2(e(S_C, bQ_B))$

$\xrightarrow{E_{K_{BC}}(Auth_{CB}), bQ_C}$

Step 2:

$K_{BC} = H_2(e(S_B, bQ_C))$

$D_{K_{BC}}(E_{K_{BC}}(Auth_{CB}))$

$T' - T_s < \Delta T$

$bQ_C \; ?= bH(ID_C)$

$e(V, P) \; ?=$

$e(W, P_{pub}) e(H_3(v^{-1}u, W) Q_T, P_{pub})$

$r \in Z_q^*$

$R' = rP_{pub}$

$U' = (rv^2 H_3(c) + v^{-1}u) S_B$

$\xleftarrow{E_{K_{BC}}(R', U')}$

Step 3:

$D_{K_{BC}}(E_{K_{BC}}(R', U'))$

$R = v^3 R'$

$U = vU'$

$e(U, P) \; ?= e(H_3(c) Q_B, R) e(uQ_B, P_{pub})$

The blind signature of the coin c

is $ecoin = (U, R, c, u)$

Fig. 9. The withdrawal phase of our proposed scheme.

Step1: Customer C chooses a random number $b \in Z_q^*$ and computes the session key $K_{BC} = H_2(e(S_C, bQ_B))$. Let $Auth_{CB} = \{ID_C, ID_B, T_s, b, v^2 H_3(c), v^{-1}u, W, V\}$. The customer sends $E_{K_{BC}}(Auth_{CB})$, bQ_C to bank B, where T_s is a timestamp, *c* is a serial number of a coin, and $Auth_{CB}$ is encrypted by K_{BC}.

Step2: After receiving the messages from C, B will run the following:

(1) computes $K_{BC} = H_2(e(S_C, bQ_B))$ and decrypts $E_{K_{BC}}(Auth_{CB})$ to obtain $Auth_{CB} = \{ID_C, ID_B, T, b, v^2 H_3(c), v^{-1}u, W, V\}$.

(2) generates a timestamp T' and checks to see if $T' - T_s$ is less than ΔT,where ΔT is the tolerant time for transmission delay. If it isn't, the request is rejected.

(3) takes b and ID_C from $Auth_{CB}$ and computes $bH_1(ID_C)$ to check if it is equal to bQ_C. If it isn't, he rejects the request.

(4) verifies if $e(V, P)$ is equal to $e(W, P_{pub}) e(H_3(v^{-1}u, W) Q_T, P_{pub})$. If it is, the data $(v^{-1}u, W, V)$ in $Auth_{CB}$ is actually from TTP. Else, he rejects the request.

(5) chooses a random numbers $r \in Z_q^*$ and computes $R' = rP_{pub}$, $U' = (rv^2 H_3(c) + uv^{-1}) S_B$.

(6) sends $E_{K_{BC}}(R', U')$ to C and deducts money, whose amount is negotiated in advance, from C's account.

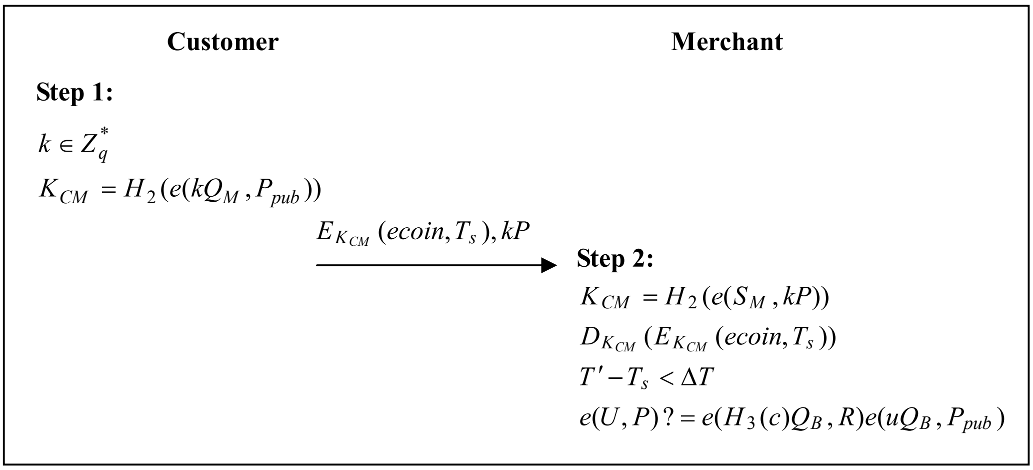
Step3: After receiving the message from B, C decrypts $E_{K_{BC}}(R', U')$ to get R', U' and computes $R = v^3 R'$, $U = vU'$. He then verifies if $e(U, P)$ is equal to $e(H_3(c)Q_B, R)e(uQ_B, P_{pub})$. If so, the blind signature of the coin c is $ecoin = (U, R, c, u)$. Otherwise, C rejects the message.

4.4 Payment phase

In this phase, we assume that there is a customer C who wants to use his ecoin for shopping in merchant M. We describe the payment phase of our protocol as follows and also illustrate it as Fig.10.

Step1: C firstly chooses a random number $k \in Z_q^*$ and computes the session key $K_{CM} = H_2(e(kQ_M, P_{pub}))$. He encrypts $(ecoin, T_s)$ with K_{CM} and computes kP. Then C sends $E_{K_{CM}}(ecoin, T_s),\ kP$ to merchant M, where T_s is C's current timestamp.

Step2: After receiving the message from C, M computes $K_{CM} = H_2(e(kQ_M, P_{pub}))$ to decrypt $E_{K_{CM}}(ecoin, T_s)$, obtaining $(ecoin, T_s)$. M checks to see if both $T' - T_s$ is less than ΔT and $e(U, P)$ is equal to $e(H_3(c)Q_B, R)e(uQ_B, P_{pub})$, where T' is M's system timestamp and ΔT is the tolerant time for transmission delay. If both hold, M accepts the ecoin and sends goods to C. Otherwise, M rejects it.



Customer **Merchant**

Step 1:

$k \in Z_q^*$

$K_{CM} = H_2(e(kQ_M, P_{pub}))$

$\xrightarrow{E_{K_{CM}}(ecoin, T_s), kP}$

Step 2:

$K_{CM} = H_2(e(S_M, kP))$

$D_{K_{CM}}(E_{K_{CM}}(ecoin, T_s))$

$T' - T_s < \Delta T$

$e(U, P) ? = e(H_3(c)Q_B, R)e(uQ_B, P_{pub})$

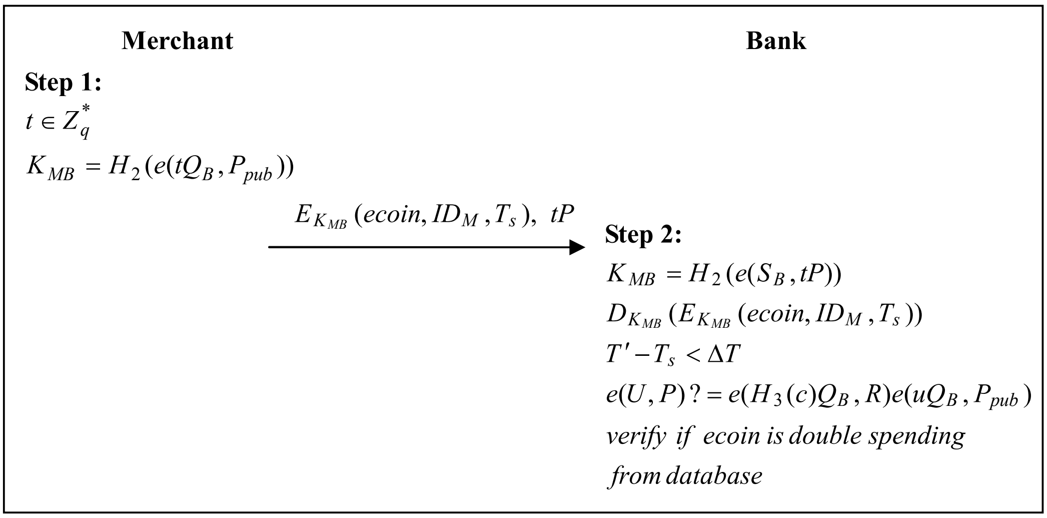
Fig. 10. The payment phase of our proposed scheme.

4.5 Deposit phase

In this phase, assume that merchant M wants to deposit an ecoin to bank B. B will add the ecoin to the merchant's account. We describe the details as follows and also illustrate it in Fig.11.

Step1: Merchant M firstly chooses a random number $t \in Z_q^*$ and computes the session key $K_{MB} = H_2(e(tQ_B, P_{pub}))$. He encrypts $(ecoin, ID_M, T_s)$ with K_{MB}, computes tP, and then sends $E_{K_{MB}}(ecoin, ID_M, T_s),\ tP$ to bank B, where T_s is M's system timestamp.

Step2: After receiving the message from M, B computes $K_{MB} = H_2(e(S_B, tP))$ to decrypt $E_{K_{MB}}(ecoin, ID_M, T_s)$, obtaining $(ecoin, ID_M, T_s)$. B checks to see if $T' - T_s$ is less than ΔT, where T' is B's current system timestamp and ΔT is the tolerant time for transmission delay. If it isn't, the request is rejected. Else, B then verifies if $e(U, P)$ is equal to $e(H_3(c)Q_B, R)e(uQ_B, P_{pub})$. If it is, B checks the ecoin to see if it is double-spending. If so, B requests TTP to reveal the identity of the dishonest customer. Else, he accepts the ecoin and adds it to M's bank account.



Merchant **Bank**

Step 1:

$t \in Z_q^*$

$K_{MB} = H_2(e(tQ_B, P_{pub}))$

$\xrightarrow{E_{K_{MB}}(ecoin, ID_M, T_s),\ tP}$

Step 2:

$K_{MB} = H_2(e(S_B, tP))$

$D_{K_{MB}}(E_{K_{MB}}(ecoin, ID_M, T_s))$

$T' - T_s < \Delta T$

$e(U, P) ? = e(H_3(c)Q_B, R)e(uQ_B, P_{pub})$

verify if ecoin is double spending from database

Fig. 11. The deposit phase of our proposed scheme.

4.6 Tracing phase

In our scheme, if a customer ID_C uses the same ecoin twice, the bank can find out the illegal transaction by checking the paid coins stored in the database. The bank then calls TTP for revealing the identity of the dishonest customer using the following equations:

$K_T = H_3(S_T)$

$ID_C = D_{K_T}(u) \oplus x$

5. Security analysis:

In this section, we will show that our scheme is secure by examining the following required properties and demonstrate that our scheme satisfies the five security properties (stated in Section 2.1) as the following.

5.1. Mutual authentication

We adopt the concept of an ID-pairing-based cryptosystem in our scheme for the advantage that it can establish a session key between two communicating parties without sharing any secret in advance. It not only can reduce the number of communicational passes but also can achieve implicit mutual authentication, if the two communicating parties can use the established session key successfully. For example, in the withdrawal phase, the customer and the bank each use his own private key and the opposite's public key to compute the session keys, $K_{CB} = H_2(e(S_C, bQ_B))$ and $K_{BC}(= K_{CB}) = H_2(e(S_B, bQ_C))$, respectively. They can communicate secretly by using this session key K_{BC} without doing any key exchange in advance. If an adversary wants to masquerade as the customer to send an encrypted message to the bank, the bank will reject it since he can't decrypt the ciphertext to obtain any meaningful information. Up to date, only our scheme provides a secure mutual authentication function in the withdrawal phase. The other proposed protocols [5-10] don't have this function.

5.2. Verifiability

We demonstrate the verification processes of our scheme by using the following two equations, (1) and (2).

$$\begin{aligned}
&e(V,P)\\
&= e(yP_{pub} + zS_T, P)\\
&= e(yP_{pub}, P)e(zS_T, P)\\
&= e(yP, P_{pub})e(zQ_T, P_{pub})\\
&= e(W, P_{pub})e(H_3(v^{-1}u, W)Q_T, P_{pub}) \ldots\ldots\ldots\ldots\ldots\ldots\ldots(1)
\end{aligned}$$

and

$$\begin{aligned}
&e(U,P)\\
&= e(vU', P)\\
&= e(v(rv^2H_3(c) + uv^{-1})S_B, P)\\
&= e(rv^3H_3(c) + u)S_B, P)\\
&= e(rv^3H_3(c)S_B, P)e(uS_B, P)\\
&= e(rv^3H_3(c)Q_B, P_{pub})e(uQ_B, P_{pub})\\
&= e(H_3(c)Q_B, rv^3P_{pub})e(uQ_B, P_{pub})\\
&= e(H_3(c)Q_B, R)e(uQ_B, P_{pub}) \ldots\ldots\ldots\ldots\ldots\ldots\ldots(2)
\end{aligned}$$

Equation (1) is used for two kinds of verifications: ① the customer verifies whether the other party is the real TTP and whether the transmitted data $(v^{-1}u, W, V)$ from TTP is calculated in a specified way since $z = H_3(v^{-1}u, W)$ (as shown in Fig. 8), and ② the bank verifies whether or not the other party is the real customer as claimed since he needs to compute K_{BC} to obtain $Auth_{CB}$ (as shown in Fig. 9) and whether or not the data $(v^{-1}u, W, V)$ in $Auth_{CB}$ from the customer is actually from TTP. This is implied by $H_3(v^{-1}u, W)Q_T$ in the equation.

Equation (2) is used for verifying whether or not the ecoin (U, R, c, u) is valid and from the claimed bank as implied in the equation. (as shown in Fig. 9, 10 and 11).

5.3. Anonymity

This is a very important security property especially for keeping the privacy of a customer's identity in an electronic cash system. To attain this purpose in our scheme, we adopt the blind signature technique. In the following, we describe why our scheme possesses this anonymous property by using two reasons.

Reason 1:

In the withdrawal phase, when customer C wants to withdraw an ecoin, he must provide both his identity ID_C and his randomized certificate $v^{-1}u$ to the bank to deduct money from his account. Although the bank knows the identity of the customer, he can't get the customer's certificate since it has been randomized by v^{-1}. Besides, the customer blinds R', U' by computing $R = v^3 R'$, $U = vU'$. This makes the bank unable to know the identity of the ecoin (U, R, c, u) owner. So, in the deposit phase, when the bank receives the ecoin (U, R, c, u) from a merchant (as shown in Fig.11), without the knowledge of *v*, he doesn't know the identity of the ecoin owner by comparing u with the stored $v^{-1}u$ which corresponds to its owner ID_C in $Auth_{CB}$ (as shown in Fig. 9). Similarly, the serial number *c* of the ecoin is embedded in value $v^2 H_3(c)$ in which *c* is first protected by a hash function H_3, then randomized by a random number v^2. It also corresponds to the identity ID_C of its owner. Therefore, in the deposit phase, even if the bank, receiving the ecoin (U, R, c, u) from the merchant, can compute $H_3(c)$, he still doesn't know the identity of the ecoin owner by comparing $H_3(c)$ with the stored $v^2 H_3(c)$.

Furthermore, our scheme can prevent the bank from knowing the identity of the ecoin owner by some mathematic operations. For example, if $e(U', R') = e(U, R)$, the bank can find the ecoin owner by this equation. Although in our scheme, after receiving the ecoin (U, R, c, u) from the merchant, the bank can know $e(U, R) = e(vU', v^3 R')$ $= e(U', R')^{v^4}$. However, without the knowledge of *v*, he can't get the customer's identity by using the point $e(U', R')^{v^4}$ in G_2.

Reason 2:

In the payment phase, when the merchant receives the ecoin (U, R, c, u) from a customer, he can't know the identity of the ecoin owner by the certificate u since $u = E_{K_T}(ID_C \oplus x)$. That is to say, certificate u is the result of the customer's identity *xor*-ed by a random *x* and then encrypted by K_T which is only known by TTP and PKG. Hence, anyone who gets the certificate u can't obtain any useful information about the identity of the customer.

5.4. Unforgeability

In our scheme, an adversary may try to fake an ecoin under the following two possible cases. In each case, we show why our scheme possesses this unforgeable property.

Case 1.

In the payment phase, the merchant can get the customer's certificate u sent in $E_{K_{CM}}(ecoin,\ T_s)$ as shown in Fig. 10. If the merchant gives u to an adversary, we must show whether or not the adversary can successfully embed the certificate u in $Auth_{AB}$ (see Fig. 9) to masquerade as ID_C for withdrawing an ecoin and pass the bank B's verification. In the following, we show why this attack fails.

Step1: The adversary may get a certificate u of customer C from a compromised merchant in which C had ever consumed.

Step2: The adversary randomly picks $v' \in Z_q$, $W' \in G_1$ and computes

$$\begin{aligned} & e(W', P_{pub})e(H_3(v'^{-1}u, W')Q_T, P_{pub}) \\ &= e(W' + H_3(v'^{-1}u, W')Q_T, P_{pub}) \\ &= e(s(W' + H_3(v'^{-1}u, W')Q_T), P) \\ &= e(V', P). \end{aligned}$$

If the adversary can find $V' = s(W' + H_3(v'^{-1}u, W')Q_T)$ to pass the bank's verification (as shown in Fig. 9 and indicated in equation (1)), the attack succeeds. However, without the knowledge of s, the adversary is doomed to fail since it is an ECDLP problem.

Case 2.

Similarly, in the payment phase, the merchant can get the certificate u of the customer and the serial number c as well. If the merchant gives them to an adversary, we must know if the adversary can use them to fake an ecoin satisfying the verification of equation (2) (also shown in Fig. 9) without communicating with the bank. If so, the adversary can use the faked ecoin for shopping and pass the verifications of: (1) merchant verifies the ecoin from customer, and (2) bank verifies the ecoin from the merchant as shown in Fig.10 and 11 respectively. We describe this attack using the following steps and show why it can not succeed.

Step1: The adversary gets a certificate u and the serial number c of an ecoin from a compromised merchant in which a legal customer consumes.

Step2: The adversary randomly chooses $n \in Z_q$ and lets $R' = nP$.

Step3: He computes

$$\begin{aligned} & e(H_3(c)Q_B, R')e(uQ_B, P_{pub}) \\ &= e(H_3(c)Q_B, nP)e(suQ_B, P) \end{aligned}$$

$$= e(nH_3(c)Q_B + usQ_B, P)$$
$$= e(U', P)$$

The adversary must let $U' = (nH_3(c) + us)Q_B$ to pass the verifications as shown in Fig.9, 10, and 11 respectively and indicated in equation (2). However, without the knowledge of PKG's secret key s, he is doomed to fail since it is an ECDLP problem. Not to mention, the adversary doesn't have the session keys K_{BC}, K_{CM}, and K_{MB} in the three scenarios.

5.5. Traceability

If any customer uses the same coin (U, R, c, u) twice, then with the help of TTP, the bank can find out this illegal transaction by checking the double-spent ecoin stored in the database because TTP can easily reveal the identity of the customer by using the following equations.

$$K_T = H_3(S_T)$$
$$ID_C = D_{K_T}(u) \oplus x$$

Since S_T is TTP's private key and u is a fixed value corresponding to each customer's identity, they can be pre-computed. Therefore, our scheme only needs one xor operation to reveal the identity of the illegal customer. Hence, the double spending tracing is very efficient in our scheme.

6. Comparisons

In this section, we compare our protocol with previously proposed schemes [5-10] by using the above-mentioned security properties. After comparing with those schemes, we can see that our scheme not only can provide mutual authentication, ecoin verification, and customer's identity protection but also can resist against double spending and ecoin forging. We summarize the comparisons of each property in Table 1 and list the reasons for why the corresponding scheme in the table can't attain some security features in Appendix A.

Table 1. The security comparisons between our proposed scheme and other schemes

	Juang et al.'s [5]	Juang's [6]	Chen's [7]	H.Wang et al.'s [8]	Popescu et al.'s [9]	S.Wang et al.'s [10]	Ours
C_0	On	Off	-	Off	Off	Off	Off
C_1	No	No	-	No	No	Yes	Yes
C_2	Yes	No	-	No	No	Yes	Yes
C_3	Yes	Yes	-	Yes	Yes	Yes	Yes
C_4	Yes	Yes	Yes	Yes	-	Yes	Yes
C_5	Yes	No	-	No	Yes	Yes	Yes
C_6	No	No	No	No	Yes	No	Yes
C_7	-	No	No	Yes	Yes	Yes	Yes

C_0: “On” represents On-line type; “Off” represents Off-line type

C_1: Mutual authentication in withdrawal phase (checking the identity to each other before communication)

C_2: The customer verifies the ecoin received from the bank

C_3: The merchant verifies the ecoin received from the customer

C_4: The bank verifies the ecoin received from the merchant

C_5: Anonymity (protecting the identity of the customer)

C_6: Unforgeability (preventing from forging a valid ecoin)

C_7: Traceability (revealing the identity of a party who uses ecoin twice in off-line type)

- : For an incomplete protocol, it lacks the consideration in the corresponding feature

From Table 1, we can see that our protocol not only is the most secure among all of the proposed protocols but also possesses complete functions which a secure electronic cash requires.

7. Conclusion

In this paper, we have reviewed and shown the attacks on schemes [5, 6, 7]. We also propose a secure ID-based e-cash scheme from parings. After analysis, we conclude that our scheme can satisfy the properties of mutual authentication, anonymity, unforgeability, traceability, and double-spending protection. After comparisons, we can see that our scheme is not only more secure but also more complete than all of the proposed schemes for electronic cash systems nowadays.

Appendix A

(1) Why are they not “mutual authentication in withdrawal phase” in [5, 6, 8, 9] ?

From schemes [5, 6, 8, 9], we can see that they don’t authenticate each other in the withdrawal phase. That’s to say, the bank may give a valid ecoin to an illegal customer. Therefore, their schemes are not secure.

(2) Why are they not “The customer verifies the ecoin received from the bank” in [6, 8, 9] ?

From schemes [6, 8, 9], we can see that the customer doesn’t verify the validity of the ecoin received from the bank in the withdrawal phase. Hence, the customer may therefore obtain a forged ecoin from an adversary.

(3) Why are they not “anonymity” in [6, 8] ?

In [6], we have demonstrated this in Section 3.2.

As for [8], in the withdrawal phase, the customer withdraws an ecoin C from the bank without employing a blind signature technology. Hence, after the deposit phase, the bank can reveal the identity of the ecoin owner by searching for the previously spent ecoins recorded in his database.

(4) Why are they not "unforgeability" in [5, 6, 7, 8, 10] ?

In [5, 6, 7], we have demonstrated this in Section 3.1, 3.2, and 3.3 respectively.

As [8], in the withdrawal phase, a user chooses $a_i, c_i, 1 \le i \le k,$ and computes $x_i = g^{a_i}$, $y_i = g_1^{a_i \oplus (I + c_i)}$, $B_i = H(x_i, y_i)$, where $I = g_1^{u_1}$ is the user's account. He then sends all B_i to the bank. Finally, the bank computes and sends ecoin $C = \prod_{1 \le i \le k/2} B_i$ to the user.

In payment phase, when the user sends an ecoin C to the shop, an adversary can intercept it and respond with a binary string $\{z_1, z_2, ..., z_{k/2}\} = \{0, 1, ..., 0\}$ to the user. After receiving the binary string, the user responds as follows, for all $1 \le i \le k/2$: If $z_i = 1$, he sends (a_i, y_i) ; else, he sends $(x_i, a_i \oplus (I \| c_i), c_i)$ to the shop. At this time, the adversary simply intercepts it to let the payment fail. He then stores these intercepted data in his storage. If the user finds it fails and wants to shop using the same C later, he must send C to a shop (either the original or another). Once seeing the same C transmitted, the adversary intercepts it and responds with a complementary binary string $\{z'_1, z'_2, ..., z'_{k/2}\} = \{1, 0, ..., 1\}$ to the user again. After receiving it, the user sends (a_i, y_i) or $(x_i, a_i \oplus (I \| c_i), c_i)$, for all $1 \le i \le k/2$ to the shop according to whether the bit is set or not. The adversary intercepts and stores them in his storage. Hence, the adversary can use the two sets of stored data to respond any shop's request to pass the shop's verification $C = \prod_{1 \le i \le k/2} B_i$. We demonstrate the detail as follows:

Step 1: the adversary sends intercepted C to any shop for shopping.

Step 2: the shop chooses a random binary string $\{z_1, z_2, ..., z_{k/2}\}$ and sends to the user. Here, we suppose the binary string is $\{z_1, z_2, ..., z_{k/2}\} = \{0, 1, 0, 1, ..., 0, 1\}$

Step 3: the adversary uses all stored data to respond the shop's random binary string. According to the binary string $\{0, 1, 0, 1, ..., 0, 1\}$, the adversary sends $\{(x_1, a_1 \oplus (I \| c_1), c_1), (a_2, y_2), (x_3, a_3 \oplus (I \| c_3), c_3), (a_4, y_4) .... (a_{k/2}, y_{k/2})\}$ to the shop.

Step 4: After receiving the data from the adversary, the shop computes the following for verification.

$$
\begin{aligned}
& H(x_1, g_1^{a_1 \oplus (I \| c_1)}) \times H(g^{a_2}, y_2) \times .... \times H(g^{a_{k/2}}, y_{k/2}) \\
&= H(x_1, y_1) \times H(x_2, y_2) \times .... \times H(x_{k/2}, y_{k/2}) \\
&= B_1 \times B_2 \times .... \times B_{k/2} \\
&= \prod_{1 \le i \le k/2} B_i \\
&= C
\end{aligned}
$$

Obviously, the adversary can pass the shop's verification successfully. Even worse, the adversary can repeatedly use C at any shop without being found. Since the embedded owner identity in C is I , not the adversary.

In [10], in their authentication protocol of withdrawal protocol, an adversary can pretend to be any legal customer to initialize a withdrawal phase by choosing a random number $x_i' \in Z_q^*$ and sending $x_i'P$ to the bank B_i. Upon receiving $x_i'P$, B_i chooses and sends a random number $\omega \in Z_q^*$ to the adversary. Upon receiving ω, the adversary chooses a random number $k' \in Z_q^*$ and computes $t' = k'P$, $c' = H_1(\omega \| P \| x_i'P \| t')$, $s' = k' - c'x_i'$. He then sends (c', s') to B_i. Upon receiving (c', s') , B_i computes $c'' = H_1(\omega \| P \| x_i'P \| s'P + c'x_i'P)$ and verifies to see if $c' = c''$. Eventually, according to their authentication protocol, the equation is doomed to hold. The adversary therefore proves that $x_i'P$ belongs to him successfully. Hence, the adversary can get a valid E-cash $(m, U', V', W', t_i, r_b\alpha_i P)$ from the bank B_i.

(5) Why are they not "traceability" in [6, 7] ?

In [6], we have demonstrated this in Section 3.2.

As for [7], their scheme just mentions the cash verification in the deposit phase without giving a detailed method. Hence, their scheme can't prevent from double spending. Thereby, it lacks the traceability mechanism.

References

[1] A. Shamir. "Identity-based cryptosystems and signature schemes," *CRYPTO'84*, volume 196 of LNCS, Springer-Verlag, 1984, pp. 47-53.

[2] D. Boneh and M. Franklin. "Identity-based encryption from the Weil pairing," *CRYPTO'01*, volume 2139 of LNCS, Springer-Verlag, 2001, pp. 213-229.

[3] D.Chaum, "Blind signatures for untraceable payments," *Crypt'82*, Plenum, NY, 1983, pp. 199-203

[4] F. Zhang and K. Kim. "Efficient ID-based blind signature and proxy signature from bilinear pairings," *ACISP 2003*, volume 2727 of LNCS, pp. 312-323, 2003.

[5] Wen-Shenq Juang, Horng-Twu Liaw, "A practical anonymous multi-authority e-cash scheme," *Applied Mathematics and Computation*, Vol. 147, No. 3, 16 January 2004, pp. 699-711.

[6] Wen-Shenq Juang, "D-cash: A flexible pre-paid e-cash scheme for date-attachment," *Electronic Commerce Research and Applications*, Vol. 6, No. 1, Spring 2007, pp. 74-80.

[7] Yu Yi Chen, Jinn Ke Jan, Chin-Ling Chen, "A novel proxy deposit protocol for e-cash systems," *Applied Mathematics and Computation* , Vol. 163 , 2005 , pp. 869–877.

[8] H. Wang, Y. Zhang, "Untraceable off-line electronic cash flow in e-commerce," *Computer Science Conference*, 2001. ACSC 2001. Proceedings. 24th Australasian 29 Jan-4 Feb 2001 pp.

191 – 198.

[9] C. Popescu, H. Oros, “An Off-line Electronic Cash System Based on Bilinear Pairings,” *Systems, Signals and Image Processing*, 2007 and 6th EURASIP Conference focused on Speech and Image Processing, Multimedia Communications and Services. 14th International Workshop on 27-30 June 2007, pp. 438 – 440

[10] Shangping Wang, Zhiqiang Chen, Xiaofeng Wang, “A new certificateless electronic cash scheme with multiple banks based on group signatures,” *IEEE International Symposium on Electronic Commerce and Security*, 2008.

[11] Yun Ling, Yiming Xiang, Xun Wang, “RSA-based Secure Electronic Cash Payment system,” *IEEE International Conference*, 2-4 December 2007, pp.1898 – 1902.

[12] Matthieu Gaud, Jacques Traore, “On the Anonymity of Fair Offline E-cash Systems,” *LNCS 2742*, 2003, pp. 34-50.

[13] Alfredo De Santis, Anna Lisa Ferrara, Barbara Masucci, “An attack on a payment scheme,” *Information Sciences*, Vol. 178, No. 5, 1 March 2008, pp. 1418-1421.

[14] Hua Wang, Jinli Cao, Yanchun Zhang, “A flexible payment scheme ant its role-based access control,” *IEEE Trans. Knowl. Data Eng*, 17 March 2005, pp. 425-436.

[15] Lin-Chuan Wu, Yi-Shiung Yeh, “Comment on traceability on RSA-based partially signature with low computation,” *Applied Mathematics and Computation*, Vol. 170, No. 2, 15 November 2005, pp. 1344-1348.

[16] Hsiang-An Wen, Kuo-Chang Lee, Sheng-Yu Hwang, Tzonelih Hwang , “On the traceability on RSA-based partially signature with low computation,” *Applied Mathematics and Computation*, Vol. 162, No. 1, 4 March 2005, pp. 421-425.

[17] Min-Shiang Hwang, Cheng-Chi Lee, Yan-Chi Lai, “Traceability on RSA-based partially signature with low computation,” *Applied Mathematics and Computation*, Vol. 145, No. 2-3, 25 December 2003, pp. 465-468.

[18] Weidong Qiu, “Converting normal DLP-based signatures into blind,” *Applied Mathematics and Computation*, Vol. 170, No. 1, 1 November 2005, pp. 657-665.

[19] Xiaofeng Chen, Fangguo Zhang, Shengli Liu, “ID-based restrictive partially blind signatures and applications,” *Journal of Systems and Software*, Vol. 80, No. 2, February 2007, pp. 164-171.

[20] Xiaoming Hu, Shangteng Huang, “Analysis of ID-based restrictive partially blind signatures and applications,” *Journal of Systems and Software*, Vol. 81, No. 11, November 2008, pp. 1951-1954.

[21] A. Menezes, T. Okamoto, S. Vanston, “Reducing elliptic curve logarithms to logarithms in a finite field,” *IEEE Transaction on Information Theory* , Vol. 39, issue 5, 1993, pp. 1639-1646.

[22] Chun-I Fan, Shi-Yuan Huang, Pei-Hsiu Ho, Chin-Laung Lei, “Fair anonymous rewarding based on electronic cash,” *Journal of Systems and Software*, in press, Corrected Proof, Available online 13 February 2009.

[23] Mafruz Zaman Ashrafi, See Kiong Ng, “Privacy-preserving e-payments using one-time payment details,” *Computer Standards & Interfaces*, Vol. 31, Issue 2, February 2009, pp. 321-328.

[24] M.Franklin, M.Yung, “Secure and efficient off-line electronic digital money,” *Lecture Notes in Computer Science*, Vol. 700, 20th Int. Colloquium on Automata, Languages and Programming (ICALP), Springer Verlag, 1993, pp. 265-276.

[25] Stefan Brands, “Untraceable Off-line Cash in Wallets with Observers,” *CRYPTO’93*, *Lecture Notes in Computer Science*, Vol.773, Springer-Verlag, pp. 302-318.