A SECURE WIRELESS PREPAID MICROPAYMENT PROTOCOL WITH EXTENSION TO POSTPAID MICROPAYMENT

Supakorn Kungpisdan¹, Bala Srinivasan¹, and Phu Dung Le²
¹School of Computer Science and Software Engineering, Monash University, Australia
²School of Network Computing, Monash University, Australia

Abstract
In this paper, a secure prepaid micropayment protocol for wireless networks is presented. The protocol deploys a secure cryptographic technique that reduces party’s computation and satisfies transaction security properties, including non-repudiation. This offers the ability to resolve disputes among parties. Compared to existing protocols, party’s private information in our protocol is well-protected. Our analysis shows that our protocol has better performance to operate on limited capability wireless devices than existing protocols. Finally, we show that our protocol can be extended to perform postpaid micropayment with better performance than existing postpaid micropayment protocols. This results in a general framework for secure wireless micropayments.

1. Introduction
Micropayment seems to be more widely accepted than other kinds of payment schemes for wireless networks due to its lightweight, low setup cost, and low transaction cost. Moreover, most payment-related applications for wireless networks are conducted with small-valued goods or services e.g. ring tone or operator logo downloads. Traditionally, micropayment protocols deploy public-key operations e.g. PayWord [6] or NetCard [2]. Although they work well for fixed environments, they are not suitable for wireless ones due to a number of limitations of wireless environments [3, 7]. Firstly, wireless devices have lower power, and computational capability compared to desktop computers. They cannot efficiently perform high computational operations such as public-key operations. Although some devices are equipped with special processors, performing such operations on them still requires longer processing time. Secondly, wireless networks have less bandwidth and reliability. Connection cost to wireless networks is also considerably higher. Thus, mobile payments with existing protocols are not acceptable by many users.

Recently, a micropayment protocol called PayFair [8] offers the ability to perform transactions on limited computational capability devices. PayFair deploys symmetric-key operations and keyed-hash functions to reduce the computation at all engaging parties. However, PayFair lacks of transaction privacy because payment information of engaging parties is transmitted in cleartext during the transaction. Moreover, the message sent from a client to a merchant in PayFair lacks of non-repudiation property which is an important property for financial transactions, in that, a bank is able to impersonate as the client. In addition, a payment token in PayFair is merchant-specific in that it is used to generate the coins to spend with only one specified merchant. Thus, the client is required to request a new payment token every time when she make a purchase to a new merchant.
In this paper, we propose a prepaid micropayment protocol which is suitable for wireless environments. It deploys a secure symmetric cryptographic technique that not only reduces the computation at all parties, especially at a mobile client, but also satisfies transaction security properties including the non-repudiation property [1]. Moreover, our protocol offers the ability to resolve disputes among parties. Furthermore, all parties’ private information such as payment information and secret keys is well-protected.

In any prepaid payment system, a client has to purchase an electronic coupon which contains spending credits from a bank, and the amount paid by the client is transferred to a specified merchant before the first transaction with the merchant. In our protocol, we present an efficient method to refund un-spending credits and coupons. This offers the practicability to the system. Moreover, the coupon in our protocol is truly general-purposed in that it can be split into a number of smaller-valued, merchant-specific coupons to spend with many merchants. In addition, we show that our protocol is general in that it can perform postpaid micropayments. Therefore, we can have a general framework for both prepaid and postpaid micropayments. We perform performance analysis of our protocol and compare the results with PayWord [6] and PayFair [8]. The results show that our protocol has better performance than others in terms of party’s computation and message passes. As a result, our protocol can be implemented in limited capability wireless devices with higher performance and security than existing protocols [6, 8].

This paper is organized as follows. Section 2 outlines PayWord and PayFair, two existing micropayment protocols. In section 3, our protocol is introduced. In section 4, we present an extension of the proposed protocol to postpaid micropayment. Section 5 discusses security and performance of the protocol. Section 6 concludes our work.

2. Overviews of Existing Micropayment Protocols

2.1 PayWord

PayWord [6] is a postpaid micropayment protocol based on public-key cryptography. Three parties are involved in the system: client C, merchant M, and bank B. Initially, C and M establish accounts with B. Before making a payment, B issues C PayWord certificate which contains authorized amount CL that C is allowed to make the payment to each merchant. To make the payment to M, C generates a set of coins \{c_0, ..., c_n\}, \( n = CL \), and \( c_i = h(c_{i+1}) \), where \( i = 0, ..., n-1 \).

In the first payment to M, C sends M a commitment, which contains the PayWord certificate and \( c_0 \), digitally signed by C. In each payment, C sends the coin \( c_i \) to M. M infers the value of the coin by applying a number of hash functions to \( c_i \). After specified period, M sends the highest value of \( c_i \), \( c_{i_{\text{max}}} \), together with the commitment to B. B then deducts \( i_{\text{max}} \) from C and transfers the money to M. However, PayWord is not suitable for applying to wireless environments because it has high client’s computation due to public-key operations. Moreover, certificate verification leads to additional communication passes [3]. In addition, payment information, \( c_0 \) and \( c_i \), is readable by any party who holds C’s public key. Thus, any party is able to trace the client’s spending.

2.2 PayFair
PayFair [8] is a prepaid micropayment protocol based on symmetric-key operations. Three parties are involved in PayFair: client $C$, merchant $M$, and bank $B$. The details of PayFair are shown below:

**Phase A: Prepaid Phase**

\[ C \rightarrow B: \ ID_C, \ O_C, \ h(O_C, \ K_C) \]  
\[ B \rightarrow C: \ \{N, \ RN\}_{SK}, \ N, \ h(\{N, \ RN\}_{SK}, \ N, \ O_C, \ K_C) \]  

Where $SK$ is the secret known only to $B$. $K_C$ is shared between $C$ and $B$. $C$ requests $B$ by sending order number $O_C$ containing the requested amount. $B$ returns the message containing a payment token $\{N, \ RN\}_{SK}$, which is later used to generate coins. $RN$ is a random number generated from the serial number $N$ and the secret $SK_{RN}$ known only by $B$. $RT$ is a nonce for replay protection. $C$ generates a set of coins $\{w_0, \ldots, w_n\}$, where $w_n = \{N, \ RN\}_{SK}$, from the process: $w_i = h(w_{i+1})$, $i = 0, \ldots, n-1$.

**Phase B: Micropayment Phase**

\[ C \rightarrow M: \ w_0, \ N, \ h(w_0, \ ID_M, \ K_C) \]  
\[ M \rightarrow B: \ w_0, \ N, \ ID_C, \ RM, \ h(w_0, \ ID_M, \ K_C) \]  
\[ B \rightarrow M: \ w_0, \ ID_C, \ RM, \ YES, \ h(w_0, \ ID_C, \ K_M, \ RM, \ YES) \]

To make a payment to $M$, $C$ sends the message (c) containing $w_0$ to $M$. $M$ then forwards $h(w_0, \ ID_M, \ K_C)$ with relevant information to $B$ in (d). After receiving the message, $B$ can generate $w_n$ from $w_0$, $N$, and its own $RN$ and $SK$. $B$ then transfers the amount $n$ to $M$’s account and sends the response to $M$ in (e). $C$ can start payment transaction with $M$ as follows:

\[ C \rightarrow M: \ w_i \quad \text{where } i = 1, \ldots, n \]

In PayFair, the problem regarding revealing payment information occurred in PayWord still exists because, in the messages (c) and (f), $w_0$ and $w_i$ are transmitted in cleartext. In addition, although Yen [8] argued that the payment token $\{N, \ RN\}_{SK}$ is general-purposed, it is still merchant-specific when used, that is, although the coins are merchant-independently generated, they still can be used to pay to only one specific merchant. $C$ needs to request $B$ to issue a new token every time she wants to make a payment to a new merchant. Moreover, in (c), $B$ can impersonate as $C$.

### 3. The Proposed Protocol

Our protocol is composed of three engaging parties: *client*, *merchant*, and *bank*. For simplicity, we assume that a client and a merchant establish accounts with the same bank. The bank is trusted by the client only to generate correct numbers and values of coins for coin validation purpose, but it is not trusted to create payment requests to any merchant by itself because it can generate the sets of coins by itself. It is possible to generate fake requests on behalf of its clients. At the beginning of the protocol, three secrets are shared among engaging parties: $X$ shared between a client and a merchant, $Y$ shared between the client and a bank, and $Z$ shared between the merchant and the bank. Such keys are updated periodically or upon the requests by engaging parties. Our protocol is composed of 6 sub-protocols: **Setup, Payment Initialization, Payment, Extra Credit Request, Coupon Cancellation, and Coin Return** protocols. Sections 3.1-3.6 present the protocols in details.

#### 3.1 Setup Protocol
A client $C$ requests a bank $B$ for an authorization on performing a micropayment transaction with the amount $CL_T$ as follows:

$$C \rightarrow B: \ ID_C, CL_T, TCP, h(CL_T, TCP, Y) \quad (1)$$

Where $CL_T$ stands for total credits that $C$ is allowed to spend in the system and $TCP$ is timestamp when generating the request. $h(CL_T, TCP, Y)$ is used to protect the integrity of the message. $B$ checks the validity of $C$’s account and then deducts the amount $CL_T$ from $C$’s account. $B$ then sends $C$ a bank coupon that can be used to perform transactions as follows:

$$B \rightarrow C: \ \{CL_T, TT, TCP, SN, c\}_Y \quad (2)$$

The bank coupon contains unique serial number $SN$ assigned by $B$ and authorized credits $CL_T$. $TT$ stands for timestamp while issuing $CL_T$, and $c$ is a random number generated by $B$ used for generating coins. With this bank coupon, $C$ can make payments to many merchants repeatedly up to $CL_T$. After running out of the credits, $C$ needs to run Setup Protocol again.

### 3.2 Payment Initialization Protocol

To make a payment to a merchant $M$, $C$ generates a set of coins $\{c_0, ..., c_n\}$, $n=CL_T$, as follows:

$$c_n = \{c, TG\} \quad \text{and} \quad c_i = h(c_{i+1}) \quad \text{where} \quad i = 0, ..., n-1$$

Where $TG$ stands for timestamp while generating the coins. $C$ specifies the amount $CL_M$ to spend with $M$. $C$ attaches $c_0$ and $CL_M$ into a merchant coupon, and sends it to $B$:

$$C \rightarrow B: \ h(c_0, TG, CL_M, X), h(ID_M, c_0, TG, CL_M, CL_T, TT, SN, Y), TG \quad (3)$$

Note that $C$ can either spend the entire credits $CL_M$ to $M$ or spend some credits to $M$ and the rest to other merchants. $h(c_0, TG, CL_M, X)$ represents the payment request from $C$ to $M$ which is unreadable by $B$. $B$ retrieves $CL_T$ and $CL_M$ from $h(ID_M, c_0, TG, CL_M, CL_T, TT, SN, Y)$ and checks if $CL_T < CL_M$. If so, $B$ rejects the request. If $CL_T > CL_M$, $B$ computes $C$’s remaining credits $CL_TR$, where $CL_TR = CL_T - CL_M$. $B$ then maintains the list of updated $CL_TR$ to prevent over-spending. At this stage, $B$ transfers $CL_M$ to $M$’s account. Then, $B$ sends the following messages to $C$ and $M$:

$$B \rightarrow M: \ h(c_0, TG, CL_M, X), \{c_0, TG, SN, CL_M, h(ID_M, SN, CL_TR, TTR, Y)\}_Z \quad (4)$$

$$B \rightarrow C: \ h(ID_M, SN, CL_TR, TTR, Y), TTR \quad (5)$$

Where $TTR$ stands for timestamp while $B$ updates $CL_TR$. Note that $T_T$ is updated to $TTR$ after calculating $CL_TR$. $M$ retrieves $c_0$ and $CL_M$ from the encrypted message. In (4), $M$ is notified that $C$ has requested for the payment to her from $h(c_0, TG, CL_M, X)$, and $C$’s request has been authorized by $B$ from the message encrypted with $Z$. The message (5) notifies $C$ that her request has been approved and processed. Later on, $C$ can use $\{CL_TR, TTR\}$ to perform transactions with other merchants. In this protocol, $M$ or $B$ cannot impersonate as $C$ because they communicate to one another with different sets of keys which are not revealed during the transactions.

### 3.3 Payment Protocol
After completing payment initialization, $C$ starts the payment to $M$ by sending the coin as follows:

$$\begin{align*}
C \rightarrow M: & \quad c_j \quad \text{(where } j = 1, \ldots, n) \\
\end{align*}$$

(6)

$M$ verifies the requested amount by comparing $c_j$ with $c_0$. $M$ checks the amount and delivers goods to $C$. After each payment, $CL_M$ is deducted by the requested amount. $C$ maintains the record of remaining credits to spend with $M$. $C$ is allowed to make payments up to $CL_M$ without payment authorization required by $B$. If the remaining credits are not sufficient to make another payment, the client can request the bank for extra credits by performing Extra Credit Request Protocol.

### 3.4 Extra Credit Request Protocol

Normally, when $C$ spends the credits up to $CL_M$, she needs to run Setup Protocol to issue a new bank coupon. In our protocol, we reduce the frequency of performing this process by running Extra Credit Request (ECR) Protocol. With ECR Protocol, the numbers of message passes are reduced. Before the next payment, $C$ checks if the requested price is greater than $CL_M$. If so, $C$ still can purchase the goods but she needs to request for extra credits. $C$ realizes that, if her request has been approved, her remaining total credits $CL_{TR}$ will be deducted by $CL_M$. To request for extra credits, $C$ sends the following message:

$$\begin{align*}
C \rightarrow M: & \quad c_j, CL_M, h(ID_M, CL_M, T_G, SN, CL_{TR}, T_{TR}, Y) \\
\end{align*}$$

(7)

$CL_M$ stands for new credits to spend with specified merchant $M$, and $T_M$ stands for timestamp when requesting for a new $CL_M$. $M$ retrieves $CL_M$ and forwards the following message to $B$:

$$\begin{align*}
M \rightarrow B: & \quad ID_M, h(ID_M, CL_M, T_G, SN, CL_{TR}, T_{TR}, Y) \\
\end{align*}$$

(8)

$B$ retrieves $CL_{TR}$ and $CL_M$, and then calculates a new $CL_{TR}$, where $new-CL_{TR} = current-CL_{TR} – CL_M$. $B$ transfers $CL_M$ to $M$’s account, and then sends the response to $M$ as follows:

$$\begin{align*}
B \rightarrow M: & \quad h(ID_M, SN, CL_{TR}, T_{TR}, Y), T_{TR}, YES, h(YES, CL_M, T_{TR}, Z) \quad \text{if approved} \\
& \quad \quad \quad \quad \quad \quad \quad \quad \quad \text{(or Rejected if } C \text{ has insufficient credits)} \\
\end{align*}$$

(9)

$M$ checks if the authorized $CL_M$ in $h(YES, CL_M, T_{TR}, Z)$ is equal to $CL_M$ received from $C$ in (7). If so, $M$ sends $C$ the following message:

$$\begin{align*}
M \rightarrow C: & \quad h(ID_M, SN, CL_{TR}, T_{TR}, Y), T_{TR} \\
\end{align*}$$

(10)

$C$ expects to receive the updated $CL_{TR}$, where $updatedCL_{TR} = currentCL_{TR} – CL_M$. She calculates $CL_{TR}$ and compares with the received $CL_{TR}$. If match occurs, $C$ can infer the updated bank coupon with $CL_{TR}$. The above message is considered as a notification of $C$’s remaining total credits. Note that, to make the payment to a new merchant, $C$ repeats Payment Initialization Protocol with the updated bank coupon without the need to run Setup Protocol as that in existing protocols.

### 3.5 Coupon Cancellation Protocol

In the proposed protocol, $C$ is able to refund an un-used bank coupon previously purchased from $B$ by sending the following message to $B$:
\[ C \rightarrow B: \ SN, T_{CR}, h(SN, CL_T, T, T_{CR}, Y) \]  \[(11)\]

Where \( T_{CR} \) is timestamp while requesting for coupon cancellation. \( B \) removes the coupon with the serial number \( SN \) from its database. This coupon is no longer used in the system. \( B \) transfers the amount \( CL_T \) to \( C \)'s account and sends the response of \( C \)'s request as follows:

\[ B \rightarrow C: \ Cancel-OK, (Cancel-OK, SN, T_{CR}, Y) \]  \[(12)\]

### 3.6 Coin Return Protocol

In some situation, \( C \) may want to end the transaction with \( M \) after spending some credits and request \( M \) to refund the un-spending credits. This process can be done in our protocol as follows:

\[ C \rightarrow M: \ c_{j_{\text{max}}}, T_G, h(ID_M, c_0, T_G, Y) \]  \[(13)\]

Where \( c_{j_{\text{max}}} \) is the highest-value coins currently spent with \( M \). \( M \) checks whether the received \( c_{j_{\text{max}}} \) is equal to \( c_{j_{\text{max}}} \) that she has. If they are matched, \( M \) forwards the following message to \( B \):

\[ M \rightarrow B: \ ID_M, c_{j_{\text{max}}}, T_G, h(ID_M, c_0, T_G, Y) \]  \[(14)\]

\( B \) retrieves \( c_{j_{\text{max}}} \) and \( c_0 \) and calculates \( \text{returned-amount} \), where \( \text{returned-amount} = CL_M - j_{\text{max}} \). \( B \) then transfers \( \text{returned-amount} \) to \( C \)'s account and updates \( C \)'s bank coupon with the updated \( CL_{TR} \), where \( \text{updated-CL}_{TR} = \text{current-CL}_{TR} + \text{returned-amount} \). \( B \) updates the entry in the list at the record containing \( T_G \) and \( c_0 \), and then sends the acknowledgement to \( M \):

\[ B \rightarrow M: \ h(\text{returned-amount}, ID_M, c_0, T_G, CL_{TR}, T_{TR}, Y), h(\text{returned-amount}, ID_C, c_0, T_G, T_{TR}, Z), T_{TR} \]  \[(15)\]

\( M \) is notified that the amount \( \text{returned-amount} \) has been withdrawn and transferred to \( C \)'s account from \( h(\text{returned-amount}, ID_C, c_0, T_G, T_{TR}, Z) \). Also, she is notified that the set of coins starting with \( c_0 \) is no longer valid in the system. \( M \) then sends the following message to \( C \):

\[ M \rightarrow C: \ h(\text{returned-amount}, c_0, T_G, CL_{TR}, T_{TR}, Y), T_{TR} \]  \[(16)\]

\( C \) expects to receive the updated \( CL_{TR} \), where \( \text{updated-CL}_{TR} = \text{current-CL}_{T} + \text{returned-amount} \), and \( \text{returned-amount} = CL_M - j_{\text{max}} \). \( C \) compares \( CL_{TR} \) with the received one. If they are matched, she can infer the updated \( CL_{TR} \). Later on, \( C \) can use the bank coupon with the updated \( CL_{TR} \) to make payment with another merchant.

### 4. Extension of the Proposed Protocol to Postpaid Micropayment

In this section, our protocol can be extended to perform postpaid micropayment. Consider PayWord [6], it is based on the assumption that a bank can accept the risk due to overspending problem because most of its clients have good behavior or the bank may require high transaction performance. Based on this assumption, we propose an extension of our protocol for postpaid micropayment by presenting \textit{Payment Initialization Protocol} and \textit{Payment Clearing Protocol}.
4.1 Payment Initialization Protocol

To make a payment to a merchant $M$, a client $C$ runs Setup Protocol to request for authorized total credits $CL_T$. However, under the same assumption as PayWord which states that $C$ is allowed to spend up to credit limit to any merchant, $CL_T$ issued by a bank $B$ stands for credit limit that $C$ is allowed to spend to each merchant instead. In this protocol, $C$, receiving $c$ from $B$ in Setup Protocol, generates a set of coins $\{c_0, \ldots, c_n\}$, where $n = CL_T$, and sends the following message to $M$:

$$C \rightarrow M: \{c_0, CL_T, T_G, h(c_0, ID_M, T_G, Y)\}^X$$  \hspace{1cm} (17)

We can see that $M$ cannot generate this message by herself because she does not have $Y$ which is shared between $C$ and $B$. After retrieving $c_0$ and $CL_T$, $C$ can start the payment with $M$ by following Payment Protocol previously presented in section 3.3.

4.2 Payment Clearing Protocol

After specified period, $M$ collects necessary information and sends the following message to $B$:

$$M \rightarrow B: ID_M, c_{j_{\text{max}}}, T_G, h(c_0, ID_M, T_G, Y)$$  \hspace{1cm} (18)

After receiving the message, $B$ calculates $TCL_c$ where $TCL_c = \{c, T_G\}$ and then calculates $c_0$ from $TCL_c$. $B$ verifies the received $c_0$ with the one it has. If they are matched, $B$ calculates $j_{\text{max}}$ and transfers the amount $j_{\text{max}}$ to $M$’s account.

5. Discussions

5.1 Transaction Security Properties

In this section, we show that the cryptographic technique applied to our protocol satisfies the above transaction security properties. The following message demonstrates how the technique works:

$$B \rightarrow M: h(c_0, T_G, CL_M, X), \{c_0, T_G, SN, CL_M, h(ID_M, SN, CL_TR, T_TR, Y)\}^Z$$  \hspace{1cm} (4)

Transaction security properties for payment systems mentioned in [1, 5] are satisfied as follows:

(i) **Party authentication** is ensured by symmetric encryption with the secret $Z$ and by the secret $Y$ shared between $C$ and $B$. The encryption ensures that either $B$ or $M$ has originated the message, and $Y$ ensures that $B$ is the originator of the message because only $B$ has both $Y$ and $Z$.

(ii) **Transaction privacy** is guaranteed by symmetric encryption.

(iii) **Transaction integrity** is guaranteed by $h(c_0, T_G, CL_M, X)$ forwarded from $C$.

(iv) **Non-repudiation of transactions** is ensured by $h(ID_M, SN, CL_TR, T_TR, Y)$ in that $B$ cannot deny that it has generated $\{c_0, T_G, SN, CL_M, h(ID_M, SN, CL_TR, T_TR, Y)\}^Z$ since $B$ is the only party that holds both $Y$ and $Z$.

5.2 Dispute Resolution

Our protocol offers the ability to resolve disputes among engaging parties in both direct and indirect manners. According to direct dispute resolution, consider the message (5), we can prove
that the bank is the originator of this message since \( h(ID_m, SN, CL_{TR}, T_{TR}, Y) \) can be retrieved by only the client and the bank, but the client does not have \( Z \). Thus, the client is not the originator of the message. However, some messages offer indirect dispute resolution. Consider the message (10) sent from the merchant to the client in \( ECR \) Protocol, although the client can generate this message by herself, she cannot modify the content of the message since it will be later detected by the bank.

5.3 Private Information

In any payment system, the information that is known only by relevant parties such as secret keys, account information, price, or goods descriptions is considered as private information [4]. Revealing such information offers the opportunity to perform attacks or to trace client’s spending. In our protocol, \( c_0 \) and \( c_{j_{\text{max}}} \) are transmitted in encrypted forms compared to signed messages in PayWord [6] and cleartext in PayFair [8]. Moreover, only \( c_j \) is sent from the client to the bank over the air interface. The bank can infer \( c_0 \) from \( c_0 = h(n, TG) \), where \( n \) is the current \( CL_{TR} \), and later sends \( c_0 \) to the merchant in the message (4). Thus, the secrecy of the requested amount is preserved.

5.4 Trust Relationships

In PayFair [8], a client is assumed to fully trust a bank in that the bank will not impersonate as the client since all the client’s secrets are known to the bank. Such assumption is too strong because, practically, the bank may be a payment gateway which is a company monitoring traffic. It may possibly have a conspiracy with an attacker or generate a fake client’s request itself. In our protocol, partial trust between the client and the bank is established. The bank cannot generate the fake request since each client’s message contains components that cannot be generated by the bank.

5.5 Performance Analyses

To demonstrate the practicability of the proposed protocol, we compare our protocol with PayWord [6] and PayFair [8] in terms of transaction performance by focusing on the computation and the numbers of message passes of engaging parties. According to the party’s computation, we mainly focus on the numbers of cryptographic operations applied to engaging parties. We compare our postpaid micropayment protocol with PayWord [6] and our prepaid micropayment protocol with PayFair [8]. Table 1 demonstrates the numbers of cryptographic operations applied to the protocols in 1 transaction. Note that \( n \) stands for the numbers of coins generated by each party.

Moreover, assume that a client receives a bank coupon and performs transactions with 10 merchants. There are 20 times that the client performs the transactions which exceed the limit specified to each merchant. Recall that in PayFair, when the requested payment order exceeds the limit specified to each merchant, the client has to repeat the entire protocol steps to get a new coupon, whereas, in our protocol, the client runs \( ECR \) Protocol to request the bank for extra credits. Table 2 demonstrates the computation of the client of each protocol under the above situation.

From Table 1 and Table 2, in our protocols, only symmetric-key operations and hash functions are deployed compared to public-key operations in PayWord. This infers that our postpaid protocol has better performance than PayWord. Compared to PayFair, our prepaid protocol still has less computation at all engaging parties. It can be seen that, the higher numbers the transactions have been performed, the higher client’s computation PayFair and PayWord have compared to our protocols. This infers that our protocols have better performance than PayFair and PayWord.
Table 1: The numbers of cryptographic operations applied to the proposed protocol, PayWord, and PayFair, respectively, in 1 transaction.

<table>
<thead>
<tr>
<th>Cryptographic Operations</th>
<th>Our Protocol</th>
<th>PayWord</th>
<th>PayFair</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Prepaid</td>
<td>Postpaid</td>
<td></td>
</tr>
<tr>
<td>1. Signature</td>
<td>C</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>M</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2. Signature verification</td>
<td>C</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>M</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3. Symmetric operations</td>
<td>C</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>M</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>4. Hash function</td>
<td>C</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td></td>
<td>M</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>n</td>
<td>n</td>
</tr>
<tr>
<td>5. Keyed-hash function</td>
<td>C</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>M</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 2: The numbers of cryptographic operations at a client applied to the proposed protocol, PayWord, and PayFair, respectively, when the client performs transactions with 10 merchants and there are 20 times that she spends the coins over the limit specified to each merchant.

<table>
<thead>
<tr>
<th>Cryptographic Operations</th>
<th>Our Protocol</th>
<th>PayWord</th>
<th>PayFair</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Prepaid</td>
<td>Postpaid</td>
<td></td>
</tr>
<tr>
<td>1. Signature</td>
<td>-</td>
<td>-</td>
<td>200</td>
</tr>
<tr>
<td>2. Signature verification</td>
<td>-</td>
<td>-</td>
<td>200</td>
</tr>
<tr>
<td>3. Symmetric operations</td>
<td>10</td>
<td>400</td>
<td>-</td>
</tr>
<tr>
<td>4. Hash function</td>
<td>10n</td>
<td>10n</td>
<td>200n</td>
</tr>
<tr>
<td>5. Keyed-hash function</td>
<td>440</td>
<td>400</td>
<td>-</td>
</tr>
</tbody>
</table>

The numbers of message passes affects transaction performance. The higher numbers the messages have been transferred, the longer time the transaction needs to be completed. Moreover, longer processing time causes a mobile user to stay online for longer period which costs higher connection cost. This may not be acceptable by mobile users.

Figure 1: The numbers of messages passes in Payment Initialization Protocol of our proposed (1) prepaid and (2) postpaid protocols, (3) PayFair, and (4) PayWord, respectively.
<table>
<thead>
<tr>
<th>Phases</th>
<th>Our Protocol</th>
<th>PayWord</th>
<th>PayFair</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Prepaid</td>
<td>Postpaid</td>
<td></td>
</tr>
<tr>
<td>Setup</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Payment Initialization</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Payment Clearing</td>
<td>-</td>
<td>1</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 3: The numbers of message passes per transaction of the protocol, PayWord, and PayFair, respectively

From Figure 1 and Table 3, our postpaid protocol has the same numbers of message passes as that of PayWord [6]. However, from Table 1, the proposed protocol has lower party’s computation than PayWord. Therefore, our protocol has better performance than PayWord. Compared to PayFair [8], our prepaid protocol has less numbers of message passes. Moreover, in PayFair, a client is required to contact a bank for issuing a new coupon and generate a new set of coins every time when she runs out of credits, whereas the coupon in our protocol is issued only once and can be used to perform transactions with many merchants. This greatly reduces the computational load at the client side. These features result in better performance than PayFair.

6. Conclusion

In this paper, we have proposed a prepaid micropayment protocol which is able to solve the problems of existing micropayment protocols due to poor performance and security flaws. We applied cryptographic technique that not only reduce parties’ computation, but also satisfies transaction security properties. Our protocol achieves full non-repudiation in that a merchant and a bank cannot impersonate as a client. Satisfying this property provides higher security to the payment system. Moreover, private information of engaging parties are well-protected. In addition, we have shown that our protocol is able to perform postpaid micropayment by proposing an extension which does not affect the existing protocol structure. This results in a general micropayment framework that a user is able to perform both micropayments under the same framework. Our analysis has shown that our protocol has better performance than PayWord [6] and PayFair [8] which results in more applicable to limited capability wireless devices.

7. References