PayFair: a prepaid internet micropayment scheme ensuring customer fairness

S.-M.Yen

Abstract: A software-based prepaid micropayment scheme is developed. As with existing prepaid micropayment schemes, the profits of the merchants are protected. Furthermore, in this proposed scheme, fairness for the customers is also assured. More precisely, in this new scheme, the merchant, after receiving prepaid money, can only claim that a customer has already spent a specific amount of money by showing the required cryptographic witness which can only be received from that customer when making a payment. Most importantly, in this new scheme, no public key-signature computation is required. Finally, it will be shown that, owing to its simplicity and high performance, the proposed scheme can also be employed by small pieces of electronic equipment for general-purpose resource-access control.

1 Introduction

Internet micropayment schemes have recently received growing attention, largely due to the fact that these schemes exhibit the potential of being embedded in numerous internet-based applications. As a special type of electronic payment [1], micropayment schemes allow a customer to transfer to a merchant a sequence of small payment amounts over the computer network, in exchange for services or electronic products from the merchant. With these services or products, it is often not quite appropriate to pay the total amount of money either in advance or afterwards. More precisely, each small amount of payment should be received and verified by the merchant when serving the customer.

Possible practical applications of the above micropayment model include digital newspapers [2], on-line journal subscriptions, on-line database queries, multimedia entertainment over the internet, and internet advertising, e.g. via lottery tickets [3]. More examples can be found in [3, 4]. In addition, accounting and pricing for internet services and mobile telecommunications may represent yet another set of promising applications for micropayments [5–9].

Until recently, there were basically two types of internet micropayment schemes: a 'prepaid' approach, e.g. Millicent [10, 11] and a 'postpaid' approach, e.g. PayWord [4] and some very similar schemes [12–14]. Each of these has its merits as well as disadvantages. The main risk of the postpaid approach is a customer having inadequate credit to pay for the items purchased, whereas the main problem in the prepaid approach is how to protect the customer's credit from unscrupulous merchants.

In this paper, a novel prepaid internet micropayment scheme is developed. The new micropayment scheme protects the profits of the merchants and the credit of the customers. As with existing prepaid micropayment schemes, the profits of the merchants are protected. The aim of the new scheme is to protect the customers' credit during the payment process. Unlike the Millicent scheme, the merchant is unable to cheat, as will become apparent during the description of the proposed scheme. More precisely, in this new scheme, the merchant after receiving prepaid money can only claim that a customer has already spent a specific amount of money by showing the required cryptographic witness, which can only be received from that customer when making a payment. This is similar to the PayWord scheme. Most importantly, the scheme has been developed to assure advantages for both the customers and the merchants. No public key-signature computation is required if a trusted third party (often, it is the bank) can be assumed. In fact, this assumption is quite common for many cryptographic schemes.

In the proposed micropayment scheme, no tamper resistant device is assumed. Therefore, a software-only implementation is possible and this is also considered by the PayWord and the Millicent micropayment schemes. Stern and Vaudenay [15] developed another efficient micropayment scheme (called the SVP scheme) based on the usage of a tamper resistant device and a cryptographic one-way hash function. However, it was recently pointed out [16] that a vulnerability of the SVP payment scheme is possible from the viewpoint of hardware transient-fault cryptanalysis, especially by manipulating the random-number generation proposed by Zheng and Matsumoto [17].

Since some parts of the PayWord scheme will be employed and a performance comparison of the proposed new micropayment scheme with both PayWord and Millicent will be given, a brief review of both schemes is included before the new proposal is set out.

2 Review and some remarks on PayWord and Millicent

In this Section, only the very fundamental ideas of both PayWord and Millicent will be reviewed. For more details, the readers are suggested to refer to the original papers [4, 10, 11] or [1]. Besides the review, we also give some remarks on the schemes, especially the Millicent scheme.

IEE Proceedings online no. 20010743

DOI: 10.1049/ip-cdt:20010743

Paper received 18th August 2000

The author is with the Laboratory of Cryptography and Information Security, Department of Computer Science and Information Engineering, National Central University, Chung-Li, Taiwan 320, Republic of China

In both schemes, there are three participants involved: they are the customer, the merchant, and the bank (usually assumed to be a trusted third party).

2.1 The PayWord system

The PayWord micropayment scheme developed by Rivest and Shamir [4] will be reviewed briefly. Before the review, some notations and symbols about a one-way hash chain are reviewed in the following.

Notation 1: When a function h is iteratively applied r times to an argument w_n, the result will be denoted as $h^r(w_n)$, that is

$$h^r(w_n) = \underbrace{h(h(\cdots(h}_{r \text{ times}}(w_n))\cdots))$$

When the function $h(\,)$ in the iteration is instantiated with a cryptographic one-way hash function, such as MD5 [18], SHA [19] and HAVAL [20], the result is a one-way hash chain, as shown below. Note that within the chain, each element w_i is computed as $h^{n-i}(w_n)$:

$$\begin{aligned} w_0 = h^n(w_n) \leftarrow w_1 = h^{n-1}(w_n) \leftarrow \cdots \leftarrow w_{n-1} \\ = h^1(w_n) \leftarrow w_n \end{aligned}$$

The fundamental cryptographic tool of the PayWord scheme is the one-way hash chain which is well known owing to its previous application by Lamport [21, 22] in the development of one-time passwords. In the PayWord micropayment scheme, a customer commits to a given merchant a payment chain consisting of values

$$w_0, w_1, w_2, \cdots, w_n$$

by signing a message containing the root value w_0 (e.g. using the RSA [23] public key-signature scheme) where $w_i = h(w_{i+1})$ for $i = 0, 1, \ldots, n-1$, and $h(\,)$ is the underlying cryptographic one-way hash function. The value w_n is a secret value selected at random by the customer. In this paper, w_i will sometimes be denoted as $h^{n-i}(w_n)$.

The number w_1 is the first coin to be paid. Each successive payment is made by releasing the next consecutive value in the payment chain, which can be verified easily by checking that it hashes to the previous element. Each element w_i in the chain is predefined to be of a fixed value for a small payment.

For example, the customer C will initially sign $Sign_C$ (Merchant-ID $\|\, w_0 \,\|$ Cert) using public key cryptography with his private key. Within the signature, 'Cert' is the certificate issued by the bank of the PayWord scheme used to indicate the credit status of the customer. The customer then sends

$$Sign_C(\text{Merchant-ID} \,\|\, w_0 \,\|\, \text{Cert}), \text{Merchant-ID}, \text{Cert}, w_0$$

to the merchant. On the customer's side, he will store the last coin w_n and the number of coins spent, say the variable j. In the merchant's machine, the message w_0 with its signature and the last received coin from the customer w_k should be kept. Each time, when the customer wishes to pay the merchant another new coin, he computes the next new coin $w_{j+1} = h^{n-(j+1)}(w_n)$ and updates j to be $j+1$. The merchant just checks whether $h(w_{j+1}) = w_k$ and updates the last received coin to be w_{j+1} if it is valid. This ensures that a sequence of small payments can be given to a specific merchant and the customer has to make only one computationally expensive public key-based digital signature which is for the purpose of commitment.

The role of the bank in the PayWord scheme is to issue a certificate for each customer and to transfer real money (although still in electronic form) from the customer to the merchant when the merchant can prove the payment via a cryptographic approach, i.e. presenting the signature received from the customer.

2.2 The Millicent system

The Millicent scheme was developed by a group of researchers in the Digital Equipment Corporation [10, 11]. The scheme is different from PayWord in that Millicent adopts the prepaid approach. Each customer should buy a bank ticket (called the 'bank scrip' in Millicent) in advance which is used to buy 'merchant scrips'. With a specific merchant scrip, a customer can pay a sequence of micropayments to that specific merchant. This two-level hierarchical payment scheme uses a similar approach for authentication to that employed by the Kerberos system [24, 25]. A multi-level payment model based on Millicent is possible where the required level number is application dependent.

Both the bank scrip and the merchant scrip use the same data format (with only minor differences), so only the merchant scrip will be shown as {Merchant-ID, Value of $, Scrip-ID, Cust-ID, Expiry, Optional_info}, Cert where 'Cert' is the certificate of this scrip and is computed as

$$h(X \,\|\, \textit{Master scrip secret})$$

where X is all the other data of the scrip except the Cert, and $h(\,)$ is the underlying one-way hash function, e.g. the MD5 system. The Master scrip secret is known only to the creator of the scrip, so both the bank and the merchant have their own scrip secrets to generate either bank scrip or merchant scrip, respectively. The Cust-ID need not have any relationship with the real identity of the customer. It, however, must be unique to every customer since this identity number will be used to generate the required customer secret for each customer, as will be seen later. All the other data in a scrip is obvious from their field names.

The bank first buys many merchant scrips from each merchant and of course the bank stores them in a secure and reliable manner. It is evident that either loss or theft of the scrips will damage the bank. In fact, in a more secure mode of Millicent to be reviewed later, the bank can only buy each required merchant scrip on line from the merchant. This is because of the inclusion of the Cust-ID in the merchant scrip. Each customer buys a bank scrip and when required uses it to buy a specific merchant scrip.

The bank re-computes and returns a modified bank scrip (with the specific merchant scrip) to the customer, containing a smaller 'value of $'. This new 'value of $' is the balance remaining of the bank scrip after deduction of the merchant scrip. Taking the received merchant scrip, the customer can pay a micropayment to that specific merchant. The merchant can check the validity of the scrip because he knows the 'Master scrip secret' used to generate the scrip. Of course, the merchant can compute a modified merchant scrip with a smaller 'value of $', then returns this new scrip to the customer with the service. This new 'value of $' in the merchant scrip is the balance remaining of the scrip after deduction of the micropayment.

The above idea is simple but with a major deficiency because both the returned/modified bank scrip and the merchant scrip are transmitted in cleartext form. This enables a passive attacker to intercept and use the scrips. To overcome this drawback, the developers of Millicent proposed another secret (called the 'Customer secret') for

the customer to protect the payment. We call this the secure mode of Millicent. In fact, two different secure modes are suggested in Millicent, and we will first review the request signature mode. The specific customer secret received by a merchant is computed as

$$h(\text{Cust-ID} \parallel \textit{Master customer secret})$$

where 'Master customer secret' is known only to the merchant. Another customer secret received from the bank is computed by a similar approach. With the knowledge of Customer secret, each time the customer wishes to spend a payment, he must compute a request signature as

$$h(\text{merchant scrip} \parallel \textit{Customer secret} \parallel \text{request info})$$

The merchant can check the validity of this request signature because he knows the customer secret, but the attacker fails to generate this signature without the knowledge of that secret.

The other secure mode of the Millicent uses a conventional encryption system, e.g. the DES [26], to protect the scrips sent and received between the customer and the merchant. However, no matter which secure mode is employed, the use of 'customer secret' complicates the implementation and maintenance on the customer's side. Loss or theft of the secrets will make the payment scheme no longer usable or secure. Note especially that the number of secrets to be stored can be large in a practical situation.

3 The proposed prepaid micropayment scheme

In this proposed micropayment scheme, each customer C opens a micropayment account and shares a secret key K_C with the trusted bank B. Each merchant M also has an account in the bank and shares a secret key K_M with B.

The payment scheme is divided into two phases: the prepaid phase and the micropayment phase. In the prepaid phase, before sending a sequence of micropayment to any merchant, the customer prepays an amount of money to the bank from his micropayment account and receives some information (we call it a 'token' hereafter) from the bank which will be required during the real payment phase. Note that the token received from the bank has no relationship with any specific merchant until the payment is conducted. This is unlike the Millicent scheme where the merchant scrip purchased from the bank has already been bound to a specific merchant. In the proposed micropayment scheme, tokens purchased in the prepaid phase are general purpose, and it will be shown in the next Section that this novel feature will be more efficient for interaction (or communication) than in the Millicent scheme.

A fresh token enables the customer to prepare and to spend the micropayment in the second phase. Fresh means that the token has not been doubly spent. It is in the payment phase that money will be transfered from the bank to a specific merchant after the validity and double-spend checks.

In order to understand each step of the payment scheme, security analysis will be given after each payment interaction description.

3.1 Phase A: the prepaid phase

The bank selects a bank secret key SK which is used to generate the 'random seed coins' as $E_{SK}(N, RN)$ where N is a non-repeated serial number, RN is a random nonce derived from the serial number N and another secret SK_{RN} (known only to the bank), and $E(\,)$ is any secure symmetric cryptographic system, e.g. the DES system [26]. It should be noted that, even if the serial number N is known, the cipher $E_{SK}(N, RN)$ behaves like a random number generated by the bank secretly. Henceforth, $E_{SK}(N, RN)$ will be called a token. The inclusion of a random nonce RN within $E_{SK}(N, RN)$ helps to prevent a known plaintext attack [27], since the serial number N is known to an attacker. If the random nonce RN is not derived from the serial number, then the bank needs to store each nonce for a corresponding token (or value N). This greatly complicates the implementation.

Before requesting service from a merchant, the customer C sends an 'authenticated' order to the bank as

$$C \rightarrow B: \quad \text{ID}_C, O_C, h(O_C, K_C) \tag{1}$$

where ID_C is the customer C's identity, O_C is C's non-repeated order number, and $h(\,)$ is the underlying one-way hash function, e.g. MD5 or SHA. Note that some well known message authentication schemes using a keyed hash technique [28, 29], e.g. HMAC-MD5 or HMAC-SHA-1, could be used to ensure secure application of the key, K_C, with the hash function. The bank maintains a table of each customer's last order number O_C and his secret key K_C; therefore it can check the authenticity of this order.

If the order is verified to be valid, the customer receives one random token $E_{SK}(N, RN)$ from the bank through a 'secure channel', i.e. under encryption using the customer's secret key K_C as the encryption key:

$$B \rightarrow C: \quad E_{K_C}(E_{SK}(N, RN), RT), N, h(E_{SK}(N, RN), N, O_C, K_C) \tag{2}$$

and of course pays the bank a specific amount of money from his micropayment account. The purpose of $h(E_{SK}(N, RN), N, O_C, K_C)$, as a keyed hash, is for a data integrity check. The inclusion of a random nonce RT in $E_{K_C}(E_{SK}(N, RN), RT)$ is to prevent a possible known plaintext attack [27], since the token $E_{SK}(N, RN)$ will finally be known to an attacker only after it has been spent.

The bank keeps a table to record which random token (in fact the serial number N is sufficient) is bought by which customer. The purpose of this table is to avoid possible double spending by the customer or double money transfer requests (to be described later) from the merchant.

3.2 Phase B: the micropayment phase

3.2.1 The coin minting process: After receiving $E_{SK}(N, RN)$, a secret number known only to B and C, the customer commits to a given merchant (say M) a payment chain consisting of values

$$w_0, w_1, w_2, \ldots, w_n = E_{SK}(N, RN)$$

by releasing the root value w_0 and the serial number N to the merchant, where $w_i = h(w_{i+1})$ for $i = 0, 1, \ldots, n-1$ (or more precisely, $w_i = h^{n-i}(w_n) = h^{n-i}(E_{SK}(N, RN))$). Since the above two values will be sent to that specific merchant through an open communication channel, any intruder intercepting this information can pretend to be the specific merchant. An authenticator $h(w_0, \text{ID}_M, K_C)$, a keyed hash, is computed and sent to the merchant for him to prove the ownership of w_0. ID_M is the identity of that specific merchant. Therefore the customer sends the following to the merchant

$$C \rightarrow M: \quad w_0, N, h(w_0, \text{ID}_M, K_C) \tag{3}$$

Each element w_i in the chain is predefined to be of a fixed value for a small payment. The above process is in fact identical to the process in the PayWord micropayment

scheme developed by Rivest and Shamir [4], except that now the last coin w_n is set to be the random token $E_{SK}(N, RN)$ instead of a randomly selected number and a digital signature, for example the RSA signature [23], for the root value w_0 is not required. It should be noted that PayWord is a postpaid scheme but the new proposal is a prepaid scheme.

3.2.2 The money transfer and payment-initialisation process: The merchant first sends the following information to the bank

$$M \rightarrow B: \quad w_0, N, \mathrm{ID}_C, \mathrm{ID}_M, R_M, h(w_0, \mathrm{ID}_M, K_C) \qquad (4)$$

to prove the ownership of the payment where ID_C is for the bank to derive the customer's secret key and R_M is a random integer selected by the merchant. Because the bank knows SK and SK_{RN}, it can reconstruct w_n as $E_{SK}(N, RN)$ (where RN is derived from both N and SK_{RN}) and therefore computes w_0 as $h^n(w_n) = h^n(E_{SK}(N, RN))$ where n is a predefined system parameter, e.g. $n = 100$. If the computed w_0 is identical to the received one and the value $h(w_0, \mathrm{ID}_M, K_C)$ is verified correctly, the bank will send an acknowledgment to the merchant through an 'authenticated channel' as

$$B \rightarrow M: \quad w_0, \mathrm{ID}_C, \mathrm{ID}_M, \mathrm{YES}, h(w_0, \mathrm{ID}_C, K_M, R_M, \mathrm{YES}) \qquad (5)$$

and transfer the money into the merchant's account (of course some service charge should be received by the bank). Sending an acknowledgment (positive or negative) to the merchant is to notify whether the payment chain to be used is valid and is not doubly spent. This can only be accomplished by the bank because the verification process needs SK and SK_{RN}. The purpose of $h(w_0, \mathrm{ID}_C, K_M, R_M, \mathrm{YES})$ is for the data integrity check of the acknowledgment. At the same time, the bank adds information to the table with which merchant the random token (the serial number N is sufficient) will be spent by a specific customer (already in the record). Since the bank maintains such a table of payer identity and payee identity for each sold token, neither a double spending from the customer nor a double money transferring request from the merchant will be possible. Of course, if the verification of w_0 fails, another acknowledgment containing NO and $h(w_0, \mathrm{ID}_C, K_M, R_M, \mathrm{NO})$ will be sent to M.

Note, especially, that without the inclusion of a random integer R_M, a 'replay attack' can be conducted which makes double spending possible. The following scenario demonstrates the attack. An attacker collecting previous interactions (exprs. 3 and 5) and all previously spent w_i ($i = 1, 2, \ldots, n$), can replay the acknowledgment information (expr. 5) to the merchant and can doubly spend all w_i. This is because, in the proposed micropayment scheme, we wish to simplify the merchant's storage requirement such that all previously spent payment-chain information (including w_0) need not be stored in the merchant's machine. Therefore, a challenge-response approach using R_M as the nonce can easily solve the above replay attack. However, if the merchant stores all the previously received payment chain, especially the item w_0, the parameter R_M is not required, but this depends on the implementation.

3.2.3 The micropayment process: Each successive payment is made by releasing the next consecutive value in the payment chain, which can be easily verified by the merchant by checking that it hashes to the previous element. This process is almost the same as the PayWord scheme [4]. During the payment, the merchant needs only to keep the last spent coin of each chain which is not fully spent. It is evident that the merchant does not even require a counter for each chain if the parameter n is a fixed value. When all the n micropayment coins of a chain are spent, the last coin w_n (or $E_{SK}(N, RN)$) and its corresponding serial number N should be stored by the merchant as a proof of spending. Of course, if we consider the expiration of the coin-chain, then the expired w_n can be removed from the memory.

Another approach to minimise the merchant's storage space needed to store the already spent payment chain, i.e. w_n and N, is suggested as follows. Since the bank is assumed to be trusted and it has already maintained a table of sold tokens and their related information, the merchant can commit the status of each payment chain to the bank after the chain has been fully spent. After receiving the last coin w_n from the customer, the merchant can send the following commitment information to the bank

$$M \rightarrow B: \quad w_n, N, \mathrm{ID}_M, h(w_n, N, K_M) \qquad (6)$$

The bank can of course check the above request by verifying whether $w_n \stackrel{?}{=} E_{SK}(N, RN)$. If the request is verified, then the bank marks the chain associated with the token $E_{SK}(N, RN)$ (or just N) as fully spent and sends the following acknowledgment back to the merchant

$$B \rightarrow M: \quad w_n, h(w_n, K_M) \qquad (7)$$

After receiving and verifying the acknowledgment, both w_n, w_0 and N are not required by the merchant and can be removed.

4 Discussions and performance analysis

The proposed micropayment scheme will be compared with the PayWord and Millicent schemes on various aspects. Through these comparisons and analyses, the practicality of the new scheme can be seen.

4.1 Postpaid versus prepaid

Although not a technical issue, the adoption of postpaid or prepaid methodology greatly affects a payment scheme construction and its properties.

Postpaid: In this payment model, in order to receive the money afterwards, the merchant should obtain a strong witness from the customer as proof to the bank, which usually involves a cryptographic computation using the customer's private information. Typical examples are a digital signature generated by the customer (an expensive approach) and secret release of a committed one-way function such as the hash chain employed in PayWord (a cheap approach). Of course, the risk of the postpaid scheme is the poor credit of customers.

Prepaid: In this model, the money has already been paid in advance either to the merchant or to the bank. The interests of the merchant and the bank are well protected. However, the big problem in this payment model is how the interests of the customer can be protected.

The proposed scheme adopts the prepaid approach, such that money is paid to the bank from the customer's account when purchasing tokens and at the same time the interests of the customer can be protected subject to the bank being trustworthy. The new micropayment scheme achieves a good balance of the prepaid and the postpaid requirements.

4.2 The fairness point of view

In the developed prepaid scheme, and also PayWord [4], each payment is achieved via releasing a secret value in the cryptographic one-way hash chain, in the computationally infeasible direction. Based on this strong cryptographic construction, the merchant can only claim that the customer has already spent a specific amount of money by showing the required secret value. Where the customer may try to double spend a coin-chain which has already been spent at other merchants or at the present merchant, the problem of double spending can be easily overcome by verifying each new chain in advance through the money transfer and payment initialisation process. Evidently, the proposed scheme is fair for both the customer and the merchant.

However, in the Millicent scheme [10, 11] (also a prepaid scheme), the original merchant scrip and all the following modified scrips (with lessening scrip value) are computed by the merchant himself. The Millicent scheme developers emphasise that money cannot be doubley spent, however, if the merchant falsely claims that an unused scrip has been sent from the customer, nothing can help the customer. The Millicent prepaid scheme is fair only for the merchant.

4.3 Elimination of the digital-signature requirement

In the original PayWord [4] scheme, the customer needs to generate a digital signature, e.g., the RSA signature [23], on the root value w_0 and some other required information, e.g. the identity of the merchant to receive the payment. This is essential because PayWord is a postpaid scheme and the digital signature is used as a witness for the customer's promise to pay afterwards. However, this requirement of a public-key-based digital signature was reported to be the main disadvantage [1, 15].

The proposed new micropayment, on the other hand, is a prepaid scheme and each new coin-chain has been verified in advance through the money transfer and payment initialisation process. Therefore no digital signature is required for witness of the payment promise.

4.4 Software-only implementation

It is evident that the proposed payment scheme can be implemented in software such as the PayWord and the Millicent schemes. The customer or the merchant needs only to know his own secret key, K_C or K_M, but not any bank's master secret key as adopted in the SVP scheme [15]. The main reason for implementing the scheme in hardware, e.g. a smart IC card, would be mainly for the reason of portability but not security. Of course, if the smart-card-implemented system allows the customer to conduct purchasing from other people's machines, security protection of the stored data would be required. But this does not violate our claim of a software-only concept.

4.5 Interaction efficiency

We shall now consider a practical situation of applying a micropayment scheme. It is reasonable to assume that the customer does not know in advance how much micropayment will be paid to a specific merchant during a period of network connection. Therefore, a new merchant scrip or payment chain will be purchased or generated when required. This situation differentiates the proposed scheme from the PayWord and the Millicent schemes.

In the PayWord scheme, each time a new payment chain is generated, the customer needs to evaluate a computationally expensive digital signature. Furthermore, since PayWord is a postpaid scheme, customers spending more than their funds would cause the merchant to risk his profit. In this paper, we only consider the prepaid situation.

It has been pointed out, when reviewing the Millicent scheme, that if either of the secure modes are used (to provide protection), owing to the inclusion of 'Cust-ID' inside the merchant scrip, the bank can only buy each required new merchant scrip (requested on line by the customer) on line from the merchant. The interaction for a customer to purchase a new scrip and to spend it is shown in Fig. 1*a*.

On the contrary, in the proposed scheme, a token purchased from the bank will be bound to a specific merchant after conducting the real payment phase. Many tokens can be purchased in advance from the bank and stored in the customer's machine. This improves the performance and also the complexity of sending another new payment chain to any specific merchant. This simplified interaction is shown in Fig. 1*b*. Of course, a complete 5-step interaction is required if each new token is separately bought from the bank. However, this situation can be avoided in the proposed payment scheme.

4.6 Storage and computational performance

Both the storage and computational complexities of the proposed micropayment scheme will be analysed and compared with other well known micropayment schemes. Since each scheme has its own message format and implementation consideration, a detailed performance comparison and listing of all the schemes would be tedious. In the following, only some key points are described and all the other results shown in two Tables.

It has already been previously remarked that, in the Millicent payment scheme, no matter whether the request signature secure mode or the encrypted scrip secure mode is employed; the adoption of 'customer secret' complicates the implementation and maintenance on the customer's side. Evidently, the customer's machine is required to store all the necessary Customer secrets received from the corresponding merchants. However, in the proposed micropayment scheme, a customer is not required to share any common secret with the merchants in order to protect the payment. Information transmitted to the merchant cannot be used to deduce any valid fresh coin due to the employment of the one-way hash chain. This greatly improves the storage requirement over Millicent. Detailed comparisons of the storage requirements can be found in Table 1.

The results of computational comparison are listed in Table 2. Since no public key digital-signature computation will be needed, performance of the proposed scheme will be better than the PayWord scheme. The Millicent scheme is the best from the view of computation, however, as described previously, it provides no fairness for the custo-

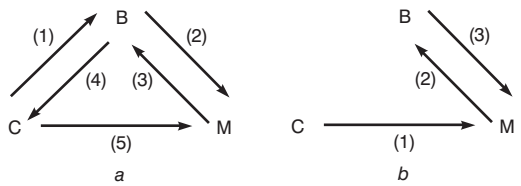

Fig. 1 *Interactive flow of initiating a new payment sequence*

a Millicent scheme
b Proposed scheme

Table 1: Storage requirement for various micropayment schemes

		Proposed new scheme	PayWord scheme	Millicent-1 with encrypted scrip	Millicent-2 with request signature
Stored shared key	C:	K_C	no	K_C	K_C
between C and B	B:	K_C/no[1]	no	no	no
Stored shared key	M:	K_M	no	no[2]	no
between M and B	B:	K_M/no[1]	no	no	no
Stored shared key	C:	no	no	K_{CM}	K_{CM}
between C and M	M:	no	no	no	no
Seed or other data stored in C per chain		$E_{SK}(N, RN)$, N: optional	w_n	merchant scrip bank scrip	merchant scrip bank scrip
Coin or other data stored in M per chain		only fresh w_i, N	fresh and spent w_i, Cert $Sign_C(\cdots)$	fresh and spent merchant scrip	fresh and spent merchant scrip
Table of spent chain or token stored in B		$\{N, ID_C, ID_M\}$ $\{ID_C, O_C\}$	$\{Sign_C(\cdots)$, Cert, $w_n\}$	no[3]	no

Note 1: If the Millicent similar implementation or the ID-based secret-key cryptography [30] are used, K_C does not need to be stored.

Note 2: Although not mentioned in the Millicent scheme, in fact the shared key between a merchant and the bank is required. This shared key will be used when sending a merchant scrip to the bank.

Note 3: However, it is better for the bank to backup already bought merchant scrips in order to prevent any possible active attack when buying any new merchant scrip from the merchant.

Table 2: Computational performance of each coin chain for various micropayment schemes

		Proposed new scheme	PayWord scheme	Millicent-1 with encrypted scrip	Millicent-2 with request signature
Cryptographic	C:	$n\delta$[1] + 3	$n\delta$	0	$n + 1$[2]
hash	M:	$n + 1$	n	$2n$	$4n$
function	B:	$n + 4$	n	2	4
Public key	C:	no	1	no	no
signature	M:	no	2	no	no
(sign or verify)	B:	no	1	no	no
Symmetric key	C:	1	no	$2n + 2$	no
encryption	M:	no	no	$2n$	no
(encrypt or decrypt)	B:	3	no	2	no

Note 1: δ = the average number of hash computations for each coin.

Note 2: Each hash performed in the Millicent scheme takes much more time since much more information will be hashed.

mer. Furthermore, for the Millicent scheme, in the Table, we do not consider the process of delivering money from the customer to the bank based on any 'macropayment', e.g. the SET system [1], when buying a merchant scrip. We also do not consider the process of sending a requested merchant scrip from the merchant to the bank, which of course takes computation. The number δ in Table 2 denotes the average number of cryptographic hash computations required to generate each micropayment coin. The number δ depends on both the number n (number of coins) and the technique employed to implement the cryptographic one-wayness. For a straightforward implementation (e.g. the conventional one-way hash chain) δ is $n/2$. However, when the novel technique of an unbalanced one-way binary tree [31] is employed, δ can be reduced to $\sqrt{n}$.

5 Conclusions

As shown from the title of this paper, in the proposed micropayment scheme, both the profit of the merchant and the customer are protected owing to the inherent property of prepaid schemes and the adoption of a one-way hash chain, respectively. The bank is assumed to be trusted and each purchase of a token from a customer can only be computed by the customer using his secret key K_C. However, in the proposed scheme, we don't consider the problem of disavowal of purchasing tokens by the customer. If this is a consideration, for the sake of the bank's profit, public key-based digital signatures can provide a simple solution. This, however, is not a real disadvantage since it has already been mentioned that many general purpose tokens can be purchased in advance and the customer can sign this batch of purchases using only one digital signature.

In fact, under the assumption of a trusted bank, the above disavowal cannot be accepted, e.g. by signing a contract when opening the micropayment account. The above remark does not break the assumption of the trusted bank but provides an adequate protection of the bank if required.

Since it is obvious from the performance analysis that the proposed scheme is efficient from both the viewpoints of storage and computation, it is possible for it to be employed in other applications such as medium or large-scale restricted-resource access control. In this application, it is assumed that a large number of small pieces of digital equipment, each with limited computing or storage capability are located over a distribution network. Access to equipment will be controlled or charged based on either usage time or other service properties, e.g. memory space occupied. A trusted centre or server assumes the role of the bank in the payment scenario. The proposed scheme can be applied in such an environment to grant a sequence of accesses and usage to each equipment by the user, but the equipment and the user don't have to share a common key. This will simplify the maintenance of the equipment.

An example of the above extended application of the proposed micropayment scheme is that service regulation or control among a large number of pieces of digital equipment within an organisation or an institute is required. Typical service regulations include: access to print servers (e.g. printer or copy machine) controlled by the number of pages printed out; access to file servers controlled by the memory space occupied; access to telecommunication servers (maybe over the internet, e.g. an internet phone) controlled by the time connected and access to application (e.g. various software programs) servers controlled by the number of items selected or the time used.

Other service regulations are possible in some complex environments. By using the proposed scheme, the system administrator can unify all the service regulations into one kind of control token (i.e. the payment token in the proposed scheme). Each user (employee) is given a number of control tokens in advance and any future resource access is controlled by delivering the required tickets, i.e. the coins, to the equipment.

Examining other possible extended applications of the proposed scheme, other than micropayments and resource access control, is an interesting future research topic.

6 Acknowledgments

This research was supported in part by the National Science Council of the Republic of China under contract NSC89-2213-E-008-049 and also by the Institute for Information Industry, R.O.C. under contract 88-EC-2A-17-0209.

7 References

1 O'MAHONY, D., PEIRCE, M., and TEWARI, H.: 'Electronic payment systems' (Artech House, Inc., 1997), Chap. 7
2 PALMER, J.W., and ERIKSEN, L.B.: 'Digital newspapers explore marketing on the internet', *Commun. ACM*, 1999, **42**, (9), pp. 33–40
3 RIVEST, R.L.: 'Electronic lottery tickets as micropayments', *Lect. Notes Comput. Sci.*, 1998, **1318**, pp. 307–314
4 RIVEST, R.L., and SHAMIR, A.: 'PayWord and MicroMint: two simple micropayment schemes', *Lect. Notes Comput. Sci.*, 1997, **1189**, pp. 69–87
5 HORN, G., and PRENEEL, B.: 'Authentication and payment in future mobile systems', *Lect. Notes Comput. Sci.*, 1998, **1485**, pp. 277–293
6 MARTIN, K.M., PRENEEL, B., MITCHELL, C.J., HITZ, H.J., HORN, G., POLIAKOVA, A., and HOWARD, P.: 'Secure billing for mobile information services in UMTS', *Lect. Notes Comput. Sci.*, 1998, **1430**, pp. 535–548
7 DASWANI, N., and BONEH, D.: 'Experimenting with electronic commerce on the PalmPilot', *Lect. Notes Comput. Sci.*, February 1999, **1648**
8 O'MAHONY, D., DOYLE, L., TEWARI, H., and PEIRCE, M.: 'NOMAD—an application to provide UMTS telephony services on fixed terminals in COBUCO'. Proceedings of the 3rd ACTS Mobile Communications Summit, 1998, Rhodes, Greece, Vol. 1, pp. 72–76
9 PEIRCE, M., and O'MAHONY, D.: 'Micropayments for mobile networks'. Technical Report of the Department of Computer Science, Trinity College Dublin, Ireland, 1999
10 GLASSMANN, S., MANASSE, M., ABADI, M., GAUTHIER, P., and SOBALVARRO, P.: 'The Millicent protocol for inexpensive electronic commerce'. Proceedings of the 4th International World Wide Web Conference, 1995, Boston, MA, pp. 603–618
11 MANASSE, M.: 'The Millicent protocols for electronic commerce'. Proceedings of the 1st USENIX workshop on *Electronic commerce*, 1995, New York
12 ANDERSON, R., MANIFAVAS, C., and SUTHERLAND, C.: 'NetCard—a practical electronic cash system', *Lect. Notes Comput. Sci.*, 1997, **1189**, pp. 49–57
13 HAUSER, R., STEINER, M., and WAIDNER, M.: 'Micro-payments based on iKP'. Proceedings of the SECURICOM '96, 14th Worldwide Congress on *Computer and communications security and protection*, 1996, pp. 67–82
14 PEDERSEN, T.: 'Electronic payments of small amounts', *Lect. Notes Comput. Sci.*, 1997, **1189**, pp. 59–68
15 STERN, J., and VAUDENAY, S.: 'SVP: a flexible micropayment scheme', *Lect. Notes Comput. Sci.*, 1997, **1318**, pp. 161–171
16 YEN, S.M.: 'Vulnerability of the SVP micropayment scheme'. LCIS technical report TR-99-2, Department of Computer Science and Information Engineering, National Central University, Taiwan, 1999
17 ZHENG, Y., and MATSUMOTO, T.: 'Breaking real-world implementations of cryptosystems by manipulating their random number generation'. Pre-proceedings of the 1997 Symposium on *Cryptography and information security*, 1997, Fukuoka, Japan
18 RIVEST, R.: 'The MD5 message digest algorithm'. RFC 1321, April 1992
19 FIPS 180-1: 'Secure hash standard', NIST, US Department of Commerce, Washington, D.C., 1995
20 ZHENG, Y., PIEPRZYK, J., and SEBERRY, J.: 'HAVAL—a one-way hashing algorithm with variable length of output', *Lect. Notes Comput. Sci.*, 1993, **718**, pp. 83–104
21 LAMPORT, L.: 'Password authentication with insecure communication', *Commun. ACM*, 1981, **24**, (11), pp. 770–772
22 HALLER, N.M.: 'The S/KEY one-time password system'. Proceedings of the ISOC Symposium on *Network and distributed system security*, 1994, San Diego, CA
23 RIVEST, R.L., SHAMIR, A., and ADLEMAN, L.: 'A method for obtaining digital signatures and public-key cryptosystem', *Commun. ACM*, 1978, **21**, (2), pp. 120–126
24 MILLER, S.P., NEUMAN, B.C., SCHILLER, J.I., and SALTZER, J.H.: 'Kerberos authentication and authorization system'. Section E.2.1 of Project Athena Technical Plan, MIT, 1987
25 STEINER, J.C., NEUMAN, B.C., and SCHILLER, J.I.: 'Kerberos: an authentication service for open network systems'. Proceedings of the Winter 1988 USENIX Conference, 1988, pp. 191–201
26 NBS FIPS PUB 46: 'Data encryption standard'. National Bureau of Standards, U.S. Department of Commerce, 1977
27 MENEZES, A.J., VAN OORSCHOT, P.C., and VANSTONE, S.A.: 'Handbook of applied cryptography' (CRC Press, 1997)
28 KRAWCZYK, H., BELLARE, M., and CANETTI, R.: 'HMAC: keyed-hashing for message authentication'. Network working group, RFC 2104, February 1997
29 KRAWCZYK, H., BELLARE, M., and CANETTI, R.: 'HMAC-MD5: keyed-MD5 for message authentication'. Network Working Group, internet draft, draft-ietf-ipsec-hmac-md5-txt.00, March 1996
30 JOYE, M., and YEN, S.M.: 'ID-based secret-key cryptography', *Oper. Syst. Rev.*, 1998, **32**, (4), pp. 33–39
31 YEN, S.M., HO, L.T., and HUANG, C.Y.: 'Internet micropayment based on unbalanced one-way binary tree'. Proceedings of the International Workshop on *Cryptographic techniques and E-commerce*, CrypTEC '99, 1999, Hong Kong, pp. 155–162