ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.20, No.3, March 2009, pp.682−691
doi: 10.3724/SP.J.1001.2009.00556

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

一种基于分组密码的 hash 函数*

林 品[1,2+], 吴文玲[2], 武传坤[2]

[1](中国科学院 软件研究所 信息安全国家重点实验室,北京 100190)

[2](中国科学院 研究生院,北京 100049)

Hash Functions Based on Block Ciphers

LIN Pin[1,2+], WU Wen-Ling[2], WU Chuan-Kun[2]

[1](State Key Laboratory of Information Security, Institute of Software, The Chinese Academy of Sciences, Beijing 100190, China)

[2](Graduate University, The Chinese Academy of Sciences, Beijing 100049, China)

+ Corresponding author: E-mail: ping_linux@163.com

Lin P, Wu WL, Wu CK. Hash functions based on block ciphers. *Journal of Software*, 2009,20(3):682−691.
http://www.jos.org.cn/1000-9825/556.htm

Abstract: In this paper, a hash function with lower *rate* but higher efficiency is proposed and it can be built on insecure compression functions. The security of this scheme is proved under black-box model and some compression function based on block ciphers are given to build this scheme. It is also shown that key schedule is a more important factor affecting the efficiency of a block-cipher-based hash function than *rate*. The new scheme only needs 2 keys and the key schedule of it can be pre-computed. It means the new scheme need not re-schedule the keys at every step during the iterations and its efficiency is improved.

Key words: block cipher, hash function, collision attack, pre-image attack, second pre-image attack

摘 要: 提出了一个基于分组密码的 hash 函数体制,它的 *rate* 小于 1 但却具有更高的效率,同时,这个 hash 函数可以使用不安全的压缩函数进行构造,降低了对压缩函数安全性的要求.首先,在黑盒子模型下对这个新的体制的安全性进行了证明,然后给出了能够用于构造该体制的使用分组密码构造的压缩函数,最后通过实验对比发现,新 hash 函数的速度比 *rate* 为 1 的 hash 函数快得多.实验结果表明,除了 *rate* 以外,密钥编排也是影响基于分组密码 hash 函数效率的重要因素,甚至比 *rate* 影响更大.该体制只有两个密钥,不需要进行大量的密钥扩展运算,大大提高了基于分组密码 hash 函数的效率,而且该体制可以使用现有的分组密码来构造.

关键词: 分组密码;hash 函数;碰撞攻击;前像攻击;第二前像攻击

中图法分类号: TP309 **文献标识码**: A

* Supported by the National Natural Science Foundation of China under Grant No.90604036 (国家自然科学基金); the National Basic Research Program of China under Grant No.2004CB318004 (国家重点基础研究发展计划(973)); the National High-Tech Research and Development Plan of China under Grant No.2007AA01Z470 (国家高技术研究发展计划(863))

Received 2007-09-06; Accepted 2008-02-27

1 Introduction

Hash function is a mapping from an input with arbitrary length to an output with fixed length which can be shown as follows.

$$H:\{0,1\}^{*}\rightarrow\{0,1\}^{n}$$

In 1989, Merkle and Damgård independently discussed a method to construct a hash function using a compression function with fixed input length[1,2]. This method is called MD method or iterated method and can be described as follows.

$$h_0 = IV$$
$$h_i = f(h_{i-1}, m_i), 1 \le i \le l$$
$$H(M) = h_l$$

here f is the compression function, IV is the fixed initial value of the hash function, $h_1, h_2, ...h_{l-1}$ are called chain values, h_l is the output, $M = m_1, m_2, ...m_l$ is the input message which is padded and divided into l blocks. One of the padding rules is to append 1 to the message and then append enough 0s to make the padded length a multiple of $|m_i|$ and finally store the length of the original message into the last block i.e. $|m_l|$. This rule is called MD-Strengthening. Lai pointed out that if a hash function is not padded with MD-Strengthening, then there is an effective attack to find its collisions[3]. Most hash functions used in practice are based on this method, such as MD4[4], MD5[5], SHA-0[6], SHA-1[7] etc. These hash functions are called dedicated-designed hash functions because the compression functions of these hash functions are specially designed. These hash functions are very fast but the compression functions need to be carefully designed and the security cannot be proved in some model. Substituted the compression functions with block ciphers, some hash functions based on block ciphers are proposed, such as MDC-2[8], PGV schemes[9]. In this way, a specially designed compression function is not necessary. Because the block ciphers are not designed specially for hash functions, the efficiency of these hash functions is lower than those dedicated- designed ones. So it is important to design a secure hash function based on block ciphers and endeavor to improve its efficiency.

In 1993, Preneel etc. considered all 64 hash functions based on block ciphers called PGV schemes[9]. The compression functions can be described as follows.

$$f(h_{i-1}, m_i) = E_a(b) \oplus C$$

here $a, b, c \in \{h_{i-1}, m_i, h_{i-1} \oplus m_i\}$ and h_0 is a fixed constant. Reference [9] gives attacks on these schemes but does not give a formal proof. In 2002, Black proved the security of these schemes in the black-box model and divided these schemes into three groups[10]. Besides the security, the efficiency is another property of block-cipher-based hash functions. To describe the efficiency of hash functions based on block ciphers, *rate* is introduced which means the number of blocks dealt with after running the block cipher one time. But *rate* is not the only factor that influences the efficiency of block-cipher-based hash functions; another factor is the key schedule. In many block ciphers, key schedule is slower than encryption and decryption. It should be noted that the *rate* of PGV schemes is 1 and all secure schemes need re-schedule keys at every step. When hash functions based on block ciphers were initially proposed, the output length of block ciphers is 64 bit. With the development of information technology, it has become insecure. So some double-block-length hash functions based on block ciphers have been proposed to improve the security, such as Parallel-DM[11], PBGV[12], LOKI[13]. But unfortunately Knudsen etc. proved that all these schemes whose *rate* is 1 are not secure[14]. The *rate* of the secure schemes is less than 1. Some *rate*-1/2 schemes are proved to be secure[15], but their efficiency is very low. To preserve security and improve efficiency simultaneously, Nandi etc. proposed a $rate-\frac{2}{3}$ scheme[16] and proved it is secure. But Knudsen proved it is as

secure as the secure single-block-length schemes[17]. Now the *rate* of secure hash double-block-length function is 1/2 or 1/4 (MDC-4). It is obvious that the double-block-length hash functions are not efficient. The emergence of AES has modified this landscape, so single-block-length hash functions may satisfy the security requirement. As mentioned above, the block-cipher-based hash functions need re-schedule the key at every step and key schedule is slower than encryption. So an efficient method to construct a block-cipher-based hash function is needed. A fixed key seems a good idea, and some schemes with a fixed key have been proposed such as Tweakable Chain Hash (TCH)[18]. This kind of hash functions is called highly efficient block-cipher-based hash functions. However, Black proved that all block-cipher-based *rate*-1 hash functions are not secure if they do not rekey the block cipher[19]. Black pointed out that TCH could not be correctly instantiated by this efficient means.

In this paper, we propose a new way to construct a single-block-length hash function based on block ciphers. All secure schemes mentioned above need a secure compression function to ensure the security of the hash functions. Unlike these schemes, the compression function used in the new scheme in this paper can be insecure but the hash function constructed with it is secure. Usually one judges by *rate* whether a block-cipher- based hash function is efficient. Larger the *rate* is and higher the efficiency. We analyze the factors that affect the efficiency of block-cipher-based hash functions and we find that besides *rate*, key schedule is another very important factor. A hash function with smaller *rate* may be more efficient than the ones with larger *rate*.

2 Definition and Preliminaries

BLOCK CIPHERS. A block cipher is a map $E:\{0,1\}^k\times\{0,1\}^n\to\{0,1\}^n$ where k is the key length in bits and n is the block length in bits. For each $K\in\{0,1\}^k, E_K=E(K,\bullet)$ is a permutation on $\{0,1\}^n$. E^{-1} is its inverse. $E_K^{-1}(y)$ returns a string x such that $E_K(x)=y$. Let *Block*(k, n) denote all block ciphers $E:\{0,1\}^k\times\{0,1\}^n\to\{0,1\}^n$. In this paper, block ciphers selected from *Block*(k, n) are used to construct a hash function.

BLACK-BOX MODEL. This model is the one dating back to Shannon[20]. Black etc. used it to prove the security of all PGV schemes. This model regards the block cipher as a black box, the adversary do not know any details about the block cipher and is only given access to oracle E^{-1} and E. Here E is a block cipher mentioned above and E^{-1} is its inverse. This model means given a fixed key $K\in\{0,1\}^k, E_K=E(K,\bullet)$ is a permutation on $\{0,1\}^n$ and the adversary queries the oracles E^{-1} and E. If the query is (K, x), E returns y such that $E_K(x)=y$. The latter on input (K, y) returns x such that $E_K(x)=y$.

SECURE HASH FUNCTIONS. In Ref.[14], Knudsen proposes seven attacks to analyze the security of hash functions. In fact, only three of these attacks are effective for hash functions, collision attack, pre-image attack and second pre-image attack. A secure hash function must be collision attack resistant, pre-image attack resistant and second pre-image attack resistant. Actually if a hash function is collision attack resistant, it is also second pre-image resistant[21]. Therefore, only collision attack and pre-image attack are considered in this paper.

Definition 1 (**collision attack**). Given a hash function H and its initial value (*IV*) h_0, find M and M' where $M\neq M'$ such that $H(h_0,M)=H(h_0,M')$.

Definition 2 (**pre-image attack**). Given a hash function H and its initial value (*IV*) h_0 and a randomly selected value σ, find M such that $H(h_0,M)=\sigma$.

D**efinition 3** (**second pre-image attack**). Given a hash function H and its initial value (*IV*) h_0 and M, find $M'\neq M$ such that $H(h_0,M)=H(h_0,M')$.

Assuming the output length of a hash function is n bit, then the complexity of finding a collision and a pre-image of a secure hash function should be $O(2^{n/2})$ oracles and $O(2^n)$ oracles respectively i.e. the hash function is collision attack resistant and pre-image attack resistant. We use the collision attack advantage and pre-image attack

advantage defined in Ref.[10] to formally describe the security of a hash function or a compression function. $x \xleftarrow{\$} S$ denotes the experiment of choosing a random element x from a set S.

Definition 4 (**collision resistance of a hash function**). Let H be a hash function constructed with a block cipher E, A be an adversary. Then the advantage of A in finding a collision in H is

$$Adv_{coll}^{H}(A) = \Pr[E \xleftarrow{\$} Block(k,n); M, M' \longleftarrow A^{E,E^{-1}} : M \neq M' \text{ and } H^{E}(M) = H^{E}(M')]$$

Let q denote the amount of the most queries that A has made, we write $Adv_{H}^{coll}(q) = \max(Adv_{H}^{coll}(A))$. Similarly we define the inversion advantage to describe the inversion resistance, i.e., pre-image resistance of a hash function.

Definition 5 (**inversion resistance of a hash function**). Let H be a hash function constructed with a block cipher E, A be an adversary. Then the advantage of A in finding a pre-image in H is

$$Adv_{inv}^{H}(A) = \Pr[E \xleftarrow{\$} Block(k,n); \sigma \xleftarrow{\$} \{0,1\}^{n}; M \longleftarrow A^{E,E^{-1}} : H^{E}(M) = \sigma]$$

The two advantages should be negligible in computation so that to get significant advantages the computation complexity needed are $O(2^{n/2})$ oracles and $O(2^{n})$ oracles respectively. The definition of collision resistance and pre-image resistance for a compression function has a little difference with the hash function.

Definition 6 (**collision resistance of a compression function**). Let f be a block-cipher-based compression function and A be an adversary. Then the advantage of A to find a collision of f is

$$Adv_{coll}^{f}(A) = \Pr[f \xleftarrow{\$} Block(k,n); ((h,m),(h',m')) \longleftarrow A^{f} : ((h,m) \neq (h',m') \wedge f(h,m) = f(h',m')) \vee f(h,m) = IV]$$

where m is the message block used by the adversary.

Definition 7 (**inversion resistance of a compression function**). Let f be a block-cipher-based compression function and A be an adversary. Then the advantage of A to invert f is

$$Adv_{inv}^{f}(A) = \Pr[f \xleftarrow{\$} Block(k,n); \sigma \xleftarrow{\$} \{0,1\}^{n}; (h,m) \longleftarrow A^{f}(\sigma) : f(h,m) = \sigma].$$

3 New Scheme and Its Security

3.1 Compression function used in this scheme

The compression function used in this scheme is defined as follows.

$$E_K(a \oplus b),$$

here K is the key of a block cipher, $a = h_{i-1}$, $b = m_i$, where h_{i-1} is the chain value and m_i is the message block. It is easy to see that the compression function is not secure. For any pair (h_{i-1}, m_i) and (h'_{i-1}, m'_i) if $h_{i-1} \oplus m_i = h'_{i-1} \oplus m'_i$, then $E_K(h_{i-1} \oplus m_i) = E_K(h'_{i-1} \oplus m'_i)$. If this function is used to construct a hash function in the MD method, the hash function is not collision resistant. This hash function is shown in Fig.1.

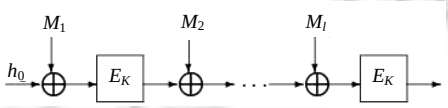

Fig.1 Single-Block-Length scheme iterated with the compression function

Any pair of messages with two blocks such that $M = h_0 \oplus c_1 \| E_K(c_1) \oplus v, M' = h_0 \oplus c_2 \| E_K(c_2) \oplus v$ will be a collision for this hash function. The output is $E_K(v)$. Here c_1 and c_2 are different values so that $M \neq M'$ and v is any constant value. It is obvious that this compression cannot be used to construct a secure hash function although it can be used to construct a secure MAC such as CBC.

3.2 New scheme description and its security analysis

It has been mentioned above that it is impossible to construct a *rate*-1 highly efficient block-cipher-based hash

function. How about *rate*-1/2? Our scheme is a *rate*-1/2 hash function iterated with the compression function mentioned above. It can be shown in Fig.2 as follows. Here $h_{0,1}$, $h_{0,2}$ are two different initial values. E_{K_1} and E_{K_2} denote two different and independent permutations by the definition. $h_{l,1}$ and $h_{l,2}$ denote the outputs of the two branches and g denotes the output transformation. The output of this scheme is $g(h_{l,1} \| h_{l,2})$. The two branches are denoted by H_1 and H_2 respectively.

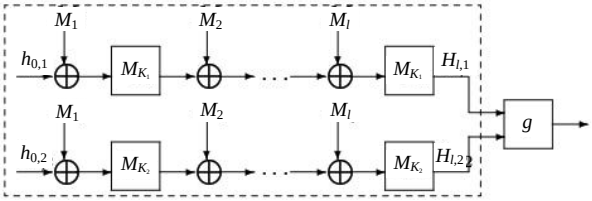

Fig.2 New scheme in this paper

Lemma 1. Let H' be the hash function shown in the dash box of Fig.2. n is the block length in bit and k is the key length in bit. Then the advantage of finding a collision for H' is

$$Adv_{coll}^{H'}(q) \le \frac{q(q+1)}{2^r}$$

where $r = Min(k,n)$ and q is the number of queries.

Proof: We define a direct graph $G_1 = (V_{G_1}, D_{G_1})$ to save the output of query E_{K_1} or $E_{K_1}^{-1}$ where V denotes the set of vertices and D denotes the set of edges. The vertex set is $V = \{0,1\}^n \times \{0,1\}^n \times \{0,1\}^n$. Let $(h_{i,1}, m_{i,1}, y_{i,1})$ denote one vertex (i denotes the i-th query and $y_{i,1} = E_{K_1}(h_{i,1} \oplus m_{i,1})$). An arc $(h_{i,1}, m_{i,1}, y_{i,1}) \to (h_{j,1}, m_{j,1}, y_{j,1})\ (i \ne j)$ is in D_{G_1} if $y_{i,1} = h_{j,1}$. Initially, each vertex in G_1 is uncolored. When adversary A asks query E_{K_1} or $E_{K_1}^{-1}$, if $h_{i,1} = h_{0,1}$ the vertex $(h_{i,1}, m_{i,1}, y_{i,1})$ is colored *red* else colored *black*. A path P in G_1 is colored if all of its vertices are colored. Two vertices $(h_{i,1}, m_{i,1}, y_{i,1})$ and $(h_{j,1}, m_{j,1}, y_{j,1})$) are colliding vertices if $y_{i,1} = y_{j,1}$. So there is a collision of H_1 if and only if there are two paths P and P' whose vertices are colored and they begin with $h_{0,1}$ and end with colliding vertices. Let $\mathbf{C}$ denote this event. We define another direct graph $G_2 = (V_{G_2}, D_{G_2})$ to save the result of queries of E_{K_2} or $E_{K_2}^{-1}$. Its conventions are the same as G_1 except that the oracle is E_{K_2} or $E_{K_2}^{-1}$ and its vertex is like $(h_{i,2}, m_{i,2}, y_{i,2})$. We color the vertices in G_2 with the same message blocks as in G_1 when the vertices in G_1 are colored. Let $\mathbf{C}'$ denote the event that there are two colliding paths. So

$$Adv_{coll}^{H'} = \Pr[\mathbf{C}' \wedge \mathbf{C}]$$

Then we give Pr[$\mathbf{C}$] and Pr[$\mathbf{C}'$]. Let $\mathbf{C}_i$ denote that the event $\mathbf{C}$ occurs at the i-th query. That is to say that the arcs queried previously form a colliding path at the i-th query. $\mathbf{C}_i$ occurs if and only if there exists an arc $(h_{p,1}, m_{p,1}, y_{p,1}) \to (h_{i,1}, m_{i,1}, y_{i,1})$ and $y_{i,1} = y$ where $y \in \{y_{1,1}, y_{2,1}, ..., y_{i-1,1}\} \cup \{h_{0,1}\}$ and $(h_{p,1}, m_{p,1}, y_{p,1})$ is the vertex queried at the p-th ($p<i$) query. It means that $y_{i,1}$ has been queried before the i-th query. So

$$\Pr[\mathbf{C}_i] = \Pr[E_{K_1}(h_{i,1} \oplus m_{i,1}) = y_{j,1}] \quad (j \in \{0,2,3,...,i-1\}).$$

It can be deduced that Pr[$\mathbf{C}_i$]=1 and Pr[$\mathbf{C}$]=1. Because for any $y \in \{y_{1,1}, y_{2,1}, ..., y_{i-1,1}\} \cup h_{0,1}$, a message block m_i can be computed by $E_{K_1}^{-1}(y) \oplus h_{i,1}$. After the vertices queried in the i-th query have been colored in G_1, we color the i-th queried vertices in G_2. Let $\mathbf{C}_i'$ denote the event that $\mathbf{C}'$ occurs. As in G_1, $\mathbf{C}_i'$ occurs if and only if there exists an arc $(h_{p,2}, m_{p,2}, y_{p,2}) \to (h_{i,2}, m_{i,2}, y_{i,2})$ and $y_{i,2} = y'$, where $y' \in \{y_{1,2}, y_{2,2}, ..., y_{i-1,2}\} \cup h_{0,2}$ and $(h_{p,2}, m_{p,2}, y_{p,2})$ is the vertex queried at the p-th ($p<i$) query. It means that $y_{i,2}$ has been queried before the i-th query. So

$$\Pr[\mathbf{C}_i'] = \Pr[E_{K_2}(h_{i,2} \oplus m_{i,2}) = y_{j,2}] \quad (j \in \{0,2,3,...,i-1\}) .$$

To get a collision for the hash function, we need $\mathbf{C}_i$ and $\mathbf{C}_i'$ occur simultaneously. So if $\mathbf{C}_i$ occurs when $y=y_{j,1}$, where $j<i$, then $\mathbf{C}_i'$ should satisfy $y' = y_{j,2}$. Let $y_{0,1}$ denote $h_{0,1}$ and $y_{0,2}$ denote $h_{0,2}$, we have

$$\begin{aligned}\Pr[\mathbf{C}_i' \mid \mathbf{C}_i] &= \sum_{j=0}^{i-1} \Pr[E_{K_2}(h_{i,2} \oplus m_{i,2}) = y_{j,2} \mid E_{K_1}(h_{i,1} \oplus m_{i,1}) = y_{j,1}] \\ &= \sum_{j=0}^{i-1} \Pr[E_{K_2}^{-1}(y_{j,2}) = h_{i,2} = E_{K_1}^{-1}(y_{j,1}) = h_{i,1}] \\ &\le \sum_{j=0}^{i-1} \frac{1}{2^r - j} \le \frac{r}{2^{r-1}}\end{aligned}$$

here $r = \mathrm{Min}(k,n)$. Let q denote the total queries of E_{K_1} or $E_{K_1}^{-1}$ and E_{K_2} or $E_{K_2}^{-1}$. Then

$$\begin{aligned}Adv_{coll}^{H'} &= \Pr[\mathbf{C}' \wedge \mathbf{C}] = \Pr[\mathbf{C}' \mid \mathbf{C}] \cdot \Pr[\mathbf{C}] \\ &\le \Pr[\mathbf{C}' \mid \mathbf{C}] \\ &= \sum_{i=0}^{q} \Pr[\mathbf{C}_i' \mid \mathbf{C}_i] \\ &= \frac{q(q+1)}{2^{r(r+1)}} .\end{aligned}$$

□

H' is a double-block-length hash function. We use a function g to transform it into a single-block-length scheme.

Lemma 2. If g is collision resistant, let H denote the whole hash function described in Fig.2. Then the advantage of finding a collision for H is

$$Adv_{coll}^{H} = Min(Adv_{coll}^{H'}, Adv_{coll}^{g}) .$$

Proof: There are two kinds of collisions for H.

1. H' has a collision. In this case, g must have a collision because the input of g is the same. $Adv_{coll}^{H} = Adv_{coll}^{H'}$.

2. The outputs of H' do not include collisions. In this case, $Adv_{coll}^{H} = Adv_{coll}^{g}$.

In conclusion, $Adv_{coll}^{H} = Min(Adv_{coll}^{H'}, Adv_{coll}^{g})$. □

Lemma 3. If g is pre-image resistant, let H denote the whole hash function described in Fig.3. Then the advantage of inverting H is

$$Adv_{inv}^{H} \le Adv_{inv}^{g} .$$

Proof: Given an output of H, We firstly need to find the pre-image of g and then find the pre-image of H'. So if g is pre-image resistant, H is pre-image resistant. □

The following theorem can be concluded from the lemmas mentioned above.

Theorem 1. If g is collision resistant and pre-image resistant, the hash function H described in Fig.2 is a secure hash function.

Proof: It can be concluded from the lemmas. □

In practice, g can be substituted by a secure compression function, for example, one of the secure PGV schemes. In this paper, we use $E_{h_{i,1}}(h_{i,2}) \oplus h_{i,2}$ to substitute g. At first glance, the new scheme looks similar with the wide-pipe schemes proposed in Ref.[22] by Lucks etc. Actually Wide-pipe schemes are proposed to resist the multi-collision attack and the compression functions used in those schemes are secure. But this scheme is not the case. We concentrate on how to construct a new hash function based on block ciphers and improve its efficiency.

3.3 Efficiency of the new scheme

At the first glance, the efficiency of this scheme is not high, because the *rate* of this scheme is only $l/(2l+1)$ which is less than 1/2 where l is the number of message blocks. As we have pointed out not only *rate* but also key

schedule will influence the efficiency of the hash functions based on block ciphers, and the key schedule may be more important. The new scheme in this paper need not re-schedule the keys at every step. It can highly improve the efficiency. We compare the efficiency of the scheme in this paper with a PGV scheme. $E_{h_{i-1}}(m_i)\oplus m_i$ is picked from PGV schemes as the compared scheme. To facilitate the comparison, we use AES-128 to implement both of the block-cipher-based schemes. The operating system is Windows XP SP2, CPU is Intel Celeron 1.7GHz and the compiler is VC++ 6.0. The result is shown in the table as follows.

Table 1 Time collapsed to process messages with different size

	16 (MB)	32 (MB)	48 (MB)	64 (MB)	80 (MB)
New scheme (s)	0.65	1.332	1.963	2.774	3.605
PGV (s)	1.181	2.454	3.545	4.636	5.708

The number from row 2 to row 3 in Table 1 denotes how much time the hash function cost to process the messages with different size. From Table 1 we note, with the increasing of message blocks, more and more key schedules are needed for PGV schemes and the time needed increases faster than our scheme. Our scheme need not key schedules except at the beginning and the last step. So our scheme is more efficient than the PGV scheme although its *rate* is less than that of the PGV scheme.

Figure 3 intuitively describes the efficiency comparison of the two schemes. The thick line denotes the PGV scheme and the thin line denotes our scheme. Actually when the message is small, PGV scheme is more efficient. But if the message has more than three blocks, then our scheme is more efficient.

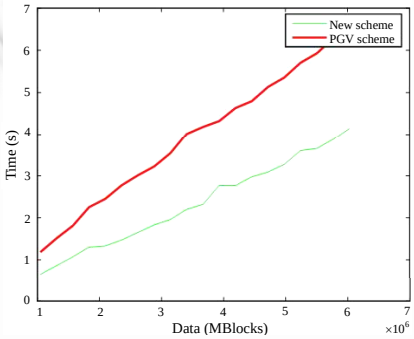

Fig.3 New scheme in this paper

It can be concluded from above that key schedule is a more important factor than *rate* which affects the efficiency of block-cipher-based hash functions. But block ciphers are not designed for hash functions, they are used in encryption and MAC where keys are kept secret and need not re-schedule. In the experiment, we find if the *rate* of a block-cipher-based hash function is 1 and it needs not re-schedule the keys, it is as efficient as SHA-1, for example the TCH scheme. Unfortunately, it has been proved not secure. The efficiency of the scheme in Fig.3 can be improved more higher. For the two branches are parallel, They can be computed simultaneously on the multi-CPU computer.

A well known method to construct a MAC with hash functions is to include a secret key as part of the input of a hash function such as $MAC(K,M)=H(IV,K\|M)$. If the hash function H is based on MD method, there exists an extension attack to forge a MAC without the secret key. Given $MAC(K,M)=H(IV,K\|M)$, one can forge

$$MAC(K,M\|y)=H(H(IV,K\|M),y)$$

Coron etc. modified MD method to resist this attack. The new scheme in this paper can also resist this attack, because the chain values of it are double-block-length while its output is single-block-length.

3.4 What kind of compression function can be used

The compression function used in the new scheme does not have to be secure. In PGV schemes, there are many insecure compression functions. Which can be used in the new scheme? Actually the compression function used in this paper is a special instance of the 36-th scheme in PGV schemes. We give a table to describe which one can be used in this scheme.

Table 2 Classification table for some PGV schemes

ind	*j*	h_i =	Secure
1	4	$E_v(m_i)\oplus v$	N
2	8	$E_v(m_i)\oplus m_i$	N
3	20	$E_v(h_{i-1})\oplus v$	N
4	28	$E_v(h_{i-1})\oplus h_{i-1}$	N
5	52	$E_v(v)\oplus v$	N
6	56	$E_v(v)\oplus m_i$	N
7	60	$E_v(v)\oplus h_{i-1}$	N
8	12	$E_v(m_i)\oplus h_{i-1}$	N
9	16	$E_v(m_i)\oplus m_i\oplus h_{i-1}$	N
10	64	$E_v(v)\oplus m_i\oplus h_{i-1}$	N
11	24	$E_v(h_{i-1})\oplus m_i$	Y
12	32	$E_v(h_{i-1})\oplus h_{i-1}\oplus m_i$	Y
13	36	$E_v(h_{i-1}\oplus m_i)\oplus v$	Y
14	40	$E_v(h_{i-1}\oplus m_i)\oplus m_i$	Y
15	44	$E_v(h_{i-1}\oplus m_i)\oplus h_{i-1}$	Y
16	48	$E_v(h_{i-1}\oplus m_i)\oplus h_{i-1}\oplus m_i$	Y

Here *ind* is the index in this paper, *j* is its corresponding index in Ref.[10], N and Y denote insecure and secure respectively. For example the 8-th scheme in this table, its output is independent of the sequence of the message blocks. If a hash function H is construct with this compression function, then

$$H(IV,(m_1,m_2,...,m_l))=H(IV,(m_{i_1},m_{i_2},...,m_{i_2}))$$
$$=E_v(m_1)\oplus E_v(m_2)...E_v(m_l)\oplus IV$$

where $\{i_1,i_2,...,i_l\}$ is a permutation on {1,2,…,*l*}. If it is used in our scheme, it is easy to find the collisions for the two branches simultaneously. The 9-th scheme and 10-th scheme are the same as the 8-th. The first 7 schemes do not include both chain values and message blocks. They are also not secure. Liskov proposed a method to construct an ideal hash function with the weak compression functions[23]. This can be described as follows.

$$Mid=f_0(m_l,...,f_0(m_3,f_0(m_2,f_0(m_1,IV)))...)$$
$$Out=f_1(m_1,...,f_1(m_{l-2},f_1(m_{l-1},f_1(m_l,IV)))...)$$

f_0 and f_1 are two weak compression functions. Weak compression function was defined in Ref.[23] as follows.

Definition 8 (**Weak compression function**). Let there be two oracles f^{-1} and f^*. On querying f^{-1} on input (*x*,*z*), the oracle returns a random value *y* such that *f*(*x*,*y*) = *z*. On querying f^* on input (*y*,*z*), the oracle return a random value *x* such that *f*(*x*,*y*) = *z*. *f* is a weak compression function.

The security of the 9-th scheme in this table is stronger than weak compression functions, but if it is used to construct Liskov's scheme, it is not secure. Let $f_0=E_{v_0}(m_i)\oplus m_i\oplus h_{i-1}^0$ and $f_1=E_{v_1}(m_i)\oplus m_i\oplus h_{i-1}^1$. Let H_W denote the hash function constructed with them in Liskov's way and $M=m_1,m_2,...,m_l$. Then there exists $M'=m_{i_1},m_{i_2},...,m_{i_l}$ such that $H_W(M)=H_W(M')$, where $(m_{i_1},m_{i_2},...,m_{i_l})$ is a permutation on $m_1,m_2,...,m_l$. Similarly, for the 8-th scheme, It is the same case. The other schemes marked by 'N' in Table 2 are weaker than weak compression function. So the compression function for Liskov's scheme should be carefully selected. It does not indicate that Liskov's scheme is not good. In fact, Liskov's scheme gave a good idea to construct a hash function with an insecure compression function.

4 Conclusion

In this paper, we proposed a new hash function based on block ciphers and analyzed its security. In practice, the scheme in this paper is easily implemented. Firstly one selects a message block as the first key K_1, then one uses a function s to transform the message block to K_2 where $K_1 \neq K_2$. Function s is easily implemented, for example $s(x) = x \oplus v$ where v is a non-zero constant. Although the *rate* of our scheme is not high, the efficiency of it is still higher than those PGV schemes. From the experiment, we find that key schedule is a more important factor that affects the efficiency of block-cipher-based hash functions, especially when the message is very large. Unlike the ordinary iterated hash functions, this hash function is constructed with an insecure compression function. We analyzed the insecure PGV schemes and divided them into two groups by the security. We found that some of these insecure compression functions cannot be used to construct our scheme.

References:

[1] Damgård I. A design principle for hash functions. In: Brassard G, ed. CRYPTO 1989. LNCS 435, Heidelberg: Springer-Verlag, 1990. 416−427.

[2] Merkle R. One way hash functions and DES. In: Brassard G, ed. CRYPTO 1989. LNCS 435, Heidelberg: Springer-Verlag, 1990. 428−446.

[3] Lai X, Massey J. Hash functions based on block ciphers. In: Ruppel R, ed. EUROCRYPT 1992. LNCS 658, Heidelberg: Springer-Verlag, 1993. 55−70.

[4] Rivest R. The MD4 message-digest algorithm. In: Menezes A, Vanstone S, eds. CRYPTO 1990. LNCS 537, Heidelberg: Springer-Verlag, 1991. 303−311.

[5] Rivest R. The MD5 message-digest algorithm. Internet Activity Board, Internet Privacy Task Force, RFC1321, 1992.

[6] FIPS 180-1. Secure Hash Standard, Federal Information Processing Standard, Publication 180-1. NIST, 1995.

[7] FIPS 180-2. Secure Hash Standard, Federal Information Processing Standard, Publication 180-2. NIST, 2003.

[8] ISO/IEC 10118-2. Hash functions using an n-bit block cipher, ISO/IEC 10118-2. ISO, 2000.

[9] Preneel B, Govaerts R, Vandewalle J. Hash functions based on block ciphers: A synthetic approach. In: Stinson D, ed. CRYPTO 1993. LNCS 773, Heidelberg: Springer-Verlag, 1994. 368−378.

[10] Black J, Rogaway P, Shrimpton T. Black-box analysis of the block-cipher based hash function constructions from PGV. In: Yung M, ed. CRYPTO 2002., LNCS 2442, Heidelberg: Springer-Verlag, 2002. 320−335.

[11] Hohl W, Lai X, Meier T, Waldvogel C. Security of iterated hash function based on block ciphers. In: Stinson D, ed. CRYPTO 1993. LNCS 773, Heidelberg: Springer-Verlag, 1993. 379−390.

[12] Preneel B, Bosselaers A, Govaerts R, Vandewalle J, Collision-Free hash functions based on block cipher algorithms. In: Rapsey A, ed. ICCST 1989. Zurich: IEEE, 1989. 203−210.

[13] Brown L, Pieprzyk J, Seberry J. LOKI—A cryptographic primitive for authentication and secrecy applications. In: Seberry J, Pieprzyk J, eds. AUSCRYPT 1990. LNCS 453, Heidelberg: Springer-Verlag, 1990. 229−236.

[14] Knudsen L, Lai X, Preneel B. Attacks on fast double block length hash functions. Journal of Cryptology, 1998,11(1):59−72.

[15] Hirose S. Provably secure double-block-length hash functions in a black-box model. In: Park C, Chee S, eds. ICISC 2004. LNCS 3506, Heidelberg: Springer-Verlag, 2005. 330−342.

[16] Nandi M, Lee W, Sakurai K, Lee S. Security analysis of a 2/3-rate double length compression function in the black-box model. In: Gilbert H, Handschuh H, eds. FSE 2005. LNCS 3557, Heidelberg: Springer-Verlag, 2005. 243−254.

[17] Knudsen L, Muller F. Some attacks against a double length hash proposal. In: Roy B, ed. ASIACRYPT 2005 LNCS 3788, Heidelberg: Springer-Verlag, 2005. 462−473.

[18] Liskov M, Rivest R, Wagner D. Tweakable block ciphers. In: Yung M, ed. CRYPTO 2002. LNCS 2442, Heidelberg: Springer-Verlag, 2002. 31−46.

[19] Black J, Cochran M, Shrimpton T. On the impossibility of highly-efficient block-cipher-based hash functions. In: Cramer R, ed. EUROCRYPT 2005. LNCS 3494, Heidelberg: Springer-Verlag, 2005. 526−541.

[20] Shannon C. Communication theory of secrecy systems. Bell Systems Technical Journal, 1949,28(4):656−715.

[21] Rogaway P, Shrimpton T. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In: Roy B, Meier W, eds. FSE 2004. LNCS 3017, Heidelberg: Springer-Verlag, 2004. 371−388.

[22] Lucks S. A Failure-Friendly design principle for hash functions. In: Roy B, ed. ASIACRYPT 2005. LNCS 3788, Heidelberg: Springer-Verlag, 2005. 474−494.

[23] Liskov M. Constructing an ideal hash function from weak ideal compression functions. In: Biham E, Youssef A, eds. Workshop Record of SAC 2006. 2006. 331−349.

LIN Pin, a doctor candidate of the Institute of Software, the Chinese Academy of Sciences, majoring in hash functions.

WU Chuan-Kun, a professor and Ph.D. supervisor at the Institute of Software, the Chinese Academy of Sciences and a CCF senior member, majoring in cryptology and information security.

WU Wen-Ling, a professor and Ph.D. supervisor at the Institute of Software, the Chinese Academy of Sciences, majoring in cryptology and information security.