A Postpaid Micropayment Scheme with Revocable Customers' Anonymity

Shin-Jia Hwang* and Chia-Wei Huang

Department of Computer Science and Information Engineering, Tamkang University, Tamsui, Taiwan 251, R.O.C.

Abstract

A new postpaid micropayment scheme is first proposed to protect customers' anonymity and provides customers' convenience. Due to customers' anonymity, customers can anonymously transact with merchants and obtain the goods/services before being charged. This scheme satisfies three properties of anonymity. First, the customer's identity is protected by a pseudonym. Second, the adversary cannot figure anonymous customers out by tracing their payments. Third, there is a trusted authority to revoke customers' anonymity when some disputes happen. On the other hand, the postpaid function provides customers with the convenience of using the credit to buy goods/services.

Key Words**:** Payment, Micropayment, Anonymity, Anonymous, Blind Signature, Digital Signature, Postpaid, Smart Card

1. Introduction

The micropayment is a special payment scheme which is arising with the development of the electronic commerce. In the electronic commerce, the prices of some goods/services are low, e.g. an image, a piece of (video), or on-line information [1]. Since the values of these payments are small, the computational cost of traditional electronic payment schemes [2–10] is more expensive than the small value payment. Therefore, many researchers have developed some micropayment schemes [11–15] to reduce the computational cost.

To reduce the computational cost, the customer's anonymity at first is not considered in the proposed micropayment schemes. Since the customers in the electronic commerce purchase goods/services through Internet, the personal information and the privacy of the customer are vulnerable to be eavesdropped or stolen. So some researchers start developing anonymous micropayment schemes.

Lin [16] and Tsou [17] proposed their anonymous micropayment schemes in 2004 and 2005, respectively. However, these two schemes do not provide the fair and efficient solutions for customers' anonymity. Lin's scheme does not provide a mechanism to revoke customer's anonymity. Without the anonymity revocation, the customers' privacy is protected completely. But the complete customers' privacy protection damages the profit of merchants and banks because disputes among customers, merchants and banks are unsolvable in the aspect of the law. Tsou's scheme provides a revocation of customer's anonymity but the anonymity recovery is not convenient for customers. To recover the anonymity, the customer has to personally apply a new bank's anonymous account. The personal application is inconvenient and inefficient for customers. To provide anonymity for customers, a micropayment scheme has to also provide an efficient anonymity revocation and efficient anonymity recovery.

Moreover, taking customers' convenience into consideration, customers may be much willing to use the micropayment schemes if they can purchase goods/services before paying. In this situation, postpaid micropayment schemes offer appropriate methods for customers. However, the postpaid scheme is not risk-free.

*Corresponding author. E-mail: sjhwang@mail.tku.edu.tw

As mentioned in Yen's scheme [18], the risk of postpaid schemes is that a customer has inadequate credit to pay for the purchased goods/services. If the postpaid scheme provides customers' anonymity, then a trace mechanism for customers' debts must exist.

Therefore a new postpaid micropayment scheme with revocable anonymity is proposed to settle the above problems. In our scheme, a customer can anonymously purchase goods/services before paying. Then only the trusted bank and the customer's smart card are responsible to maintain customers' anonymity. In the new scheme, the benefit of merchants is protected by the anonymity revocation while the privacy of customers is protected by anonymous and untraceable payments.

Blind signature schemes are first introduced by Chaum, Fiat, and Naor [19,20]. Then Pointcheval and Jacques proposed their blind signature schemes [21] that are proved to be secure. Then the blind signature schemes are used to construct the anonymity in some micropayment schemes.

The next section states the review of PayWord scheme. Section 3 describes the underlying model of our scheme, some cryptographic primitives, and notations used in our scheme. Our scheme is proposed in Section 4. The security analysis of our scheme is given in Section 5. The comparison among our scheme, Lin's scheme, and Tsou's scheme is given in Section 6. The final section is our conclusions.

2. Related Works

In this section, the PayWord scheme [14] developed by Rivest and Shamir is briefly described here. Rivest and Shamir use the one-way property of hash functions to implement a technique which is called the payword chain.

In the PayWord scheme, if the customer wants to spend n paywords in a specific merchant. The customer first chooses the last payword w_n at random. Then he/she generates the payword chain in reverse order by computing $w_i = H(w_{i+1})$ for $i = n-1, n-2, \ldots, 0$, where H is a secure one-way hash function. After generating the payword chain $\{w_0, w_1, \ldots, w_n\}$, the customer signs w_0, concatenated with the certificate of the customer's public key, the merchant's identification, and some additional information to generate the commitment. After the merchant verifies the certificate of the customer's public key and the commitment, the customer is permitted to spend his/her paywords in the merchant.

Assume that the customer's latest spent payword in merchant's database is w_0 and the customer wants to spend w_i this time. The customer sends the merchant w_i, the index i, and the customer's public key. The merchant first retrieves w_0 from its database according to the customer's public key, and hashes w_i i times iteratively. If the hashed value matches w_0, the merchant is convinced that the payword w_i is valid.

When the merchant wants to deposit the paywords, it first provides the bank with the commitment and the certificate of the customer's public key. If the bank verifies the certificate and the commitment as valid, the merchant starts depositing the paywords. The process of depositing the paywords is similar to the customer's payment. The merchant sends the paywords to the bank. If the paywords are valid, the bank adds the same amount of money to the merchant's account.

3. Model, Cryptographic Primitives and Notations of Our Scheme

3.1 Model of Our Scheme

The postpaid micropayment model in our scheme is stated first. In this model, there are five kinds of members: A bank, merchants, customers, a trusted bank, and smart cards. The responsibility of each member is described below.

Bank

The bank is responsible to issue the certificate of smart card's public key and redeem the valid paywords. Each merchant in this scheme must open an account with the bank before the transaction.

Trusted Bank

The trusted bank is responsible to issue one smart card to one customer and revoke the customer's anonymity. Each customer has to open an account with the trusted bank if the customer wants to transact with merchants. In addition, the trusted bank is responsible to redeem the valid paywords from banks and transfer real money to banks.

Merchants

Merchants offer services or goods to the customer in exchange for the paywords.

Customers

Customers use paywords to purchase services or goods from the merchant.

Smart cards

Smart cards act as agents for the trusted bank. During the transaction between customers and merchants, the smart card provides the protection of the customer's anonymity. With the help of the smart card, the trusted bank transfers money from customer's account to the bank.

The model of our scheme is described below. The transaction flows are shown in Figure 1. This scheme consists of seven phases: the setup phase, the registration phase, the key updating phase, the commitment phase, the payment phase, the deposit phase, and the anonymity revocation phase. In the following, the phases are briefly described one by one.

Setup phase

The trusted bank prepares lots of smart cards and initializes those smart cards before the customers open accounts in the trusted bank.

Registration phase

In the registration phase, the customer opens an account in the trusted bank. After authenticating the customer's identity, the trusted bank issues one smart card to the customer and records necessary information about the customer. In order to buy goods/services with small

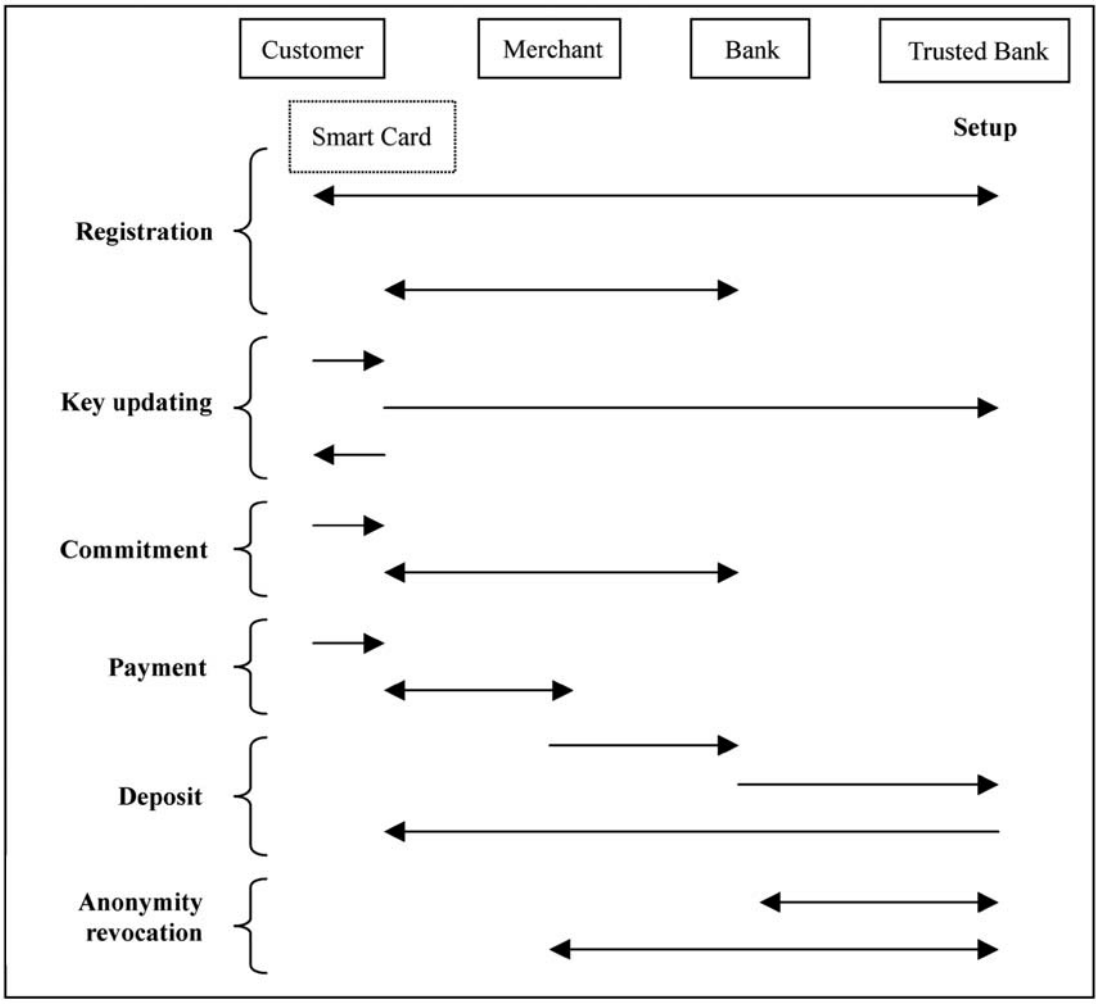

Figure 1. The transaction flows of our scheme.

values over the Internet, the customer has to perform the registration on the trusted bank and the bank in sequential way.

Key updating phase

In the key updating phase, the customer requests his/her smart card to update the smart card's current public and private key. After updating the key pair, the smart card sends its new current public key to the trusted bank. If the trusted bank confirms that this new current public key is unused, the trusted bank records the necessary information about the smart card's new current public key, and informs the smart card that this new current public key is unused. The current public key of the smart card is replaced with this new current public key. To achieve payer untraceability, it is better that the customer updates the smart card's public key for each payword chain.

Commitment phase

After updating the smart card's public key, the smart card first needs the certificate of its public key issued by the bank. Therefore, this commitment is started for each new payowrd chain.

In the commitment phase, the customer first requests the smart card to apply for a certificate of the smart card's current public key from the bank. The bank sends a *token* to the smart card in order to securely transmit the certificate of the smart card's current public key. As long as the smart card obtains the certificate from the bank, the smart card is authorized to generate a payword chain and the commitment to this payword chain. By using the commitment and the payword chain, the customer is able to purchase goods or services in one specific merchant.

Payment phase

By using the smart card, the customer sends the merchant the smart card's current public key, the bank's certificate for smart card's public key, the paywords, the commitment to the payword chain, and an order. After validating the certificate and the commitment, the merchant could only validate the payword sent form an authorized customer. If the payword is legal, then the merchant sends the goods or services to the customer and records necessary information.

Deposit phase

This phase consists of three parts. The merchant asks the bank to redeem the paywords in the first part, the bank asks the trusted bank to redeem the paywords in the second part, and trusted bank informs the smart card to recover customer's debts in the last part. In order to reduce the financial risk and the communication cost, the merchant redeems paywords periodically. Similarly, the bank also redeems paywords periodically. The length of the redeeming period depends on financial consideration and efficiency.

Anonymity revocation phase

To resolve disputes between the customer and the bank (or the merchant), the trusted bank is responsible to revoke this customer's anonymity. After the anonymity revocation phase, the customer can easily regain his anonymity back by executing the key updating phase. The regained anonymity must be independent of the revoked anonymity. So the anonymity revocation does not influence customer's anonymity seriously.

3.2 Cryptographic Primitives and Notations

The notations used in the new scheme are defined as follows:

- B, TB, M, C, SC: The notations B, TB, M, C, and SC represent the bank, the trusted bank, the merchant, the customer, and the smart card, respectively.
- ID_B, ID_T, ID_M, ID_C, ID_S: The notations ID_B, ID_T, ID_M, ID_C, and ID_S are the identifications of the bank, the trusted bank, the merchant, the customer, and the smart card, respectively.
- p, q: Two large public prime numbers p and q randomly chosen by TB such that $q|(p-1)$.
- g: The element g in Z^*_p is a generator of order q.
- TN_B, TN_{SC}: TN_B is the serial numbers of the *token* recorded by the bank B while TN_{SC} is the serial number of the *token* recorded by some smart card SC.
- PK_B, SK_B: PK_B and SK_B are the public key and private key of a bank B.
- PK_C, SK_C: PK_C and SK_C are the public key and private key of some customer C.
- PK_{SC}, SK_{SC}: PK_{SC} and SK_{SC} are the initial public key and private key of some smart card SC. PK_{SC} and SK_{SC} will be updated by the customer for keeping customer's anonymity.

- K_{BM}: K_{BM} is the secret key shared by the bank and one merchant for some symmetric cryptosystem.
- K_{BS}: K_{BS} is the secret key shared by the bank and one smart card for some symmetric cryptosystem.
- K_{BT}: K_{BT} is the secret key shared by the bank and the trusted bank for some symmetric cryptosystem.
- K_{TS}: K_{TS} is the secret key shared by TB and one smart card for some symmetric cryptosystem.
- $Sig_{SK}(m)$: The notation $Sig_{SK}(m)$ denotes a digital signature being generated by using a private key SK on message m.
- $E_K(m)$: A symmetric encryption on message m by using a symmetric key K.
- $PE_{PK}(m)$: An asymmetric encryption on message m by using a public key PK.
- $H(\bullet)$: H is a secure one-way hash function.
- $H^n(m)$: The notation $H^n(m)$ denotes the result by iteratively applying hash function H on the input *m* in n times.
- BLIND(), UNBLIND(): Functions BLIND() and UNBLIND() are the blinding and the unblinding functions in some blind signature scheme, respectively.
- r, r^{-1}: The notation r is the blinding factor used in the blinding function BLIND() while r^{-1} is the inverse of r to remove the blinding effect caused by the blind factor r.
- OWE: The notation OWE is a field maintained by each smart card to records the customer's debts. Each time the smart card generates the payword chain and the commitment successfully, the smart card adds the length of the payword chain to OWE.

4. Our Scheme

Our scheme consists of seven phases: the setup phase, the registration phase, the key updating phase, the commitment phase, the payment phase, and the deposit phase. The details of each phase are described one by one here.

4.1 Setup Phase

In this phase, TB prepares a lot of smart cards in advance, and initializes each smart card. TB initializes the values of ID_S, K_{TS}, the key pair $SK_{SC} = x \bmod q$ and $PK_{SC} = g^x \bmod p$, where $x \in Z^*_p$, OWE = 0, and $TN_{SC} = 0$ for each smart card. TB also issues a certificate Cert_of_PK_{SC} for the current public key PK_{SC} for each smart card.

These smart cards should be tamper-resistant to protect the secret data stored on them. So the stored secret data on smart cards is assumed to be secure. Since the smart cards are issued to customers, customers need the smart card reader to interactive with their smart cards. In our scheme, the smart cards have to interactive with the bank B and the trusted bank TB over networks. Although smart cards do not have networking ability, the customers have to help the smart cards to communicate with B or TB.

In our scheme, the communication channel between any two entities provides authentication function to authenticate the talking entities with one another. Moreover, the common secret keys shared by two entities also provide the authentication function.

4.2 Registration Phase

If a customer C wants to purchase goods or services from merchants over Internet, he/she first opens an account with TB through secure channels. Then he/she personally obtains one smart card SC. After issuing the smart card to the customer, TB stores ID_C, PK_C, ID_S, PK_{SC}, and K_{TS} in its local database.

After obtaining smart cards, the customer C first uses the smart card to apply for a symmetric secret K_{BS} from the bank B using secure authenticated channels. The application process is described below:

Step 1: The smart card uses its private key SK_{SC} to sign the data ID_S, *Application*, Cert_of_PK_{SC} as $Sig_{SK_{SC}}(ID_S, Application, Cert\_of\_PK_{SC})$, where *Application* records some requestor's personal information. Then the smart card sends $\{PK_{SC}, ID_S, Application, Cert\_of\_PK_{SC}, Sig_{SK_{SC}}(ID_S, Application, Cert\_of\_PK_{SC})\}$ to the bank.

Step 2: The bank uses TB's public key to verify the certificate Cert_of_PK_{SC} of smart card's public key PK_{SC}. Then the PK_{SC} is used to verify the signature $Sig_{SK_{SC}}(ID_S, Application, Cert\_of\_PK_{SC})$.

Step 3: If both the validations in **Step 2** are valid, the bank B believes that this smart card is certified by TB. The bank B sets up a symmetric secret key K_{BS} shared by SC and B, and sends a ciphertext $PE_{PK_{SC}}(K_{BS}, TN_B)$ to the smart card, where TN_B is the newest serial number of the to-

ken. Besides, the bank stores ID_S and K_{BS} in its local database.

Step 4: Upon receiving $PE_{PK_{SC}}(K_{BS}, TN_B)$, SC decrypts this ciphertext by using its own private key SK_{SC}. Then SC stores K_{BS} and sets $TN_{SC} = TN_B$.

The above application for K_{BS} is only executed once for each smart card.

4.3 Key Updating Phase

For the purpose of anonymity, the customer randomly updates the smart card's current public key PK_{SC} and the corresponding private key SK_{SC} at times. The following protocol is used for key updating.

Step 1: C chooses x' randomly from Z_q^*, and computes $g^{x'} \bmod p$. C also uses the private key SK_C to generate a signature on x' as $Sig_{SK_C}(H(g^{x'}))$. Then C sends $(x', g^{x'}, Sig_{SK_C}(H(g^{x'})))$ to the smart card.

Step 2: SC computes its new current private key as $SK'_{SC} = SK_{SC} + x' \bmod q$, and its new current public key as $PK'_{SC} = PK_{SC} \times g^{x'} \bmod p$. After updating the key pair (PK'_{SC}, SK'_{SC}), SC sends $\{ID_S, E_{K_{TS}}(ID_S, g^{x'}, Sig_{SK_C}(H(g^{x'})), nonce_{TS})\}$ to TB, where $nonce_{TS}$ is a fresh nonce generated by SC to remove replay attacks.

Step 3: TB looks for the corresponding PK_{SC} and K_{TS} according to ID_S, and the uses K_{TS} to decrypt the ciphertext $E_{K_{TS}}(ID_S, g^{x'}, Sig_{SK_C}(H(g^{x'})), nonce_{TS})$. Then TB computes $PK''_{SC} = PK_{SC} \times g^{x'} \bmod p$.

Step 4: If the value of PK''_{SC} is ever used, TB replies SC a failed message to inform the customer C choosing another x' to repeat **Step 1**. Otherwise, TB finds PK_C according to ID_S and uses PK_C to verify $Sig_{SK_C}(H(g^{x'}))$. If the verification is valid, TB replies SC the successful message containing $nonce_{TS}+1$ that the smart card's current key public key PK''_{SC} is unused. SC also separately stores SK_{SC} and SK'_{SC} without modifying SK_{SC}.

Step 5: TB finally stores PK''_{SC} and $Sig_{SK_C}(H(g^{x'}))$ with C's recorded data in the local database.

4.4 The Commitment Phase

In this phase, via smart cards, the customer C first applies for a certificate of smart card's current public key from the bank B. Assume that there is a fixed value *LIMIT* which limits the total value of transaction amounts that C can apply. The details are described below:

Step 1: C randomly chooses a blinding factor r and uses r to blind PK'_{SC} by $BLIND(PK'_{SC})$. Then C sends $\{Amount, r^{-1}, BLIND(PK'_{SC})\}$ to SC, where *Amount* is the transaction amount C wants to apply.

Step 2: After obtaining $\{Amount, r^{-1}, BLIND(PK'_{SC})\}$, SC first checks whether $OWE + Amount \geq LIMIT$. If the inequality is true, the smart card rejects C's request; otherwise the smart card sends the data $\{ID_S, E_{K_{BS}}(ID_S, Amount, BLIND(PK'_{SC}), nonce_{BS})\}$ to the bank B, where $nonce_{BS}$ is a fresh nonce generated by SC to remove replay attacks.

Step 3: Upon receiving $\{ID_S, E_{K_{BS}}(ID_S, Amount, BLIND(PK'_{SC}), nonce_{BS})\}$, the bank B finds K_{BS} out according to ID_S, and uses K_{BS} to decrypt the ciphertext $E_{K_{BS}}(ID_S, Amount, BLIND(PK'_{SC}), nonce_{BS})$. Then B checks whether or not the decrypted ID_S is equal to the received ID_S. If the above check is true, B signs $BLIND(PK'_{SC})$ as $Sig_{SK_B}(BLIND(PK'_{SC}))$ and increases TN_B by 1.

Step 4: The bank B generates the *token* as **TOKEN** = $E_{K_{BS}}(Amount, Sig_{SK_B}(BLIND(PK'_{SC})), TN_B, nonce_{BS}+1)$, where the token means a certificate for the smart card's public key. Then B sends $\{ID_B, \textbf{TOKEN}\}$ back to SC.

Step 5: After obtaining **TOKEN**, SC uses K_{BS} to decrypt **TOKEN** and checks whether $nonce'_{BS}+1 = nonce_{BS}+1$ and $TN_B > TN_{SC}$, where $nonce'_{BS}+1$ denotes the decrypted nonce. If $nonce'_{BS}+1 = nonce_{BS}+1$ and $TN_B > TN_{SC}$ are both true, SC uses r^{-1} to unblind $Sig_{SK_B}(BLIND(PK'_{SC}))$ by $UNBLIND(Sig_{SK_B}(BLIND(PK'_{SC}))) = Sig_{SK_B}(PK'_{SC})$. SC also informs C that the application for the certificate of PK'_{SC} is permitted by the bank.

After the smart card obtains the certificate of PK'_{SC}, the customer requests the smart card to generate a payword chain and the corresponding commitment for some specific merchant M by the following steps.

Step 1: The customer C randomly chooses a number w_n and sends $\{w_n, ID_M\}$ to SC, where ID_M is the

identification of this merchant.

Step 2: Upon receiving w_n, SC generates the payword chain as $w_i = H(w_{i+1})$, where i = n–1, n–2,…, 0, and n = *Amount*. Then SC generates the commitment as **CMT** = $Sig_{SK'_{SC}}(w_0, ID_M, Amount, Sig_{SK_B}(PK'_{SC}))$. After generating the payword chain and the commitment **CMT**, SC computes OWE = OWE + *Amount*. At last, the SC replies to the customer C the successful message. Now the customer C is able to shop at the merchant M in the payment phase.

4.5 Payment Phase

In this phase, the customer uses his/her smart card to purchase goods or services from the merchant. This phase comprises two protocols: **setup payment protocol** and **further payment protocol**.

Setup payment protocol

Step 1: The smart card SC sends $\{PK'_{SC}, Sig_{SK_B}(PK'_{SC}), w_0, ID_M, Amount, \mathbf{CMT}\}$ to the merchant M.

Step 2: M validates the certificate $Sig_{SK_B}(PK'_{SC})$ on PK'_{SC} by using the bank's public key PK_B and validates the commitment **CMT** by using the smart card's public key PK'_{SC}. If both validations are correct, M replies to SC that the customer C can start shopping at M. Besides, M stores $\{PK'_{SC}, \boldsymbol{CMT}, w_0, index = 0\}$ in his/her local database.

Further payment protocol

Without losing generality, suppose that the current payword and index in the merchant's database are w_i and i, respectively. The customer C wants to buy some goods or services worth L paywords. The payment protocol is described below.

Step 1: The smart card sends $\{PK'_{SC}, w_{i+L}, i+L\}$ to the merchant M.

Step 2: M retrieves $\{PK'_{SC}, \mathbf{CMT}, w_i, index = i\}$ from its local database according to the received PK'_{SC}.

Step 3: M checks whether or not $i+L \le Amount$. If $i+L > Amount$, M cancels the payment.

Step 4: M validates the payword w_{i+L} by computes $H^L(w_{i+L}) = w_i$. If $H^L(w_{i+L}) = w_i$, M is convinced that the payword w_{i+L} is valid and sends the goods or service to C.

Step 5: M replaces w_i with w_{i+L}, and sets index = i+L in its local database.

4.6 Deposit Phase

The deposit phase consists of three parts. In Part 1, M requests B to redeem the paywords. In Part 2, B requests TB to redeem the amount of money which B paid to M. In Part 3, TB informs SC to recover customer's debts. The details are described below.

Part 1: Deposit request from the merchant

Part 1 comprises two protocols: **setup deposit protocol** and **further deposit protocol**. In this part, **CMT** = $Sig_{SK'_{SC}}(w_0, ID_M, Amount, Sig_{SK_B}(PK'_{SC}))$ plays an important role. If $Sig_{SK_B}(PK'_{SC})$ is a certificate of PK'_{SC} issued by the bank B, B has to redeem the merchant's request based on the redeeming guarantee of TB. The certificate $Sig_{SK_B}(PK'_{SC})$ is also adopting K_{TS} and K_{BS} to guarantee that the owner of $Sig_{SK_B}(PK'_{SC})$ is a legal smart card issued from TB. Then TB provides the redeemable guarantee for the bank B to reduce the financial risk.

Setup deposit protocol

Step 1: The merchant M sends $\{ID_M, E_{K_{BM}}(PK'_{SC}, \mathbf{CMT})\}$ to the bank B.

Step 2: The bank B finds K_{BM} to decrypt the ciphertext $E_{K_{BM}}(PK'_{SC}, \mathbf{CMT})$ according to ID_M.

Step 3: The bank B verifies the certificate $Sig_{SK_B}(PK'_{SC})$ by his/her own public key PK_B, and then verifies **CMT** by PK'_{SC}. If both the two verifications are valid, B informs the merchant M that M is authorized to redeem the paywords.

Step 4: The bank B stores the record $(PK'_{SC}, \mathbf{CMT}, ID_M, w_0, index = 0)$ in his/her local database.

Further deposit protocol

Without losing generality, suppose that the current payword and index in the bank's database are w_i and i, respectively. The merchant M wants to redeem L paywords. The deposit protocol is described below.

Step 1: M sends $\{ID_M, E_{K_{BM}}(PK'_{SC}, L, w_{i+L})\}$ to the bank B.

Step 2: B first decrypt the ciphertext $E_{K_{BM}}(PK'_{SC}, L, w_{i+L})$ by K_{BM}. Then B finds current payword w_i and index = i in the bank's database according to PK'_{SC}.

Step 3: B computes L hashes on w_{i+L} and compares the hashed value $H^L(w_{L+i})$ with the stored w_i. If $H^L(w_{i+L}) = w_i$, B increases merchant's account by the amounts which are worth L paywords.

Step 4: The bank replaces w_i with w_{i+L}, and sets index = i+L in his local database.

Part 2: Deposit request from the bank

Part 2 also comprises two protocols: **setup deposit protocol** and **further deposit protocol**.

Setup Deposit Protocol

Step 1: B sends $\{ID_B, E_{K_{BT}}(PK'_{SC}, \mathbf{CMT})\}$ to TB.

Step 2: TB decrypts the ciphertext $E_{K_{BT}}(PK'_{SC}, \mathbf{CMT})$ by K_{BT}. Then TB verifies the certificate by the public key PK_B and verifies **CMT** by PK'_{SC}. If both verifications are valid and PK'_{SC} has been stored in TB's database, TB stores (ID_C, ID_S, K_{TS}, PK_C, PK_{SC}, PK'_{SC}, $Sig_{SK_C}(H(g^{x'}))$, **CMT**, w_0, index = 0) in his local database.

Step 3: TB informs B that B is authorized to redeem money.

Further Deposit Protocol

Without losing generality, suppose that the current payword and index in TB's database are w_i and i, respectively, where i is a nonnegative integer. Suppose that the bank wants to redeem L paywords, where L is an integer. The further deposit protocol is executed by the bank B and TB.

Step 1: B sends $\{ID_B, E_{K_{BT}}(PK'_{SC}, L, w_{i+L})\}$ to TB.

Step 2: TB decrypts the ciphertext $E_{K_{BT}}(PK'_{SC}, L, w_{i+L})$ by K_{BT}. Then TB finds the current payword w_i and index = i in TB's database according to PK'_{SC}.

Step 3: TB computes L hashes on w_{i+L} to obtain the hashed value $H^L(w_{i+L})$.

Step 4: If $H^L(w_{i+L})$ matches w_i, TB pays B the amounts worth L paywords. Besides, TB knows the owner C of the SC whose current public key is PK'_{SC}, so TB decreases C's account by the amounts worth L paywords.

Part 3: Recovery

As mentioned in the previous section, SC computes OWE = OWE+*Amount* after it successfully generates the commitment to the payword chain. When OWE+*Amount* ≥ LIMIT is true, the customer cannot apply for the *token*. Hence, after TB debits C's spent paywords from C's account, TB must inform SC to decrease C's debts. Suppose that TB debits the amounts worth K paywords from C's account.

Step 1: TB sends $\{ID_T, E_{K_{TS}}(ID_T, CLEAR\_MSG, K)\}$ to the smart card SC, where CLEAR_MSG is a message that asks SC to decrease the value of OWE.

Step 2: SC decrypts the message $E_{K_{TS}}(ID_T, CLEAR\_MSG, K)$ by the secret key K_{TS} and confirms the contents of CLEAR_MSG, SC computes OWE = OWE−K.

4.7 Anonymity Revocation Phase

If a customer is involved in a dispute with the bank or the merchant, the bank or the merchant would request TB to revoke the customer's anonymity by sending the smart card's current public key PK'_{SC} to TB. Upon receiving PK'_{SC}, TB finds the customer's identity out from its database. TB then notifies the bank or the merchant that the identity of PK'_{SC}'s owner is ID_C.

After the dispute is resolved, the customer obtains his/her anonymity back by executing the key updating phase. No one knows the relation between PK'_{SC} and ID_C except TB.

5. Security Analysis

The analyses of security and anonymity issues of our scheme are given.

5.1 Double Spending Prevention

Suppose that a malicious customer C′ wants to spend one payword in different merchants. In this scheme, he/she is not able to do this attack. Because the commitment **CMT** generated by SC is merchant-specific, C′ cannot use the same **CMT** in different merchants.

On the other hand, the malicious customer C′ cannot spend one payword twice or more times in the same merchant. In our scheme, since the merchant records the latest payword w_i and the index of w_i in his/her local database, the merchant is able to detect the used payword immediately.

5.2 Unforgeability

Suppose that an adversary C′ wants to forge paywords to purchase goods or services in the merchant without bank's permission and the smart card's monitor. The security of the private keys is considered first. Due to the computationally infeasible property of the discrete logarithm problem (DLP), C′ cannot derive smart card's private keys SK'_{SC} from its public key PK'_{SC} and bank's private key SK_B from the bank's public key PK_B. Therefore, all private keys are secure. Moreover, assume that the smart card is tamper-resistant without leaking its private key SK'_{SC}.

With the aim of arising forgery attacks, the adversary C′ must forge the certificate of an illegal PK^*_{SC} or forge the commitment to his forged payword chain for some legal PK'_{SC}. It is also hard to forge the certificate of an illegal PK^*_{SC} since the certificate is a (blind) signature of the bank. Although the payword chain can be easily created by C′, it is hard to forge the commitment because the commitment is actually SC's signature being verified by PK'_{SC}. Based on the analysis that all private keys are secure, the adversary cannot obtain these private keys to forge commitments or certificates. Therefore, the adversary C′ cannot forge paywords.

5.3 Anonymity

The anonymity is discussed in two aspects: payment anonymity and payer untraceability [22]. The payment anonymity means that anyone except TB only knows the customer's pseudonym. The payer untraceability means that anyone except TB cannot trace the payer's identity of some payments.

Assume that TB is trustworthy and SC is temper-resistant without leaking customer's identity. The bank B, the merchant M, and malicious customers C′ are the possible adversaries who wants to recognize C's identity in this scheme.

The Bank B

In the commitment phase of our scheme, B receives SC's blinded public key $BLIND(PK'_{SC})$ and sends $Sig_{SK_B}(BLIND(PK'_{SC}))$ back to the smart SC. Because PK'_{SC} is blinded in a secure blind signature scheme, the bank B does not know what the value of PK'_{SC} is. After the unblinding process, the relation between the blinded public key $BLIND(PK'_{SC})$ and the original public key PK'_{SC} disappears in a secure blind signature scheme. In other words, the bank B cannot build the relation between PK'_{SC} and the customer C alone. The bank B cannot identify the customer as the owner of PK'_{SC}. On the other hand, B cannot trace customer's transactions via the smart card's public key PK'_{SC} since the customer randomly updates PK'_{SC} each time.

The merchant M

M cannot learn C's identity or trace C's payments in this scheme. In the payment phase, M receives PK'_{SC}, **CMT**, the payword w_i, and index of the payword from SC. PK'_{SC} is the only useful information for M. However, PK'_{SC} is a randomized public key, so M cannot use it to trace the smart card's owner.

A malicious customer C′

Since messages sent from C are all encrypted by some symmetric encryption, the only work C′ can do is to trace C's transactions by PK'_{SC}. As mentioning in the above, the traceability of PK'_{SC} is removed because the public key PK'_{SC} is randomly updated each time.

As discussing in the above, our scheme achieves both payment anonymity and payer untraceability.

6. Comparisons

Our scheme is compared with Lin's and Tsou's schemes for the security requirements in Section 6.1. Then the computational performance comparison among our scheme, Lin's scheme and Tsou's scheme is given in Section 6.2.

6.1 Comparisons of Security Requirements

Some common security requirements for micropayment schemes are described below. The first requirement is double spending prevention that the customer cannot use a payword twice or more times. The divisibility requirement means that it is possible to use multiple denominations (in token-based system) [22]. The transferability requirement means that the customer can spend the token in different merchants without contacting the bank. The prepaid requirement means that the customer's account is decreased when his/her withdrawals occur. The postpaid requirement means that the customer's account is decreased when the merchant re-

deem payments. The payment anonymity means that all other users in the scheme can only know the customer's pseudonym except the trusted bank. The payer untraceability means that customer's payments cannot be traced. The anonymity revocation requirement means that there exists a trusted bank to revoke customer's anonymity if the customer is involved in a dispute with the bank or the merchant.

Table 1 demonstrates the security requirement comparisons among our new scheme, Lin's scheme, and Tsou's scheme. Our scheme offers better solution to anonymity than Lin's and Tsou's. Lin's scheme does not provide anonymity revocation to resolve disputes among customers, banks, and merchants. Although Tsou's scheme provides anonymity revocation, the anonymity recovery is not convenient for the customers in Tsou's scheme. In Tsou's scheme, to regain the customer's anonymity after the anonymity revocation phase, the customer has to register a new bank account. Moreover, Tsou's scheme doesn't provide payer untraceability, so customers' payments in their scheme may be traced.

Only our new scheme is postpaid while the other two schemes are both prepaid. For customers, the postpaid micropayment scheme is more popular than the prepaid micropayment scheme because the customers enjoy shopping before actual payment. On the other hand, the profits of merchants are protected by the revocable customers' anonymity. The new scheme provides a fair protection both for customers and merchants.

6.2 Performance Analysis

In this section, the computational performance is measured by four kinds of cryptographic functions: the signature generation and verification, the symmetric encryption and decryption, the hash function, and the blind signature generation. Some notations used to representing computational cost are defined below. The notation *sign* denotes the cost to generate one signature. The notation "*verify*" denotes the cost to verify one signature. The notation EN denotes the execution cost of one symmetric encryption while the notation DE denotes the execution cost of one symmetric decryption. The notation hash denotes the computational cost of performing hash function once. The notation blind denotes the computational cost of performing a blinding function.

In the following, the performance of Lin's scheme and the new scheme is given because the fundamental ideas of micropayment of Lin's scheme and the new scheme are similar. Since the new scheme is designed for the micropayment scheme for specific merchants, the specific merchant version of Lin's scheme is considered. Being compared with Lin's scheme, our scheme provides additional anonymity revocation function and the postpaid mode which benefit customers. Therefore, our scheme additionally needs 2 signature generations, n+1 hash computations, 2 symmetric encryptions, 3 symmetric decryptions, 5 signature verifications. Details of the extra costs are described below.

For the anonymity revocation, the customer has to provide evidence each time he/she executes the key updating phase, therefore, the customer has to cost one signature generation and one hash operation. The smart card uses one symmetric encryption to securely transmit the customer's signature and other information to TB. TB costs one symmetric decryption to decrypt the ciphertext, and one signature verification to verify the customer's signature.

To provide the postpaid function in our scheme, TB and the smart card cooperate to settle the customer's pay-

Table 1. Security requirement comparisons among various anonymous micropayment schemes

Functions \ Schemes	Our scheme	Lin's scheme	Tsou's scheme
Double spending prevention	Yes	Yes	Yes
Divisibility	No	No	No
Transferability	No	Yes	Yes
Payment anonymity	Yes	Yes	Yes
Anonymity revocation	Yes	No	Yes
Payer untraceability	Yes	Yes	No
Prepaid/postpaid	Postpaid	Prepaid	Prepaid

Table 2. Computational performance of anonymous micropayment schemes

	Our scheme	Lin's scheme (basic version)
Registration	**B**: 2 verifies **SC**: 1 sign	
Key updating	**TB**: 1 DE + 1 verify **C**: 1 hash + 1 sign **SC**: 1 EN	
Commitment	**B**: 1 DE +1 EN + 1 sign **C**: 1 blind **SC**: 1 EN + 1 DE + n hashes + 1 sign	**B**: 1 DE + 1 EN + 1 sign **C**: 1 EN + 1 blind **SC**: 1 DE + n hashes + 1 sign
Payment	**M**: 2 verifies + n hashes	**M**: 2 verifys + n hashes
Deposit	**TB**: 1 DE + 2 verifys + n hashes + 1 EN **B**: 1 DE + 2 verifys + n hashes + 1 EN **M**: 1 EN **SC**: 1 DE	**B**: 1 DE + 2 verifys + n hashes + 1 EN **M**: 1 EN

ments and debts. The customer does not need to make contact with the bank. In the registration phase, the smart card sends the bank one signature $Sig_{SK_{SC}}$ (ID_S, *Application*, Cert_of_PK_{SC}) to apply for one symmetric secret key K_{BS} which lets the smart card securely contact with the bank as a substitute for the customer. The bank first verifies the certificate Cert_of_PK_{SC} for the smart card's public key, and then verifies the signature $Sig_{SK_{SC}}$ (ID_S, *Application*, Cert_of_PK_{SC}). Besides, in the deposit phase, TB has to pay off the customer's debts, TB costs 1 symmetric encryption, 2 signature verification, and n hash to this work. After TB decreases the customer's account, TB informs the smart card to recovery the customer's debts, and this work costs TB one symmetric encryption and the smart card one symmetric decryption.

Table 2 demonstrates the detailed comparison among Lin's scheme and our scheme. For fairness, the comparison is under the assumption that one customer purchases goods/services in one specific merchant.

7. Conclusion

A postpaid micropayment scheme with revocable customers' anonymity is proposed in this paper. This scheme achieves both payment anonymity and payer untraceability. The customer in this scheme can efficiently obtain anonymity. Besides, this scheme provides the revoking function of customers' anonymity. Once the customer is in dispute with the merchant or the bank, the trusted bank can revoke the customer's anonymity to resolve the dispute. After disputes are resolved, the customer efficiently obtains his/her anonymity again by only updating the smart card's key pair. This scheme also provides postpaid payment in which the customer can pay the cash after receiving their goods/services. This characteristic may attract more people using the micropayment scheme. However, our scheme needs the help of a trust bank to achieve both payment anonymity and payer untraceability. To remove the trust bank is a future research topic.

References

[1] Herberg, Amir, "Micropayment," *Payment Technologies for E-Commerce*, Kou, Weidong Ed., New York: Springer, pp. 245–282 (1998).

[2] Anand, R. Sai and Madhavan, C. E. Veni, "An Online, Transferable E-Cash Payment System," *Advance in Cryptology – INDOCRYPT 2000,* LNCS, Vol. 1977, New York: Springer-Verlag, pp. 93–103 (2000).

[3] Brands, Stefan, "Untraceable Off-Line Cash in Wallet with Observers," *Advances in Cryptology – CRYPTO '93*, LNCS, Vol. 773, New York: Springer-Verlag, pp. 302–318 (1993).

[4] Chan, Agnes, Frankel, Yair, and Tsiounis, Yiannis, "Easy Come – Easy Go Divisible Cash," *Advances in Cryptology – EUROCRYPT '98*, LNCS, Vol. 1403, New York: Springer-Verlag, pp. 561–575 (1998).

[5] Frankel, Yair, Tsiounis, Yiannis and Yung, Moti, "Indirect Discourse Proofs: Achieving Efficient Fair

Off-Line E-Cash System," *Advance in Cryptology – ASIACRYPT '96*, LNCS, Vol. 1163, New York: Springer-Verlag, pp. 286–300 (1996).

[6] Jakobossn, Markus and Yung, Moti, "Revokable and Versatile Electronic Money," *Proceeding of the 3rd ACM Conference on Computer and Communications Security*, India: ACM press, pp.79–87 (1996).

[7] MasterCard and VISA, *Secure Electronic Transactions Specification* (books 1, 2, 3), June (1996).

[8] Mu, Yi, Nguyen, Khanh Quoc and Varadharajan, Vijay, "A Fair Electronic Cash Scheme," *Topics in Electronic Commerce*: *Second International Symposium – ISEC 2001,* LNCS, Vol. 2040, Springer-Verlag, pp. 20–32 (2001).

[9] Neuman, B. C. and Medvinsky, G., "NetCheque, NetCash and the Characteristics of Internet Payment Services," *Proceeding of MIT Workshop on Internet Economics 1995* (1995).

[10] Sirbu, M. and Tyger, T. J., "NetBill: An Electronic Commerce System Optimized for Network Delivered Information and Services," *Proceeding of IEEE CompCon '95*, pp. 20–25 (1995).

[11] Bellare, M., Garay, J., Hauser, R., Herzberg, A., Krawczyk, H., Steiner, M., Tsudik, G. and Waidner, M., "iKP – A Family of Secure Electronic Payment Protocols," *Proceeding of 1st USENIX Workshop on Electronic Commerce,* pp. 89–106 (1995).

[12] Glassman, S., Manasse, M. S., Abadi, M., Gauthier, P. and Sobalvarro, P., "The Millicent Protocol for Inexpensive Electronic Commerce," *World Wide Web Journal, Proceeding of 4th International World Wide Web Conference,* O'Reilly, pp. 603–618 (1996).

[13] Manasee, M. S., "The Millicent Protocols for Electronic Commerce," *Proceeding of 1st USENIX Workshop on Electronic Commerce,* pp. 117–123 (1995).

[14] Rivest, R. L. and Shamir, A., "PayWord and MicroMint: Two Simple Micropayment Schemes," *Proceedings of 1996 International Workshop on Security Protocols* (*Cambridge, United Kingdom, April 10–12*), LNCS, Vol. 1189, Berlin: Springer-Verlag, pp. 69–87 (1997).

[15] Stern, J. and Vaudenay, S., "SVP: A Flexible Micropayment Scheme," *Proceeding of Financial Cryptography*, LNCS, Vol. 1318, New York: Springer-Verlag, pp. 161–172 (1997).

[16] Lin, S.-Y., "Design and Cryptanalysis of Micropayment Schemes," Master Thesis, National Central University, Taiwan, R.O.C. (2004).

[17] Tsou, J.-H., "The Study of Electronic Payment Scheme," Master Thesis, Tamkang University, Taiwan, R.O.C. (2005).

[18] Yen, S.-M., "PayFair: A Prepaid Internet Micropayment Scheme Ensuring Customer Fairness," *Computers and Digital Techniques, IEE Proceedings,* Vol. 148, pp. 207–213 (2001).

[19] Chaum, D., "Blind Signatures for Untraceable Payments," *Advances in Cryptography – Proceeding of Crypto '82*, New York: Springer-Verlag, pp. 199–203 (1983).

[20] Chaum, D., Fiat, A. and Naor, M. "Untraceable Electronic Cash," *Advances in Crytology – CRYPTO '88*, LNCS, Vol. 403, New York: Springer-Verlag, pp. 21–25 (1988).

[21] Pointcheval, David and Stern, Jacques, "Provably Secure Blind Signature Schemes," *Advances in Cryptology – ASIACRYPT '96*, LNCS, Vol. 1163, Berlin: Springer-Verlag, pp. 252–265 (1996).

[22] Tsiakis, T. and Sthephanides, G. "The Concept of Security and Trust in Electronic Payments," *Computers & Security,* Vol. 24, pp. 10–15 (2005).

Manuscript Received: Jun. 5, 2007
Accepted: Apr. 28, 2008