A Set of Protocols for Micropayments in Distributed Systems

Lei Tang
Graduate School of Industrial Administration
Carnegie Mellon University

A Set of Protocols for Micropayments in Distributed Systems

(extended abstract)

Lei Tang
Graduate School of Industrial Administration
Carnegie Mellon University
Email: ltang@cs.cmu.edu

Abstract

Micropayments refer to low-value financial transactions ranging from several pennies to a few dollars. A big portion of electronic commerce occurring in the Internet belong to the category of micropayments. The cost of micropayments should be kept as low as possible in order for the service provider (the merchant) to profit from the low-value transactions. We propose several protocols for micropayments in distributed systems. Our main goal is to reduce the charging cost by choosing a suitable security model, a charging model, and cryptographic algorithms; and by employing the properties unique to the information goods. Our protocols satisfy the requirements of a payment system and are "cheap" in terms of computation costs, communication costs, and key management costs. We select the debit model for designing our protocols and base our protocols on the private key cryptosystems. We show that the existing authentication protocols and systems can be extended to handle micropayments in distributed systems. We also present possible extensions to our protocols.

1 Introduction

With the rapid growth of the Internet, more and more computer users rely on computer networks for every kind of information ranging from daily news and journal papers to movies. Most of the information items on the Internet have a low value, ranging from pennies to several dollars. A low-cost electronic charging mechanism should be provided to the intellectual property owners so they can profit from providing these services, and also to stimulate them to provide quality services to the Internet community.

There exists several electronic payment protocols [12, 3]. They are based on different charging models and use different cryptographic algorithms. Security and privacy are main concerns of these protocols. Less is studied about how to address the issue of lowering the transaction cost. We analyze trade-offs of different charging models and different cryptographic algorithms. We also observe that the information goods have two unique properties: they are easy to duplicate without extra cost; and they are unreturnable once purchased. Based on this observation, a set of "cheap" payment protocols is designed. The charging model we use is the debit model. We use only the private key cryptosystems, with the option extended to use the public key cryptosystems. We also give a detailed security analysis of our protocols. Authentication in distributed system has been studied for almost twenty years and a lot of practical systems have been built [17, 18]. Is it possible to extend those systems to process micropayments in distributed systems so that we do not have to build our system from scratch, in turn, the costs of building the system are reduced? we describe the relationship between our protocols and the protocols for authentication in distributed systems and show that it is possible to extend those authentication systems for the purpose of micropayments in distributed systems.

In section 2 we define the parties involved in our payment protocols, and notations used throughout this paper. In section 3 we analyze the elements affecting the transaction cost, the trade-offs among them. Then we choose the on-line debit model to design our protocols. In section 4 we describe the structure of our payment systems. In section 5 we present our protocols, giving a detailed security analysis of these protocols. We also discuss the relationship between our protocols and authentication protocols. We conclude in section 6.

2 Principals and Notations

Every payment system involves at least two parties exchanging goods (or services) and money. We call the parties involved in the electronic payment system the *principals*. All principals communicate through a computer network. The basic units carrying out the communication in favor of the principals are *processes*. We call the processes used to export services the *servers* and processes used to import services the *clients*. The principal who provides (or sells) services is called the *merchant*, and the principal who imports (or buys) services, the *customer*. The merchant sells his services through the merchant server and the customer buys the services through the customer client. The principal who wants to illegally benefit from the electronic payments by sabotaging or eavesdropping the communication channel is called the *adversary*.

All principals communicate exclusively by passing messages over a network. We assume that the network is not secure. Not only can an adversary mount passive attacks in which the adversary merely observes the messages passing on the network without interfering with them; the adversary can also mount active attacks, performing a variety of processing on the messages passing on the network. These messages can be selectively modified, deleted, delayed, reordered, duplicated, and inserted into the communication at a later point in time; or be allowed to pass through unaffected [19].

We adopt notation common in the literature for authentication protocols, especially the notation used by Abadi and Needham [2]. We will not make a distinction between the principal and the process which the principal creates to accomplish the transaction on behalf of himself. We use the symbols C, M and B to denote the true identities of the principals (the customer, the merchant and the billing service center [1]). T_a represents the timestamp read from principal A's clock. N_a denotes the nonce [15, 16] generated by the principal A. We also use the symbols below to represent the following.

K_a A's public key.

K_{ab} The key shared between A and B.

$\{x\}_K$ x encrypted with key K.

x, y x concatenated with y.

$A \rightarrow B : x$ A sending message x to B.

I_x Item x.

Id_a The pseudonym of A's identity.

P_x Offered price of item x by the merchant.

ET_x Expiration time of the offered price for I_x.

[1] The definition is given in Section 3.3

3 Elements affecting the transaction cost

3.1 System Security

The key for the success of an electronic payment system is *security*. We use security to refer to the most important properties of an electronic payment system: secrecy, authenticity and integrity. Secrecy refers to denial of access to information by unauthorized principals. (e.g., eavesdropper on the network.) Authenticity refers to validating the source of a message; that is, the message was transmitted by a proper identified sender and is not a replay of a previously transmitted message. Integrity refers to assurance that a message was not modified accidentally or deliberately in transmit by replacement, insertion, or deletion. Nonrepudiation of origin is also a desired characteristic for protection against a sender of a message later denying transmission. The electronic payment system should be able to prevent dishonest principals from illegally gaining financial benefits by deviating from the protocols or colluding with other malicious adversaries. The electronic payment system should also be secure against attacks from the malicious adversaries and the dishonest customers (or merchants).

The merchants and the electronic payment service provider will lose revenue due to the breach of the payment system. Finally, the lost revenue is transferred to the honest customers and the cost will increase. Therefore, security is the key for keeping the transaction cost low.

3.2 Public key vs. private key

The public key cryptosystems [6] provide the mechanism for non-repudiation signatures, and are secure against known plaintext attacks [8], which can not be provided by the private key cryptosystems. However, the public key cryptosystems are unsuitable for electronic micropayments for the following reasons.

First, the cost of public key management is very expensive compared with that of private key management. A trusted key certification authority should be established to register and certify all principals who have the public keys in the electronic payment system. The key certification authority also revokes lost or stolen keys. The key certification authority must make sure that revoked keys are known to all principals. Every principal involved in the payment has to check the revocation list whenever a transaction is processed. Any unavailability of the revocation list or delay in delivery of the revocation list, which is common in the present distributed environment, will make the principals fail to detect attacks from the adversary. However, in the private key cryptosystem, if a customer (or a merchant) loses the secret key shared between him and the electronic payment service provider, he just needs to notify the electronic payment service provider. No further operation is needed for either party.

Second, the public key cryptosystems are computationally intensive since most of the widely used public key cryptosystems (e.g., RSA [14], ElGamal [7]) involve many exponentiation and multiplication operations in an Abelian group with very big cardinality. Usually a public key algorithm (e.g., RSA) is much slower than a private key algorithm (e.g., DES). The performance degrades dramatically when a payment service provider processes these expensive computations for all its customers in one central server.

Third, most of the practical public key cryptographic algorithms are patented. An electronic payment service provider has to pay royalty for using these algorithms. This will increase the transaction cost.

A lack of a public key for the customer and the merchant does introduce some problems. First, it is impossible for customers to inquire about the price of an item from a merchant privately unless both of them share a secret. Second, it is impossible for the merchant to sign a receipt to the customer which is verifiable by a third party. For the first problem, we solve it in another way by using the *pseudonym* concept introduced by Chaum [4]. For the second problem, we provide a weak form of non-repudiation, which we think is sufficient for micropayment. We will elaborate our method in section 5.

Although the public key methods have advantages over the private key methods, they are expensive. It is proper to make distinction between high value transactions and low value transactions. The security mechanism for a transaction worth several pennies should be different from that for a transaction worth several hundred dollars. Therefore, we should use as few public key operations as possible in our protocol design. But our protocol can be easily extended to be based on the public key cryptosystems.

3.3 Charging models

There are several commonly used models available to us to design our protocols for micropayments. They are the *billing (or subscription) model*, the *credit card model*, the *electronic check model*, the *electronic currency model*, and the *debit model*.

In the billing model, every customer registers with different merchants and obtains services (or goods) from the merchants. The transaction cost is low for this model. However, it is very inconvenient to the customers. If a customer wants to purchase goods from one hundred different merchants, he has to open accounts with these merchants and remember one hundred different encryption (decryption) keys. This is a very tedious task for the customers.

For the credit card model, the customer sends his credit card number to the merchant through some secure channel between them. But the high credit card transaction cost makes this model unsuitable for the micropayments.

The electronic check model is also a candidate. The electronic check model depends on a hierarchical banking infrastructure to transfer funds along a path inside the hierarchical structure. If the system is based on the public key cryptosystems, the banks have to provide on-line key revocation servers for their customers whose secret keys are compromised. These servers must be available to all merchants at any time. Any unavailability of these servers will cause the compromised secret keys to be abused by the adversaries who obtain these secret keys. If the electronic check system is based on the private key cryptosystems, a hierarchical accounting structure has to be established so funds can be transferred. Incorporating such an accounting infrastructure into the existing banking system will be expensive. Moreover, a fund transfer operation involves at least three accounting servers if the customer and the merchant do not share a common accounting server. Furthermore, the merchant has to clear the check on-line before he honors the customer's purchasing request. This means that the computer systems and the communication channels along the check clearing path must be reliable all time, which is not the case in a distributed environment.

The electronic currency is not considered since the electronic check model is just a simplified (or special) electronic currency model.

Based on the above observations, we consider the debit model as the model for our electronic payment protocol design. The money debit model is an on-line system. The customer's bank debits the customer's account, transfers the funds to the merchant's bank; the merchant's bank credits the merchant's account. This is the scenario of the debit model. It is not realistic for us to assume that every bank will provide on-line transfer service to its customers at the present time. Instead, a trusted electronic payment service provider is established to handle all fund transfers between the customers and the merchants. We call it the *billing service center*. Although the billing service center is a security bottleneck and performance bottleneck of the whole system, the simplicity of the debit model structure can reduce the transaction cost significantly.

4 The structure

Our protocol consists of three principals: the billing service center, the customer and the merchant. We assume that the billing service center is honest and is trusted by both the customer and the merchant. The customer and the merchant may be dishonest. The merchant and the customer open accounts and deposit funds in the billing service center.

The billing service center consists of many servers to handle The authentication and authorization of fund transfers between the customer's account and the merchant's account. The protocols for these operations are described in the next section. Besides the fund transfer operations, the billing service center also handles the following functions:

- Account administration for the customer and the merchant. This includes account openings, closures, fund transfers, balance inquiries, account statements, dispute resolutions, etc.
- Key generation and distribution for the customer and the merchant. The billing service center can also certify public keys for certain classes of customers and merchants at an additional fee.
- Administration of the merchant's services and handling complaints from the customers.

There may be more functions needed for the billing service center. They are not covered in this paper. We will only design protocols that transfer funds between the customer and the merchant securely and cheaply.

The billing service center shares a secret key $K_{cb}(K_{mb})$ with the customer (the merchant) and this key is known only to the customer (the merchant) and the billing service center. The billing service center has a certified public key K_b with the corresponding secret key K_b^{-1}.

A customer maintains a positive balance and authorizes charges against this account. When the billing service center receives an authorization by the customer, it checks the validity of the authorization, and debits the customer's account by the amount of the price agreed to by both the customer and the merchant. The billing service center then credits the merchant's account and sends an acknowledgement to the merchant.

Since the customer shares a secret with the billing service center and does not have a certified public key, the billing service center may forge the authorization by a customer, which can not be judged by a third party. This kind of attack is more likely from an insider (e.g., a malicious system administrator of the billing service center) than from an outsider. It can be prevented using the audit method. Moreover, since the amount of the payment is very low, the small cost due to this fraud can be covered by the billing service center, or the customer can change to another electronic payment service provider offering high quality services [2].

5 Our protocols

Before the transaction begins, the customer queries the merchant about the price of specific goods. Since we assume that neither the merchant nor the customer has a certified public key associated with him, it is impossible for both of them to establish a secure channel for the price information. Sometimes it is important to prevent a merchant from knowing the identity of his customers. It is also important to prevent the eavesdropper from knowing what the customer is purchasing and what the price is. We provide a weak form of a secure channel between the customer and the merchant by adopting the *pseudonym* concept [4]. A pseudonym is a nickname of a principal (not his true identity). A principal can have many pseudonyms. When a customer opens an account in the billing service center, several pseudonyms are assigned to the customer and these pseudonyms are known only to the customer and the billing service center. These pseudonyms are used by the customer to inquire about price information and to order goods from the merchants. Since the eavesdroppers and the merchants do not know the mapping between the pseudonyms and the true identity of a customer, the customer's privacy is protected.

This is a weak form of secrecy compared with the

[2] There are many such billing service centers in the Internet. They compete with each other.

1. $C \rightarrow M : Id_c, I_x, P_x?$
2. $M \rightarrow C : \{Id_c, M, I_x, P_x, T_m, ET_x\}_{K_{mb}}, I_x, P_x, T_m, ET_x$

Figure 1: Price Negotiation Protocol

public key approach. But it is more efficient. The price negotiation protocol is shown in Figure 1.

P_x is either a number denoting the price of the item x; or a list denoting the price policy of setting the price for the customer with pseudonym Id_c. For example, P_x can denote the list (*student* \$0.5 *member* \$1.0 *nonmember* \$1.5). In the later case, the customer knows what price he should pay since it knows which class he belongs to, while the adversary can not figure out what the price is for the customer with pseudonym Id_c. This is also a weak form of privacy for the price information. The price policy is enforced by the billing service center since it knows which category the customer belongs to. We call the message $\{Id_c, M, I_x, P_x, T_m, ET_x\}_{K_{mb}}$ the *price-form*. In case the merchant has a public key, then the price-form can be signed by his secret key K_m^{-1}. We may send a checksum on (I_x, P_x, T_m, ET_x) to prevent the tuple from being modified by the adversaries during its transmission. Even without the checksum, any modification on the tuple will be detected by the billing service center and the adversaries gain nothing from their malicious attacks.

Based on different communication paths among these three parties shown in Fig 2. we propose three protocols.

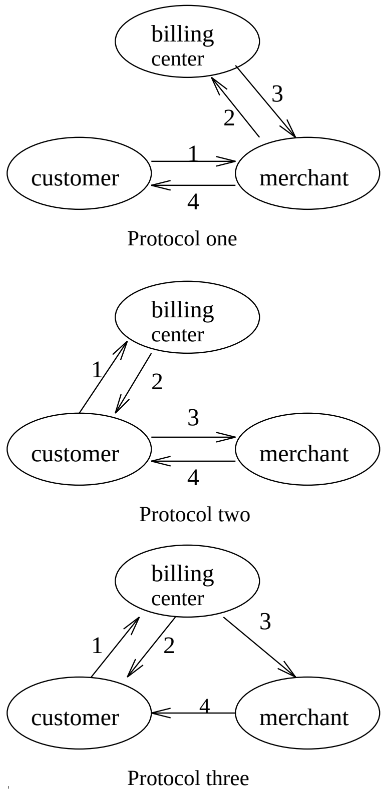

Figure 2: Design alternatives

1. $C \rightarrow M : orderform$
2. $M \rightarrow B : orderform, Id_c, M, \{Id_c, M, K_{cm}, T'_m\}_{K_{mb}}$
3. $B \rightarrow M : \{ok, Id_c, M, I_x, P_x, T_b, \{C, M, K_{cm}, T_b\}_{K_{cb}}\}_{K_{mb}}$
4. $M \rightarrow C : \{content(I_x)\}_{K_{cm}}, \{C, M, K_{cm}, T_b\}_{K_{cb}}$
   if ok from B.

Figure 3: Protocol one

5.1 Protocol one

Before the transaction begins, the customer obtains the price-form from the merchant through the price negotiation protocol. Our first protocol includes the following steps: The customer sends his payment authorization together with the price-form to the merchant; the merchant forwards this payment authorization to the billing service center. The billing service center checks the validity of the authorization, debits the customer's account by the amount specified in the authorization and credits the merchant's account. After the merchant receives an acknowledgement from the billing service center, the merchant provides the goods (or services) to the customer. The billing service center also generates a random key used by the customer and the merchant to encrypt the communication between them. In this protocol, the merchant communicates with both the customer and the billing service center. There is no communication between the customer and the merchant.

After running the protocol, C and M share key K_{cm}, C can then decrypt the information sent to him by M; or M rejects C's order since he does not have enough funds in his account or there is an inconsistency between the order-form and the price-form. We call the token $\{C, M, I_x, P_x, ET_x, T_c, serialnum, priceform\}_{K_{cb}}$ the *order-form*. The protocol is explained step by step:

1. The customer creates a fund transfer authorization containing

   –the merchant's and the customer's identities;

   –the item x and the associated price P_x;

   –a timestamp T_c read from the customer's clock;

   –the serial number for the order; and

–the price-form signed by the merchant during the price negotiation procedure.

Double encryptions are used to guarantee that P_x is agreed upon by both C and M. The merchant can not change it after it is signed. This is the order-form from C to M and is sent to B later for verification. A timestamp is used to guarantee the freshness of the price-form and the order-form [5]. The *serialnum* serves two functions here: one is to serve as a "confounder" to stop password guessing attacks from malicious merchants or eavesdroppers [11]; another one is to make it easy for the customers to record their purchases (like the serial number in personal checks).

2. M sends to the billing service center the payment agreement between C and M signed by both parties. K_{cm} is the decryption key of C which is used to decrypt the encrypted information sent to C by M. It is encrypted together with a timestamp by K_{mb} to guarantee the secrecy and freshness of K_{cm}. K_{cm} will be recorded with other records of this transaction by B. When B receives the authorization from C, it extracts the order-form and the price-form using the secret key K_{cb}. Then it checks if the price-form is consistent with the order-form. This includes checking whether the prices are the same, and whether the price offer is still in effect. Moreover, it verifies the timeliness of the order-form. If the message is determined to be valid, B debits the customer's account by the amount specified in P_x and credits the merchant's account. In case P_x is a price list, B sets the price for the merchant according to the price policy. If the customer has not enough funds in his account or the customer is not eligible to purchase the goods (e.g., a customer under eighteen years old is not allow to buy pornographic materials), B acknowledges to C and the transaction aborts. Otherwise, B goes to the next step.

3. B acknowledges to M that he has transferred funds to the merchant's account by including the following fields in the acknowledgement form: the merchant's identity and the customer's pseudonym; the item and the corresponding price; the timestamp and the encrypted secure communication key between C and M. Double encryption here is to prevent C from getting $\{C, M, K_{cm}, T_b\}_{K_{cb}}$ and obtaining the key K_{cm} directly from B. It guarantees that C obtains the decryption key from M directly. If this is not a requirement, then the double encryption is not necessary here. Since we include M inside $\{C, M, K_{cm}, T_b\}_{K_{cb}}$, M's identity is also authenticated to the customer.

4. M sends the decryption key to C. The encryption file can be sent to C by M at any step during the protocol.

Usually, at the end of transaction, the merchant should give the customer a receipt. The receipt serves as a function that the customer can prove to the merchant or a third party that the transaction between the merchant and him did happen. In most of the cases the receipt is used by the customer to return goods to the merchant or to present the receipt to a third party for refund. But information goods have the special property that they can not be returned once purchased. Since the merchant does not have a public key, it is impossible for him to sign a verifiable receipt to the customer. Under our structure, the merchant can provide the customer a weak form of receipt. Because all transactions go through the billing service center, the receipt can be provided by the billing service center to the customer if necessary. The receipt has the form $\{Id_c, M, I_x, P_x, ET_x, T_c\}_{K_b^{-1}}$. The billing service center can also serve as a "witness" if a dispute occurs between the customer and the merchant. In this case, the billing service center does not have to sign the receipt.

At the final step of the protocol, the merchant sends $\{content(I_x)\}_{K_{cm}}$ to the customer. A dishonest customer may claim to the billing service center that he has not received the goods even through he did. The billing service center can ask the merchant to send the goods again. Unlike physical goods, information goods are easy to replicate, and the merchant can send duplication of a given good many times.

Neither the customer nor the merchant can modify the price in the price-form or the order-form during the transaction, since otherwise the billing service center can detect the inconsistency and abort the transaction.

If a dishonest merchant sells low-quality goods to the customer, the customer can complain to the billing service center. The billing service center who administrates all merchants can stop providing service to those merchants with poor reputations.

A dishonest merchant may collude with other dishonest merchants or dishonest customers by showing them the order-form from his customers. Since the merchant's identity and a timestamp are included in

1. $C \rightarrow M : \{C, M, I_x, P_x, ET_x, \{C, serialnum, T_c\}_{K_{cb}}\}_{K_b}$
2. $M \rightarrow B : \{C, M, I_x, P_x, ET_x, \{C, serialnum, T_c\}_{K_{cb}}\}_{K_b},$
$\{M, Id_c, I_x, P_x, ET_x, \{Id_c, M, K_{cm}, T_m\}_{K_{mb}}\}_{K_b}$
3. $B \rightarrow M : \{\{ok, Id_c, M, I_x, P_x, T_b,$
$\{C, M, K_{cm}, T_b\}_{K_{cb}}\}_{K_b^{-1}}\}_{K_{mb}}$
4. $M \rightarrow C : \{content(I_x)\}_{K_{cm}}, \{C, M, K_{cm}, T_b\}_{K_{cb}}$

Figure 4: Protocol one based on public key method

the order-form, the order-form can not be reused by other merchants.

The biggest threat from the adversary is the key guessing attack, since the format of the price-form and the order-form are predetermined. Although the exact value of T and $serialnum$ may not be known in advance, a plausible range of values can be determined. This structure property generally increases the vulnerability of these messages to a guessing attack. The vulnerability is worsened by the generation of the keys such as K_{cb} and K_{mb}. Usually, a customer of the billing service center (here the merchant is a customer of the billing service center) chooses a password or a PIN number. This password or pin number, together with a random number, are applied to a one-way hash function, and thus the secret keys shared between the customer and the billing service center are generated. The structure properties and the way that the keys are generated make the protocol weak against password-guessing attacks since all the attacks can be done off-line without being perceived. For example, an eavesdropper can collect the order-forms from the customer C and take a guess of the key K'_{cb}. If $\{orderform\}_{K'_{cb}}$ contains C and M, this means that his guess is right with high probability. To prevent this kind of attack, we extend our protocol based on the public key cryptosystem. (only the billing service center has a public key.) The public key cryptosystem is known to be secure against the known plain-text attacks [8]. The modified protocol is shown in Figure 4.

5.2 Protocol two and protocol three

Before the transaction begins, the customer obtains the price-form from the merchant through the price negotiation procedure. Protocol two is shown in Fig 5. The protocol consists of the following steps: In message 1, the customer sends the price-form from and his order-form to B directly. B checks the consistency between the order-form and the price-form which are signed by C and M respectively. If they are consistent, B debits C's account and credits M's account, and generates a key K_{cm} for secure communication between C and M. He then sends the key to C in message 2, and sends the acknowledgement form $\{ok, Id_c, M, I_x, P_x, K_{cm}, T_b\}_{K_{mb}}$ to M. In message 3, C forwards the acknowledgement form to M so that M can check the validity of the form. M then sends the content of I_x to C using the encryption key K_{cm} in message 4.

1. $C \rightarrow B : \{M, Id_c, I_x, P_x, ET_x, T_m\}_{K_{mb}},$
$\{C, M, I_x, P_x, ET_x, T_c\}_{K_{cb}}, Id_c, M$
2. $B \rightarrow C : \{C, M, I_x, P_x, K_{cm}, T_b, balance\}_{K_{cb}},$
$\{ok, Id_c, M, I_x, P_x, K_{cm}, T_b\}_{K_{mb}}$
3. $C \rightarrow M : \{ok, Id_c, M, I_x, P_x, K_{cm}, T_b\}_{K_{mb}},$
$\{Id_c, addr, T_c\}_{K_{cm}}$
4. $M \rightarrow C : \{content(I_x)\}_{K_{cm}}$

Figure 5: Protocol two

1. $C \rightarrow B : \{M, Id_c, I_x, P_x, ET_x, T_m\}_{K_{mb}},$
$\{C, M, I_x, P_x, ET_x, T_c\}_{K_{cb}}, Id_c, M$
2. $B \rightarrow C : \{C, M, I_x, P_x, K_{cm}, T_b, balance\}_{K_{cb}}$
3. $B \rightarrow M : \{ok, Id_c, M, I_x, P_x, K_{cm}, T_b\}_{K_{mb}}$
4. $M \rightarrow C : \{content(I_x)\}_{K_{cm}}$

Figure 6: Protocol three

The only difference between protocol three and protocol two is that B sends the acknowledgement to the merchant directly without going through the customer. Protocol three is shown in Fig 6. Security analysis for these two protocols is exactly the same as that we did for protocol one.

5.3 Discussion

Our protocols are quite similar to those protocols designed for authentication in distributed systems. If the price policy set by the merchant is that the price is zero for every customer in a specific group, our payment protocols can function as authentication protocols (to authenticate if a customer belongs to a group). In this case, protocol two is in the style of the protocol designed by Needham and Schroeder [15] and is the same as the Kerberos protocol [17]. Protocol one is in the style of the protocol designed by Otway and Rees [13]. Our electronic payment protocols can be regarded as extensions of the protocols for authentication in distributed systems. The formal method [1] can also be applied to our protocols. Some practical authentication system such as Kerberos can be extended to handle micropayments if we add additional fields into the Kerberos message structure. This means that we do not have to build an electronic payment system from scratch, which will save us a lot of work for implementation and therefore reduce the transaction costs.

Protocol one is suitable for an electronic payment system in which the merchants are put together in the same realm. Protocol two is suitable for an electronic payment system in which the customers are in the same realm.

The "cheapness" of our protocols is achieved in the following aspects.

- *Communication complexity.* The number of messages passing among the customer, the merchant, and the billing service center is optimal [9, 10].

- *Computation complexity and key management complexity.* We base our protocols on the private key cryptosystems. Our protocol requires significantly less computation than its public key counterparts. The disadvantages introduced by using the private key cryptosystems are remedied by employing the properties unique to the information goods. The secrecy of the price negotiation between the customer and the merchant is guaranteed by the pseudonym concept, which can be implemented cheaply. The receipt of the purchase is achieved in a cheap way due to the following reasons. First, the information goods can not be returned once purchased. Second, the information goods are easy to copy. The merchant can send the information goods to a customer several times if the customer claims that he has not received the information goods which has been paid.

- *Simple banking structure.* We use a debit model consisting of three parties. This model reduces the interactions with the existing banking system, hence also reduces the transaction costs.

6 Conclusions

We have presented a set of protocols suitable for micropayments in distributed systems. Our protocols are "cheap" and satisfy the requirements that a payment system should have. We reduce the charging cost in two ways. First, we analyze the tradeoffs among different charging models and different cryptographic algorithms and base our choice on this analysis. Second, we observe the special properties unique to the information goods and optimize our protocols based on these observations. Our protocols are also natural extensions to the protocols for authentication in distributed systems. The scalability of the system based on our protocol suffers if the system grows. Further study is needed to address this problem.

7 Acknowledgements

I am grateful to Robert Carr for his helps to improve the presentation of this paper. I also want to thank the following people for their comments and constructive criticisms: Li Gong (SRI International), David Kristol (AT&T Bell Labs), and Bennet Yee (Microsoft).

References

[1] M. Abadi, M. Burrows, and R. Needham. A Logic of Authentication. *ACM Transaction on Computing Systems*, 8(1):18–36, 1990.

[2] M. Abadi and R.M. Needham. Prudent Engineering Practice for Cryptographic Protocols. Technical Report SRC 125, DEC System Research Center, 1994.

[3] M. Bellare and et al. iKP–A Family Secure Electronic Payment Protocols. Technical report, IBM Zurich Research Lab, 1995.

[4] D. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Comm. of ACM*, 24(2):84–88, 1981.

[5] D.E.Denning and G.M.Sacco. Timestamps in Key Distribution Protocols. *Communications of ACM*, 24(8), 1981.

[6] W. Diffie and M.E. Hellman. New directions in Cryptography. *IEEE Transaction on Information Theory*, 22(6):644–654, 1976.

[7] T. El Gamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Trans. Information Theory*, IT-31(4):469–472, 1985.

[8] S. Goldwasser, S. Micali, and R. Rivest. A Digital Signature Scheme Secure Against Adaptive-Chosen Message Attacks. *SIAM J. of Computing*, 17(2), 1988.

[9] Li Gong. Lower Bounds on Messages and Rounds for Network Authentication Protocols. In *Proceedings of the 1st ACM conference on Computer and Communication Security*, Nov 1993.

[10] Li Gong. Efficient Network Authentication Protocols: Lower Bounds and Optimal Implementations. Technical report, SRI International, Computer Science Lab, 1994.

[11] L.Gong, M. Lomas, R.M. Needham, and J.H. Saltzer. Protecting Poorly Chosen Secrets from Guessing Attacks. *IEEE Journal on Selected Areas in Communication*, 11(5), 1993.

[12] C. Neuman and G. Medvinsky. Requirements for Network Payment: The Netcheque Perspective. In *Proceedings of IEEE Compcon'95*, March 1995.

[13] D. Otway and O. Rees. Efficient and Timely Mutual Authentication. *Operating System Review*, 21(1), 1987.

[14] R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-key Cryptosystems. *Communications of ACM*, 21(2), 1978.

[15] R.M.Needham and M.D.Schroeder. Using encryption for Authentication in Large Networks of Computers. *Communications of ACM*, 21(12), 1978.

[16] R.M.Needham and M.D.Schroeder. Authentication Revisted. *Operating System Review*, 21(1), 1987.

[17] J. Steiner, C. Neuman, and J. Schiller. Kerberos: An Authentication Service for Open Network Systems. In *Proceedings of the USENIX Winter Conference*, pages 191–202, February 1988.

[18] J. Tardo and K. Alagappan. SPX: Global Authentication using public key certificates. In *Proceedings of the 1991 IEEE Symposium on Research in Security and Privacy*, pages 232–244, 1992.

[19] V.L. Voydock and S.T. Kent. Security mechanisms in High-level Network Protocols. *ACM Computing Survey*, 15(2):135–171, June 1983.