PAYSTAR: A DENOMINATION FLEXIBLE MICROPAYMENT SCHEME

Sung-Ming Yen, Hsi-Chung Lin, Yen-Chang Chen
Dept of Computer Science and Information Engineering
National Central University, Chung-Li, Taiwan 320, R.O.C.

Jia-Jun Hung, Jui-Ming Wu
Networks and Multimedia Institute, Institute for Information Industry, Taiwan, R.O.C.

Keywords: Cryptography, Electronic payment, Micropayment, One-time password, One-way hash chain.

Abstract: In this paper, a micropayment scheme with flexible coin denomination mintage is proposed. This varying denomination approach might be more reasonable for most micropayment applications. Furthermore, the proposed scheme improves extensively the computational performance of both the customer as well as the merchant. Storage cost of the customer is exactly the same as in the original PayWord micropayment scheme and is thus applicable to implementations on small portable devices, e.g., smart card.

1 INTRODUCTION

Micropayment schemes have received growing attention recently, primarily because that these schemes provide solutions for numerous Internet based applications. Micropayment schemes allow a customer to transfer to a merchant a sequence of small amount payments in exchange for services or electronic products from the merchant. Usually, it is inappropriate to pay the total amount of money either in advance or afterwards in some applications, and this makes micropayment schemes be especially useful. The result is that efficient and cryptographically strong micropayment schemes are highly demanded to provide acceptable performance and also to solve dispute when occurred.

For most application scenarios of micropayment, both the computational cost and the storage cost should be reduced to maintain acceptable performance.

- Computational cost reduction: Since the payment is small and might be very frequent, this cost should be reduced. Evidently, the employment of public key cryptography should be minimized if possible.
- Storage cost reduction: For most micropayment applications, huge amount of payment records might need to take care. Therefore, minimization on storage cost will be helpful.

Furthermore, administrative cost that includes the interactions with the trusted third party (usually the bank) and the frequency of doing withdraw and deposit should also be minimized.

Due to the potential applications, many research results about micropayment schemes design and their performance enhancement have been published in the past few years (Glassmann et al., 1995; Rivest and Shamir, 1997; Jutla and Yung, 1996; Pedersen, 1997; Stern and Vaudenay, 1998; Yen et al., 1999; Micali and Rivest, 2002). One large category of micropayment schemes (e.g., the PayWord scheme (Rivest and Shamir, 1997) by Rivest and Shamir) use one-way hash chain as the primary cryptographic tool to develop the schemes.

A conventional one-way hash chain is recursively defined from a secret value by performing a sequence of cryptographic hash computations, e.g., (SHA-1, 1995). This simple design of linear one-way hash chain makes the PayWord-like micropayment schemes have their intrinsic disadvantage that more longer a one-way hash chain will lead to a less efficient performance of generating a micropayment coin. Some approaches (Jutla and Yung, 1996; Yen et al., 1999) had been proposed trying to overcome the above mentioned disadvantage, however each of them has its merit and also limitation or even disadvantage.

In the conventional PayWord-like schemes and the modified versions (e.g., (Jutla and Yung, 1996; Yen et al., 1999)), each coin is defined to be of the same smallest denomination, but this brings some performance overhead for the merchant and also for the customer if each payment will be a random times of the smallest denomination.

Our Contribution. The main contribution of this paper is that we propose a new micropayment scheme, called the PayStar scheme, that employs the proposed merged one-way hash chain and provides a flexible design of varying denomination. The result is that average computational cost of generating each smallest denomination for the customer is reduced extensively when compared with the conventional PayWord scheme and the enhanced tree-based scheme (Yen et al., 1999).

This performance enhancement is important for micropayment schemes that will be implemented based on small portable devices with limited resource, e.g., smart card. Furthermore, the computational cost of coin validation performed by the merchant is also reduced extensively on average for most of the cases, and the cost is basically independent to the amount to pay.

2 RELATED WORK

Some notations and symbols used in this paper about a one-way hash chain are given in the following.

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

$$h^r(v_n) = \underbrace{h(h(\cdots(h}_{r\ times}(v_n))\cdots)).$$

When the function $h()$ in the iteration is instantiated with a one-way hash function, such as SHA (SHA-1, 1995), the result is a *one-way hash chain* as shown below

$$\begin{aligned} v_0 = h^n(v_n) \quad &\leftarrow \quad v_1 = h^{n-1}(v_n) \leftarrow \cdots \\ \cdots \quad &\leftarrow v_{n-1} = h^1(v_n) \leftarrow v_n. \end{aligned}$$

Note that within the chain, each element v_i is computed as $h^{n-i}(v_n)$.

2.1 Brief Review of the PayWord Scheme

PayWord is a credit-based micropayment scheme proposed by Rivest and Shamir (Rivest and Shamir, 1997). Prior to the first transaction taking place between a customer C and a merchant M, the following preparatory steps need to be carried out.

(1) The customer generates a payword chain as follows:

$$v_0 \leftarrow v_1 \leftarrow v_2 \leftarrow \cdots \leftarrow v_{n-1} \leftarrow v_n$$

where $v_i = h(v_{i+1})$ for $i = n-1, n-2, \cdots, 1, 0$, and $h()$ is a cryptographic one-way hash function. The value v_n is a secret value selected at random by the customer.

(2) The customer signs, e.g., using RSA (Rivest et al., 1978), on the *root* v_0, together with the merchant's identity and other information:

$$Sign_C(\text{Merchant-ID}||v_0||\text{Cert})$$

where "Cert" used as a proof of credentials is a digital certificate issued to the customer by the bank B. The certificate authorizes the customer to mint his own payword chains. Note that the signature on $(\text{Merchant-ID}||v_0||\text{Cert})$ acts as a commitment.

(3) The customer then sends

$$Sign_C(\text{Merchant-ID}||v_0||\text{Cert}),$$
$$\text{Merchant-ID}, v_0, \text{Cert}$$

to the merchant.

After completing successfully the above steps between the customer and the particular merchant, the number v_i ($i = 1, 2, \ldots, n$) can now be used as the ith coin to be paid. Each coin v_i in the sequence is predefined to be of one smallest denomination. When receiving a new coin v_i from the customer, the merchant verifies whether $v_{i-1} \overset{?}{=} h(v_i)$. The merchant accepts v_i as a valid payment only if the verification is successful. Note that the merchant can store a valid v_i in place of v_{i-1}. The above process ensures that a sequence of small payments can be paid to a specific merchant and the customer has to perform only one computationally expensive public key based digital signature which is for the purpose of commitment.

Suppose v_x is the previously spent coin and the customer now needs to spend t times of the smallest denomination, then the customer computes and sends $v_i = v_{x+t}$ to the merchant. When receiving v_i from the customer, the merchant verifies whether $v_x \overset{?}{=} h^t(v_i)$.

2.2 Micropayment based on Unbalanced One-way Binary Tree

At the customer's side, suppose that he only stores the last coin v_n, then he has to compute each required new coin by a sequence of hash function computations. It is evident that on average each new coin generation will cost $((n-1)+(n-2)+\cdots+1)/n=(n-1)/2$ (or roughly $n/2$ for large n) hash computations.

In (Yen et al., 1999), the technique of one-way hash chain was extended into a multi-dimensional tree-based approach which is called the *unbalanced one-way binary tree* (UOBT). An unbalanced one-way binary tree is essentially a bundle of many one-way hash chains, with their secret values (called the seed nodes in PayWord scheme) being tied together via another separate one-way hash chain.

In the unbalanced one-way binary tree with n nodes (coins), given the seed value v_n, each v_i can be computed using on average $\mathrm{O}(n^{1/2})$ hash computations (Yen et al., 1999). Therefore, the usage of unbalanced one-way binary tree brings extensive performance improvement over the conventional linear one-way hash chain.

3 THE PROPOSED DENOMINATION FLEXIBLE MICROPAYMENT SCHEME

In most applications, each payment request for different purchase might differ from each other. Suppose that each payment is a random amount between one time to 2^{k+1} times of the smallest denomination (say one cent), and the expected average payment is roughly 2^k cents. In the conventional one-way hash chain and the unbalanced one-way binary tree designs, each coin v_i is defined to represent the smallest denomination. In such fixed denomination approaches, the merchant needs to perform 2^k hash computations to validate the correctness of a payment with 2^k cents.

In this section, a new micropayment scheme with variable denomination is proposed in which the computational cost of payment validation performed by the merchant is reduced for the average case. Basically, the computational cost is independent to the amount being paid.

3.1 Varying Denomination Mintage Scheme

Each payment will be assigned a denomination between $[1,2^{k+1}]$. Suppose the denomination of the ith payment is D cents and $D-1$ be binary encoded as $(d_k,\cdots,d_1,d_0)_2$. For this payment of D cents, a *merged one-way hash chain* will be defined in the following

$$\begin{aligned} c_{i,k} &= h(c_{i,k-1}\|h(x_{i,k})) \\ &\leftarrow\ c_{i,k-1}=h(c_{i,k-2}\|h(x_{i,k-1})) \\ &\leftarrow\ \cdots\leftarrow c_{i,0}=h(c\|h(x_{i,0}))\leftarrow c \end{aligned}$$

where $x_{i,j}$ ($j=k,k-1,\cdots,0$) are secret values and c is a public information. To pay D cents, the sequence of values below (called the *trapdoor sequence*) will be released to the merchant

$$(h^{1-d_k}(x_{i,k}),\cdots,h^{1-d_1}(x_{i,1}),h^{1-d_0}(x_{i,0}))$$

where $h^0(x_{i,j})=x_{i,j}$.

The basic idea is to encode the denomination D (a positive integer) into a merged one-way hash chain. In the trapdoor sequence, $h^{1-d_k}(x_{i,k})$ and $h^{1-d_0}(x_{i,0})$ will be used to encode the most significant bit d_k and the least significant bit d_0 of $D-1$, respectively. If $d_j=1$ then the secret $x_{i,j}$ will be released to the merchant, otherwise $h(x_{i,j})$ will be released. So, given c, $c_{i,k}$, and the corresponding trapdoor sequence, a unique denomination D can be specified by the merged one-way hash chain.

For example, let $k=2$, the denomination of six cents can be minted by releasing $(x_{i,2},h(x_{i,1}),x_{i,0})$.

3.2 PayStar Scheme with Varying Denomination

Let the payment of each purchase be a random amount between one cent to 2^{k+1} cents with the expected average payment of $(2^k+\frac{1}{2})$ cents, and we wish to prepare a payment token of totally n cents on average. The proposed PayStar scheme with $\ell=n/(2^k+\frac{1}{2})$ merged one-way hash chains is shown below.

(1) The customer prepares $\ell=n/(2^k+\frac{1}{2})$ independent merged one-way hash chains all with the same public information c and each being capable of encoding denomination between $[1,2^{k+1}]$. All the last values $c_{i,k}$ ($i=1,2,\cdots,\ell$) of all the corresponding merged one-way hash chains are computed. Fig. 1 illustrates the construction.

All the secret values $x_{i,j}$ of the merged one-way hash chains can be defined as $x_{i,j}=h(x,i,j)$ or alternatively by using conventional block cipher $x_{i,j}=E_x(i,j)$ where x is a secret value selected at random by the customer. So, the customer only needs to store a secret value, i.e., x.

(2) The customer signs on the merchant's identity, the public information c (which can be any constant), the roots $(c_{1,k},\cdots,c_{\ell,k})$ of all the merged one-way

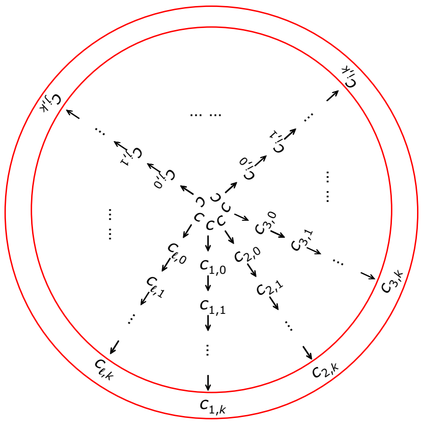

Figure 1: PayStar scheme with varying denomination.

hash chains, an integer parameter k, and the certificate issued by the bank

$$Sign_C(\text{Merchant-ID}\|c\|(c_{1,k},\cdots,c_{\ell,k})\|k\|\text{Cert})$$

where k is used to indicate that the denomination of payment is between $[1,2^{k+1}]$ cents. The above signature (used as the payment commitment), Merchant-ID, c, $(c_{1,k},\cdots,c_{\ell,k})$, k, and Cert will be sent to the merchant before the first purchase.

The merchant validates the correctness of the payment commitment and stores the commitment as well as the following values for further transaction verification

$$\{c,(c_{1,k},\cdots,c_{\ell,k}),k\}.$$

(3a) To pay D cents to the merchant as the ith payment, the customer prepares the corresponding trapdoor sequence $\mathcal{S}$

$$\begin{aligned} & (s_k,\cdots,s_1,s_0) \\ = \quad & (h^{1-d_k}(x_{i,k}),\cdots,h^{1-d_1}(x_{i,1}),h^{1-d_0}(x_{i,0})) \end{aligned}$$

of the denomination D where $(d_k,\cdots,d_1,d_0)_2$ is the binary representation of $D-1$.

(3b) After receiving the trapdoor sequence $\mathcal{S}$, the merchant verifies the validity of $\mathcal{S}$ by computing

$$c_{i,k} \overset{?}{=} h(\cdots h(h(c\|h^{d_0}(s_0))\|h^{d_1}(s_1))\cdots\|h^{d_k}(s_k))$$

based on the definition of the merged one-way hash chain and the stored root $c_{i,k}$. Correct validation of $\mathcal{S}$ confirms the committed denomination D. In fact, after verifying and storing the trapdoor sequence $\mathcal{S}$, the root $c_{i,k}$ is no longer necessary.

Therefore, a PayStar token is essentially a bundle of many merged one-way hash chains, with their public information c being tied together by selecting a common value.

To avoid the merchant searching among all the previously received payment from a specific PayStar token to detect any possible double spending, the root $c_{i,k}$ of each merged one-way hash chain can be redefined as $c_{i,k}=h(i\|c_{i,k-1}\|h(x_{i,k}))$.

3.3 Security Analysis of the PayStar Scheme

In the PayStar scheme, it is impossible for a dishonest customer to double spend, as long as the merchant ensures that each merged one-way hash chain of a PayStar token can only be used once. In practice, since the customer is forced to spend his merged one-way hash chains in sequence and the merchant promptly deletes the root $c_{i,k}$ after the i-th payment is validated, the PayStar scheme is evidently double spending free.

Similarly, it is impossible for a dishonest merchant to double encash at the redemption, as long as the bank ensures that each payment corresponds to a different merged one-way hash chain. It is also impossible for a dishonest merchant to over encash at the redemption since enlarging the denomination of a used merged one-way hash chain requires the ability of extracting pre-images of some (at least one) items in the trapdoor sequence. Clearly, the merchant's ability of extracting pre-images contradict the one-way property of the underlying hash function.

4 PERFORMANCE ANALYSIS

4.1 Computational and Storage Cost Analysis

We assume that the probability of appearance of denomination D $(1 \le D \le 2^{k+1})$ for each payment is the same. So, the expected amount of money (in cent) that will be embedded in a PayStar token with ℓ merged one way hash chains is

$$E=(1+2^{k+1})\ell/2$$

(roughly $\ell \times 2^k$ for large k). The customer takes on average $1.5\times(k+1)$ hash computations to obtain each necessary trapdoor sequence. Therefore, to spend the whole PayStar token, the customer needs totally $1.5\times\ell\times(k+1)$ hash computations, and the spending of each "cent" costs on average

$$(1.5\times\ell\times(k+1))/E \approx (1.5\times(k+1))/2^k$$

hash computations. Here, we ignore the computational cost to prepare the PayStar token since this can be off-line performed. Totally, $3 \times \ell \times (k+1)$ hash computations are necessary to prepare the PayStar token.

Like in the original PayWord scheme and the UOBT-based micropayment scheme, the customer needs only to store the secret value x.

For the merchant, after receiving the trapdoor sequence representing a committed denomination D, the related merged one-way hash chain will be employed to validate the denomination with on average $1.5 \times (k+1)$ hash computations. Therefore, for the whole PayStar token, the merchant needs totally $1.5 \times \ell \times (k+1)$ hash computations, and the verification of each "cent" costs on average

$$(1.5 \times \ell \times (k+1))/E \approx (1.5 \times (k+1))/2^k$$

hash computations which is exactly the same cost of spending each cent by the customer.

Knowledge of the trapdoor sequence (with $k+1$ hash values) is sufficient for the committed denomination D, so now the merchant can remove the related root $c_{i,k}$. If the whole PayStar token is spent, the merchant needs to store ℓ trapdoor sequences, i.e., $\ell \times (k+1)$ hash values, for all of the committed denominations.

Suppose that $\ell = 100$, $k = 5$, and the bit length of each hash value is 160, then the expected total amount of money embedded in the PayStar token is $E = 3250$ cents and it costs the merchant about 12 kilobytes to store all of the 100 trapdoor sequences. In this example, the amount of money embedded in the PayStar token is quite sufficient for micropayment applications and the storage cost remains acceptable and practical.

4.2 Alternative Merged One-way Hash Chain

An alternative (generalized) merged one-way hash chain is possible which can be used to improve the performance of both the customer and the merchant. Furthermore, this generalized merged one-way hash chain will reduce the storage cost of the merchant.

To assign a denomination D, let $D-1$ be represented with radix r as $(d_k, \cdots, d_1, d_0)_r$ where $0 \le d_i \le r-1$. With this design, the denomination can be in the range $1 \le D \le r^{k+1}$. Now, a merged one-way hash chain will be defined as

$$\begin{aligned} c_{i,k} &= h(c_{i,k-1} \| h^{r-1}(x_{i,k})) \\ &\leftarrow c_{i,k-1} = h(c_{i,k-2} \| h^{r-1}(x_{i,k-1})) \\ &\leftarrow \cdots \leftarrow c_{i,0} = h(c \| h^{r-1}(x_{i,0})) \leftarrow c \end{aligned}$$

and the trapdoor sequence to be released to the merchant becomes

$$(h^{r-1-d_k}(x_{i,k}), \cdots, h^{r-1-d_1}(x_{i,1}), h^{r-1-d_0}(x_{i,0})).$$

In the trapdoor sequence, $h^{r-1-d_k}(x_{i,k})$ and $h^{r-1-d_0}(x_{i,0})$ will be used to encode the most significant digit d_k and the least significant digit d_0 of $D-1$, respectively. Following the same reasoning in Sect. 4.1, it is clear that the average cost for the customer to spend and for the merchant to validate one cent are both

$$(\frac{r+1}{2} \times \ell \times (k+1))/E = (r+1)(k+1)/(1+r^{k+1})$$

hash computations, where $E = (1+r^{k+1})\ell/2$. As for the storage cost, the customer needs only to store one secret value and the merchant need to store ℓ trapdoor sequences, i.e., $\ell \times (k+1)$ hash values.

Theorem 1 *Let each $x_{i,j}$ be computed by $x_{i,j} = h(x,i,j)$ with one hash computation. Then, the computationally optimal design of merged one-way hash chain for any predetermined maximum possible denomination is to select $r = 4$.*

Proof: Let $\mathcal{D}$ be the predetermined maximum possible denomination to be encoded, r the radix, $k+1$ the length of the merged one-way hash chain, then $r^{k+1} = \mathcal{D}$. The average number of hash computations that the customer needs to prepare the trapdoor sequence is $(k+1) \times (r+1)/2$, and this is also the same amount of computation that the merchant needs to verify the trapdoor sequence. For a specific radix r, $k+1 = \log_r \mathcal{D}$, so the computational cost becomes $(k+1) \times \frac{r+1}{2} = \frac{r+1}{\ln r} \times \frac{\ln \mathcal{D}}{2}$.

For any predetermined maximum possible denomination $\mathcal{D}$, to minimize the computational cost is to minimize $\frac{r+1}{\ln r}$ and $m(r) = \frac{d(\frac{r+1}{\ln r})}{dr} = \frac{1}{\ln r} - \frac{r+1}{(\ln r)^2 r}$. We have $m(3.59114) = 0$ when $r \ge 2$, and the minimum value of $\frac{r+1}{\ln r}$ (for $r \in \mathbb{N}$ and $r \ge 2$) becomes $\frac{4+1}{\ln 4} = 3.6067375$ (see Fig. 2).

The result is that $r = 4$ for any $\mathcal{D}$ is the best selection in the proposed merged one-way hash chain. $\square$

The result obtained in Theorem 1 shows that the implementation of the optimized merged one-way hash chain can be very easy since a representation with radix of four can be obtained easily from a representation with radix of two. With the same example of

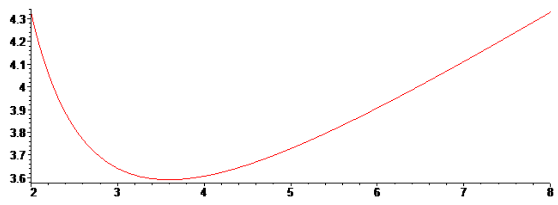

Figure 2: The best selection of radix r.

$\ell = 100$ and $1 \leq D \leq 64$ ($E = 3250$) mentioned previously, the length of each merged one-way hash chain will be reduced to 3 ($k = 2$) when the radix $r = 4$ is selected. Therefore, the storage cost of the merchant will be substantially reduced to about 6 kilobytes (i.e., 50% reduction) and the computational cost will be reduced to 83% for both of the customer and the merchant.

In some cases where the denomination D is designed to be in the range $[1 \leq D \leq r^{k+1} - \delta]$ (where δ is a large positive integer and $\delta < r^k$), to assign a denomination D into the merged one-way hash chain, radix r representation of $(D-1+\delta)$ instead of $(D-1)$ can be used to compute the trapdoor sequence. In this modified version, the most significant digit d_k will now usually have a larger value and $h^{r-1-d_k}(x_{i,k})$ will take fewer hash computations. On the other hand, the merchant will take more hash computations (when compared with the customer) to verify the received trapdoor sequence since he needs more hash computations to obtain $c_{i,k}$. This design is of course reasonable because the customer might have only a resource limited portable device, e.g., smart card, but the merchant can be equipped with a powerful computer. This modified design becomes especially important due to the result obtained in Theorem 1 that the optimal radix r is four but not two, so δ might be a large value.

4.3 Comparisons with other Schemes

In this section, comparisons of performance among the PayWord scheme (Rivest and Shamir, 1997), the UOBT scheme (Yen et al., 1999), and the proposed PayStar will be given. It is assumed that the smallest denomination is one cent and the cost of each payment is uniformly distributed over the range of $[1,t]$. On average, each payment costs $A = \frac{1+t}{2}$ cents.

Lemma 1 *Let n be the length of a conventional one-way hash chain and each payment costs A cents on average. Then, in the PayWord scheme, the expected computational costs for the customer to spend and for the merchant to validate each cent are* $(\frac{n}{A} - 1)/2$ *and one hash computations, respectively.*

Proof: The expected total number of hash computations required to be done by the customer during the $\frac{n}{A}$ payment transactions (an average case) is $T = \sum_{i=1}^{\frac{n}{A}} (i-1) \cdot A = \frac{n \cdot (\frac{n}{A} - 1)}{2}$.
Therefore, the expected computational cost for the customer to spend each cent is $T/n = (\frac{n}{A} - 1)/2$ hash computations. Apparently, the computational cost for the merchant to validate each cent is one hash computation. $\square$

Table 1: Comparison of computational cost per cent among PayWord, UOBT, and PayStar.

	PayWord	UOBT	PayStar	
			$r=2$ $k=5$	$r=4$ $k=2$
Customer	49.5	1.72	0.28	0.23
Merchant	1	1	0.28	0.23

Lemma 2 *Suppose that there are $p \cdot q$ nodes (coins) in an Unbalanced One-way Binary Tree (i.e., q one-way hash chains each of length p, with their seeds tied together via another one-way hash chain of length q) and each payment costs A cents on average. Then, the expected computational costs for the customer to spend and for the merchant to validate each cent are* $\frac{p+q-2}{2A}$ *and one hash computations, respectively.*

Proof: The expected total number of hash computations required to be done by the customer during the $\frac{p \cdot q}{A}$ payment transactions (an average case) is $T = \left(\frac{p-1}{2} + \frac{q-1}{2}\right) \times \frac{p \cdot q}{A} = \frac{p \cdot q \cdot (p+q-2)}{2A}$. In this analysis, for the sake of simplicity, we do not explicitly consider the case in which a payment may need to deliver two hash values from two adjacent one-way hash chains. However, the above brief estimation reflects the usual cases in a UOBT setting.
Therefore, the expected computational cost for the customer to spend each cent is $T/(p \cdot q) = \frac{p+q-2}{2A}$ hash computations. Apparently, the computational cost for the merchant to validate each cent is one hash computation. $\square$

Following the example used in Sect. 4.1, a concrete comparison can be made by letting the cost of each payment be a random value in the range $[1,64]$ with the average cost $A = 32.5$, and the expected total amount embedded in a payment token be about 3250 cents. So, each payment token can be used for approximately 100 payment transactions before being exhausted. Parameters corresponding to the above scenario are $n = 3250$ for the PayWord scheme, $p = q = 57$ ($p \times q = 3249$) for the UOBT scheme, and $\ell = 100$ for the proposed PayStar. Performance comparisons of the expected computational cost (number of hash computations per cent) among these four payment schemes with the above setting are summarized in the Table 1.

Table 1 shows that for the computational performance of the customer, the proposed PayStar scheme is substantially superior to all other schemes; and for the computational performance of the merchant, the proposed PayStar scheme is also superior to the PayWord scheme and the UOBT scheme.

Table 2: Overall comparisons among PayWord, UOBT, and PayStar.

	PayWord	UOBT	PayStar
Building block	one-way hash chain	unbalanced one-way binary tree	merged one-way hash chain
Denomi-nation	fixed	fixed	flexible
Customer computation cost (per cent)	inefficient	efficient	very efficient
Customer storage cost	very small	very small	very small
Merchant computation cost (per cent)	efficient	efficient	very efficient
Merchant validation cost (each transaction)	depend on total amount	depend on total amount	independent to total amount
Merchant storage cost	very small	very small	reasonable

4.4 Further Extension of the PayStar Scheme

One further extension of the PayStar scheme is possible by collecting a few groups of merged one-way hash chains together and each group with their own predetermined maximum possible denomination. This design is especially useful when the cost of each payment might be diverse with large values.

On the other hand, if the PayStar token consists of a few groups of merged one-way hash chains and each group with their own predetermined merchant identity, then a micropayment scheme for multiple merchants can be readily constructed. One possible approach of encoding the merchant's identity into the merged one-way hash chain is shown below

$$c_{i,k} = h(\text{Merchant-ID} \| c_{i,k-1} \| h(x_{i,k})).$$

5 CONCLUSIONS

The proposed PayStar micropayment scheme with varying denomination by employing the merged one-way hash chain performs substantially superior to both the original PayWord scheme and also the UOBT-based scheme. Computational performances of both the customer as well as the merchant have been enhanced. In the PayStar scheme, storage cost of the customer is very small which enables the implementation of the scheme on small portable devices. Although the merchant suffers from the penalty of more storage cost, but the overhead is in fact acceptable according to our analysis on reasonable examples. Overall comparisons among the PayWord scheme, the UOBT-based scheme, and the proposed PayStar scheme are summarized in the Table 2.

ACKNOWLEDGEMENTS

This work was supported in part by Institute for Information Industry, R.O.C.

REFERENCES

Glassmann, S., Manasse, M., Abadi, M., Gauthier, P., and Sobalvarro, P. (1995). The millicent protocol for inexpensive electronic commerce. In *Proc. of 4th International World Wide Web Conference*, pages 603–618.

Jutla, C. and Yung, M. (1996). Paytree: Amortized-signature for flexible micropayments. In *Proc. of 2nd USENIX Workshop on Electronic Commerce*, pages 213–221.

Micali, S. and Rivest, R. (2002). Micropayments revisited. In *Proc. of Cryptographer's Track at the RSA Conference, CT-RSA '02*, volume Lecture Notes in Computer Science 2271, pages 149–163. Springer-Verlag.

Pedersen, T. (1997). Electronic payments of small amounts. In *Proc. of Security Protocols Workshop*, volume Lecture Notes in Computer Science 1189, pages 59–68,. Springer-Verlag.

Rivest, R. and Shamir, A. (1997). Payword and micromint: Two simple micropayment schemes. In *Proc. of Security Protocols Workshop*, volume Lecture Notes in Computer Science 1189, pages 69–87. Springer-Verlag. Also in *CryptoBytes*, Pressed by RSA Laboratories, Vol. 2, No. 1, pp. 7–11, 1996.

Rivest, R., Shamir, A., and Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystem. *Commun. of ACM*, 21(2):120–126.

SHA-1 (1995). FIPS 180-1, secure hash standard. Technical report, NIST, US Department of Commerce, Washington D.C.

Stern, J. and Vaudenay, S. (1998). Svp: A flexible micropayment scheme. In *Proc. of Financial Cryptography Conference, FC '97*, volume Lecture Notes in Computer Science 1318, pages 161–172. Springer-Verlag.

Yen, S., Ho, L., and Huang, C. (1999). Internet micropayment based on unbalanced one-way binary tree. In *Proc. of International Workshop on Cryptographic Techniques and E-Commerce, CrypTEC '99*, pages 155–162.