The General Pay-Word: A Micro-payment Scheme Based on n-dimension One-way Hash Chain

Iuon-Chang Lin[‡] Min-Shiang Hwang[†] Chin-Chen Chang[‡]

Department of Management Information System[†]
National Chung Hsing University
250 Kuo Kuang Road, 402 Taichung, Taiwan, R.O.C.
Email: mshwang@nchu.edu.tw
Fax: 886-4-22857173

Department of Computer Science and Information Engineering[‡]
National Chung Cheng University
160, San-Hsing, Min-Hsiung, Chaiyi 621, Taiwan, R. O. C.
Email: {iclin;ccc}@nchu.edu.tw
Fax: 886-5-2720859

October 31, 2012

[†]Responsible for correspondence: Prof. Min-Shiang Hwang

The General Pay-Word: A Micro-payment Scheme Based on n-dimension One-way Hash Chain

Abstract

In this paper, we extend the Pay-Word micro-payment scheme using the 1-dimension one-way hash chain to generate an n-dimension one-way hash chain. According to the system requirements, a user can flexibly choose the number of dimensions to gain the best time/space trade off. The proposed scheme is based on an n-dimension one-way hash chain, which can improve the efficiency of deriving the pay-words but also increase the temporary storage space. In addition, this scheme is fit for real-time payment.

Keywords: Micro-payment, one-way hash function

1 Introduction

Electronic payments can make electronic commerce more prevalent if it can be achieved efficiently and securely. However, many businesses over the Internet usually charge for this service. The macro-payment system currently in use is not suitable for micro-payments because the cost for macro-payment processes may probably be higher than the value of the businesses themselves. Therefore, much attention has been drawn to the research of micro-payments [4]. Micro-payment is a special type of electronic payment where it is possible to make a payment of a small amount through a computer network. Such a payment system is used for purchases of information goods over the open network. The payment system is also used in other electronic commerce, such as electronic auction systems [5]. These businesses have three main characteristics: (1) customers get the information goods or services in real time, (2) the transaction amounts are small, and (3) the transactions occur repeatedly.

Due to these characteristics, the fundamental goals of a micro-payment system design are high efficiency and low cost. The one-way hash function (e.g. SHA [9] or MD5 [11]) is a simple and efficient technique that is suitable for a micro-payment system. So far, there have been many micro-payment schemes based on the one-way hash function chain [3, 10, 12, 13]. Similar schemes [1, 8] have also been used for micro-payments. They generalize this chain for using hash trees.

Pay-Word [12] is a famous micro-payment scheme proposed by Rivest and Shamir. In the Pay-Word scheme, the customer computes a pay-word chain to pay for merchandise. The pay-word chain is derived from a single dimension one-way hash function. Recently, Yen et al. proposed an Internet micro-payment system [13] based on an unbalanced one-way binary tree to improve the performance of Pay-Word. In their scheme, the customer uses an unbalanced one-way binary tree to generate the pay-word chain. The unbalanced one-way binary tree is a 2-dimension one-way hash function chain that employs two different one-way hash functions. If the chain has a length m, the scheme reduces the computational complexity from $O(m)$ down to $O(m^{\frac{1}{2}})$. However, the storage of temporary parameters increases from $O(1)$ to $O(m^{\frac{1}{2}})$.

In this paper, we shall propose a general micro-payment scheme based on an n-dimension one-way hash function chain. The computational complexity of our micro-payment scheme is $O(m^{\frac{1}{n}})$, and the complexity of storing temporary parameters is $O(m^{\frac{n-1}{n}})$. The user can determine the trade off and choose the right number of dimensions for her/himself. The organization of this paper is as follows: In the next section, we will briefly review the two micro-payment schemes: Pay-Word and micro-payments based on the unbalanced one-way binary tree. In Section 3, we well introduce the development of a hash chain traversal protocol and propose a novel hash chain traversal protocol, called an n-dimension one-way hash chain, that will be used to design the micro-payment scheme. Furthermore, the n-dimensional one-way hash chain traversal protocol has a computational complexity of $O(m^{\frac{1}{n}})$ and has a storage complexity of $O(1)$. The application of the Pay-Word micro-payment scheme using the n-dimension one-way hash chain is described in Section 4. The performance of our proposed general Pay-Word micro-payment scheme is analyzed and discussed in Section 5. Finally, we will summarize the benefits of our scheme in the last section of this paper.

2 A Review of Pay-Word and Internet Micro-payment Scheme Based on an Unbalanced One-way Binary Tree

The Pay-Word scheme is a post-paid, and off-line verification micro-payment system [12]. The scheme involves three entities: the customer, the merchant, and the broker (e.g. bank). The customer opens an account with the broker. The broker issues to the customer a Pay-Word certificate which, containing parameters such as the broker's identity, the customer's identity, the customer's public key, the expiration date, and other information, is signed by the broker's secret key.

For making a payment to the merchant, the customer picks a random number W_m at will as the seed to generate the chain of pay-words $W_0, W_1, \cdots, W_m$ using the following equation:

$$W_i = h(W_{i+1}), \quad \text{for} \quad i = m-1, m-2, \cdots, 0.$$

The notation $h(\cdot)$ denotes a one-way hash function. Thus, each pay-word is a hash value. Except for W_0, which is the root of the chain, each pay-word can be paid for and is worth one cent. For the first request, the customer sends a *"commitment"* and its signature to the merchant. The commitment contains the root of the chain W_0, the customer's certificate, and the merchant's ID.

During the payment process, the customer computes W_i from W_m ($W_i =$

$h^{m-i}(W_m))$ and sends the pair (W_i, i) as the i-th payment to the merchant. The merchant can verify the validity of the payment by examining whether it hashes to previous element or not, i.e. $W_{i-1} \stackrel{?}{=} h(W_i)$. If the payment is correct, the merchant updates the record with the last payment. If the customer does not yet use up all the pay-words, he/she can pay with the remaining pay-words in order in the next transactions.

In the merchant-customer settlement phase, the merchant delivers the record of the last payment (W_l, l) and the commitment to the broker. The broker confirms (W_l, l) by checking if the equation $h^l(W_l) = W_0$ holds. Then broker withdraws l cents from the customer's account and deposits l cents to the merchant's account.

In 1999, Yen et al. proposed a novel micro-payment scheme [13] to improve the complexity of the hash function operation of Pay-Word. Their scheme is based on an unbalanced one-way binary tree that is a 2-dimensions one-way hash function chain.

In Yen's scheme [13], the user also randomly chooses the seed W_m to generate the pay-word chain. However, unlike the Pay-Word scheme, the chain is generated from two different hash functions. The generation algorithm is shown as follows, and the construction of the unbalanced one-way binary tree is demonstrated in Figure 1.

Assume that m denotes the total number of the pay-words in the chain, and $m = p \times q$, where p is the number of the nodes along axis X and q is the number of the nodes along axis Y in Figure 1.

In this technique, two different hash functions are required. The nodes along axis X are generated using hash function h_1, and the nodes along axis Y are generated using hash function h_2. Each node represents a pay-word as a hash value. Moreover, the nodes W'_1, W'_{p+1}, $\cdots$, W'_{pq-p+1} represent the roots of the chain. Therefore, the computing rules of each node can be concluded as

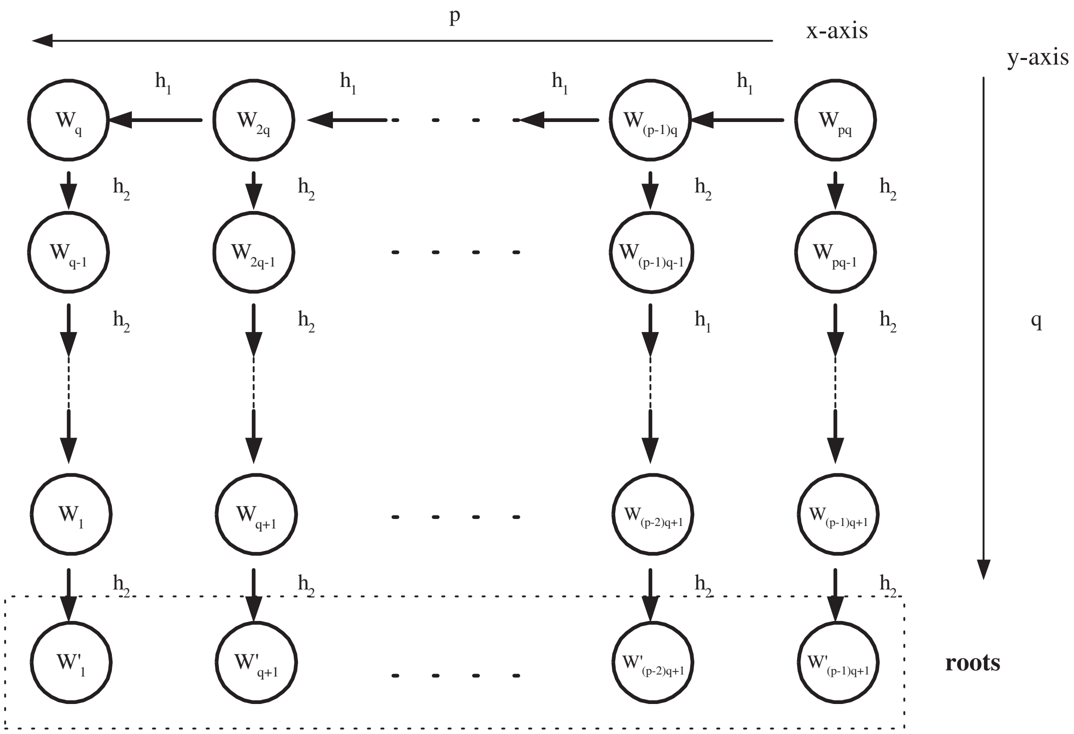

Figure 1: The construction of the unbalanced one-way binary tree

follows.

1. To input the parameters W_m, p, q, and i ($i \in 1, 2, \cdots, m$), where i means the ith pay-word.

2. To compute

$$\begin{aligned} a &= p - \lceil \frac{i}{q} \rceil, \\ b &= (q - (i \mod q)) \mod q, \ and \\ W_i &= h_2^b(h_1^a(W_{pq})). \end{aligned}$$

3. To produce the output value W_i.

Similar to the Pay-Word scheme, the new payment protocol is described in the following:

1. *The Initialization Phase:* First, the customer sends the commitment to the merchant. The commitment contains the consumer's certificate, the

merchant's identification, the signature of H, and the roots W'_1, W'_{q+1}, $\cdots$, W'_{pq-q+1}, where $H = h_2(W'_1 \| W'_{q+1} \| \cdots \| W'_{pq-q+1})$. Then the merchant validates the signature and creates two temporary parameters S and T, whose default values are W'_1 and $null$, respectively.

2. *The Payment Phase:* The customer computes the i-th pay-word W_i and sends the payment $\{W_i, i\}$ to the merchant. Upon receiving the pay-word, the merchant verifies the pay-word by doing the following examinations:

   (a) When $i = cq$ and $c \geq 2$, check $h_2(W_i) \stackrel{?}{=} S$ and $h_1(W_i) \stackrel{?}{=} T$;

   (b) Otherwise, check $h_2(W_i) \stackrel{?}{=} S$.

   Here c is an integer. If the above verification is confirmed, the merchant updates the parameters

   (a) $S = W'_{i+1}$ and $T = W_i$, if $i = cq$;

   (b) $S = W_i$, otherwise.

3. *The Clearance Phase:* In this phase, the merchant addresses the last payment (W_l, l) to the bank. If the verification is positive using the 2-dimension hash chain, the bank deposits l cents to the merchant's account.

We give a simple example for Yen's scheme. Assume that the parameters are $m = 9$, $p = 3$, and $q = 3$. Therefore, the roots of the chain are W'_1, W'_4, and W'_7. And the default values of S and T are W'_1 and $null$, respectively. The construction is shown in Figure 2. For the first request transaction, if a purchase costs two cents, the customer must deliver the commitment and the payment $[(W_1, 1), (W_2, 2)]$ to the merchant. If the equation $h_2(W_1) = S$ holds, the merchant updates S to W_1. Then the merchant verifies $h_2(W_2) = S$ (now $S = W_1$) and updates $S = W_2$ if the verification is valid. In the

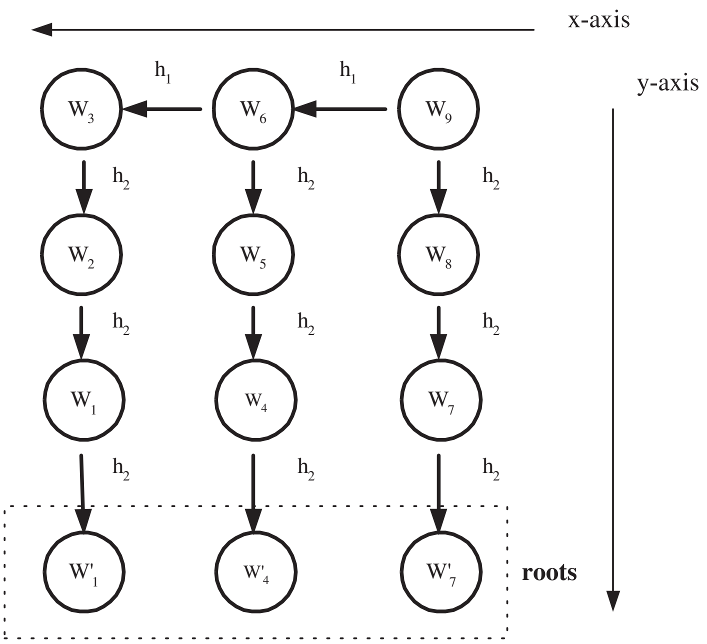

Figure 2: The construction of the 3×3 unbalanced one-way binary tree

second transaction, assume the purchase costs four cents. The customer sends the payment $[(W_6, 6)]$ to the merchant. The merchant can easily compute $W_3, 3$, $W_4, 4$, and $W_5, 5$ from $W_6, 6$. Because $3 = cq$, where $c = 1$ and $q = 3$, the merchant verifies whether $h_2(W_3) = S = W_2$ and updates $S = W_4'$ and $T = W_3$. The verifications of $(W_4, 4)$ and $(W_5, 5)$ are similar to those in the first transaction. To verify $(W_6, 6)$, the merchant must check whether $h_2(W_6) = S = W_5$ and $h_1(W_6) = T = W_3$. If the verification turns out positive correct, the merchant updates $S = W_7'$ and $T = W_6$. For the deposit date, the merchant addresses the last payment $(W_6, 6)$ to the bank. The bank validates the last payment by checking $h_2^3(h_1(W_6)) = W_1'$ and $h_2^3(W_6) = W_4'$ as Figure 2 shows.

The difference between the above two schemes is that Rivest's scheme uses a 1-dimension hash chain while Yen's scheme uses a 2-dimensions hash chain. The complexity analysis for these two schemes is as follows. Assume that the number of the nodes in the chain is m. In order to minimize the total number of hash operations, assume $p = q = m^{1/2}$ in Yen's scheme. On average, a customer needs $\frac{m-1}{2}$ and $\sqrt{m}-1$ hash function operations to generate a node in Rivest's and Yen 's scheme, respectively. However, in Rivest's scheme, the stored temporary parameter is only the root of the pay-word W_0; on the other hand, in Yen's scheme, the storage of temporary parameters includes $W'_1, W'_{q+1}, \cdots, W'_{(p-1)q+1}$. Furthermore, Rivest's scheme only requires a single hash function, while Yen's scheme is based on an unbalanced binary tree that requires two different hash functions.

3 Hash Sequence Traversal and n-dimension One-way Hash Chain Traversal Protocol

In this section, we first introduce the development of traversing a hash sequence. Then, we will propose a novel and efficient hash chain traversal protocol, called n-dimension one-way traversal protocol, for the computation of consecutive hash values.

3.1 Hash sequence traversal

Because the one-way hash function has low computational complexity, it has been widely used in several low-cost applications, such as micro-payments, authentication, and signature mechanisms. However, some applications, such as micro-payments, that were described in previous section require a large number of hash values as the money used to pay for merchandise. Therefore, the performance of traversing a hash chain has become an interesting issue and has attracted many researchers to this area of investigation. Suppose that a hash sequence is from v_0 to v_m, where v_m is a seed of the hash chain, and each element v_i is derived from v_m, such that $v_i = h^{m-i}(v_m)$. If we have to store the entire hash values v_i's for use later, the method has a storage complexity of $O(m)$ for a chain of size m. This is impractical, especially when the storage of the devices used is small. Conversely, if we only store the seed v_m of the chain, the method has a computational complexity of $O(m)$ for computing the entire hash values. Both methods have a memory-times-computational complexity of $O(m)$.

In order to minimize the memory and computational requirements of traversing a hash chain, two efficient hash sequence traversal schemes have been proposed recently [2, 7]. Both schemes are based on the principle of "pebbling," similar in spirit to [6]. In Jakobsson's scheme, he first provided an attractive solution [7] that only performs $\lceil \log_2 m \rceil$ hash function operations per output element and uses $\lceil \log_2 m \rceil$ memory cells to store some hash chain values with some short state information. Thus, both the computational and storage complexities of Jakobsson's scheme are $O(\log m)$. After that, Coppersmith and Jakobsson proposed a novel scheme [2] that further reduces the computational cost to about $\lfloor \frac{1}{2} \log_2 m \rfloor$. However, the scheme requires $\lceil \log_2 m \rceil + \lceil \log_2(\log_2 m + 1) \rceil$ memory cells. Obviously, this scheme requires more memory storage than Jakobsson's scheme.

In the following subsection, we will propose a novel n-dimension one-way hash chain traversal protocol that only stores a seed of the hash chain and performs $\lceil n \times (m^{\frac{1}{n}} - 1) \rceil$ hash function operations at most to derive the hash values. Thus, our traversal protocol has a storage complexity of $O(1)$ and has a computational complexity of $O(m^{\frac{1}{n}})$. In addition, the n-dimension one-way hash chain traversal protocol is much different than [7] and [2]. Their approaches use only one hash function, and our approach applies multiple hash functions. This feature is very useful in micro-payment schemes because each hash function can be represented as a different amount of money.

3.2 Hash sequence traversal using the n-dimensions one-way hash chain

The construction of the n-dimension one-way hash chain has the following characteristics.

1. Different one-way hash functions are used to derive the nodes in different

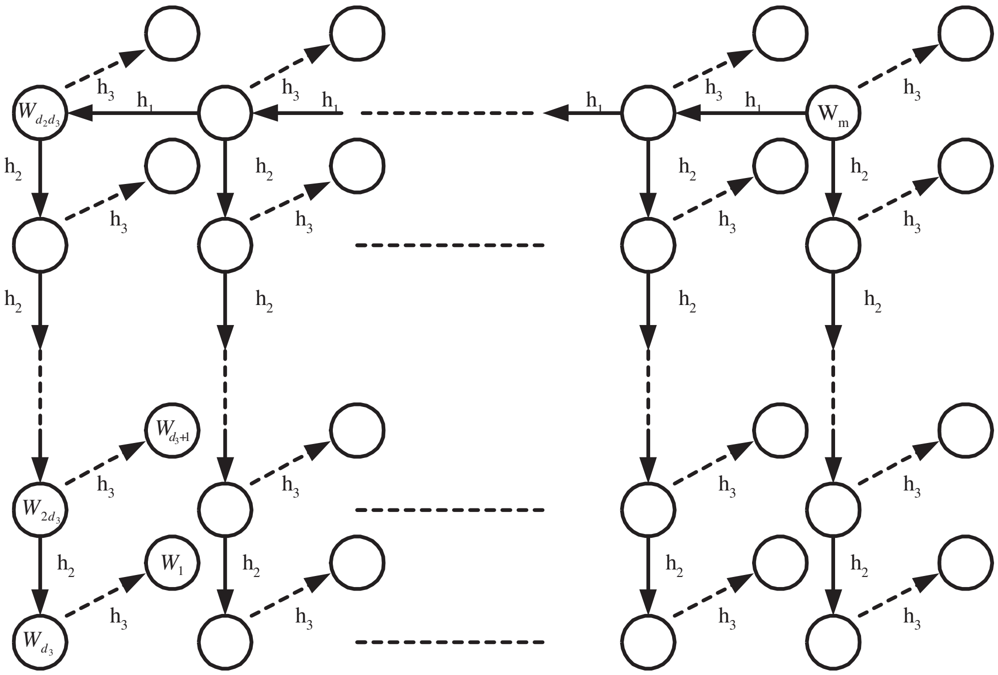

Figure 3: The structure of the 3-dimensions one-way hash chain

dimensions.

2. The nodes in one dimension are the seeds for another dimension.

3. Each node can be derived from the previous nodes easily, but it is difficult to derive the previous nodes from the current nodes.

In the n-dimension hash chain, each node represents a hash value. The numbers of nodes in the dimensions are, respectively, $d_1, d_2, \cdots, d_n$. And the nodes in the each dimension are derived by using the different hash functions $h_1, h_2, \cdots, h_n$, respectively. As an example, the construction of the 3-dimensions one-way hash chain is shown in Figure 3. In this construction, d_1, d_2, and d_3 express the numbers of nodes along the axes x, y, and z, respectively. First, the user chooses and stores an original seed W_m, where $m = d_1 \times d_2 \times d_3$. Then the hash value W_i in dimensions x, y, and z are derived from W_m by using the hash functions h_1, h_2, and h_3, respectively.

After the construction of the n-dimension hash chain is defined, the algorithm for computing node W_i in the n-dimensions hash chain can be generated as follows.

1. Input the seed W_m of the chain and the index i of the hash value W_i currently under computation.

2. Let $a_0 = i$, $c_0 = W_m$ and $d_{n+1} = 1$.

   For $j = 1$ to n, compute

   $$a_j = a_{j-1} \bmod (d_j \times d_{j+1} \times \cdots \times d_n).$$

   If $a_j \neq 0$, compute

   $$\begin{aligned} b_j &= d_j - \lceil \frac{a_j}{d_{j+1} \times d_{j+2} \times \cdots \times d_n \times d_{n+1}} \rceil, \text{ and} \\ c_j &= h_j^{b_j}(c_{j-1}). \end{aligned}$$

   If $a_j = 0$, break the for loop and the current node $W_i = c_{j-1}$.

   Until $j = n$, the current node $W_i = c_n$.

3. The output is the computing node W_i.

Theorem 1: In the worst case scenario, the computational complexity of our hash chain traversal algorithm is $O(m^{\frac{1}{n}})$, where m is the length of the hash chain and n is the number of dimensions.

Proof of Theorem 1: According to the construction of the n-dimension hash chain, the worst case scenario occurs when we want to derive the first hash value W_1 on the hash chain. Thus, $\sum_{k=1}^{n}(d_k - 1)$ hash function operations are required to derive the hash value W_1. Let $d_1 = d_2 = \cdots = d_n = M^{\frac{1}{n}}$. The required hash function operations become $n \times (m^{\frac{1}{n}} - 1)$. That implies that the performed hash function operations is bound at $n \times (m^{\frac{1}{n}} - 1)$ for each output element in the hash chain. Thus, the algorithm has a computational complexity of $O(m^{\frac{1}{n}})$ per output element in the worst case scenario.

Theorem 2: In the average case, our algorithm performs $n \times (\frac{m^{\frac{1}{n}}-1}{2})$ hash function operations per output element in the hash chain.

Proof of Theorem 2: The amount of required hash function operations to derive all the hash elements in the hash chain can be formulated as follows:

$$\begin{aligned} T &= \sum_{s_1=0}^{d_1-1} \sum_{s_2=0}^{d_2-1} \cdots \sum_{s_n=0}^{d_n-1} (s_1 + s_2 + \cdots + s_n), \\ &= \sum_{j=1}^{n} (\frac{(d_j-1)d_j}{2}) L_j, \end{aligned} \tag{1}$$

where T is the total number of required hash function operations and

$$L_j = \prod_{k=1,\cdots,n,k\neq j} d_k.$$

Let $d_1 = d_2 = \cdots = d_n = m^{\frac{1}{n}}$, Equation (1) becomes $T = m \times n \times (\frac{m^{\frac{1}{n}}-1}{2})$. Thus, our algorithm performs $\frac{T}{m} = n \times (\frac{m^{\frac{1}{n}}-1}{2})$ hash function operations per hash element on average.

Table 1 illustrates the comparisons of the computational complexity, the storage complexity, and the number of required hash functions in Jakobsson's scheme, Coppersmith and Jakobsson's scheme, Yen's unbalanced one-way binary tree, and our n-dimension one-way hash chains. Note that the value of m is usually much larger than the value of n.

4 The General Implementation of Pay-Word

The proposed micro-payment system is flexible and can be implemented on various devices. According to the memory space and the computational ability of

Table 1: The comparisons of related approaches on hash sequence traversal

Approaches	Computational complexity	Storage complexity	The required hash functions
Jakobsson's approach	$O(\log_2 m)$	$O(\log_2 m)$	1
Coppersmith and Jakobsson's approach	$O(\log_2 m)$	$O(\log_2 m)$	1
Yen's approach	$O(m^{\frac{1}{2}})$	$O(1)$	2
Our approach	$O(m^{\frac{1}{n}})$	$O(1)$	n

the device, the user can choose suitable number of dimensions for implementing the micro-payment system efficiently.

4.1 The micro-payment scheme based on the n-dimension one-way hash chain

Extending Rivest' and Yen's 1- and 2-dimensions micro-payment scheme, we come to a more flexible micro-payment scheme. The fundamental design of the proposed scheme involves implementing the n-dimension one-way hash chain technique in Rivest's Pay-Word micro-payment scheme. Each node in the n-dimension hash chain represents a pay-word that is worth one cent. Furthermore, the roots of the chain can be computed by using the hash function h_n. For example, in Figure 3, the roots of this three-dimension hash chain are W'_1, W'_{d_3+1}, W'_{2d_3+1}, $\cdots$, $W'_{(d_1 \times d_2 - 1) \times d_3 + 1}$. Our scheme can be divided into three phases: the initialization phase, the payment phase and the clearance phase. There are three kinds of participants in our scheme: the customer C, the merchant M, and the bank B. The customer and the merchant must first open accounts with the bank. The customer then generates the pay-words and spends the pay-words in order. The detailed process of the proposed micro-payment scheme is described as follows.

The Initialization Phase:

1. B submits a certificate to an authorized C. Each certificate contains the following information: the customer's public key, the bank's identification, the customer's identification, the valid dates, and the signature for this information.

2. C indicates the numbers $d_1, d_2, \cdots, d_n$ of all the dimensions and determines an original seed W_m, where $m = d_1 \times d_2 \times \cdots \times d_n$. W_m is used to generate the pay-word and its roots as described in Subsection 3.2.

3. C submits a "*commitment*" to M when C first requests a purchase from M. The commitment includes a set of roots R, the customer's certificate, the merchant's ID, and $d_1, d_2, \cdots, d_n$.

4. C sends the commitment and the commitment signature to M.

5. M verifies the signature and creates the temporary verification parameters $V_1, V_2, \cdots, V_n$. The default of V_n equals W_1', and all the others are null.

The Payment Phase:

1. C computes the $i - th$ pay-word W_i. The generation algorithm is described in Subsection 3.2.

2. C sends the payment $\{W_i, i\}$ to M, in which i is the index of the pay-word.

3. M verifies the accuracy of the pay-word using the following equations. Let $q_j = \prod_{k=j+1}^{n} d_k$ and r is a integer.

For $j = 1$ to $n-1$,

if the index $i = r \times q_j$ and $r \geq 2$, check

$$V_j \stackrel{?}{=} h_j(W_i);$$

otherwise, only check the equation

$$V_n = h_n(W_i). \tag{2}$$

4. If the above verification is positive, M updates the temporary verification parameters as

   (a) For $j = 1, 2, \cdots, n-1$, if $i = r \times q_j$ and $r < d_j$, update $V_j = W_i$ and $V_n = W'_{i+1}$.

   (b) Otherwise, only update $V_n = W_i$.

The Clearance Phase:

1. At the time of clearance, M sends the commitment and the last payment $\{W_l, l\}$ received from C.

2. B checks the signature and the valid dates.

3. Using W_l to compute the roots used, B then verifies if the roots are identical with the roots of the commitment.

4. If the above verifications are positive, B withdraws l cents from $C's$ account and deposits l cents to $M's$ account.

4.2 Example

In this subsection, we give an example to illustrate how the proposed scheme works. We use a 3-dimensions hash chain in this example, and the structure of this example is shown in Figure 4.

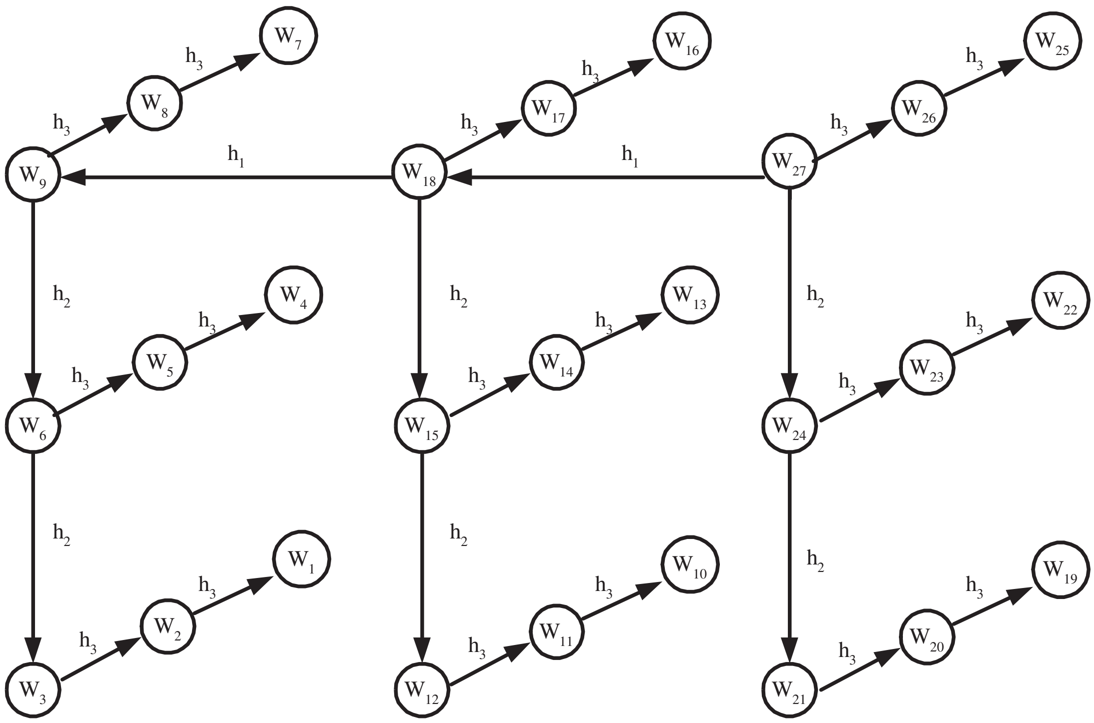

Figure 4: The example of the 3-dimensions one-way hash chain

Assume that C wants to generate a 3-dimensions hash chain which contains 27 pay-words. Therefore, C denotes $d_1 = d_2 = d_3 = 3$. C then selects a seed W_{27} and generates the roots $W'_1, W'_4, W'_7, W'_{10}, W'_{13}, W'_{16},$ *and* W'_{19}. For the first request transaction, C sends the commitment and the pay-words $(W_1, 1)$, $(W_2, 2)$, $(W_3, 3)$ to M. M first verifies the validity of the commitment and then sets $V_1 = null$, $V_2 = null$, and $V_3 = W'_1$.

As Figure 4 shows, the verification of W_1, W_2, W_3, are checked by Equation 2. The parameters $V_3 = W_1$ and $V_3 = W_2$ are updated after the verification of the pay-words W_1 and W_2, respectively. Moreover, C needs to update $V_3 = W'_4$ and $V_2 = W_3$ after verifying the pay-word W_3 since $3 = 1 \times q_2$.

Considering the special nodes of the chain, for the pay-word W_6, both the equations $V_3 = W_5 = h_3(W_6)$ and $V_2 = W_3 = h_2(W_6)$ need to be verified. The parameters $V_3 = W'_7$ and $V_2 = W_6$ also need to be updated. The pay-word W_{18} requires the verification of the three equations $V_1 = W_9 = h_1(W_{18})$,

$V_2 = W_{15} = h_2(W_{18})$, and $V_3 = W_{17} = h_3(W_{18})$ since $18 = 2 \times q_1 = 2 \times 9$, $18 = 6 \times q_2 = 6 \times 3$. It also requires that the parameters $V_1 = W_{18}$ and $V_3 = W'_{19}$ be updated.

Assume that the customer spends 18 cents in this merchant. For the deposit date, the merchant sends the last payment $(W_{18}, 18)$ to the bank. The bank computes W'_{16}, W'_{13}, W'_{10}, W'_7, W'_4, and W'_1 from W_{18}, and checks if the roots equal the roots in the commitment. If the verification is positive, the bank deposits the money to the merchant's account.

5 Performance Analysis

In this section, the total number of hash operations, the required roots, and the required different hash functions in the general Pay-Word scheme are summarized. In order to minimize the total number of hash operations, we assume that the numbers of nodes in all the dimensions are equal to each other. Let $m = d_1 \times d_2 \times \cdots \times d_n$.

First, we consider the required hash operations in the proposed micropayment scheme. In this scheme, since the customer holds a seed W_m of the chain, he/she can derive all the pay-words in the chain by using the seed and hash operations. The computation of the total number of hash operations is similar to the n-dimension hash chain traversal protocol. According to Equation 1, the total number of required hash operations is $T = m \times n \times (\frac{m^{\frac{1}{n}}-1}{2})$, and the average number of hash operations for computing each payword is $E = \frac{T}{m} = n \times (\frac{m^{\frac{1}{n}}-1}{2})$.

Second, we consider the required roots in the general Pay-Word scheme. In the n-dimension hash chain, the number of required roots equals $d_1 \times d_2 \times \cdots \times d_{n-1}$. If all the dimensions have the same number of nodes, the total number of required roots becomes $m^{\frac{n-1}{n}}$. Furthermore, n different hash functions are required. A summary of the performance analysis is shown in Table 2.

Table 2: Summary of the n-dimensions one-way hash chain performance

the average hash operations	the required roots	the required hash functions
$\frac{n(m^{\frac{1}{n}}-1)}{2}$	$m^{\frac{n-1}{n}}$	n

Table 3: The comparisons among different numbers of dimensions used to generate 1000 pay-words

# of dimensions	1	2	3	4	5
the nodes in each dimension	1000	32	10	6	4
the total pay-words	1000	1024	1000	1296	1024
the required hash operations (average case)	499.5	31	13.5	10	7.5
the required hash operations (worst case)	999	62	27	20	15
the required roots	1	32	100	216	256
the required hash functions	1	2	3	4	5

Suppose that now we want to generate 1000 pay-words. We can use different numbers of dimensions to construct the hash chain. The comparison among different numbers of dimensions is shown in Table 3. Rivest and Shamir's scheme [12] is a special case of our general Pay-Word scheme when $n = 1$. Yen et al.'s scheme [13] is based on a two-dimension hash chain that is also a special case of our scheme when $n = 2$. From Table 3, when the length of a hash chain is close, we can see that when the number of dimensions increases, the computational cost is reduced, but the storage cost increases. Conversely, when the number of dimensions decreases, the storage cost is reduced, but the computational cost increases. A large amount of required hash operations means that more time is required to derive a pay-word. Thus, it is not suitable for real-time systems. For instance, Rivest and Shamir's scheme ($n = 1$) has a least storage cost but it requires a largest amount of hash operations to derive the pay-words. Thus, Rivest and Shamir's scheme is suitable for less-storage devices but it is not suitable for real time system because it needs to perform

999 hash operations to derive a pay-word in the worst case scenario. According to the system's requirements, our proposed scheme allows a user to flexibly choose an appropriate n so that the optimum time/memory trade-off can be achieved.

6 Conclusions

Pay-Word is an off-line micro-payment scheme. The merchant verifies the payment without involving the bank. The merchant batch settles the payments of all customers at the end of each settlement period. This reduces the amount of required communication with the bank. Furthermore, the pay-words of the chain can express various types of currency. The scheme uses a one-dimensional hash chain. This scheme is an efficient and low-cost micro-payment system.

In this article, we extend the one dimensional hash chain to n dimensions. In our proposed general Pay-Word scheme, the computational complexity is $O(m^{\frac{1}{n}})$, and the complexity of storing temporary parameters is $O(m^{\frac{n-1}{n}})$. A user can easily implement this system in any device and achieve high computation efficiency. The user can determine the memory/time trade off and choose the right number of dimensions for her-/himself.

Acknowledgements

The authors wish to thank many anonymous referees for their suggestions to improve this paper.

References

[1] R. Anderson, H. Manifavas, and C. Sutherland, "NetCard – A Practical Electronic Cash System," *Security Protocols—International Workshop*, Cambridge, United Kingdom, vol. 1189, April, 1997.

[2] D. Coppersmith and M. Jakobsson, "Almost optimal hash sequence traversal," *Financial Cryptography (FC'02)*, Lecture Notes in Computer Science, Springer-Verlag, 2002.

[3] R. Hauser, M. Steiner, and M. Waidner, "Micro-payments based on iKP," in *Proceedings of SECURICOM 96, 14th Worldwide Congresson Computer and Communication Security and Protection*, pp. 67–82, 1996.

[4] M. S. Hwang, I. C. Lin, and L.H. Li, "A simple micro-payment scheme," *International Journal of Systems and Software*, vol. 55, no. 3, pp. 221–229, 2001.

[5] M. S. Hwang, E. J. L. Lu, and I.C. Lin, "Adding timestamps to the secure electronic auction protocol," *Data & Knowledge Engineering*, vol. 40, no. 2, pp. 155–162, 2002.

[6] G. Itkis and L. Reyzin, "Forward-secure signatures with optimal signing and verifing," *Advances in Cryptology-CRYPTO 2001*, Lecture Notes in Computer Science, Vol. 2139, Springer-Verlag, pp.19-23, 2001.

[7] M. Jakobsson, "Fractal hash sequence representation and traversal," in *Proceedings of the IEEE International Symposium on Information Theory (ISIT02)*, pp. 437–444, 2002.

[8] C. S. Jutla and M. Yung, "PayTree: Amoritized-Signature for Flexible MicroPayments," *Second USENIX Workshop on Electronic Commerce*, pp. 213–221, 1996.

[9] NITS. "Secure hash standard,". Tech. Rep. FIPS 180-1, NITS, US Department Commerce, April 1995.

[10] T. Pedersen, "Electronic payments of small amounts," in *LNCS 1189, Proceedings of Security Protocols Workshop*, pp. 59–68, 1997.

[11] R. Rivest. "The MD5 message digest algorithm,". Tech. Rep. RFC 1321, April 1992.

[12] R. Rivest and A. Shamir. "PayWord and MicroMint: Two simple micro payment schemes," in *Proceedings of 1996 International Workshop on Security Protocols*, pp. 69–87, 1996.

[13] S. M. Yen, L. T. Ho, and C. Y. Huang, "Internet micropayment based on unbalanced one-way binary tree," in *Proceedings of the Ninth National Conference on Information Security*, pp. 66–73, 1999.

I.-C. Lin received the B.S. in Computer and Information Sciences from Tung Hai University, Taichung, Taiwan, Republic of China, in 1998; the M.S. in Information Management from Chaoyang University of Technology, Taiwan, in 2000. He is currently pursuing his Ph.D. degree in Computer Science and Information Engineering from National Chung Cheng University, Taiwan. His current research interests include electronic commerce, information security, cryptography, and mobile communications.

M.-S. Hwang was born on August 27, 1960 in Tainan, Taiwan, Republic of China (ROC.). He received the B.S. in Electronic Engineering from National Taipei Institute of Technology, Taipei, Taiwan, ROC, in 1980; the M.S. in Industrial Engineering from National Tsing Hua University, Taiwan, in 1988; and the Ph.D. in Computer and Information Science from National Chiao Tung University, Taiwan, in 1995. He also studied Applied Mathematics at National Cheng Kung University, Taiwan, from 1984-1986. Dr. Hwang passed the National Higher Examination in field "Electronic Engineer" in 1988. He also passed the National Telecommunication Special Examination in field "Information Engineering", qualified as advanced technician the first class in 1990. From 1988 to 1991, he was the leader of the Computer Center at Telecommunication Laboratories (TL), Ministry of Transportation and Communications, ROC. He was also a chairman of the Department of Information Management, Chaoyang University of Technology (CYUT), Taiwan, during 1999-2002. He was a professor and chairman of the Graduate Institute of Networking and Communications, CYUT, during 2002-2003. He obtained the 1997, 1998, 1999, 2000, and 2001 Distinguished Research Awards of the National Science Council of the Republic of China. He is currently a professor of the department of Management Information System, National Chung Hsing University, Taiwan, ROC. He is a member of IEEE, ACM, and Chinese Information Security Association. His current research interests include electronic commerce, database and data security, cryptography, image compression, and mobile computing. Dr. Hwang had published 90 articles on the above research fields in international journals.

Chin-Chen Chang received his BS degree in applied mathematics in 1977 and the MS degree in computer and decision sciences in 1979, both from the National Tsing Hua University, Hsinchu, Taiwan. He received his Ph.D in computer engineering in 1982 from the National Chiao Tung University, Hsinchu, Taiwan. During the academic years of 1980-1983, he was on the faculty of the Department of Computer Engineering at the National Chiao Tung University. From 1983-1989, he was on the faculty of the Institute of Applied Mathematics, National Chung Hsing University, Taichung, Taiwan. From August 1989 to July 1992, he was the head of, and a professor in, the Institute of Computer Science and Information Engineering at the National Chung Cheng University, Chiayi, Taiwan. From August 1992 to July 1995, he was the dean of the college of Engineering at the same university. From August 1995 to October 1997, he was the provost at the National Chung Cheng University. From September 1996 to October 1997, Dr. Chang was the Acting President at the National Chung Cheng University. From July 1998 to June 2000, he was the director of the Ministry of Education of the R.O.C.. Since 2002, he has been a Chair Professor of National Chung Cheng University. In addition, he has served as a consultant to several research institutes and government departments. His current research interests include database design, computer cryptography, image compression and data structures. Dr. Chang has published more than 250 papers in well-known journals, more than 200 papers in international conference proceedings, and 21 books in the field of database design, data structures, information security and cryptography, in Chinese. Dr. Chang is a fellow of the IEEE, a fellow of IEE, a research fellow of National Science Council of R.O.C., and a member of the Chinese Language Computer Society, the Chinese Institute of Engineers of the Republic of China, the International Association for Cryptologic Research, the Computer Society of the Republic of China, and the Phi Tau Phi Honorary Society of the Republic of China. Dr. Chang was the chair and is the honorary chair of the executive committee of the Chinese Cryptography and Information Security Association of the Republic of China.