國 立 交 通 大 學

資訊科學與工程研究所

碩 士 論 文

適用於可移動裝置的輕量公平付費協定

A Lightweight Fair Payment Protocol for Mobile Devices

研 究 生：李政仲

指導教授：謝續平　教授

中 華 民 國 九 十 七 年 九 月

適用於可移動裝置的輕量公平付費協定

A Lightweight Fair Payment Protocol for Mobile Devices

研　究　生：李政仲　　　　Student：Cheng-Chung Lee

指導教授：謝續平　博士　　Advisor：Dr. Shiuhpyng Shieh

國　立　交　通　大　學

資　訊　科　學　與　工　程　研　究　所

碩　士　論　文

A Thesis
Submitted to Institute of Computer Science and Engineering
College of Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
in

Computer Science

July 2008

Hsinchu, Taiwan, Republic of China

中華民國九十七年七月

適用於可移動裝置的輕量公平付費協定

研究生：李政仲　　　　　　　　　　　　　指導教授：謝續平

國立交通大學　資訊科學與工程研究所

摘　要

大部分的付費協定利用非對稱式加密系統對訊息加密或簽章來達到不可否認性及公平性。由於可移動裝置的計算能力較弱且記憶體較小，所以可能不能做非對稱式加密或是太複雜的運算。為了達到交易的不可否認性及公平性，一個輕量的加密及簽章方法是需要的。在此篇論文中，我們將提出一個適用於移動式裝式的輕量公平付費協定，我們也將分析我們的協定達到幾個重要的性質，包含了公平性、不可否認性、產品驗證、商家認證及隱私性。在我們的方法中，需要一個可信賴的第三方去幫忙初始我們的付費協定且可以排解交易中產生的爭議。

A Lightweight Fair Payment Protocol for Mobile Devices

Student: Cheng-Chung Lee   Advisor: Shiuhpyng Shieh

Department of Computer Science
National Chiao Tung University

Abstract

Most payment protocols use public key cryptosystems to encrypt and sign messages for non-repudiation and fairness. Due to low computational power and limited memory, mobile devices cannot work well with public key cryptosystems or complex operations, such as exponential operations. A lightweight scheme is desirable for both encryption and signature, to achieve non-repudiation and fairness for mobile devices. In this paper, we propose a lightweight fair payment protocol for mobile devices. The proposed protocol is able to attain the important properties, namely fairness, non-repudiation, product validation, and merchant authentication. In our scheme, a trusted third party is needed to initialize the payment protocol, and resolve disputation automatically.

誌　謝

首先感謝指導教授謝續平教授兩年來的諄諄教誨，從一開始對資訊安全的領域一竅不通，慢慢的了解到安全的重要性。除了研究上外，由於計畫的關係，也學習到如何跟別人合作，以及如何領導一個團隊。對於未來，謝老師也提供許多寶貴的意見和人生經歷，我會記得老師的教誨，做自己有興趣的事情並且享受它，努力、不要輕易放棄。另外，感謝實驗室的碩二同學們(繼偉、鼎鈞、佳貞和經偉)，在我研究的過程中給予許多寶貴的意見，在我論文遇到瓶頸的時候，幫我指引方向，讓我能順利完成；也很感謝學長們分享許多經驗、給我信心；感謝碩一的學弟妹們(雨芊、家維、家銘、秉翰和蘇偉)、可愛的助理們(明華和順瑩)和工程師(Michael 和季榆)，多虧了你們撐住實驗室，讓我們能專心準備論文口試。

最後要感謝我的家人，在這段時間照顧我、包容我，讓我無後顧之憂的專注於研究。祝福所有人，事事順心如意！

Table of Contents

List of Figures

List of Tables

1. Introduction

With the development of wireless communications technology, we can access the Internet via wireless devices, e.g. mobile phones or PDAs. It is more convenient for users to use online services on the go. We can download MP3s, send emails, browse web pages, and so forth. Electronic commerce is also a common service. It consists of the buying and selling of products or services over the Internet.

To support electronic payments over the Internet, we need online payment protocols. A payment scenario commonly involves a customer and a merchant. The customer browses a sold list through the Internet and purchase products from the merchant using online payment protocols. SET [3] and iKP [2] are the most well-known payment protocols.

An important issue is how to achieve fairness. That is, either the customer obtains products he or she wants and the merchant obtains money, or neither of them does. Many fair payment protocols had also been proposed [21][22][23][24][25][26]. They utilize public key cryptosystems to sign on the messages as evidences of payment. However, some limitations of mobile devices cause inefficient performance using these payment protocols. Due to low computation power of mobile devices, public key cryptography may not be applied. Because of limitative memory, the key length and the amount of keys are restricted. Lastly, wireless networks are more dangerous than wired networks since it’s easy to eavesdrop and forge messages.

In mobile commerce (m-commerce), we separate the existing payment protocols for mobile devices into three classifications: (1) Mobile Agent Based Payment Protocols [9][10][11][12][13][14][15], (2) Smart Card Based Payment Protocols [16][17][18][19], and (3) Lightweight Cryptography Based Payment Protocols

[1][4][5][6][7][27]. Mobile agent based payment protocols utilize mobile agents to help mobile users to fulfill transactions. Smart card based payment protocols employ smart cards to sign messages. Lightweight cryptography based payment protocols only use symmetric key cryptographies and hash functions, so they are suitable for mobile devices. Moreover, lightweight cryptography based payment protocols don't need additional infrastructures (e.g. mobile agents or certificate authorities (CA)) or devices (e.g. smart cards). However, the existing lightweight cryptography based payment protocols are not satisfying fairness or non-repudiation.

In this paper, we present a lightweight fair payment protocol which, in addition to being lightweight, satisfies the following security requirements, including the fairness property. Assume A trades with B, and A behaves correctly.

R1. Non-repudiation: After completing a payment protocol, A will be able to prove

R1a. Non-repudiation of Origin: the origin of all messages sent from B, and

R1b. Non-repudiation of Receipt: that B received messages form A.

R2. Fairness: Two different level of fairness are possible,

R2a. Strong Fairness: When the protocol has completed, either each party receives the expected items, or neither party obtains any items.

R2b. Weak Fairness: When the protocol has completed, either strong fairness is achieved, or A can prove to an arbiter that B has received A's items without any further intervention from A if A doesn't obtain the expected items.

R3. Validation: One of the following requirements must be satisfied.

R3a. Product Validation (also known as Validated Receipt proposed by

[20][21]): (If a customer pays first) The product validation property is satisfied if a customer is able to ensure with the scope of the protocol and before the customer pays for a product, that the product the customer is about to receive from a merchant, is the same as the product the customer intended to purchase.

R3b. Payment Receipt Validation: (If a merchant sends products first) The product receipt validation property is satisfied if a merchant is able to ensure with the scope of the protocol and before the merchant sends products, that the payment receipt the merchant is about to receive from a customer, is the same as the payment receipt they both agree before.

R4. Authentication:

R4a. Customer Authentication: The customer authentication property is satisfied if a merchant is able to ensure with the scope of the protocol and before the merchant sends products, that the customer is the customer to pay for products.

R4b. Merchant Authentication: The merchant authentication property is satisfied if a customer is able to ensure with the scope of the protocol and before the customer pays for a product, that the merchant is the authority to sell the product.

R5. Privacy: In a payment protocol, customers' privacy, which includes purchase information, secrete information (e.g. customers' credit-card information), and payment information, isn't revealed.

2. Related Works

In this section, we provide brief related works about fair exchange protocols and payment protocols for mobile devices.

2.1. Fair exchange protocol

Payment protocols are a kind of exchange protocols. Exchange protocols can be used to exchange digital contents, while payment protocols are for electronic commerce. Fair exchange guarantees that at the end of the protocol, either both parities receive each other’s items or none do. A fair exchange protocol is impossible without trusted third parties [34]. Fair exchange protocols can be broadly categorized into two types: (1) Protocols with an online trusted third party (TTP) [28][29], (2) Protocols using an offline TTP (known as optimistic fair exchange protocols) [26][30][31][32].

Fair exchange protocols with an online TTP need a TTP directly involved in every exchange. In online third party protocols, the exchange is achieved via a trusted third party. Each party submits its own item to the TTP and the TTP passes the item to the recipient. The third party is always online. If the protocol needs an online TTP, the protocol can’t work when the TTP fails.

In fair exchange protocols using an offline TTP, two parties want to exchange something. One party takes a risk by sending its own item to the other first. The receiver either delivers its own item or tries to cheat by sending nothing. If both parties behave honestly, the protocol is complete and they obtain their expected items. Otherwise, a dispute happens and the trusted third party resolves this dispute by sending the correct items to them. By this way, it lightens the loading of the TTP and reduces network traffic.

2.2. Payment protocols for mobile devices

Due to some limitations of mobile devices and wireless environment, Internet applications cannot be applied. Mobile devices are assumed to have low-powered and computational capability. The bandwidth and reliability of wireless networks are lower than these of the fixed ones. We separate payment protocols for mobile devices into three classifications: (1) Mobile Agent Based Payment Protocols [9][10][11][12][13][14][15], (2) Smart Card Based Payment Protocols [16][17][18][19], and (3) Lightweight Cryptography Based Payment Protocols [1][4][5][6][7][27].

A mobile agent can be defined as a software element, capable of migrating from one computer to another, to execute a set of task on behalf of its owner. Mobile agent based payment protocols migrate some computing from mobile devices to another ones. The customer sends a request including account information and purchase order to a mobile agent, and then the mobile agent fulfills the transaction with the merchant. Mobile agents lift the computation burden of mobile devices and reduce communication overhead. However, mobile agents must be trusted.

Data on a smart card is said to be relatively secure because it is difficult to extract encrypted data from the outside and to alter it. The mobile phone with its integrated SIM card is an ideal bearer for the digital signature or encryption of a PKI system. To use smart card to sign payment messages is secure, but computation overhead and power consumption are a problem.

Lightweight cryptography based payment protocols employ symmetric key operations and hash functions, not public key cryptosystems. Mobile devices don’t need complex operations, but the strong fairness property is not satisfied. We are about to proposed a lightweight cryptography based payment protocol and it achieve the strong fairness property.

3. Basic Schemes

In this section, we introduce the three foundations of our proposed scheme for our security requirement. They are the lightweight encryption scheme, the lightweight signature scheme, and the lightweight merchant authentication scheme. The lightweight encryption scheme protects the customer’s purchase information and secret information, and ensures that each transaction uses the different session key, and it doesn’t affect another transaction even if the certain session key is compromised. The lightweight signature scheme is used to authenticate messages without expensive cost. The merchant authentication scheme is used to ensure that the merchant have rights of selling goods, and the receiver is about to obtain correct products.

3.1. Lightweight encryption Scheme

In order that don’t to expose some information during transactions, we need a lightweight encryption scheme to encrypt all messages. The information may be the customer’s purchase information, secrete information (e.g. the customer’s credit-card information (CCI)), or payment information. The lightweight encryption scheme consists of the lightweight encryption and decryption technique, and the lightweight key generation technique. We adopt a symmetric-key cryptosystem, such as AES, as the lightweight encryption and decryption technique. The lightweight key generation technique regards the security of shared keys. We hope that the security of overall systems and customers’ privacy are threatened if a session key is compromised.

S. Kungpisdan, P. D. Le, and B. Srinivasan [8] presented a session key generation technique for internet transactions that eliminates the need of storing long-term shared key which makes the system insecure against key compromise during transactions. In this scheme, if a certain session key is compromised, the security of the system is not affected because the master key (possible the customer's credit-card information) is not revealed. Therefore, we feel this scheme also resists Man-in-The-Middle attacks and replay attacks because the session keys are different on different transactions.

We will introduce briefly the limited-used key generation scheme. Alice and Bob share the master key, K_{AB}. K_{AB} can be assumed to be never expired. The distributed key, DK is anther shared key between Alice and Bob, but DK needs to be updated periodically or upon their request. The following steps show how to generate session keys.

1. Alice generates the key DK and distributes it to Bob through a secure channel.
2. Alice and Bob generate a set of preference keys K_i, where i=1,…,m.

   $K_1=h(DK,K_{AB}),K_2=h(DK,K_1),\ldots,K_m=h(DK,K_{m-1})$
3. Alice generates a random number r and sends it to Bob. They both select two preference keys. One is K_{Mid1}, where K_{Mid1} is the middle key among $\{K_1,\ldots K_w\}$, $w=r$ mod m. The other is K_{Mid2}, where K_{Mid2} is the middle key among $\{K_1,\ldots,K_{Mid1}\}$. Then, they calculates the session initialization key, SIK, where

   $SIK=h(K_{Mid1},K_{Mid2})$
4. Alice and Bob generate a set of session keys SK_j, where j=1,…,n.

   $SK_1=h(SIK,DK),SK_2=h(SIK,SK_1),\ldots,SK_n=h(SIK,SK_{n-1})$

After the set of SK_j has been generated, Alice and Bob can make use of them in several ways. Firstly, they can send SK_j to each other to authenticate themselves. Secondly, SK_j can also be used as an encrypting key. Thirdly, SK_j can be used as a key for keyed-hash function, HMAC, to verify the message from each other. The session key update and the distributed key update are also presented in detail in [8].

We will employ this method on our proposed scheme. Although the session key, SK_j, can be used to authenticate himself/herself or verify messages, they can't be as evidence to an arbiter. Alice sends SK_j to Bob when Alice wants to authenticate herself, but Bob also is able to generate SK_j so that SK_j can't be as the evidence to authenticate Alice. In the same way, Bob can't say that messages are sent by Alice because he verifies them with HMAC. We only use the second use (to encrypt messages) described above. By this way, every transaction uses different keys to encrypt messages and the master key (possible the customer's CCI) is not compromised or revealed even if a session key is compromised.

3.2. Lightweight Signature

Everyone first raises the asymmetric cryptographic mechanisms like RSA digital signatures when we discuss how to sign messages. However, the asymmetric cryptographic mechanisms cost too expensive that some mobile devices with low computation power don't work.

A message authentication code (MAC) is another way. An authentication tag derives by applying an authentication scheme, together with a secret key, to a message. MAC is an efficient symmetric cryptographic primitive for two-party authentication. However, the receivers also knows the MAC key and could impersonate the sender and forge messages. This problem brings out that MAC tags are not sufficient to be as evidence because two parties are able to forge messages. That is, both sender and receivers can sign messages.

S. M. Chang, S. P. Shieh, W. W. Lin, and C. M. Hsieh [33] proposed an efficient broadcast authentication scheme. The scheme also is one time signature scheme. We is about to introduce the scheme briefly.

The scheme uses Merkle hash trees for authenticating messages. First, the sender must generate his/her private keys and public keys. The private keys are *t* random numbers generated by a pseudorandom generator. We take *t* random numbers as the input to one way hash function to generate *t* hash values. Then, we separate *t* hash values into *d* group. So there are *t/d* values in each group. Finally, we use these *t/d* values as the leaves of a binary tree and compute each intermediate node as the hash of the concatenation of the two child values. Thus, we can get *d* Merkle hash trees, whose roots are our public keys. The public keys are distributed to receivers. In Figure 1, we can see one of *d* Merkle hash trees.

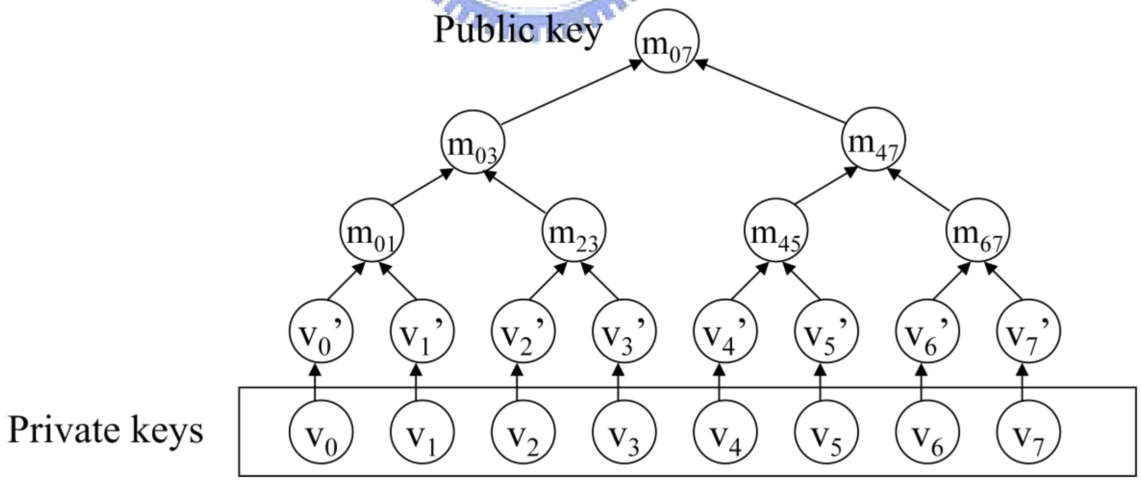

Figure 1. Merkle hash tree

When to sign the message *m*, we first compute *h* = *hash(m)*. Then, we separate the hash value *h* into *k* pieces and regard these pieces as integers, so we get $(i_1, i_2, \ldots, i_k)$, where i_x is between 0 and *t*-1. Each integer is an index of private keys,

$(v_0, v_1, \ldots, v_{t-1})$. Therefore, we can pick *k* private keys. These *k* picked private keys $(v_{i_1}, v_{i_2}, \ldots, v_{i_k})$ and their authentication paths are used as the signature of this message *m*. The authentication path of the leaf is the values of all nodes that are siblings of nodes on the path between the leaf and the root. For example, in the Figure 2, if there is $i_y\ (i_y = 0)$ in $(i_1, i_2, \ldots, i_k)$, its authentication path is (v_1', m_{23}, m_{47}). In this case, the private key, v_0, and its authentication path, (v_1', m_{23}, m_{47}) are sent to the receiver.

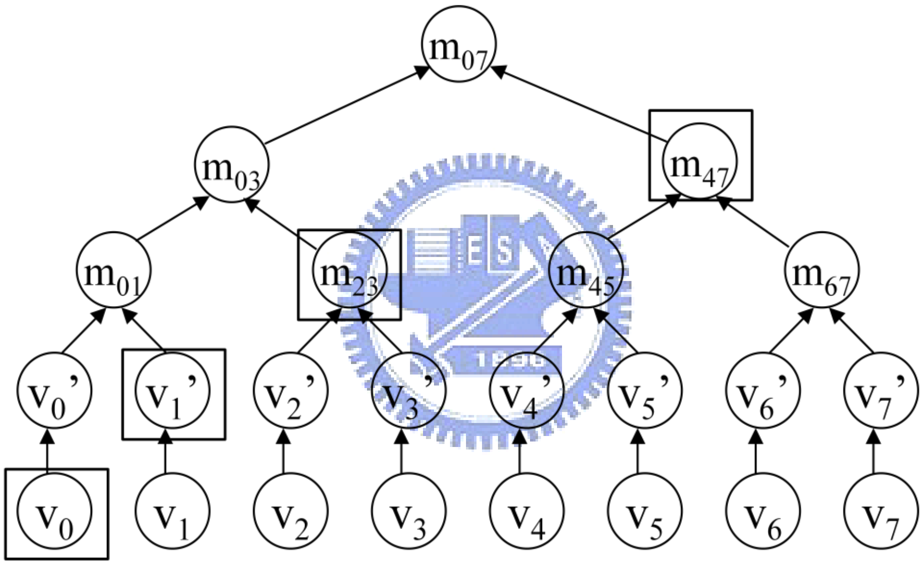

Figure 2. private key and its authentication path

Once the receiver receives a message *m'* and its lightweight signature, he needs to ensure that the message come from the authenticated sender by verifying the signature. First, he computes *h'=hash(m').* Then, we separate the hash value *h* into *k* pieces and regard these pieces as integers, $(i_1, i_2, \ldots, i_k)$. Each integer is an index of private balls, $(v_0, v_1, \ldots, v_{t-1})$. Second, receivers verify each corresponding private key with its authentication path and the corresponding public key. For example, in Figure 2, if there is $i_y\ (i_y = 0)$ in $(i_1, i_2, \ldots, i_k)$, its authentication path is (v_1', m_{23}, m_{47}). The receiver compute v_0*=*hash*(v_0), m_{01}*=*hash*(v_0*,v_1'), m_{03}*=*hash*(m_{01}*,m_{23}), and m_{07}*=*hash*(m_{03}*,m_{47}). If m_{07}*=m_{07}, where m_{07} is the one of the public keys given previously by the sender, the piece is correct. If all pieces are correct, the receiver believes the message is sent form the authenticated sender.

We will use the scheme to get the lightweight signature. As the certain public keys are used, we are about to replace them by sending the new public keys from the sender. In this way, it avoids an attacker collects the private keys and their authentication path and then fakes messages.

3.3. Lightweight Merchant Authentication Scheme

Before the customer is about to pay for the product, he must check two properties. One is the validated receipt property. The other is the merchant authentication property. Thus, the customer verifies that the merchant indeed possesses the product he would like to buy, and the merchant indeed has the authority of the product.

I. Ray, I. Ray, and N. Natarajan [20][21] proposed the validated receipt property and proposed a resolution. The definition of the validated receipt property is shown above (see Chapter 1). Before introducing the resolution and a possible attack, we now present the theory of cross validation. The theory of cross validation also proposed by I. Ray, I. Ray, and N. Natarajan [20][21] is based on RSA-like cryptography to achieve the validated receipt property.

Definition 1. The Eular's totient function $\phi(N)$ is defined as the number of integers that are less than N.

1. $\phi(N) = N - 1$ if N is prime.

2. $\phi(N) = \phi(N_1)\phi(N_2)\cdots\phi(N_k)$ if $N = N_1 N_2 \cdots N_k$ and N_1, N_2, …, N_k are pairwise relatively prime.

Theorem 1. (Euler's theorem) For every *a* and *N* that are relatively prime,

$$a^{\phi(N)} \equiv 1 \bmod N$$

Corollary 1. If $0 < m < N$ and $N = N_1 N_2 \cdots N_k$ and N_1, N_2, …, N_k are primes,

$$m^{x\phi(N)+1} \equiv m \bmod N$$

Definition 2. A key *K* is defined to be the ordered pair <*e, N*>, where *N* is a product of distinct primes, and e is relatively prime to $\phi(N)$. The inverse of a key *K*, denoted by K^{-1}, is an ordered pair <*d, N*>, satisfying

$$ed \equiv 1 \bmod \phi(N)$$

Definition 3. The encryption of a message *m* with the *K*=<*e, N*>, denoted as [*m, K*], is defined as

$$[m, <e, N>] = m^e \bmod N$$

Definition 4. Two keys K_1=<e_1, N_1> and K_2=<e_2, N_2> are said to be compatible if $e_1 = e_2$, and N_1 and N_2 are relatively prime.

Definition 5. If two keys K_1=<e_1, N_1> and K_2=<e_2, N_2> are compatible, the product key, $K_1 \times K_2$, is defined as <*e*, N_1N_2>

Lemma 1. For positive integers *a*, N_1 and N_2,

$$(a \bmod N_1 N_2) \equiv a \bmod N_1$$

Theorem 2. For any two messages m_1 and m_2, such that

$$[m_1, K_1 \times K_2] \equiv [m_2, K_1] \bmod N_1 \text{ if and only if } m_1 = m_2$$

$$[m_1, K_1 \times K_2] \equiv [m_2, K_2] \bmod N_1 \text{ if and only if } m_1 = m_2$$

where two keys K_1=<e, N_1>, K_2=<e, N_2> are compatible, and $K_1 \times K_2$ is the product key <e, N_1N_2>.

The proof of Theorem 2 is shown in [21] and to refer interested readers to [21].

The proposed protocol by [20][21] achieves the validated receipt property using the results of Theorem 2. The simplified protocol is shown in Figure 3. The trusted third party TP generates keys K_1=<e, N_1> and K_1^{-1} and provides K_1 to the merchant M. The trusted third party also calculates L=[m, K_1] and stores it. If the customer C would like to purchase the product m, the trusted third party gives L to the customer, where [m, K_1] is the encryption of the product. The customer keeps it for future validation of the product received during the transaction.

When the customer begins to trade with the merchant, the customer sends his purchase order to the merchant. The merchant chooses the second set of keys (K_2, K_2^{-1}) such that K_2=<e, N_2> is compatible with K_1 according to Definition 4, and selects a random number r, where r is relatively prime to N_2. Then, the merchant provides the customer with L_1=[mr, $K_1 \times K_2$], L_2=[r, K_1]. When the customer obtains L_1 and L_2, he verifies

$$[mr,\ K_1 \times K_2] \equiv [m, K_1] \cdot [r, K_1] \bmod N_1$$

If satisfied, the customer believes the merchant has the product ordered by him/her according to Theorem 2. After the customer has paid for the product, the merchant sends K_2^{-1} and r^{-1}, where r^{-1} is the multiplicative inverse of r modulo N_1. Using the decrypting key K_2^{-1} (to decrypt (L_1 mod N_2)), the customer obtains mr mod N_1. Multiplying this by r^{-1}, the customer can retrieve m.

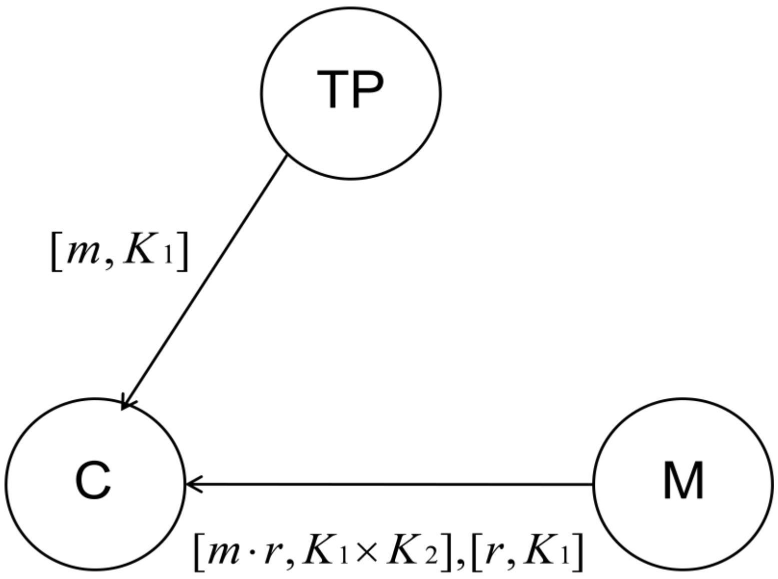

Figure 3. protocol presented by I. Ray, I. Ray, and N. Natarajan

However, the protocol is attacked by a malicious merchant. The malicious merchant first pretends to want to buy the purchase, and then gets L=[m, K_1] form the trusted third party TP. After the malicious merchant obtains L, he runs Algorithm 1 follows.

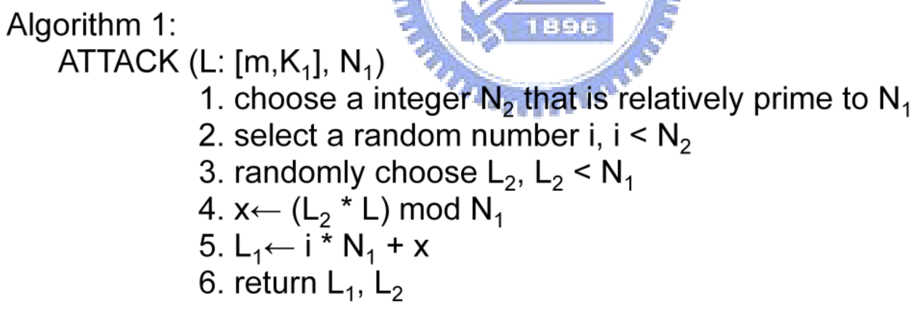
```
Algorithm 1:
    ATTACK (L: [m,$K_1$], $N_1$)
            1. choose a integer $N_2$ that is relatively prime to $N_1$
            2. select a random number i, i < $N_2$
            3. randomly choose $L_2$, $L_2$ < $N_1$
            4. x← ($L_2$ * L) mod $N_1$
            5. $L_1$← i * $N_1$ + x
            6. return $L_1$, $L_2$
```

Figure 4. attack algorithm

After Algorithm 1, we can have $L_1 < N_1N_2$, and $L_1 \equiv L \cdot L_2$ mod N_1. The malicious merchant can send L_1 and L_2 to honest customers. Thus, the malicious merchant can cheat customers, and customers believe the malicious merchant has the real product by verifying $L_1 \equiv L \cdot L_2$ mod N_1.

In order to solve this problem, we improve the protocol. We also replace m with the product key k, which is a symmetric key to encrypt the product m. When the merchant register the product to the trusted third party, the trusted third party generates keys K_1=<e, N_1> and K_1^{-1} and provides K_1 to the merchant. Once the customer would like to buy the product, the trusted third party generates a random number r, calculates $[k \cdot r, K_1]$, and sends r to the merchant and $[k \cdot r, K_1]$ to the customer. When the customer begins to trade with the merchant, the merchant chooses the second set of keys (K_2, K_2^{-1}) such that K_2=<e, N_2> is compatible with K_1 and N_2 is relatively prime to r. Then, the merchant computes $[k \cdot r, K_1 \times K_2]$, and transmits it to the customer. The customer verifies that $[k \cdot r, K_1 \times K_2] \equiv [k \cdot r, K_1] \bmod N_1$. If satisfied, the customer not only believes the merchant has the true product, and believes the merchant has the authority of the product because the merchant doesn't generate $[k \cdot r, K_1 \times K_2]$ if not to possess the product key k and the key K_1. Finally, the merchant gives k to the customer after the customer has paid for the product. If the device the customer uses is not restricted with low computation power, the merchant could send K_2^{-1} and r^{-1} to the customer. Because K_1 may be used to encrypt many products, we can't send K_1^{-1} to the customer.

4. Proposed Payment Protocol

There are four parties involved in our proposed protocol. They are the customer C, the merchant M, the trusted third party T, and the payment gateway P. The customer buys something from the merchant. The trusted third party plays an arbiter and the payment gateway copes with all matters about payment.

4.1. Assumption

We make the following assumptions in our payment protocol.

1. The trusted third party T and the payment gateway P are trusted by all parties.
2. The customer and the merchant do not trust each other.
3. The communication channel between any party and T (or P) is resilient, but the channel between C and M may controlled by a attacker.
4. The public keys of the merchant, the trusted third party, and the payment gateway are known by all parties.
5. All encryption are strong enough that the receiver can't decrypt encrypted messages without the appropriate key.
6. The hash function *h()* is a collision-resistant one-way hash function.
7. The digital signature $(m_1, m_2, \cdots)_{K_A^{-1}}$ is $(h(m1, m2, \cdots))_{K_A^{-1}}$.
8. Each party keeps a copy of transaction information that is the evidence till the information is no longer needed.

4.2. Notation

Before to introduce our proposed scheme, we introduce the notations used in the protocol first. Table 1 lists the notations and their interpretations.

C, M, P, T	customer, merchant, payment gateway, and trusted third party
ID_C, ID_M	identities of C and M
TID, *TIDReq*	identity of transaction, and request for TID
k	theproduct key to encrypt the product
r	random number chosed by T
m	theproduct the customer purchases encrypted with the product key *k*
PI	identity of product *m*
PO	purchase order used by the customer to order product *m* (*PO* contain a time stamp and *TID*)
a,*b*,*c*	random numbers for key generation s
K_{AB}	the master key A share with B
SK_{AB}	session key between A and B
K_A, K_A^{-1}	public key and private key of A
Ack	the status of transaction approved/rejected (*yes* or *no*)
Xaccount	X's bank account
h()	the one - way hash fuction
$MH_{AB}()$	lightweigh t signature sent to B by A
MH_{AB}-*INIT*	the public keys of the Merkle hash trees sent to B by A
MH_{AB}-*NEW*	the new public keys of the Merkle hash trees replace used public keys

Table 1. notations

4.3. Prelude

$M \rightarrow T$	$m,(k,PI,ID_M)_{K_T},(h(m),k,PI,ID_M)_{K_M^{-1}}$
$T \rightarrow M$	$(ID_M,PI, Ack,K_1)_{K_M},(ID_M,PI, Ack,K_1)_{K_T^{-1}}$
$B \rightarrow C$	$CCI,DK,MH_{PC}\text{-}INIT$
$C \rightarrow B$	$MH_{CP}\text{-}INIT$
$T \rightarrow C$	$DK,K_{CT},MH_{TC}\text{-}INIT$
$C \rightarrow T$	$MH_{CT}\text{-}INIT$

Before the payment protocol begins, we assume that some steps have already completed to set up the environment in which the protocol works. The prelude is shown in Figure 2. The merchant M must register its product to the TTP T. M send its identity ID_M, the encrypted product *m* with the product key *k*, the product key *k*, the product identity *PI*, and the signature represent that M, not other one, really want to register. T verifies the signature, and checks if *m* is encrypted with the *k* and *h(m)* is correct. T generates the key pair (K_1, K_1^{-1}), and stores the product identity *PI*, the product key *k*, and the key pair (K_1, K_1^{-1}), that is used to encrypt the key *k*. T sends an acknowledgement and K_1 to M.

In addition to merchant registration, the customer, who wants to trade with his/her mobile devices, must apply to his/her bank. The bank and the customer exchange secrets, such as credit-card information (*CCI*), the distributed key *DK*, and the public keys of Merkle hash trees used to generate lightweight signatures, through a secure channel. In the same way, the TTP and the customer also exchange information, which contains the distributed key *DK*, and the public keys of Merkle hash trees. Finally, the TTP, the bank, and the customer would compute the preference keys.

4.4. Payment Protocol

Our proposed protocol includes four parties: the customer C, the merchant M, the payment gateway P, and the on-line trusted third party T. It is composed of three sub-protocols: exchange sub-protocol and resolve sub-protocol.

Exchange Sub-protocol

A customer browses an Internet e-commerce website, and reviews the products. Once he/she find the products he/she would like to buy, the customer clicks the "check out" button to initiate the exchange sub-protocol. All messages of the exchange sub-protocol are shown in the Figure 3. The exchange sub-protocol includes four steps.

$$\begin{array}{ll}
C \rightarrow T: & ID_C, a, TIDReq, (ID_M, PI, MH_{CM}\text{-}INIT, K_{CM}, MH_{CT}\text{-}NEW)_{SK_{CT}}, \\
 & MH_{CT}(ID_M, PI, MH_{CM} - INIT, K_{CM}, MH_{CT} - NEW) \\
T \rightarrow M: & (TID, MH_{CM}\text{-}INIT, K_{CM})_{K_M}, (TID, MH_{CM}\text{-}INIT, K_{CM})_{K_T^{-1}} \\
M \rightarrow T: & (TID, b, MH_{MC}\text{-}INIT)_{K_T}, (TID, b, MH_{MC}\text{-}INIT)_{K_M^{-1}} \\
T \rightarrow C: & (TID, r, [k \cdot r, K_1], h(m), b, MH_{MC}\text{-}INIT, MH_{TC}\text{-}NEW)_{SK_{CT}}, \\
 & MH_{TC}(TID, r, [k \cdot r, K_1], h(m), b, MH_{MC}\text{-}INIT, MH_{TC}\text{-}NEW) \\
C \rightarrow M: & TID, (r, PO, MH_{CM}\text{-}NEW)_{SK_{CM}}, MH_{CM}(TID, r, h(PO)) \\
M \rightarrow C: & m, (M_{account}, TID)_{K_P}, (TID, h(PO))_{K_M^{-1}}, (TID, [k \cdot r, K_1 \times K_2], MH_{MC}\text{-}NEW)_{SK_{CM}}, \\
 & MH_{MC}(h(PO), (M_{account}, TID)_{K_P}, [k \cdot r, K_1 \times K_2], MH_{MC}\text{-}NEW) \\
C \rightarrow P: & ID_C, c, (M_{account}, TID)_{K_P}, (TID, h(PO))_{K_M^{-1}}, (TID, ID_M, PO, MH_{CP}\text{-}NEW)_{SK_{CP}}, \\
 & MH_{CP}(TID, h(PO), MH_{CP}\text{-}NEW) \\
P \rightarrow C: & (TID, Ack, h(PO))_{K_P^{-1}}, (TID, Ack, MH_{PC}\text{-}NEW)_{SK_{CP}}, MH_{PC}(TID, Ack, h(PO)) \\
C \rightarrow M: & (TID, Ack)_{SK_{CM}}, (TID, Ack, h(PO))_{K_P^{-1}} \\
M \rightarrow C: & (k)_{SK_{CM}}
\end{array}$$

Step 1 includes four messages. In step 1, the customer C and the merchant M exchange some secrets used in following steps through the TTP T, and T give the identity of the transaction *TID* to C and M.

Message 1:

$$\begin{aligned} C \rightarrow T: \quad & ID_C, a, TIDReq, (ID_M, PI, MH_{CM}\text{-}INIT, K_{CM}, MH_{CT}\text{-}NEW)_{SK_{CT}}, \\ & MH_{CT}(ID_M, PI, MH_{CM} - INIT, K_{CM}, MH_{CT} - NEW) \end{aligned}$$

C initiates the exchange sub-protocol by sending T a message, which contains major things: (1) a TID request, *TIDReq*, (2) the Merchant *ID*, ID_M, (3) the identity of the product he wants to buy, *PI*, (4) the master key shared by C and M, K_{CM}, and (5) C's public keys of Merkle hash trees he want to share with M, *MHCM-INIT*.

a is a random number used to generate the session key between C and T. In following steps, b and c are same as a.

MHCT-NEW is the new public keys to replace old public keys that have been used to sign this message to T by C. MHAB-NEWs in the following messages are the same, so we don't illustrate them.

Message 2:

$$T \rightarrow M: \quad (TID, MH_{CM}\text{-}INIT, K_{CM})_{K_M}, (TID, MH_{CM}\text{-}INIT, K_{CM})_{K_T^{-1}}$$

After T receives Message 1, he checks the lightweight signature first. If the signature is correct, T stores ID_C, IDM, PI, K_{CM}, and MHCM-INIT for resolving the dispute in the future, and sends TID, K_{CM}, and MHCM-INIT to M.

Message 3:

$$M \rightarrow T: \quad (TID, b, MH_{MC}\text{-}INIT)_{K_T}, (TID, b, MH_{MC}\text{-}INIT)_{K_M^{-1}}$$

When the merchant receives Message 2 from T, it verifies the signature of T.

Subsequently, M sends its public keys of Merkle hash trees to TTP T and endorses it if M agrees to trade.

Message 4:

$$\begin{aligned} T \rightarrow C: \quad & (TID,[k\cdot r,K_1\times K_2],h(m),b,MH_{MC}\text{-}INIT,MH_{TC}\text{-}NEW)_{SK_{CT}}, \\ & MH_{TC}(TID,[k\cdot r,K_1\times K_2],h(m),b,MH_{MC}\text{-}INIT,MH_{TC}\text{-}NEW) \end{aligned}$$

It store M's public keys of Merkle hash trees, MHMC-INIT, for dealing with the debate in the future once T receives the message from M. The TTP would generate a random number r and calculate $[k\cdot r,K_1]$. This $[k\cdot r,K_1]$ will be used to verify that the merchant is the authority to sell the product and it really has the product. h(m) will be used to ensure the integrity of the encrypted product, m, later. Subsequently, T sends r, $[k\cdot r,K_1]$, h(m), and MHMC-INIT to C.

After Step 1 is completed, the customer starts to trade with the merchant. In Step 2, the customer sends its order for goods, and check that the merchant is the authority of the product. The merchant sends its bank account and signs the order of the customer if it accepts the order.

Message 5:

$$C \rightarrow M: \quad TID,(r,PO,MH_{CM}\text{-}NEW)_{SK_{CM}},MH_{CM}(TID,r,h(PO))$$

C gets TID, b, r, $[k\cdot r,K_1]$, h(m), and MHMC-INIT from Message 4. Because C doesn't have the key, K_1, he doesn't derive the product key, k. The customer utilizes K_{CM} and b to compute the session key, SK_{CM}. SK_{CM} is used to encrypt messages between C and M. If C wants to buy the product indeed, he sends TID, r, and the purchase order (PO). C also uses lightweight signature scheme to sign h(PO), representing the agreement on the order. The purchase order, PO, includes the follow information:

(1) the identity of the product, PI

(2) the amount

(3) the price

(4) the time stamp and the nonce

(5) other order information

Message 6:

$$\begin{aligned} \mathrm{M} \rightarrow \mathrm{C}: \quad & m, (M_{account}, TID)_{K_P}, (TID, h(PO))_{K_M^{-1}}, (TID, [k \cdot r, K_1 \times K_2], MH_{MC}\text{-}NEW)_{SK_{CM}}, \\ & MH_{MC}(h(PO)), (M_{account}, TID)_{K_P}, [k \cdot r, K_1 \times K_2], MH_{MC}\text{-}NEW) \end{aligned}$$

When M receives Message 5 from the customer, it checks to see if the purchase order is to its satisfaction─that is, the merchant agrees to all its contents, including the amount and the price. In addition, the merchant also verify the lightweight signature. If not, the merchant informs the customer of the rejection. Otherwise, the merchant endorses the purchase order as its agreement by signing h(PO). Then, the merchant generates the key pair (K_2, K_2^{-1}), where K_2=<e, N_2> is compatible key with K_1 and N_2 is relatively prime to r, and computes the product key ($K_1 \times K_2$). Subsequently, the merchant calculates $[k \cdot r, K_1 \times K_2]$. If an evil man would like to impersonate the merchant, he doesn't succeed because he doesn't know the key, K_1 and all primes, that construct N1. The same, if the merchant doesn't have the true product key, k, it does not calculate $[k \cdot r, K_1 \times K_2]$ either. Finally, M conveys the signature, $(TID, h(PO))_{K_M^{-1}}$, the lightweight signature, $MH_{MC}(h(PO))$, $[k \cdot r, K_1 \times K_2]$, and the merchant's bank account to C.

After the customer and the merchant make an agreement, the customer is about to pay for the product. In Step 3, the customer sends the contract (the merchant's signature for h(PO) and the customer's lightweight signature for h(PO)), and the merchant's bank account to the payment gateway. The payment gateway requests the customer's bank to deduct the money from the customer's account or to credit the money, and transfer money to the merchant's account.

Message 7:

$$C \rightarrow P: \quad ID_C, c, (M_{account}, TID)_{K_P}, (TID, h(PO))_{K_M^{-1}}, (TID, ID_M, PO, MH_{CP}\text{-}NEW)_{SK_{CP}}, \\ MH_{CP}(TID, h(PO), MH_{CP}\text{-}NEW)$$

The customer validates the product key and authenticates the merchant by verifying that $[k \cdot r, K_1 \times K_2] \equiv [k \cdot r, K_1] \bmod N_1$. If not, the customer cancels the transaction. Otherwise, he delivers the purchase order, two parties' signatures for the purchase order and the merchant's account to the payment gateway, P.

Message 8:

$$P \rightarrow C: \quad (TID, Ack, h(PO))_{K_P^{-1}}, (TID, Ack, MH_{PC}\text{-}NEW)_{SK_{CP}}, MH_{PC}(TID, Ack, h(PO))$$

When P receives Message 7, it checks two parities' signatures. If two signatures for the purchase order are consistent, the payment gateway begins to communicate with two parties' banks. If two parties' banks both approve the transaction, the payment gateway acknowledges the customer that the transaction is approved. If the contract is inconsistent, or any bank rejects the transaction, the payment gateway is about to answer the customer that the transaction fails as well as the reason.

The customer asks the merchant of the purchase key if he has completed the payment with the payment gateway. In Step 4, the customer sends the payment receipt to the merchant, and then the merchant delivers the product key.

Message 9:

$C \rightarrow M: \quad (TID, Ack)_{SK_{CM}}, (TID, Ack, h(PO))_{K_P^{-1}}$

The customer checks the lightweight signature once he receives the acknowledgement from the payment gateway. Subsequently, the customer sends the payment receipt signed by P, $(TID, Ack, h(PO))_{K_P^{-1}}$. Simultaneously, the customer starts a timer, and waits for the product key to arrive from the merchant. If the timer expires before the product key arrives from the merchant, the customer is about to execute the resolve sub-protocol.

Message 10:

$M \rightarrow C: \quad (k)_{SK_{CM}}$

After the merchant obtains the payment receipt from the customer, it first verifies the signed receipt by P. If the receipt is correct, the merchant conveys the product key, k, to the customer. The customer is about to decrypt m by k. If the merchant gives a wrong key or denies sending k to the customer intentionally, the resolve sub-protocol is initiated by the customer. Otherwise, the payment protocol is finished.

Resolve Sub-protocol

The resolve sub-protocol is executed if any party misbehaves or if there is a communication failure. The sub-protocol is to resolve disputes automatically without manual interventions. The merchant gets the payment receipt before sending the product key, so it has an advantage in the exchange sub-protocol. Therefore, the resolve sub-protocol is initiated only by the customer if the timer is expired or if the customer receives a wrong key. In Figure 4, we can see all messages in the resolve sub-protocol.

$C \rightarrow T$ $\quad$ $TID,x,(PO,(TID,h(PO))_{K_M^{-1}},(TID,Ack,h(PO))_{K_P^{-1}})_{SK_{CT}}$

$T \rightarrow C$ $\quad$ $(k)_{SK_{CT}}$

$T \rightarrow M$ $\quad$ $(TID,(TID,Ack,h(PO))_{K_P^{-1}})_{K_M}$

Message S1:

$C \rightarrow T$ $\quad$ $TID,x,(PO,(TID,h(PO))_{K_M^{-1}},(TID,Ack,h(PO))_{K_P^{-1}})_{SK_{CT}}$

If the merchant cheats the customer by giving a fake key or rejects to send the product key when receiving the payment receipt, the customer starts the resolve sub-protocol by delivering Message S1 to the TTP. The Message S1 contains all evidences: merchant's agreement of the purchase order, $(TID,h(PO))_{K_M^{-1}}$, and the payment receipt, $(TID,Ack,h(PO))_{K_P^{-1}}$. He needn't to send his agreement on the purchase order because the payment receipt interprets that as the payment gateway check two parties' agreements.

Message S2 & S3:

$T \rightarrow C$ $\quad$ $(k)_{SK_{CT}}$

$T \rightarrow M$ $\quad$ $(TID,(TID,Ack,h(PO))_{K_P^{-1}})_{K_M}$

When the TTP receive an application for a arbitration from a customer, it examines the agreement on the purchase order and the payment receipt. If no problem, T sends the product key to C. At the same time, T also sends the merchant the payment receipt for avoiding that M doesn't receive the payment receipt or receives the wrong receipt from C.

5. Analysis

We analyze the properties mentioned before of our proposed protocol. They are (R1) non-repudiation, (R2) fairness, (R3) product validation, (R4) merchant authentication, (R5) privacy respectively. Finally, we also analyze efficiency of our protocol.

5.1. Non-repudiation

In order to achieve the non-repudiation of origin and the non-repudiation of receipt, the two signatures are important. One is an evidence to represent the origin of the message. The other is an evidence to represent the receipt of the message.

Repudiation of origin. In our protocol, each important item is signed by a digital signature or a lightweight signature to ensure the origin of information. If the customer denies confessing his/her purchase behavior later or paying for it, the merchant provides the evidences to the TTP to prove the customer's repudiation. The evidences include the customer's agreement on the purchase order and the payment receipt. The customer's agreement is the lightweight signature of the purchase order signed by the customer, and the payment receipt is the digital signature of the payment information.

If the merchant denies confessing the agreement of the purchase order (possible the price or the amount) later, the customer can provide the merchant's agreement signed by the merchant to TTP. The merchant agreement could be the lightweight signature or the digital signature.

Digital signatures and lightweight signatures are non-repudiation, because they are only signed by the origin of messages and not forged if not knowing private keys. (The security strength of the lightweight signature is presented in [33])

Repudiation of receipt. This property is more difficult to achieve than the repudiation of origin. It can be achieved by a receipt signature or secret information to represent the receipt evidence, but there is a problem if a malicious party rejects sending the receipt evidence. Although this issue can be solved by the TTP involved, it isn't practical as the TTP can aid both parties to obtain their expected items. That is, no party can say that he/she has not received the expected item because he/she gets it from the TTP if he/she really acquires nothing from the corresponding party. Therefore, the repudiation of receipt is not need in our protocol.

5.2. Fairness

We start showing that our protocol provides strong fairness here. Strong fairness is that when the protocol has completed, either each party receives the expected items, or neither party obtains any items. In our protocol, the customer exchanges the payment receipt, $(TID,Ack,h(PO))_{K_P^{-1}}$, for the product, including the encrypted product, m, and the product key, k, with the merchant. Before to prove strong fairness is satisfied, we first prove two theorems in the following.

Theorem 3. No party gains anything before Message 9. That is, the customer can't derive the product key, k, from $[k \cdot r, K_1]$ and $[k \cdot r, K_1 \times K_2]$, and the merchant can't obtain the payment receipt, $(TID,Ack,h(PO))_{K_P^{-1}}$.

Proof:

The customer doesn't have the key, K_1^{-1}, K_2^{-1}, or $(K_1 \times K_2)^{-1}$, so he/she can't derive anything about the product key, k. The product is encrypted with the product key, k, so the customer doesn't decrypt without the product key, k.

The merchant can't generate the payment receipt, $(TID,Ack,h(PO))_{K_P^{-1}}$ without the payment gateway's private key. Moreover, it is impossible that a payment receipt is the same as another one. TID may be reduplicate in the future and Ack is also the same, but PO can't be the same. The purchase order contains a time stamp.

Theorem 4. The customer obtains the product key, k, if and only if the merchant obtains the payment receipt, $(TID,Ack,h(PO))_{K_P^{-1}}$.

Proof:

[If part] The customer obtains the product key, k, if the merchant obtains the payment receipt, $(TID,Ack,h(PO))_{K_P^{-1}}$.

There are only two ways that the merchant obtains the payment receipt, $(TID,Ack,h(PO))_{K_P^{-1}}$, in our protocol. One is Message 9 from the customer in the exchange sub-protocol, and the other is Message S3 from the TTP in the resolve sub-protocol. If the honest merchant gets the payment receipt form Message 9, it sends the product key, k, to the customer. If the merchant behaves incorrectly, or there is a failure in the communication channel, the customer applies for the TTP to obtain the product key. Thus, the merchant obtains the payment receipt form Message 9, and then the customer also gets the product key.

If the merchant obtains the payment receipt from Message S3, the resolve sub-protocol must have executed and the customer has received the product key from the TTP. Therefore, the customer obtains the product key, k, if the merchant obtains the payment receipt, $(TID, Ack, h(PO))_{K_P^{-1}}$.

[Only if part] The customer obtains the product key, k, only if the merchant obtains the payment receipt, $(TID, Ack, h(PO))_{K_P^{-1}}$.

There are two ways that the customer obtains the product key, k, in our protocol. One is Message 10 from the merchant in the exchange sub-protocol, and the other is Message S2 from the TTP in the resolve sub-protocol. For case one, before Message 10, the customer must send Message 9, including the payment receipt, to the merchant, and the payment receipt is valid. Thus, the merchant sends Message 10, including the product key, only if it receives the payment receipt.

For case two, the customer obtains the product key from Message S2. Obviously, the resolve sub-protocol is executed and the merchant gets the payment receipt from the TTP. Hence, the customer obtains the product key, k, only if the merchant obtains the payment receipt, $(TID, Ack, h(PO))_{K_P^{-1}}$.

From Theorem 3, we can know that two parties obtain nothing before Message 9, and due to Theorem 4, we can realize that if any party gains the expected item, the other does, too. That is, when the protocol has completed, either each party receives the expected items, or neither party obtains any items. We have shown that the strong fairness is satisfied in our proposed protocol above.

5.3. Product Validation and Merchant Authentication

The purpose of these two properties is in order to authenticate merchants and products, and hence they protect that customers are cheated. Subsequently, we show our protocol satisfies the product validation property and the merchant authentication property.

We consider the Dolev-Yao intruder. It can store all messages over the network, but it only decrypt and sign messages if it knows the corresponding key. It can compose new messages after analyzing its knowledge. It can remove or delay messages over the network. The intruder also acts as customers (or merchants), intruders, and network.

Theorem 5. If the intruder doesn't possess both the product key, k, and the key, K1, it can't generate x where x $\equiv [k \cdot r, K_1]$ mod N_1.

Proof:

Clearly, if the intruder doesn't possess the product key, k, and the key, K1, it can't "compute" x where x $\equiv [k \cdot r, K_1]$ mod N_1 by itself. (If it can, it can also break RSA cryptosystems.) It must analyze normal messages stored in its database to obtain useful information. In the following, we separately prove that the intruder can't generate x if it possess k or k1 only.

[Possessing k only]

As all messages over the network are encrypted, the intruder only decrypts some messages encrypted with some keys known to it. When it acts as a customer, it can obtain $[k \cdot r, K_1]$ and r. It can compute $k \cdot r$ and get the pair ($k \cdot r$, $[k \cdot r, K_1]$). The intruder can't derive K1 from the pair ($k \cdot r$, $[k \cdot r, K_1]$) due to the discrete logarithm problem. Thus, the intruder gains nothing except many pairs of r and $[k \cdot r, K_1]$. If TTP uses the same random number, r, collected by the intruder before, the intruder can generate x where x $\equiv [k \cdot r, K_1]$ mod N_1. However, it happens with small probability if r varies in an enough large range and K1 is updated periodically.

[Possessing K1 only]

As all messages over the network are encrypted, the intruder only decrypts some messages encrypted with some keys known to it. When it acts as a customer, it can obtain $[k \cdot r, K_1]$ and r. It can't derive $k \cdot r$ with K1 due to RSA assumption. If it guesses $k \cdot r$, it doesn't derive k due to factoring assumption. Thus, the intruder gains nothing except many pairs of r and $[k \cdot r, K_1]$. If TTP uses the same random number, r, collected by the intruder before, the intruder can generate x where x $\equiv [k \cdot r, K_1]$ mod N_1. However, it happens with small probability if r varies in an enough large range and K1 is updated periodically.

From Theorem 5, we can realize that the merchant can't be authenticated without K1 and the product can't be validated without k by the customer. Therefore, the validated receipt property and the merchant authentication property are satisfied in our protocol.

5.4. Privacy

None of the exchanged messages between the merchant and the customer contains information to identify the customer. (The merchant only knows the identity of the transaction.) Moreover, even if a customer performs multiple exchanges with the same merchant, the merchant can’t link those exchanges together using the protocol messages since session keys are different and identities of the transactions are unique on every transaction. (IP address linking or similar means might be possible, but our protocol doesn’t create any additional means of identification and linking.) Only the TTP and the payment gateway know who the buyer is, but they are trusted by everyone.

6. Conclusion

We pointed out the problems of existing payment protocols when applied to wireless environments. We proposed the first lightweight fair payment protocol for mobile devices without any extra devices or infrastructures. We applied symmetric key cryptosystems and hash functions not only to reduce customers' computation, but also to satisfy security requirements including strong fairness.

The TTP helps to initiate our payment protocol and to manage transactions. As our future works, we can employ distributed servers and secrete sharing scheme lightens loading of TTPs and enhances security of TTPs.

7. Reference

[1] S. Kungpisdan, B. Srinivasan, and P. D. Le, "Lightweight Mobile Credit-Card Payment Protocol," Lecture Note in Computer Science, Vol. 2904, 2003.

[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 iKP Secure Electronic Payment System," IEEE Journal of Selected Areas in Communications, 2000.

[3] Mastercard and Visa, "SET Protocol Specifications," 1997.

[4] S. Kungpisdan, B. Srinivasan, and P. D. Le, "A Secure Prepaid Wireless Micropayment Protocol," Proceedings of the Second International Workshop on Security in Information Systems, 2004.

[5] X. Wu, O. Dandash, and P. D. Le, "The Design and Implementation of a Smartphone Payment System based on Limited-used Key Generation Scheme," Proceedings of the Third International Conference on Information Technology: New Generations, 2006.

[6] B. T. S. Toh, S. Kungpisdan, and P. D. Le, "KSL Protocol: Design and Implementation," Proceedings of the 2004 IEEE Conference on Cybernetics and Intelligent Systems, 2004.

[7] S. Kungpisdan, B. Srinivasan, and P. D. Le, "A Secure Account-Based Mobile Payment Protocol," Proceedings of the International Conference on Information Technology: Code and Computing

[8] S. Kungpisdan, P. D. Le, and B. Srinivasan, "A Limited-Used Key Generation Scheme for Internet Transactions," Lecture Note in Computer Science, Vol. 3325, 2004.

[9] Artur Romao, Miguel Mira da Silva, "An Agent-Based Secure Internet Payment System for Mobile Computing," Proceedings of IFIP International Conference on Trends in E-commerce, 1998.

[10] T. O. Lee, Y. L. Yip, C. M. Tsang, and K. W. Ng, "An Agent-Based Micropayment System for E-Commerce," E-commerce agent, Lecture Notes in Artificial Intelligence 2033, 2001.

[11] Y. Wang and T. Li, "LITESET/A++: A New Agent-assisted Secure Payment Protocol," Proceedings of the IEEE International Conference on E-Commerce Technology, 2004.

[12] C. C. Liew, W. K. Ng, E. P. Lim, B. S. Tan, and K. L. Ong, "Non-repudiation in An Agent-Based Electronic Commerce System," 10th International Workshop on Database and Expert Systems Applications, 1999.

[13] X. Yi, C. K. Siew, X. F. Wang, and E. Okamoto, "A Secure Agent-based Framework for Internet Trading in Mobile Computing Environments," Distributed and Parallel Databases 8, 2000.

[14] D. H. Shih, S. Yi Huang, and D. C. Yen, "A New Reverse Auction Agent System for M-commerce Using Mobile Agents," Computer Standards & Interfaces 27, 2005.

[15] F. M. Matos and E. R. M. Madeira, "An Automated Negotiation Model for M-commerce Using Mobile Agents," Lecture Note in Computer Science, Vol. 2722, 2003.

[16] J. Hall, S. Kilbank, M. Barbeau, and E. Kranakis, "WPP: A Secure Payment Protocol for Supporting Credit- and Debit-card Transactions over Wireless Networks," IEEE International Conference on Telecommunications, 2001.

[17] H. Wang and E. Kranakis, "Secure Wireless payment Protocol," International

Conference on Wireless Networks, 2003.

[18] H. W. Lee, I. Y. Lee, and D. I. Oh, “Smart Card Based Mobile Payment with Fairness Revocation Mechanism,” Lecture Note in Computer Science, Vol. 2738, 2003.

[19] R. Abbadasari, R. Mukkamala, and V. V. Kumari, “MobiCoin: Digital Cash for M-Commerce,” Lecture Note in Computer Science, Vol. 3347, 2004.

[20] I. Ray and I. Ray, “An Anonymous Fair Exchange E-commerce Protocol,” Proceedings 15th International Parallel and Distributed Processing Symposium, 2001.

[21] I. Ray, I. Ray, and N. Natarajan, “An Anonymous and Failure Resilient Fair-exchange E-commerce Protocol,” Decision Support Systems 39, 2005.

[22] C. Boyd and E. Foo, “Off-line Fair Payment Protocols Using Convertible Signatures,” Advances in Cryptology, 1998.

[23] S. Kim and H. Oh, “Fair Offline Payment Using Verifiable Encryption,” Lecture Note in Computer Science, Vol. 3325, 2004.

[24] M. Changshe, L. Feiyu, and C. Kefei, “Optimistic Fair Exchange E-commerce Protocol Based on Secret Sharing,” Journal of Systems Engineering and Electronics, Vol. 17, 2006.

[25] C. H. Wang, “Untraceable Fair Network Payment Protocols with Off-Line TTP,” Advances in Cryptology, 2003.

[26] X. Liang, Z. Cao, R. Lu, and L. Qin, “Efficient and Secure Protocol in Fair Document Exchange,” Computer Standards & Interfaces 30, 2008.

[27] S. M. Yen, “PayFair: APrepaid Internet Micropayment Scheme Ensuring Customer Fairness,” IEE Computers and Digital Techniques, Vol. 148(6), 2001.

[28] R. H. Deng, L. Gong, A. A. Lazar, and W. Wang, “Practical Protocol for Certified

Electronic Mail," Journal of Network and Systems Management, Vol. 4, No. 3, 1996.

[29] J. Zhou and D. Gollmann, "A Fair Non-repudiation Protocol," Proceedings of the 1996 IEEE Symposium on Security and Privacy, 1996.

[30] N. Asokan, V. Shoup, and M. Waidner, "Optimistic Fair Exchange of Digital Signatures," IEEE Journal on Selected Areas in Communication, 2000.

[31] N. Asokan, V. Shoup, and M. Waidner, "Optimistic Protocols for Fair Exchange," Proceeding of the 4th ACM Conference on Computer and Communications Security, 1997.

[32] G. Ateniese, "Efficient Verifiable Encryption and Fair Exchange of Digital Signatures," Proceedings of the 6th ACM Conference on Computer and Communications Security, 1999.

[33] S. M. Chang, S. P. Shieh, W. L. Lin and C. M. Hsieh, "An efficient broadcast authentication scheme in wireless sensor networks" ACM Symposium on Information, Computer and Communications Security, 2006.

[34] H. Pagnia and F. C. Gartner, "On the impossibility of fair exchange without a trusted third party," Technical Report, 1999.