SCIS 2002 The 2002 Symposium on Cryptography and Information Security
Shirahama,Japan, Jan.29-Feb.1,2002
The Institute of Electronics,
Information and Communication Engineers

A Micro-payment System for Multiple-Shopping

Manho Lee * Kwangjo Kim *
lmh@icu.ac.kr kkj@icu.ac.kr

Abstract— In this paper, we propose a micro-payment system that is able to apply to several vendors using only one hash chain. Explicitly, this scheme can be considered as an improvement and extension of PayWord [8] suggested by Rivest and Shamir in 1996. We focus on the scheme that guarantees atomicity and non-repudiation among requirements for payment systems. Atomicity and non-repudiation play an important role in payment systems. In particular, atomicity allows customers to use the payment system without loss of their money in any disaster. Finally, when a customer uses the micro-payment over several vendors,we analyze not only the efficiency in terms of the storage size but also the security issues.

Keywords: Micro-payment, Atomicity, Non-repudiation, PayWord

1 Introduction

Many electronic cash ("e-cash" in short) protocols have been proposed with the proliferation of the Internet and the activation of e-commerce.

One of them is a system that uses a well-established credit card infrastructure like SET (Secure Electronic Transaction) [11] that is proposed by VISA and MasterCard. Since Chaum proposed untractable e-cash protocol based on blind signature in 1982 [3], various extended schemes and systems have been proposed, which provide functionalities such as anonymity, double spending prevention, unforgeability, untraceability, and efficiency. These protocols are designed as medium for large transaction (macro-payment) over 10$.

On the other hand, micro-payment systems have been proposed for small and frequent payment such as database query, software, and e-news. Because the overhead cost for card company is higher than price of goods, and the processing time related to public key computation is required too much, most of the previous systems exclude public key computation or admit the minimal computational overhead for efficiency. Several protocols that use computationally secure one-way hash function were proposed in [4, 5, 7, 8]. Stern and Vaudenay's system [12] used MAC (Message Authentication Code) and an unkeyed one-way hash function instead of using public key cryptography and adopted tamper resistant devices for both customer and merchant sides. MilliCent [6] used "scrips", and hash functions extensively. MicroMint [8] used k-way hash function collision to make money. Micro-payment system is considered to be proper for areas where efficiency is the most critical factor.

Even if a customer loses his e-cash, it is not critical to him due to the small amount. Though an attacker forfeits the e-cash of a customer, computational cost is too big to gain a profit from the attack. Although damage by system errors or attacks is trivial, customers do not want to lose his money, or goods. To solve these problems, protocols in [1, 2, 10, 13, 14] were proposed to satisfy atomicity.

From the security point of view, micro-payment system is mainly concerned about preventing or detecting from double spending and forgery.

In this paper, we propose a new micro-payment system, in which a customer generates and maintains only one hash chain for every vendor. Since PayWord requires to use each hash chain for each vendor, the customer has to maintain all indices after dealing with vendors. To solve this problem, we use a receipt which includes a signature of vendor. The customer uses this signature for different vendors to enable them to verify whether the start index of payment is correct or not. Moreover, we design each phase of protocol to guarantee atomicity.

The organization of this paper is as follows: In Section 2, we present various micro-payment systems. Especially, we describe PayWord which is one of the most popular micro-payment systems. In Section 3, we introduce a new micro-payment system. In Section 4, we analyze our new system in terms of atomicity, security and efficiency. Finally, we summarize our results and suggest future works in Section 5.

2 Related works

2.1 Basic Architecture of Micro-payment

Basically, a micro-payment system is composed of three entities, *i.e.*, customer, vendor, and broker.

Fig. 1 shows a basic architecture of micro-payment system.

- Customer : does business with vendors by using micro-payment system.

* International Research center for Information Security (IRIS), Information and Communications Univ., 58-4 Hwaamdong, Yuseong-gu, Daejon, 305-732, Korea.

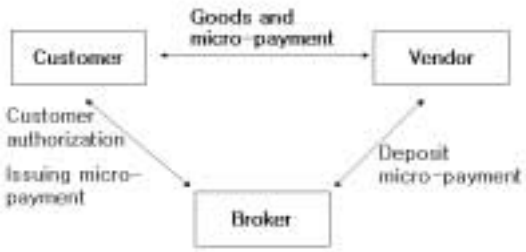

Fig. 1: A basic architecture of micro-payment

- Vendor : offers customers goods and deposits e-cash through a broker.
- Broker : issues e-cash, validates customers, and manages accounts of customers and vendors.

2.2 Basic Requirements

- Efficiency
  - *Minimizing of public key computation* : We do not use public key computation at all or use as few as possible to reduce high cost of public key computation.
  - *Off-Line Verification* : A broker doesn't carry on on-line verification due to limitation of the bandwidth of communication line during the transaction.
  - *Minimum usage of message* : The number of messages among entities, especially between customers and vendors, should be kept lower during transaction.
  - *Minimum usage of memory space* : Permanent variables should be used as few as possible.
- Security
  - *Double spending prevention or detection* : If the same e-cash is used more than once, then it should be detected.
  - *Unforgeability* : Only publishers who have a authority should be able to issue e-cash.

2.3 Atomicity

In addition to basic requirements, atomicity is also of another importance in serving a complete payment system. If network crashes in the middle of payment protocol, some disputes such as loss of money, repudiation, and no procurement of goods even after a complete payment will happen. Under a network model where customers purchase electronic goods and receive its service, Tygar [13] introduced the problem of atomicity in electronic transactions and defined three classes of atomicity; money atomic, goods atomic, and certified delivery.

- *Money atomicity* : The transfer of funds from one entity to another should be done without creation and loss of money.
- *Goods atomicity* : With the money atomic, it should be ensured that customers will receive goods if and only if the money is transferred.
- *Certified delivery* : Including goods and money atomic, both customers and vendors should be able to prove exactly what was delivered. If there is a dispute, this evidence can be shown to a judge to prove exactly what goods were delivered.

Camp *et al.* [2] proposed a method to support atomicity using on-line TTP (Trusted Third Party). Moreover, Sirbu and Tygar [10], and Bellare *et al.* [1] presented on-line account-based atomicity, and token-based atomicity, respectively. Xu *et al.* [14] addressed money conservation via atomicity in fair off-line e-cash.

2.4 PayWord

This system employs the cryptographic properties of digital signature and hash chain. Principal entities consist of customer, broker and vendor. Customers open an account with a broker. The broker issues a digitally-signed certificate, which authorizes the customer to make PayWord chain and assures vendors that the customer's PayWords are redeemable. Customers create the PayWord chain in reverse order by picking the last Payword w_n at random, and then

$$w_{i-1} = h(w_i),$$

where h is a collision-resistant hash function, and $i = 1, \ldots, n$. Let denote w_0 be the root of the PayWord chain. The commitment contains the root w_0. Let customer, broker and vendor be denoted by C, B and V, respectively.

2.4.1 Certificate

The certificate is defined by

$$C_c = \{B, C, A_C, PK_C, E, I_C\}_{SK_B}$$

$-B$: Broker identity
$-C$: Customer identity
$-A_C$: Customer's IP-address
$-PK_C$: Customer's public key
$-E$: Expiration date
$-I_C$: Other customer specific information

The customer's certificate has to be regularly renewed by the broker; the broker will do so only if the customer's account is in good standing.

2.4.2 Protocol steps

Three components execute each step as follows:

- C receives the certificate C_C generated by B.
- When C contacts a new vendor V, she computes a new PayWord chain with root w_0, and computes her commitment for that chain:

  $$M = \{V, C_C, w_0, D, I_M\}_{SK_C},$$

  where V identifies the vendor, C_C is C's certificate, w_0 is the root of the chain, D is the current date, and I_M is any additional information. M is signed by C.

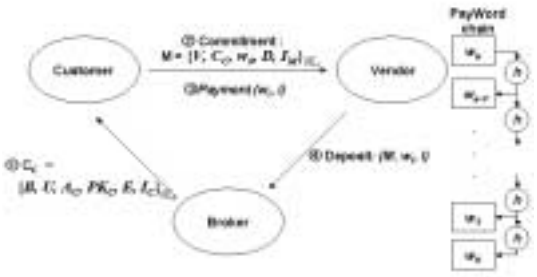

Fig. 2: PayWord protocols

- In advance of business, C commits M to V. Then V verifies C's signature on M and B's signature on C_C, and checks their expiration date. If valid, V stores it.
- When C purchases goods from V, C transfers (w_i, i) to V as its payment.
- V computes $h(w_i)$ and confirms it, if valid then keeps it.
- V sends the last payment information (w_l, l) and its commitment M to B and requests redemption. After verifying, B withdraws eligible money from C's account and deposits it in B's account.

2.4.3 Problems of PayWord system

PayWord has several problems as follows:

- A customer should keep a different hash chain according to each vendor. Hence, in the customer's side, she must compute expensive public key operations as many as the number of vendors that she wants to transact.
- A customer should preserve all of last indices that she used to deal with vendors.
- This system doesn't present the complete steps of business to sale goods.
- When a customer sends the index with respect to goods' price to a vendor, the vendor issues no receipt on it. Due to absence of verification of receiving, it occurs the loss of money.
- A customer can make payment exceeding her credit limit.

In order to address these problems, we propose a new payment protocol in which we make use of only one hash chain with respect to a number of transactions. The new protocol assures atomicity as well.

3 Our Proposed System

We assume that customers and vendors completely trust the broker. It is also assumed that a new protocol deals with digital goods.

3.1 Notation

Afterwards, we use the following notation to describe the proposed system.

- C, B, V : Customer, Broker, Vendor
- Z : entity $Z \in \{C, B, V\}$
- Z_{ID} : Z's identity
- PK_Z, SK_Z : Z's public key, private key
- $[Data]_K$: symmetric encryption scheme
- $\{Data\}_{PK_z}$, $\{Data\}_{SK_z}$: public key encryption scheme with Z's public key and decryption scheme with Z's private key
- $Sign_Z\{Data\}$: signature of Z
- $X \Rightarrow Y$: X sends one message to Y
- $h()$: collision-free hash function
- C_Z : Z's certification made from broker
- E : expiration period of certificate
- r_Z : random number made by Z

3.2 Details of Protocol

The proposed system is composed of four phases: initialization, negotiation, payment-delivery, and deposit phase. Fig. 3 shows an overall configuration of each phase.

Fig. 3: Overall configuration

3.2.1 Initialization phase

C sends an encrypted message that contains the root value of hash chain w_0, the length of chain n, customer identity C_{ID}, and broker identity B_{ID}.

$$C \Rightarrow B : \{w_0, n, C_{ID}, B_{ID}\}_{PK_B} \quad (1)$$

This message is encrypted by B's public key. B decrypts the received message by his private key and checks whether the generation of payment is available in the customer's account standing or not. If the length of hash chain does not exceed the limit, B issues a certificate C_C to C.

$$B \Rightarrow C : C_C = Sign_B\{C_{ID}, B_{ID}, w_0, n, E\} \quad (2)$$

V receives the certificate, and uses it in transaction with C.

$$B \Rightarrow V : C_V = Sign_B\{V_{ID}, B_{ID}, PK_V, E\} \quad (3)$$

3.2.2 Negotiation phase

C and V participate this phase to agree on the price, goods, and predetermined time limit. C searches information of product which she wants to purchase through the Internet. If C wants to purchase a product, then she sends a product request message as follow:

$$C \Rightarrow V : Product\ request \tag{4}$$
$$\{V_{ID}, C_{ID}, C_C, \mathsf{ProductID}, \mathsf{Price}, r_C, t\}_{PK_V}$$

Here, ProductID is a product identity which V deals with. It is a signed value by V, $Sign_V\{h(\mathsf{goods})\}$. C utilizes this by downloading from V's web site and identifies the product. Also C can use it when she claims that she receives goods different from what she ordered. The random number r_C identifies the negotiation. The value t is the predetermined time limit from receiving coin of V to deposit it in his own account. V verifies the expiration period of C's certificate, root value of hash chain, and the length of hash chain from C's certificate.

3.2.3 Payment-Delivery phase

V encrypts digital goods by symmetric key and delivers it to C.

$$V \Rightarrow C : Goods\ Delivery \tag{5}$$
$$[\mathsf{goods}]_K,$$
$$\{h([\mathsf{goods}]_K), r_V, h(r_C, r_V, \mathsf{Price})\}_{PK_C}$$

V transfers the product's hash value, a random number generated by himself and the certificate from broker. When C receives encrypted goods, she decrypts it and checks if it is the same goods as she previously requested by computing $h(r_C, r_V, \mathsf{Price})$, then identifies the price of goods. C pays a designated amount. Payment is made by transferring hash chain value and its index.

$$C \Rightarrow V : Payment = (w_i, i) \tag{6}$$

V verifies the length of hash chain and makes sure if it exceeds the limit. V verifies the transferred hash value using the root value. After verification of payment, V returns receipt and symmetric decryption key, and K of sold goods. Also, V includes a signature for remaining index length and his certificate. When C uses the same hash chain to another vendor, it will be sent to the next V to verify the previous used index.

$$V \Rightarrow C : Receipt \tag{7}$$
$$\{K, n-i, C_V, w_i, Sign_V\{h(C_V, n-i)\}\}_{PK_C}$$

C decrypts purchased goods using decryption key from V and identifies its contents.

3.2.4 Deposit phase

V sends the latest payment and hash chain index with the certificate of C to B. V should request within the predetermined specific time limit during negotiation phase.

B verifies C's certificate and confirms the limit length of hash chain. Using the root value, B verifies hash chain value. If the value is valid, B deposits the corresponding amount of money in V's account.

3.3 Multiple Shopping

PayWord should issues a new hash chain whenever a customer carries out business with different vendors. Also a customer should preserve all of last indices that she used to deal with different vendors. On the other hand, in the new protocol, customer holds only a unique hash chain and does business with a number of vendors using the same hash chain. The customer also sends the same certificate as one from the previous vendor to the current vendor.

3.3.1 Payment-Delivery phase

Assume that C has a hash chain of length n and uses index, i to $(k-1)$th V and index, j to kth V. In the case that C pays to kth V, she provides the certificate from $(k-1)$th V and indices i and j together to kth V.

$$C \Rightarrow V_k : Payment \tag{8}$$
$$(w_{i+j}, j, n-i, C_{V_{k-1}}, Sign_{V_{k-1}}\{h(C_{V_{k-1}}, n-i)\})$$

V verifies the certificate of $(k-1)$th vendor and transfers his certificate and index verification value along with the decryption key of sold goods.

$$V \Rightarrow C : Receipt \tag{9}$$
$$\{K, n-i-j, w_{i+j}, C_{V_k}, Sign_{V_k}\{h(C_{V_k}, n-i-j)\}\}_{PK_C}$$

3.3.2 Deposit phase

When V requests redemption, V sends the certificate of $(k-1)$th V, $C_{V_{k-1}}$, hash chain value, w_i, his own hash chain value w_{i+j}, the index of redemption request, j and certificate, C_{V_k} to B.

$$V \Rightarrow B : Request\ deposit \tag{10}$$
$$(C_{V_{k-1}}, w_i, w_{i+j}, j, C_{V_k})$$

By the verification of each hash chain value, B can make a decision on redemption to the requested index. If the requested index ranges within the predetermined limit, the redemption will be done. In a similar way, B can check the excess of the limit.

4 Analysis of Our Proposed System

4.1 Atomicity

4.1.1 Money atomicity

Money atomicity is preserved by;

- Only after a customer pays for goods, the customer can receive the key used for the decryption for goods, and the receipt. Therefore, loss of the money can be prevented.
- In the negotiation phase between a customer and a vendor, the predetermined time limit is established over money from the beginning of transaction to the deposit. Therefore, the available period of money can be established.
- Since a broker certifies information on the length of money, a customer can create money at her own discretion.

4.1.2 Goods atomicity

Goods atomicity is guaranteed only when money atomicity is guaranteed for a customer who receives the goods that has been already paid. In the case of the proposed system, if a customer doesn't pay for the ordered goods, she can't obtain the key for the decryption. Therefore, she can't get the decrypted goods.

On the other hand, although a customer has already completed payment for goods, a customer cannot get the key for the decryption until the predetermine time limit is over. Then, the customer processes as following ways:

- The customer sends the index value (w_i, i) of payment to the vendor, the random number r_C used for the transaction, certification C_V of the vendor, the value of predetermined time limit t to the broker, and requests for the confirmation of whether the deposition into the vendor's account is completed.
- If the deposition is completed, the broker sends the index value, and the random number to the vendor in order to request for the customer to send the value of receipt for the transaction. If the vendor refuses, the broker terminates the account with the vendor, registers a blacklist, and considers legal action.
- If the deposition has not yet completed, notify the customer that the transaction with the vendor was cancelled.

Goods atomicity follows from the above procedure.

4.1.3 Certified delivery

Now, suppose that a customer claims that she receives goods different from what she ordered. In this case the following steps will be taken to guarantee certified delivery.

- The customer requests the validation of goods to a broker by sending ProductID value, an encrypted product value, a decrypted key K, and the good's encrypted value signed by a vendor.
- The vendor can't deny the transaction because he signed the product's hash value with his private key. However, in reality, the vendor will not commit such a misconduct because of the little profit from cheating is the risk of losing his credit in the case that the misconduct is revealed, and the risk of the termination of his account from the broker. On the other hand, if it turns out that the customer received the right product, it tells that it was the customer's misconduct.

4.2 Security Issues

4.2.1 Double spending detection

Our system makes use of a unique hash chain to each vendor so that it has the following advantages:

- When a customer uses an identical index to a different vendor once more, a broker is able to detect double spending since he verifies the customer's certificate transferred from the vendor to be paid redemption. For the purpose of preserving this property, the broker should keep both the root value and the length information in the certificate that is issued to the customer.
- In case that the previously used index range is reused in a different vendor, for instance, a customer used index range from i to j to kth vendor, and he paid from i to j to arbitrary $(k+1)$th vendor once more. As the same way above, the broker can detect two indices which were spent in k and $(k+1)$th vendors identically.

Broker is able to keep blacklist of the double spenders. The broker also has a flexibility on a term of "validity" according to the length of hash chain and the credit of customer when she issues a certificate.

4.2.2 Overspending prevention

The length of hash chain is sent to a broker and included in the certificate so that a vendor can verify the excess of the limit by identifying a customer's certificate.

4.2.3 Forgery preventing

An unused hash chain value cannot be inferred from a used one. If malicious vendors conspired each other, they could know unused hash values. But, the right redemption will not occur since a broker can verify double spending when it requested.

4.3 Efficiency

In PayWord, it doesn't provide a complete protocol for electronic commerce so that the loss of money happens and atomicity is not assured. On the contrary, our proposed system presents a complete protocol to deal with digital goods. This expansion makes transaction steps and the length of message increase slightly. But, in customers' side, they utilize only one unique hash chain to a number of vendors. Thus, it is not necessary for customers to select a new hash chain to do a business with a different vendor. Furthermore, since the remaining hash index can be used with the signature of previous vendor, all what customers do is to handle the index of one hash chain.

Table 1 compares the efficiency between PayWord and our proposed system, where m denotes the number of vendors and w_m denotes the hash chain itself, respectively.

Table 1: Comparison of efficiency

System	PayWord	Proposed System
Public key computation	1	3
Indices to be saved with m vendors	$m \times w_m$, $m \times index$	$1 \times w_m$, $1 \times receipt$

Table 2 compares the several properties between PayWord and our proposed system.

Table 2: Comparison of properties

Property	PayWord	Proposed System
Atomicity		√
Double spending detecting	√	√
Forgery preventing	√	√
Overspending prevention		√
Non-repudiation		√
Multiple shopping		√

5 Conclusion and Further Works

In this paper, we propose a new micro-payment system that is able to apply to multiple vendors using only one hash chain. Explicitly, this scheme is an improvement and extension of PayWord suggested by Rivest and Shamir. In PayWord, the loss of money happens and atomicity is not assured since it doesn't provide a complete protocol of electronic commerce. But, our proposed system presents a complete protocol to transact digital goods, and at the same time, preserves atomicity. Furthermore, it employs only one hash chain to deal with multiple vendors so that it provides efficiency and convenience of index management and addresses the general security characteristics of micro-payment. However, the frequency of public key computation gets so increased that it compromises computational efficiency that PayWord holds.

As further works, we need to design micro-payment scheme that is available in mobile communication. It requires more efficiency of computation and data store and availability.

References

[1] M. Bellare, J. Garay, C. Jutla, and M. Yung, "VarietyCash: A Multi-Purpose Electronic Payment System", *Proc. of the 3rd USENIX Workshop on Electronic Commerce*, pp. 9-24, 1998.

[2] J. Camp, M. Harkavy, J. D. Tygar, and B. Yee, "Anonymous Atomic Transactions", *Proc. of the 2nd USENIX Workshop on Electronic Commerce*, pp. 123-133, 1996.

[3] D. Chaum, "Blind Signatures for Untraceable Payments", *Advances in Cryptology-Proc. of CRYPTO'82*, Plenum Press, pp. 199-203, 1983.

[4] G. Horn, B. Preneel, "Authentication and Payment in Future Mobile Systems", *Computer Security ESORICS 98*, LNCS 1485, pp. 277-293, 1998.

[5] R. Hauser, M. Steiner, and M. Waidner, "Micro-Payments based on iKP", *Proc. of the 14th Worldwide Congress on Computer and Communications Security Protection*, pp. 67-82, 1996.

[6] M. S. Manasse, "The Millicent Protocol for Electronic Commerce", *Proc. of the 1st USENIX Workshop on Electronic Commerce*, 1995.

[7] T. P. Pedersen, "Electronic Payments of Small Amounts", *Proc. of Cambridge Workshop on Security Protocols*, LNCS 1189, pp. 59-68, 1996.

[8] R. L. Rivest and A. Shamir, "PayWord and MicroMint: Two Simple Micropayment Schemes", *CryptoBytes*, pp. 7-11, 1996.

[9] M. Satyanarayanan, "Fundamental Challenges in Mobile Computing", *Proc. of the 15th ACM Symp. on Principle of Distributed Computing*, pp. 1-7, 1996.

[10] M. Sirbu and J. D. Tygar, "NetBill: An Internet Commerce System Optimized for Networked Delivered Services", *IEEE COMPCON'95*, pp. 20-25, 1995.

[11] SET Secure Electronic Transaction LLC, "SET Secure Electronic Transacrion Specification", 1999, `http://www.setco.org`

[12] J. Stern and S. Vaudenay, "SVP : a Flexible Micropayment Scheme", *Financial Crypto 97*, LNCS 1318, pp. 161-172, 1997.

[13] J. D. Tygar, "Atomicity in Electronic Commerce", *Proc. of the 15th ACM Symp. on Principles of Distributed Computing*, pp. 8-26, 1996.

[14] S. Xu, M. Yung, G. Zhang, and H. Zhu, "Money Conservation via Atomicity in Fair Off-Line E-Cash", *Proc. of the 2nd Int. Information Security Workshop*, LNCS 1729, pp. 14-31, 1999.