A Secure Account-Based Mobile Payment Protocol

Supakorn Kungpisdan[1], Bala Srinivasan[2], and Phu Dung Le[3]
School of Network Computing, Monash Univesity, Australia
[1]keng@beast.csse.monash.edu.au
{[2]Bala.Srinivasan, [3]Phu.Dung.Le}@infotech.monash.edu.au

Abstract

In this paper, we propose a secure account-based payment protocol which is suitable for wireless networks. The proposed protocol employs symmetric-key operations which require lower computation at all engaging parties than existing payment protocols. The proposed protocol also satisfies transaction security properties provided by public-key based payment protocols such as SET and iKP. The formal analysis illustrates that our protocol achieves the goals of payment protocols. Moreover, the credit-card information is not required to be sent during transactions which results in a security enhancement of the system.

Keywords: *Electronic commerce protocol, mobile payment, cryptographic protocol, credit-card payment*

1. Introduction

Recently, the emergence of wireless communications technology raises concerns about performance and securities of payment systems. Such concerns come from limitations of wireless environments [4, 7, 10]. Firstly, mobile devices are considered to have lower power, storage, and computational capabilities compared to desktop computers. They cannot efficiently perform high computational operations such as public-key encryptions. Although some mobile devices are equipped with special processors [12], performing such operations on them still requires longer processing time. Secondly, wireless networks have less bandwidth and reliability, and higher latencies. Furthermore, the connection cost to wireless networks is considerably higher. Therefore, mobile payments with existing payment protocols are not acceptable by many users.

Several payment protocols were proposed for fixed networks [2, 11]. Nevertheless, they are based on public-key infrastructure (PKI) which is not efficiently applied to wireless networks, that is, a client needs to perform high computational operations, and her mobile device is required to have considerable storage to store public-key certificates. Moreover, during a transaction, each certificate sent to the client has to be verified by a Certificate Authority (CA) located in a fixed network which results in additional communication passes.

In this paper, we propose an account-based payment protocol for wireless networks which overcomes the above limitations. Our protocol satisfies all transaction security properties of payment systems [1] without any public-key operations at engaging parties as in SET [11] and iKP [2] protocols. This results in the reduction of all parties' computation and communication passes. Also, without PKI-based operations, the setup cost for payment infrastructure and the transaction cost are reduced.

We perform an analysis to show that our protocol satisfies all party's requirements for payment transactions which are the goals of iKP protocol. Moreover, the client is not required to trust a payment gateway as required in SET and iKP. Therefore, the client can ensure that her account information will not be leaked to other parties. In addition, we analyze performance of our protocol to show that our protocol has better performance than SET and iKP when applied to wireless networks.

Section 2 describes the existing approaches. Section 3 introduces the proposed protocol. Section 4 analyzes the proposed protocol on security and performance. In section 5, concludes our work.

2. Background

2.1 Symmetric VS Asymmetric Cryptography for Mobile Payment Transactions

Symmetric and asymmetric cryptography are normally used for secure communications among engaging parties. Symmetric cryptography which employs a shared key between each of two parties provides message confidentiality, message integrity, and party authentication whereas asymmetric cryptography which employs private/public keys provides all above security properties including non-repudiation. The non-repudiation property ensures that a party cannot deny the transaction she has originated. This property is very important for financial transactions that are relevant to fund transfers and goods ordering. Normally, it can be achieved by using digital signature. In symmetric-key based protocols, we cannot prove the originator of an encrypted message because the secret is shared between two parties.

However, symmetric-key operations that can be processed much faster than asymmetric ones are likely to be more suitable for wireless networks. Moreover, in any asymmetric-key protocol, public-key certificates have to be verified by CAs which cause additional communication passes.

2.2 General Model for Payment Transactions

A general account-based payment model [5] is composed of 4 involved parties: *client*, *merchant*, *issuer* (client's financial institution), and *acquirer* (merchant's financial institution). An additional party called *payment gateway* acts as a medium between issuers/acquirers at banking private network side and clients/merchants at the Internet side for clearing purpose.

The model introduces 3 primitive payment transactions (see Figure 1). *Payment* is made by a client about the payment to a merchant. *Value Subtraction* is made by the client in order to request a payment gateway (on behalf of an issuer) to deduct the money from the client's account. *Value Claim* is made by the merchant in order to request the payment gateway (on behalf of an acquirer) to transfer money to the merchant's account.

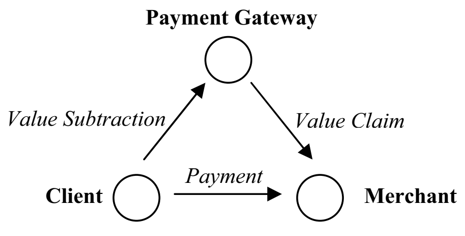

Figure 1. Primitive transactions

2.3 Notations

- *{C, M, PG, I, A}* : the set of client, merchant, payment gateway, issuer, and acquirer, respectively.
- ID_P : the identity of a party *P*. It contains the contact information of *P*.
- *TID* : the identity of transaction including time and the date of the transaction.
- *OI* : order information. *OI = {TID, h(OD, Price)*, where *OD* and *Price* are order descriptions and its amount.
- *Yes/No* : the status of transaction *approved/rejected.*
- *TIDReq* : the request for *TID*.
- *MIDReq* : the request for ID_M.
- $\{M\}_X$: the message *M* symmetrically encrypted with the shared key *X*.
- *h(X)* : the one-way hash function of the message *X*.
- *MAC(X, K)* : *Message Authentication Code* (MAC) of the message *X* with the key *K*.

2.4 Mobile Payments with SET and iKP

Although SET [11] and iKP [2] protocols are successfully implemented on fixed networks, they do not apply well to wireless ones mainly because a client with limited capability mobile device has to perform high computational operations such as public-key encryptions. Moreover, certificates sent to the client have to be verified by a CA. It results in additional message passes. In addition, the client needs to trust a PG since their credit-card information is revealed to the PG in order to be passed to her issuer. In fact, the PG may be a company which is responsible for monitoring the system. It may have a conspiracy with an attacker, or even the merchant.

3. Our Approach

3.1 Initial Assumptions

1) A client has an internet-accessible mobile device.
2) The client shares her credit-card information (*CCI*) with her issuer. *CCI* contains the long-term secret *CCISec* known only by the client and her issuer.
3) The issuer is trusted by the client in that it will not reveal the client's *CCI* to any merchant.
4) A merchant registers herself to a PG. The PG shares the secret *Z* with the merchant. Both of them then generate a set of secrets Z_j, where $j = 1,\ldots, n$ and store them in their terminals. Also, the issuer shares the secret *Y* with the client. Both of them then generate a set of secrets Y_i, where $i = 1,\ldots, n$ and store them in their terminals. *Y* and *Z* can be distributed by using an authenticated-key exchange (AKE) protocol for wireless networks [3, 6] that do not use public-key cryptography. The details of existing AKE protocols for wireless networks can be found in [3, 6].
5) It is easy to compute the hash function *h(x)* from the given *x*, and it is computational infeasible to compute *x* which $y = h(x)$ from the given *y*. Moreover, the MAC algorithm is a fast and secure version.

3.2 Key Generation Techniques

In order to generate the sets of shared keys, we present two efficient key generation techniques. One is used for generating the sets of X_i (shared between a client and a merchant), $i = 1,\ldots, n$, and Z_j, $j = 1,\ldots,n$, from *X* and *Z*, respectively. The others is used for generating the set of Y_i, $i = 1,\ldots,n$, from *Y*. The main concept is to apply one hash algorithm with one-bit cyclic shift of a master secret each time a session key is generated. The details are shown as follows:

Generating X_i and Z_j

$X_1 = h(1\text{-bit-shift-of-}X)$, $X_2 = h(2\text{-bit-shift-of-}X)$,...,
$X_n = h(n\text{-bit-shift-of-}X)$
$Z_1 = h(1\text{-bit-shift-of-}Z)$, $Z_2 = h(2\text{-bit-shift-of-}Z)$,...,
$Z_n = h(n\text{-bit-shift-of-}Z)$

Generating Y_i

$Y_1 = h(1\text{-bit-shift-of-}(CCI, Y))$,
$Y_2 = h(2\text{-bit-shift-of-}(CCI, Y))$, ...,
$Y_n = h(n\text{-bit-shift-of-}(CCI, Y))$

Before making a payment, a client needs to register herself to her issuer and a merchant. After the registration, the client may receive client's wallet software by mail or downloading from the issuer's site. This wallet contains both key generation and payment software. After the wallet is successfully installed, a set of Y_i is generated. To generate a set of X_i, the client needs to run *Merchant Registration Protocol* to register herself to merchant in order to share the secret X with merchant. The details of this protocol will be given in section 3.4.

3.3 Cryptographic Technique

We apply the techniques proposed by [4] and [10] to mobile payment scenario. The concept is that a party can authenticate herself to others by adding a secret which cannot be generated by those participants in a message. The technique can be formalized as the following:

$$\{ Message, Y, MAC[(Message, Y), X_2] \}_{X1}$$

The following example describes how our technique works and its properties. Let $\{X_1, X_2\}$, where $X_1 \neq X_2$, is a set of secrets shared between a client and a merchant, and an issuer has given a secret Y to the client. Note that Y can be any kind of message (hashed or MAC). It can be seen that this message is originated by the client who holds X_1, X_2, and Y whereas the merchant does not have Y. Moreover, it cannot be generated by the issuer since it does not have X_1 and X_2. In the case that a dispute occurs, with the assistance of the issuer, on one hand, the client is able to prove to a verifier that she is the originator of the message. On the other hand, the client cannot deny that she has originated this message.

3.4 The Proposed Protocol

Our protocol is composed of two sub-protocols. In *Merchant Registration Protocol*, a client shares the secret X with a merchant when she newly registers herself to the merchant or updates a set of X_i. After the sets of X_i, Y_i, and Z_j are successfully generated, the client can start *Payment Protocol*. The details of the protocols are given below:

Merchant Registration Protocol

C→M: $\{ID_C, X, n\}_K$
M→C: $h(n, X)$

A client **C** generates X which is to be shared with a merchant **M**. **C** then sends **M** ID_C, a nonce n, and X encrypted with the session key K, generated by running AKE protocol with **M**. **M** then confirms **C**'s registration by sending $h(n, X)$ to **C**. After the completion of the protocol, both of them can generate a new set of X_i by using the same key generation technique.

Payment Protocol

1) **C→M:** ID_C, i, $TIDReq$, $MIDReq$
   M→C: $\{TID, ID_M\}_{Xi}$
2) **C→M:** $\{OI, Price, ID_C, ID_I,$
   $MAC[(Price, h(OI), ID_M), Y_i] \}_{Xi}$,
   $MAC[(OI, Price, ID_C, ID_I), X_{i+1}]$
3) **M→PG:** $\{ MAC[(Price, h(OI), ID_M), Y_i], h(OI), i,$
   $TID, Price, ID_C, ID_I \}_{Zj}$, j, ID_M,
   $MAC [(h(OI), i, TID, ID_C, ID_I), Z_{j+1}]$
4) Under banking private network,
4.1) PG→I: $MAC[(Price, h(OI), ID_M), Y_i]$, $h(OI)$, i,
   TID, $Price$, ID_C, ID_M, $h(Z_{j+1})$
4.2) PG→A: $Price$, ID_M
4.3) I, A→PG: Yes/No, $\{h(OI), Yes/No, h(Z_{j+1})\}_{Yi}$,
   $h(Yes/No, h(OI), h(Y_i))$
5) **PG→M:** $\{ Yes/No, \{h(OI), Yes/No, h(Z_{j+1})\}_{Yi},$
   $h(Yes/No, h(OI), h(Y_i)) \}_{Zj+1}$
6) **M→C:** $\{ \{h(OI), Yes/No, h(Z_{j+1})\}_{Yi} \}_{Xi+1}$

Step 1: **C** and **M** exchange the information necessary to start the protocol.

Step 2: **C** starts making the payment by sending *Payment Request* (referred to the model in section 2.2) to **M**. *Payment Request* contains *OI* used to inform **M** about the goods and price requested. It also contains $MAC[(Price, h(OI), ID_M, h(X_i)), Y_i]$ which is *Value-Subtraction Request* that is to be forwarded to an issuer **I**. It can be noted that, although **M** has X_i, she cannot generate this message since she does not have Y_i used for constructing $MAC[(Price, h(OI), ID_M, h(X_i)), Y_i]$. Thus, we can ensure that the message is really sent from **C**.

Step 3: **M** decrypts the message to retrieve *OI*. **M** then sends a payment gateway **PG** *Value-Claim Request* encrypted with Z_j. *Value-Claim Request* contains the forwarded *Value-Subtraction Request*. Note that ID_C and ID_I are used to identify **C** and **I**, and the index i is used identify the current session key in the set of Y_i.

Step 4: **PG** then sends *Value-Subtraction Request* to **I**. Also, **PG** also sends ID_M and *Price* to notify an acquirer **A** that **M** is the person whom the requested amount *Price* will be transferred to. After the approval, **I** and **A** send the result *approved/rejected* to **PG**. Note that this step is done under the banking private network. Thus, we do not need to concern about its security issue.

Step 5: PG sends *Value-Claim Response*, the approval result, to **M**. Note that the whole message encrypted with Z_{j+1} represents *Value-Claim Response* whereas $\{h(OI), Yes/No, h(Z_{j+1})\}_{Yi}$ represents *Value-Subtraction Response* which will be forwarded to **C**. **M** can check whether or not the message is the response of her request by comparing the received *h(OI)* with her own *OI*. If they are not matched, **M** can reject the transaction. Note that **PG** cannot fool **M** by modifying the approval result because it does not have $h(Y_i)$. **M** can verify *Yes/No* from $h(Yes/No, h(OI), h(Y_i))$ by using *OI* and $h(Y_i)$ previously received from **Step 2**.

Step 6: M encrypts $\{h(OI), Yes/No, h(Z_{j+1})\}_{Yi}$ which represents *Value-Subtraction Response*, with X_{i+1} and sends to **C** as *Payment Response*. **C** decrypts the message to retrieve the results of her requests from both responses.

In next purchases, the client does not have to run *Merchant Registration Protocol* again since she can use other values in the set of X_i to perform transactions until being notified to update the secret *X*. Then the client runs *Merchant Registration Protocol* to get a new *X*. To update *Y* and *Z*, the issuer and the PG send the new secrets to the client and the merchant, respectively, by using an AKE protocol for wireless networks [3, 6].

Note that, after each X_i and Y_i has been used, they are put into all parties' revocation lists in order to prevent the replay of the secrets from both client and merchant.

4. Analyses and Discussions

4.1 Analysis of Our Protocol on Party's Requirements for Payment Transactions

Party's requirements for payment transactions can be considered as the goals of iKP protocol [2]. They are relevant to the ability to prove the authorizations of transactions by particular parties in each party's point of view. The details of party's requirements for payment transactions can be found in [2].

We perform analysis by using Kungpisdan *et al.* (KSL)'s accountability logic [8] to show that our protocol satisfies the party's requirements. Accountability of KSL's logic is based on the belief of a party that a prover can convince a verifier that she is responsible for a transaction. We formalize the party's requirements into proof statements of KSL's logic and use them as proof goals. For example, the client's requirements '**B1**: *a client must not be charged on the payment she has never made.*' can be formalized into:

A1, B1: *I believes I CanProve (*
C authorized value-subtraction(C, I, Price, Date)) to V

The above statement states that the issuer believes that it can prove to a verifier that the client has requested it to deduct the money from her account. If this proof is successful, the issuer can use it to show that the client has really requested it to charge the money from her account, and the issuer will deduct the money if it receives the request from only the client. Other party's requirements can be formalized into the following statements:

A2: *PG believes PG CanProve (*
M authorized value-claim(M, PG, Price, Date)) to V
S1: *M believes M CanProve (*
PG authorized value-claim(M, PG, Price, Date)) to V
S2: *M believes M CanProve (*
C authorized payment(C, M, Price, Date)) to V
B2, B3: *C believes C CanProve (*
I authorized value-subtraction(C, I, Price, Date)) to V
B4: *C believes C CanProve (*
M authorized payment(C, M, Price, Date)) to V

The formal definitions of all above statements were presented in [8, 9]. We do not show the details of the proofs because of the limited space. The results show that our protocol satisfies all party's requirements. As stated in [9], achievement of proving those requirements infers the ability to resolve disputes among parties.

4.2 Security Issues

4.2.1 Transaction Security

From the message format discussed in section 3.3, our protocol satisfies the following transaction securities [1]; *(i) Party authentication* is ensured by symmetric encryption and the secret *Y*. The encryption ensures that either client or merchant has originated the message and *Y* authenticate the client, *(ii) Transaction privacy* is ensured by symmetric encryption, *(iii) Transaction integrity* is ensured by MAC, and *(iv) Non-repudiation of transactions* is ensured by *Y* in that the merchant is able to provide a non-repudiable evidence to prove to other parties that the client has originated the message.

4.2.2 Secret Keys

The concern about key distribution arises in our proposed protocol since the keys shared between parties are needed to be updated periodically or upon requests. Normally, each time when a new key is distributed, even in an encrypted form, it is possible to be retrieved by an attacker. In our protocol, although an attacker successfully retrieves both *X* and *Y*, he cannot impersonate as a client since he does not have *CCI*. In addition, the client is not required to send *CCI* during transactions. *CCI* is used only for generating Y_i which can be done offline.

4.2.3 Trust Relationships

In any transaction, a party should not trust others unless they can provide a proof of trustworthiness. However, in SET [11] and iKP [2] protocols, the client's credit-card information has to be revealed to a PG to be forwarded to an issuer/acquirer. Unfortunately the PG may be a company that monitors the system. It may have a conspiracy with an attacker, or even merchant.

In our protocol, we state the trust relationship between the client and the issuer instead of between the client and the PG since the issuer issues a credit card to the client. Therefore, we do not need to concern about the honesty of the PG as that in SET and iKP protocols.

4.3 Performance Analysis

In this section, we compare our protocol with SET [11] and iKP [2] protocols in terms of performance by focusing on the numbers of cryptographic operations at each party. Table 1 demonstrates the numbers of cryptographic operations at involved parties of each protocol.

Cryptographic Operations		SET	iKP	Ours
1. Public-key encryptions	C	1	1	-
	M	1	-	-
	PG	1	-	-
2. Public-key decryptions	C	-	-	-
	M	1	-	-
	PG	2	1	-
3. Signature generations	C	1	1	-
	M	3	1	-
	PG	1	1	-
4. Signature verifications	C	2	3	-
	M	2	2	-
	PG	1	2	-
5. Symmetric-key encryptions/decryptions	C	2	-	4
	M	-	-	5
	PG	1	-	2
6. Hash functions	C	3	2	2
	M	2	4	-
	PG	-	1	-
7. Keyed-hash functions	C	-	-	2
	M	-	1	2
	PG	-	-	1
8. Key generations	C	-	-	2
	M	-	-	1
	PG	-	-	1

Table 1. The numbers of cryptographic operations of SET, iKP, and our protocol, respectively

We can see that, in our protocol, only symmetric-key operations, including MAC and hash functions are applied, compared to public-key operations in SET and iKP. Less party's computation in our protocol infers better performance than SET and iKP.

In the proposed protocol, the key generation process is required to update keys regularly. However, this would not cause the time consumption problem because the key generation processes can be done offline.

5. Conclusion

We have proposed an account-based payment protocol which is applicable to wireless networks. The symmetric cryptographic technique applied to our protocol not only reduces all parties' computation, but also satisfies transaction security properties, including non-repudiation, provided by public-key payment protocols. It offers the abilities to deal with failures and disputes among parties. We have shown that the proposed protocol has advantages over SET [11] and iKP [2] protocols in that it has lower computation at each party since no public-key operation is required. In our protocol, clients can ensure that their account information will not be compromised by a payment gateway.

As a result, with our proposed protocol, mobile users can have efficient and secure payments, and it may gain more acceptability than existing protocols.

6. References

[1] V. Ahuja, *Secure Commerce on the Internet*, Academic Press, 1996.

[2] M. Bellare, J. A. Garay, R. Hauser, A. Herzberg, H. Krawczyk, M. Steiner, G. Tsudik, E. V. Herreweghen, and M. Waidner, Design, Implementation, and Deployment of the *i*KP Secure Electronic Payment System, *IEEE Journal of Selected Areas in Communications*, 2000.

[3] C. Boyd, P. Montague, and K. Nguyen, Elliptic Curve Based Password Authenticated Key Exchange Protocols, *LNCS Vol. 2119*, 2001, pp. 487-501.

[4] S. Cimato, Design of an Authentication Protocol for GSM Javacards, *LNCS Vol. 2288*, 2002, pp. 355-368.

[5] E. V. Herreweghen, Non-Repudiation in SET: Open Issues, *LNCS Vol. 1962*, 2001, pp. 140-156.

[6] G. Horn and B. Preneel, Authentication and Payment in Future Mobile Systems, *Proceedings of 5th ESORICS'98*, Belgium, 1998, pp. 277-293.

[7] S. Kungpisdan, B. Srinivasan, and P. D. Le, A Practical Framework for Mobile SET Payment, *Proceedings of International E-Society Conference 2003*, pp. 321-328.

[8] S. Kungpisdan, B. Srinivasan, and P. D. Le, Accountability Logic for Mobile Payment Protocols, *To appear in ITCC'2004*, Las Vegas, 2004.

[9] S. Kungpisdan, and Y. Permpoontanalarp, Practical Reasoning about Accountability in Electronic Commerce Protocols, *LNCS Vol. 2288*, 2002, pp. 268-284.

[10] L. M. Marvel, Authentication for Low Power Systems, *Proceedings of IEEE MILCOM 2001.*

[11] Mastercard and Visa, SET Protocol Specifications, 1997. http://www.setco.org/set_specifications.html

[12] B. S. Yee, *Using Secure Coprocessor*, PhD thesis, Carnegie Mellon University, 1994.