Preimage Attacks on Reduced DHA-256*

Jinmin Zhong and Xuejia Lai

Department of Computer Science and Engineering
Shanghai Jiao Tong University
Shanghai, 200240 China
zjm_new@sjtu.edu.cn

DHA-256 (Double Hash Algorithm) was proposed at the Cryptographic Hash Workshop hosted by NIST in November 2005. DHA-256 is a dedicated hash function with output length of 256 bits and 64 steps of operations designed to enhance SHA-256 security. In this paper, we show two attacks on reduced DHA-256. The first attack finds one-block second preimage and preimage of 26-step DHA-256 with time complexity of $2^{223.82}$ compression function operations and 2^{32} x 9 words memory. The second attack finds pseudo-preimage and preimage of 35-step DHA-256 with time complexity of $2^{239.63}$ and $2^{248.82}$ compression function operations, respectively, and 2^{16} x 11 words memory. To the best of our knowledge, this is the first paper that analyzes second preimage resistance and preimage resistance of DHA-256.

Keywords: DHA-256, meet-in-the-middle, second preimage, preimage, hash function

1. INTRODUCTION

Hash function is an important cryptographic primitive. It is applied in many cryptographic applications, such as digital signature and message authentication. Since 2004, Wang et al. has made tremendous progress in the cryptanalysis of hash functions [24, 25, 26, 27] by showing theoretical attacks on SHA-1. NIST plans to replace SHA-1 with SHA-2 [1]. The competition for SHA-3 is due to recent advances in cryptanalysis of hash functions [22].

The meet-in-the-middle technique is used to construct preimage attacks on MD4 [4, 14], MD5 [4, 5, 17, 18], reduced RIPEMD [19, 23], HAVAL [5, 16], reduced SHA-0/1 [2] and reduced HAS-160 [20]. Constructing the second preimage using hash collision was presented in [28]. Reversing the inversion problem and P^3 graphs techniques are applied to analyze the preimage resistance of reduced SHA-0/1 [6]. The most step number of current collision attack on reduced SHA-2 is 24 [10, 15]. The first result about preimage attack on 24-step SHA-256 was proposed in [11]. Recently, preimage attack on

43-step SHA-256 and 46-step SHA-512 was presented in [3, 9, 21].

DHA-256 was proposed in the first cryptographic hash workshop hosted by NIST in 2005 by Jesang Lee, et al. [12]. The compression function of DHA-256 has 64 step operations and outputs length 256 bits, and uses the two same message words to update two intermediate words every step. In order to enhance the security of SHA-256, the designers claimed that DHA-256 uses the same resource as SHA-2 for the message expansion and step function, but it is more secure than SHA-2 against known attack methods, especially Wang et al.'s attack [12]. IAIK Krypto Group presented a preliminary analysis on the message expansion of DHA-256 and gave a 9-step local collision for DHA-256 in 2005 [8]. So far, it is the only paper about the analysis of DHA-256.

Our Results. We show two preimage attacks on reduced DHA-256. A summary of our results is shown in Table 1.

The first attack is a one-block preimage attack on 26-step DHA-256. This attack is based upon strategy of Isobe and Shibutani [11] and uses two-way message expansion technique proposed in [3, 9].

We present a 2-word initial structure with two message words, which is a variant of so-called initial structure technique shown in [3, 2, 18, 21] with only one message word in every step. We propose pseudo-preimage and preimage attack on 35-step DHA-256. This attack is based on the general framework proposed by Sasaki and Aoki and uses two-way message expansion and message compensation technique proposed in [3, 9].

In addition, we find that the matrix of the inverse function σ_1^{-1} of SHA-2 showed in [11] is not correct, and we provide a correct one and the corresponding program code in appendix B, A respectively.

Table 1. Preimage attacks on reduced DHA-256.

	Attack Type	Time Complexity	Memory Complexity
26-step DHA-256 (one block)	(Second)Preimage	$2^{223.82}$	$2^{32} \times 9$ words
35-step DHA-256	Pseudo-preimage	$2^{239.63}$	$2^{16} \times 11$ words
	Preimage	$2^{248.82}$	

2. PRELIMINARIES

2.1. Specification of DHA-256

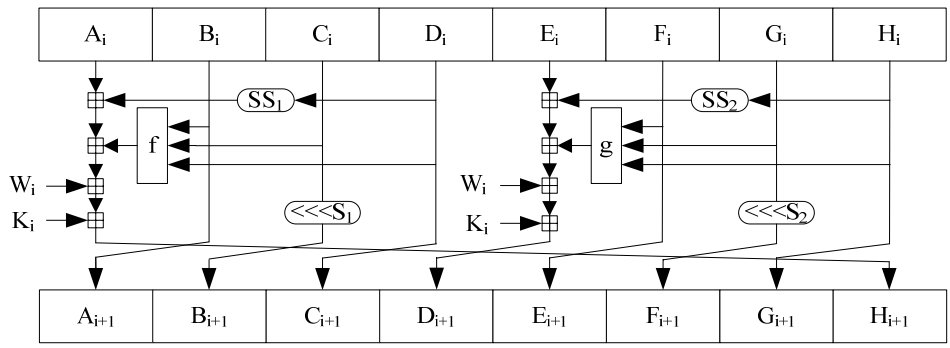

Fig. 1. Step Operation for DHA-256.

DHA-256 is a dedicated hash function. The structure of DHA-256 is similar to that of SHA-256. But their message expansion functions are different from each other. Moreover, it is more important that the two same message words are used in the state update function of DHA-256, while the only one message word is used in that of SHA-256. So the hash function is called double hash algorithm (DHA) because of the two message words. The step operation for DHA-256 is depicted in Figure 1.

The initial values, the input block length，the padding rule and the constants of DHA-256 are the same as those of SHA-256.

The Boolean functions of DHA-256 are as follows.

$$f(x, y, z) = (x \wedge y) \vee (\neg x \wedge z)$$
$$g(x, y, z) = (x \wedge y) \vee (y \wedge z) \vee (x \wedge z)$$

The shift rotations of DHA-256 are as follows.

$$SS_1(x) = x \oplus (x <<< 11) \oplus (x <<< 25),\ SS_2(x) = x \oplus (x <<< 19) \oplus (x <<< 29)$$
$$S_1(x) = x <<< 17,\ S_2(x) = x <<< 2$$

We divide the 512 bits message block into 16 message words, $M_0||M_1||\ldots||M_{15}$. The algorithm of message expansion is

$$W_i = \begin{cases} M_i, & 0 \le i \le 15 \\ \sigma_1(W_{i-1}) + W_{i-9} + \sigma_2(W_{i-15}) + W_{i-16}, & 16 \le i \le 63 \end{cases}$$

$$\sigma_1(x) = x \oplus (x <<< 7) \oplus (x <<< 22),\ \sigma_2(x) = x \oplus (x <<< 13) \oplus (x <<< 27)\ .$$

The step operation of DHA-256 is

$$(1)A_{i+1} = B_i,\ (2)B_{i+1} = C_i \lll S_1,\ (3)C_{i+1} = D_i,\ (4)D_{i+1} = E_i + SS_2(H_i) + g(F_i, G_i, H_i) + W_i + K_i,$$
$$(5)E_{i+1} = F_i,\ (6)F_{i+1} = G_i \lll S_2,\ (7)G_{i+1} = H_i,\ (8)H_{i+1} = A_i + SS_1(D_i) + f(B_i, C_i, D_i) + W_i + K_i.$$

The output of the compression function of DHA-256 is

$$(A_0+A_{64},\ B_0+B_{64},\ C_0+C_{64},\ D_0+D_{64},\ E_0+E_{64},\ F_0+F_{64},\ G_0+G_{64},\ H_0+H_{64})\ .$$

2.2. Related techniques

The following are some important techniques used to analyze the preimage resistance of MD4-family, HAS-160 and DHA-256.

2.2.1. Meet-in-the-middle attack (MITM)

The meet-in-the-middle attack is a type of birthday attack and makes use of a space-time tradeoff. The balanced meet-in-the-middle attack appeared in [7], and the unbalanced one was proposed first in [13]. When it is applied to a hash function, the compression function computes forward to the given step and gets a set of results for intermediate chaining, and then compression function computes backward to the same given step and gets another set of results for the same intermediate chaining. The two sets of results are matched to find an equal value according to the birthday attack rule. So the two sets of results must be independent of each other in order to satisfy the birthday attack rule. The key issue for the meet-in-the-middle attack is to find the two message parts independent of each other.

2.2.2. Converting pseudo-preimage attack into preimage attack

The method of converting pseudo-preimage attack into preimage attack was proposed first in [13]. The initial chaining values of pseudo-preimage are not the fixed IV, so a message block is needed to hash and connect the fixed IV with the initial chaining value of pseudo-preimage. The method of the birthday attack is used to search the proper message block. Assume that it takes 2^k complexity to produce a pseudo-preimage. $2^{(n-k)/2}$ pseudo-preimages cost 2^k x $2^{(n-k)/2}$ complexity. It needs $2^{(n+k)/2}$ complexity to compute the hash values of $2^{(n+k)/2}$ one-block messages. Then, $2^{(n+k)/2}$ hash values are compared with the initial chaining values of $2^{(n-k)/2}$ pseudo-preimages, and thus $2^{(n+k)/2+(n-k)/2} = 2^n$ pairs are compared. A pair will succeed in matching according to birthday attack rule with high probability. The total complexity is 2^k x $2^{(n-k)/2} + 2^{(n+k)/2} = 2^{1+(n+k)/2}$. Thus, if k is less than n-2, it will succeed in transforming pseudo-preimage attack into preimage attack.

2.2.3. Splice-and-cut technique

The hash values equal the values to which the initial values are added for the final

chaining values in the Davies-Meyer mode. Splice-and-cut technique proposed first in [4] regards the first step and the last step as consecutive steps. So the neutral words and independent chunk can be searched in all of the range.

2.2.4. Partial-matching technique, Partial-fixing technique and Indirect-partial-matching technique

The state update function only updates a chaining word for MD4-family hash function (updates two chaining words for SHA-2) and the corresponding chaining words of the next step equal to or are easy to compute based on other chaining words. For example, two words out of eight chaining words are updated every step in SHA-2 and other six chaining words equal to the corresponding words of the next step and one chaining word of the next step can be derived from the eight chaining words. If eight chaining words in the same step are known, seven chaining words in the next step will be known for SHA-2 while we do not need to know the corresponding message words. In a similar way, there is a chaining word that can be derived from eight words in each current step, in every step of the succeeding seven steps. So we only need to match the corresponding chaining words in the meet-in-the-middle attack, while we do not care about the seven message words. This property helps us to find the neutral words and independent chunks. The technique is so-called partial-matching technique proposed first in [4]. The state update function of SHA-2 is different from that of DHA-256. The main difference is that the state update function of DHA-256 includes two message words and updates two intermediate words at the same time, while that of SHA-2 only includes one message word and updates two intermediate words. Thus, the partial-matching technique in DHA-256 can match at most three steps while that in SHA-2 can match up to seven steps.

The partial-fixing technique proposed first in [4] is essentially the partial-matching technique. The number of matched bits in the partial-matching technique is the number of bits in a chaining word at least, while the one in the partial-fixing technique can be several bits, and any bit. During the meet-in-the-middle attack, a chunk meets the opposite neutral word and has to stop computing the intermediate values. If the part values of the opposite neutral word are fixed and known, the chunk can continue to compute the intermediate values with the known part values of the opposite neutral word and thus we can attack more steps.

Indirect-partial-matching proposed first in [3, 9] is an extension of partial-matching.

The computation of the match point can be decomposed the independent computation of the function of the two neutral words. Indirect-partial-matching makes use of the property so as to transfer the problem of the computation of the match point into the problem of the independent computation of the function of the two neutral words. Indirect-partial-matching can get two more steps for SHA-2.

2.2.5. Initial structure

If the two independent chunks overlap at the beginning of the meet-in-the-middle attack, we will meet the problem which is how to compute forward and backward in these overlapped steps, respectively. Initial structure proposed first in [18] is designed to solve this problem. Initial structure with deterministic way for SHA-0 introduced in [2] can reduce the memory requirement by a factor of 2^{32}. The 4-step initial structure for SHA-2 proposed in [3, 9] consumes a lot of message freedom. Until now all of proposed initial structures only contain one message word every state update step.

2.2.6. Two-way expansion, Message modification and Message compensation

Any consecutive 16 message words can determine other message words for SHA-2 in two directions. This property helps to find the longer chunks independent of each other, and the neutral words. The two-way expansion proposed in [3, 9] makes use of the above property.

The message modification introduced in [11] is used to construct two independent chunks in the message expansion of SHA-2. Note that the message modification technique is different from the one proposed in [24].

The message compensation proposed in [3, 9] is used to attain the goal that the change of the neutral words in an independent chunk does not affect the message words in another independent chunk. The message compensation is similar to the message modification.

2.2.7. Finding kernel and neutral words technique

Finding kernel and neutral words technique proposed in [2] is used to analyze the linear message schedule of SHA-0/1 and to find the two independent chunks and the neutral words for SHA-0 (bits for SHA-1 instead of words). The strategy uses a matrix to represent the expanded message of SHA-0/1. The message expansion functions of SHA-2 and DHA-256 are not linear and are different from those of SHA-0 and SHA-1.

3. ONE-BLOCK (SECOND) PREIMAGE ATTACK ON 26-STEP DHA-256

A one-block preimage attack on 26-step DHA-256 is also a one-block second preimage attack on 26-step DHA-256. We combine the strategy of analysis of SHA-2 in [11] with the two-way message expansion technique, and then apply them to DHA-256. Only the 24-step SHA-2 in [11] can be analyzed. If we use the same strategy in [11], only the 24-step DHA-256 can be analyzed. So we extend to the 26-step DHA-256 using the two-way expansion technique.

3.1. Construct two message chunks independent of each other

The values of message words W_i, $i \in \{0, 1, \ldots, 15\}$ determine those of W_j, $j \in \{16, 17, \ldots, 25\}$ according to the rule of message expansion of DHA-256.

$$W_{16} \leftarrow \sigma_1(W_{15}) + W_7 + const1\ (const1 \leftarrow \sigma_2(W_1) + W_0) \quad (1)$$

$$W_{17} \leftarrow \sigma_1(W_{16}) + W_8 + const2\ (const2 \leftarrow \sigma_2(W_2) + W_1) \quad (2)$$

$$W_{18} \leftarrow \sigma_1(W_{17}) + W_9 + const3\ (const3 \leftarrow \sigma_2(W_3) + W_2) \quad (3)$$

$$W_{19} \leftarrow \sigma_1(W_{18}) + W_{10} + const4\ (const4 \leftarrow \sigma_2(W_4) + W_3) \quad (4)$$

$$W_{20} \leftarrow \sigma_1(W_{19}) + \sigma_2(W_5) + const5\ (const5 \leftarrow W_{11} + W_4) \quad (5)$$

$$W_{21} \leftarrow \sigma_1(W_{20}) + W_{12} + \sigma_2(W_6) + W_5 \quad (6)$$

$$W_{22} \leftarrow \sigma_1(W_{21}) + W_{13} + \sigma_2(W_7) + W_6 \quad (7)$$

$$W_{23} \leftarrow \sigma_1(W_{22}) + W_{14} + \sigma_2(W_8) + W_7 \quad (8)$$

$$W_{24} \leftarrow \sigma_1(W_{23}) + W_{15} + \sigma_2(W_9) + W_8 \quad (9)$$

$$W_{25} \leftarrow \sigma_1(W_{24}) + W_{16} + \sigma_2(W_{10}) + W_9 \quad (10)$$

The following description shows that the forward computation and the backward computation can produce the two sets of intermediate chaining values independent of each other to match using the rule of birthday attack.

The neutral word of forward computation is W_{11}, which is used to compute W_{20} in equation (5) and is not used in other expansion message words before step 26 of DHA-256. So W_{20} will change with changes in W_{11}. In order to keep W_{20} unchanged, we use W_4 to absorb the change of W_{11} using the message modification technique proposed in [11], that is to say, the sum of W_{11} and W_4 (const5), does not change. Hence the value of W_{20} does not change with changes in W_{11}. The change of W_4 can be absorbed by the

change of W_3 in equation (4) in a similar way. The same analysis is true for W_1 and W_0.

The neutral word of backward computation is W_{25}. The change of W_{25} is absorbed by the change of W_{24} in equation (10) so that other message words do not change in equation (10). The change of W_{21} can be absorbed by the change of W_{12} in equation (6) as the same reason. W_{12} is not used in other expansion message words before step 26, thereby the change of W_{12} will not affect other message words. Moreover, W_{12}, W_{13} and W_{14} will be neglected through the partial-matching technique.

Because the initial values can be any value, this attack directly produces a one-block preimage and a one-block second preimage.

This attack is also easy to extend to a one-block (second) preimage attack on 27-step DHA-256, which needs to deal with the most significant bit of message word W_{13} to satisfy the padding rule. The two message chunks independent of each other are $\{W_0, W_1,\ldots,W_{12}\}$ and $\{W_{13}, W_{14},\ldots,W_{26}\}$, and the corresponding neutral words are W_0 or W_{12} and W_{13} or W_{26}.

We discuss the two-block preimage attack on 35-step DHA-256 in Section 4.

3.2. Attack procedure

1. Initialization

(a) Set the values of W_i, $i \in \{13, 14, 15\}$ to satisfy the padding rule.

(b) Select the values of W_i, $i \in \{5, 6, 7, 8, 9, 10\}$ randomly.

(c) Select the values of const1, const2, const3, const4 and const5 randomly.

(d) Compute the values of W_i, $i \in \{16, 17, 18, 19, 20\}$ using equation (1)~(5), respectively.

2. Forward Computation

For all possible value of W_{11} (32 free bits in total),

(a) Compute the following equations,

$$W_4 \leftarrow const5 - W_{11}, W_3 \leftarrow const4 - \sigma_2(W_4), W_2 \leftarrow const3 - \sigma_2(W_3),$$
$$W_1 \leftarrow const2 - \sigma_2(W_2), W_0 \leftarrow const1 - \sigma_2(W_1)$$

(b) Compute $p_{j+1} \leftarrow R_j(p_j, W_j)$ for j=0, 1, ..., 11. Store the values of (p_{12}, W_{11}) into the list L.

3. Backward Computation and Comparison

For all possible values of W_{25} (32 free bits in total),

(a) Compute the following equations (The program code about computing σ_1^{-1} and the value of σ_1^{-1} is in appendix A and C),

$$W_{24} \leftarrow \sigma_1^{-1}(W_{25} - W_{16} - \sigma_2(W_{10}) - W_9), W_{23} \leftarrow \sigma_1^{-1}(W_{24} - W_{15} - \sigma_2(W_9) - W_8),$$
$$W_{22} \leftarrow \sigma_1^{-1}(W_{23} - W_{14} - \sigma_2(W_8) - W_7), W_{21} \leftarrow \sigma_1^{-1}(W_{22} - W_{13} - \sigma_2(W_7) - W_6),$$
$$W_{12} \leftarrow W_{21} - \sigma_1(W_{20}) - \sigma_2(W_6) - W_5$$

(b) Compute $p_j \leftarrow R_j^{-1}(p_{j+1}, W_j)$ for j=25, 24, ..., 15. Compare D_{12}, H_{12} in the list L with A_{15}>>>17, E_{15}>>>2.

(c) If matched, compute p_{14} using p_{15} and W_{14}. Compare C_{12}, G_{12} in the list L with A_{14}>>>17, E_{14}>>>2.

(d) If matched, compute p_{13} using p_{14} and W_{13}. Compare B_{12}, F_{12} in the list L with A_{13}, E_{13}.

(e) If matched, compute p_{12} and W_{12}. Compare A_{12}, D_{12} in the list L with A_{12}, E_{12}.

(f) If matched, return (W_0, W_1, …, W_{15}) preimage of 26-step DHA-256, otherwise, repeat the attack procedure.

4. Complexity analysis

The complexity analysis of the one-block (second) preimage attack on 26-step DHA-256 is as follows.

1. The cost of initialization compared to that of other steps is negligible. (ME represents the computation of message expansion and CF represents the computation of compression function. We use the same notations hereafter.)

2. For forward computation, the complexity is about $2^{32} \cdot \frac{2}{26}$ ME in step 2a. The complexity is $2^{32} \cdot \frac{12}{26}$ CF in step 2b. The memory complexity is $2^{32} \cdot 9$ words.

3. For backward computation, the complexity is $2^{32} \cdot \frac{5}{26}$ ME in step 3a. The complexity is $2^{32} \cdot \frac{11}{26}$ CF in step 3b. The cost of comparison compared to that of other steps is negligible. 2^{64} $(= 2^{32} \cdot 2^{32})$ pairs will take part in comparison. The number of bits for matching is 64 in total. If matched, 2^0 $(= 2^{64} \cdot 2^{-64})$ pair remains, and thus, the complexity is negligible in step 3c~3f. Hereto, the complexity is $2^{32} \cdot \frac{7}{26}$ $(= 2^{32} \cdot (\frac{2}{26} + \frac{5}{26}))$ ME and $2^{32} \cdot \frac{23}{26}$ $(= 2^{32} \cdot (\frac{12}{26} + \frac{11}{26}))$ CF in total. It is about $2^{32} \cdot \frac{23}{26} \approx 2^{31.82}$ 26-step DHA-256 opera-

tions. The rate of success to find a one-block preimage of 26-step DHA-256 is $2^{-192}(=2^{64}\bullet 2^{-256})$. So it needs to repeat 2^{192} times.

Hence, the complexity for finding a one-block preimage of 26-step DHA-256 is $2^{223.82}$ $(=2^{192}\bullet 2^{31.82})$ 26-step DHA-256 operations. The memory complexity is $2^{32}\cdot 9$ words.

4. PREIMAGE ATTACK ON 35-STEP DHA-256

4.1. Construct two message chunks independent of each other

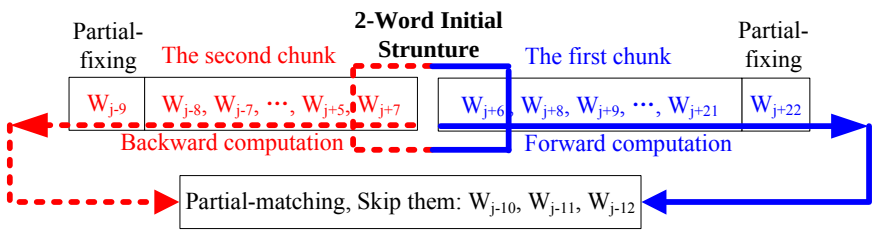

Fig. 2. The roles of message words in the meet-in-the-middle attack on 35-step DHA-256.

How to find two independent message chunks is the key issue of meet-in-the-middle attack. The message expansion rule of DHA-256 is similar to that of SHA-2 and it is also a bijective mapping. The consecutive 16 message words out of 64 message words of DHA-256 can compute other message words determinately. This property allows us to use two-way message expansion technique proposed in [3, 9] to find the two independent message chunks with long steps. We select $\mathbf{W_{j+6}}$ and $\mathbf{W_{j+7}}$ as the neutral words by our manual attempts, and thus the two independent message chunks are $\{W_{j-7}, W_{j-6}, \ldots, \mathbf{W_{j+6}}\}$ and $\{\mathbf{W_{j+7}}, W_{j+8}, \ldots, W_{j+20}\}$.

In order to extend the independent message chunks, we exchange the two neutral message words through a 2-word initial structure which will be explained in next subsection. This makes us to get 2 more steps. The two longer independent message chunks are $\{W_{j-8}, W_{j-7}, \ldots, \mathbf{W_{j+7}}\}$ and $\{\mathbf{W_{j+6}}, W_{j+8}, \ldots, W_{j+21}\}$, and the extended corresponding message words are W_{j-8} and W_{j+21} respectively. But W_{j+16} in equation (11) is derived from W_{j+7} which is the neutral word. Then, the value of W_{j+16} will be affected by W_{j+7} when the value of W_{j+7} changes. Such will break the rule of meet-in-the-middle. We use message compensation technique proposed in [3, 9] to solve the problem. Let $W_j = -W_{j+7}$,

such that the change of W_{j+7} can be absorbed by the corresponding change of W_j and will not affect the value of W_{j+16} in (11). The change of W_j will not break the rule of meet-in-the-middle, because W_j is not a neutral word and is in the same message chunk with W_{j+7}. The similar operations for W_{j+6} and W_{j+15} in another message chunk are carried out for the same reason. Therefore, we make $W_{j+15} = W_{j+6}$ in equation (18).

The first chunk $\{W_{j+6}, W_{j+8}, W_{j+9}, ..., W_{j+21}\}$ is used to compute forward.

$$W_{j+16} = \sigma_1(W_{j+15}) + W_{j+7} + \sigma_2(W_{j+1}) + W_j \quad (\Rightarrow W_j = -W_{j+7}) \tag{11}$$

$$W_{j+17} = \sigma_1(W_{j+16}) + W_{j+8} + \sigma_2(W_{j+2}) + W_{j+1} \tag{12}$$

$$W_{j+18} = \sigma_1(W_{j+17}) + W_{j+9} + \sigma_2(W_{j+3}) + W_{j+2} \tag{13}$$

$$W_{j+19} = \sigma_1(W_{j+18}) + W_{j+10} + \sigma_2(W_{j+4}) + W_{j+3} \tag{14}$$

$$W_{j+20} = \sigma_1(W_{j+19}) + W_{j+11} + \sigma_2(W_{j+5}) + W_{j+4} \tag{15}$$

$$W_{j+21} = \sigma_1(W_{j+20}) + W_{j+12} + \sigma_2(W_{j+6}) + W_{j+5} \tag{16}$$

$$W_{j+22} = \sigma_1(W_{j+21}) + W_{j+13} + \sigma_2(W_{j+7}) + W_{j+6} \quad (Partial-fixing) \tag{17}$$

The second chunk $\{W_{j+7}, W_{j+5}, W_{j+4}, ..., W_{j-8}\}$ is used to compute backward.

$$W_{j-1} = W_{j+15} - \sigma_1(W_{j+14}) - W_{j+6} - \sigma_2(W_j) \quad (\Rightarrow W_{j+15} = W_{j+6}) \tag{18}$$

$$W_{j-2} = W_{j+14} - \sigma_1(W_{j+13}) - W_{j+5} - \sigma_2(W_{j-1}) \tag{19}$$

$$W_{j-3} = W_{j+13} - \sigma_1(W_{j+12}) - W_{j+4} - \sigma_2(W_{j-2}) \tag{20}$$

$$W_{j-4} = W_{j+12} - \sigma_1(W_{j+11}) - W_{j+3} - \sigma_2(W_{j-3}) \tag{21}$$

$$W_{j-5} = W_{j+11} - \sigma_1(W_{j+10}) - W_{j+2} - \sigma_2(W_{j-4}) \tag{22}$$

$$W_{j-6} = W_{j+10} - \sigma_1(W_{j+9}) - W_{j+1} - \sigma_2(W_{j-5}) \tag{23}$$

$$W_{j-7} = W_{j+9} - \sigma_1(W_{j+8}) - W_j - \sigma_2(W_{j-6}) \tag{24}$$

$$W_{j-8} = W_{j+8} - \sigma_1(W_{j+7}) - W_{j-1} - \sigma_2(W_{j-7}) \tag{25}$$

$$W_{j-9} = W_{j+7} - \sigma_1(W_{j+6}) - W_{j-2} - \sigma_2(W_{j-8}) \quad (Partial-fixing) \tag{26}$$

$$W_{j-10} = W_{j+6} - \sigma_1(W_{j+5}) - W_{j-3} - \sigma_2(W_{j-9}) \quad (Partial-matching) \tag{27}$$

$$W_{j-11} = W_{j+5} - \sigma_1(W_{j+4}) - W_{j-4} - \sigma_2(W_{j-10}) \quad (Partial-matching) \tag{28}$$

$$W_{j-12} = W_{j+4} - \sigma_1(W_{j+3}) - W_{j-5} - \sigma_2(W_{j-11}) \quad (Partial-matching) \tag{29}$$

4.2. The description of 2-word initial structure

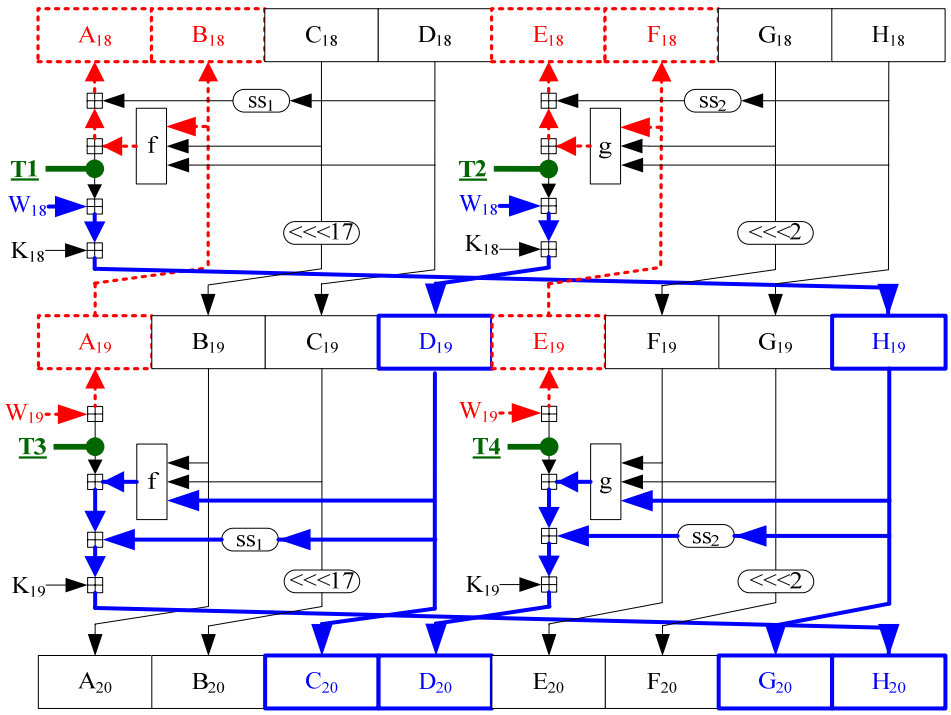

Fig. 3. Description of 2-word initial structure for 35-step DHA-256.

The section shows how to construct the 2-word initial structure.

The intention of the initial structure is to prevent the change of the opposed neutral word from the computation of the corresponding intermediate state values. For example, the change of the neutral word W_{18} is not allowed to affect the computation of the intermediate state values A_{18}, B_{18}, …, H_{18} (i.e. p_{18}) shown in Figure 3 in order to carry out the meet-in-the-middle attack. The neutral word W_{18} must be independent of p_{18}. The neutral word W_{19} should satisfy the same relation with p_{20}.

The step update functions of MD5, SHA-0/1 and SHA-2 only use one message word to update one (or two) chaining values in every step, while that of DHA-256 uses two same message words to update two chaining values in every step. Thereby only one message word is needed to deal with in every step of the initial structure of MD5, SHA-0/1 and SHA-2. The 2-word initial structure for 35-step DHA-256 is depicted in Figure 3.

The points T1, T2, T3 and T4 in Figure 3 are set to four fixed values. $A_{18} = T1 - f(B_{18}, C_{18}, D_{18}) - SS_1(D_{18}) + K_{18}$. Because T1 is fixed, the change of W_{18} does not affect A_{18}. The change of A_{18} is independent of G_{20} because of $G_{20} = H_{19} = T1 + W_{18} + K_{18}$ and the

fixed value of T1. The effect of T2, T3 and T4 is similar to that of T1.

4.3. Attack procedure

The attack procedure for 35-step DHA-256 includes initialization, forward computation, backward computation and comparison. The description of partial-fixing and partial-matching procedure is depicted in Figure 4.

1. Initialization

(a) Set the values of W_i, i=13,14,15 to satisfy the padding rule for a 2-block message.

(b) Select the values of T1, T2, T3, T4, C_{18}, D_{18}, G_{18} and H_{18} in the 2-word initial structure randomly. Let $A_{20}=B_{19}=C_{18}<<<17$, $C_{19}=D_{18}$, $B_{20}=C_{19}<<<17$, $E_{20}=F_{19}=G_{18}<<<2$, $G_{19}=H_{18}$, $F_{20}=G_{19}<<<2$.

(c) Select the values of W_i, $i \in \{16, 17, 20, 21, 22, 23, 24, 25, 26\}$, const1and const2 randomly. Select the values from the 16th bit to the 31st bit of X ($=\sigma_1(W_{18})$) and the values from the 0th bit to the 15th bit of Y ($=\sigma_2(W_{19})$) randomly.

2. Forward Computation

For the free bits of X ($=\sigma_1(W_{18})$) (from the 0th bit to the 15th), do the following,

(a) Compute $W_{18}=\sigma_1^{-1}(X)$. Let $W_{27}=W_{18}+$ const2. Compute the values of W_{28}, W_{29}, W_{30}, W_{31}, W_{32}, W_{33} and the values from the 0th bit to the 15th bit of W_{34} using the values from the 0th bit to the 15th bit of Y. ($\overset{15-0}{W_{34}} = \sigma_1(\overset{ALL}{W_{33}}) + W_{25} + \overset{15-0}{Y} + W_{18}$).

(b) Compute $G_{20}=H_{19}=T1+W_{18}+K_{18}$, $C_{20}=D_{19}=T2+W_{18}+K_{18}$, $H_{20}=T3+f(B_{19}, C_{19}, D_{19})+SS_1(D_{19})+K_{19}$, $D_{20}=T4+g(F_{19}, G_{19}, H_{19})+SS_2(H_{19})+K_{19}$.

(c) Compute $p_{j+1} \leftarrow R_j(p_j, W_j)$ for j=28, 29, ..., 33. Compute p_{35} using p_{34} and W_{34}, then only get the values from the 0th bit to the 15th bit for D_{35} and H_{35}, all the bit values for other intermediate variables. Let $\overset{31-17,\,0(0\ unused)}{m} = (\overset{ALL}{d} - \overset{15-0}{D_{35}}) <<< 17$, $\overset{17-2}{n} = (\overset{ALL}{h} - \overset{15-0}{H_{35}}) <<< 2$. Store the values of $(p_{34}, \overset{31-17}{m}, \overset{17-2}{n}, W_{18})$ into the list L. The memory requirement of the list L is $2^{16} \cdot 11$ words. Note that because $A_3 = B_2 = C_1 <<< 17$ and $C_1 = D_0 = d - D_{35}$, we can get the equation $A_3 = m = (d - D_{35}) <<< 17$. We can also get the equation $E_3 = n = (h - H_{35}) <<< 2$ in a similar way. We store m and n, i.e. A_3 and E_3, into the list, instead of D_0 and H_0 due to the convenience of comparison next subsection. This is slightly different from the description in Figure 4, but they are the same in essence.

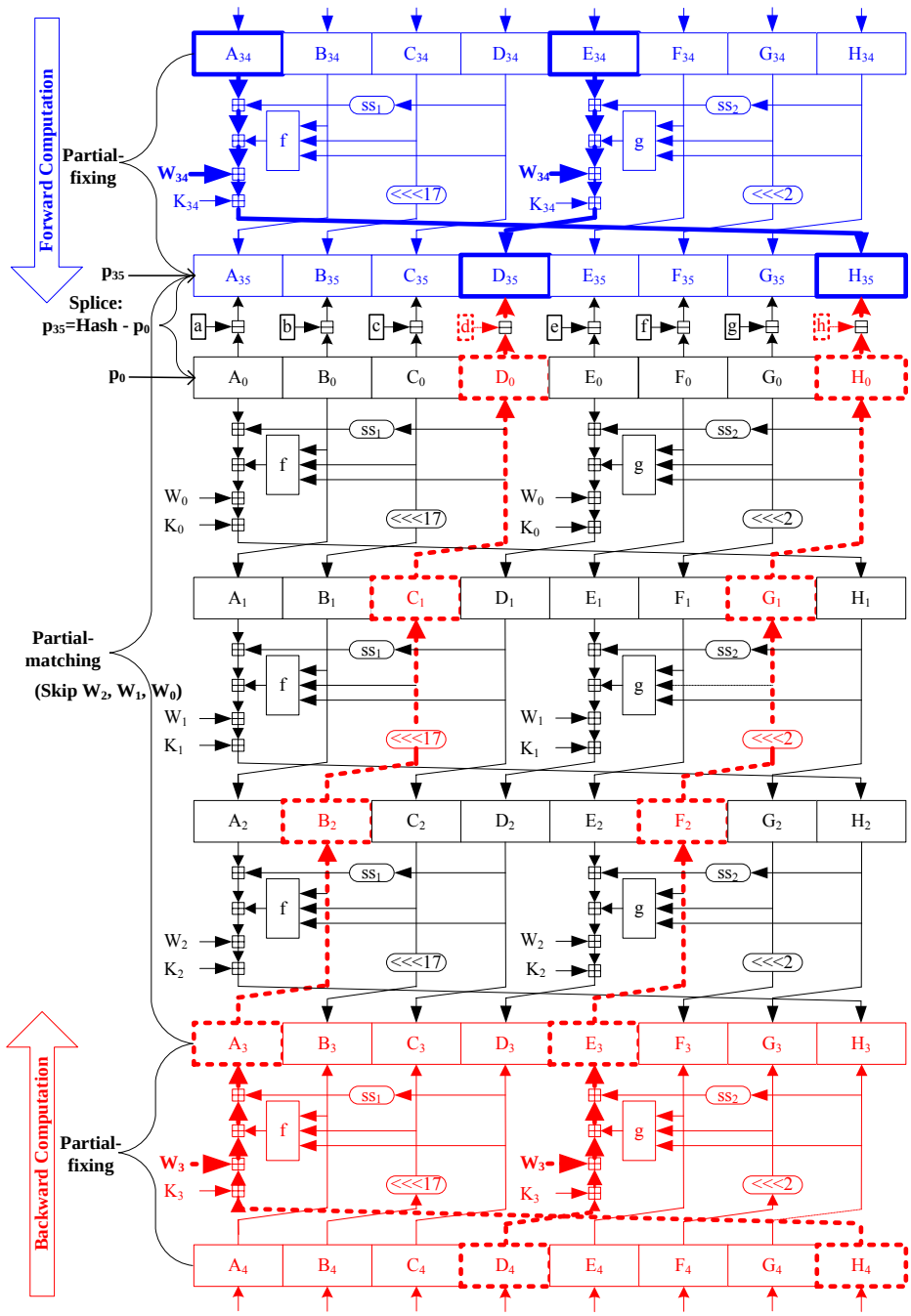

Fig. 4. The partial-fixing and partial-matching.

3. Backward Computation and Comparison

For the free bits of Y ($=\sigma_2(W_{19})$) (from the 16th bit to the 31st bit), do the following (The program about computing σ_2^{-1} and the value of σ_2^{-1} is in appendix A and D),

(a) Compute $W_{19}=\sigma_2^{-1}(Y)$. Let $W_{12}=$ const1-W_{19}. Compute the values of W_{11}, W_{10}, …, W_4 and the values from the 16th bit to the 31st bit of W_3 using the values from the 16th bit to the 31st bit of X. ($W_3^{31-16(2^1)} = W_{19}^{ALL} - X^{31-16} - W_{10}^{ALL} - \sigma_2(W_4)^{ALL}$).

(b) Compute $B_{18}=A_{19}=T3 - W_{19}$, $F_{18}=E_{19}=T4 - W_{19}$, $A_{18}=T1-f(B_{18}, C_{18}, D_{18})-SS_1(D_{18})$ and $E_{18}=T2-g(F_{18}, G_{18}, H_{18})-SS_2(H_{18})$.

(c) Compute $p_j \leftarrow R_j^{-1}(p_{j+1}, W_j)$ for j=17, 16, ..., 4. Compute A_3 and E_3 using p_4 and $W_3^{31-16(2^1)}$, then only get the values from the 16th bit to the 31st bit for A_3 and E_3. Compare m^{31-17}, n^{17-2} in the list L with $A_3^{31-16(2^1)}$, $E_3^{31-16(2^1)}$ respectively.

(d) If matched, compute W_3 using the corresponding W_{18} in the list L and compute p_3 using W_3, then all bits of A_3 and E_3 are known. Compare m^{31-17}, n^{17-2} in the list L with A_3^{ALL}, E_3^{ALL} respectively.

(e) If matched, compute W_2 and p_2 successively. Let $m^{ALL} = c - A_2^{ALL} >>> 17$, $n^{ALL} = g - E_2^{ALL} >>> 2$. Compare $D_{34}^{ALL}, H_{34}^{ALL}$ in the list L with m^{ALL}, n^{ALL} respectively.

(f) If matched, compute W_1 and p_1 successively. Let $m^{ALL} = b - A_1^{ALL}$, $n^{ALL} = f - E_1^{ALL}$. Compare $C_{34}^{ALL}, H_{34}^{ALL}$ in the list L with $m^{ALL} >>> 17$, $n^{ALL} >>> 2$ respectively.

(g) If matched, compute W_0 and p_0 successively. Let $m^{ALL} = a - A_0^{ALL}$, $n^{ALL} = e - E_0^{ALL}$. Compare $B_{34}^{ALL}, F_{34}^{ALL}$ in the list L with m^{ALL}, n^{ALL} respectively.

(h) If matched, compute W_{34} and compute $A_{34}^{ALL}, E_{34}^{ALL}$ using W_{34} and p_{35}=Hash - p_0. Compare $A_{34}^{ALL}, E_{34}^{ALL}$ in the list L with $A_{34}^{ALL}, E_{34}^{ALL}$ respectively.

(i) If matched, return (p_0, W_0, W_1, …, W_{15}) as a pseudo-preimage of 35-step DHA-256, otherwise, repeat the attack procedure.

4. Complexity analysis

The complexity analysis of pseudo-preimage and preimage attack on 35-step DHA-256 is as follows.

1. The cost of initialization compared to that of other steps is negligible.
2. The number of the free bits for forward computation is 16. In step 2a, the complexity for computing W_{18} and W_{27} is negligible. For others, the complexity is $2^{16} \cdot \frac{7}{35}$ ME. In step 2b, the complexity is $2^{16} \cdot \frac{1}{35}$ CF. In step 2c, the complexity is $2^{16} \cdot \frac{7}{35}$ CF. The memory complexity is $2^{16} \cdot 11$ words.
3. For backward computation and comparison, the number of the free bits is 16. In step 3a, we get two candidates of $^{31-16}W_3$ for every $^{31-16}X$ because of the two possible carried number patterns from bit 15 to bit 16, so the complexity is $2^{16} \cdot (\frac{8}{35} + \frac{2}{35})$ ME and we get 2^{17} pairs $^{31-16}W_3$. In step 3b, the complexity is $2^{16} \cdot \frac{1}{35}$ CF. In step 3c, we obtain 2^{17} pairs $^{31-16}A_3$ and $^{31-16}E_3$, thus the complexity is $2^{16} \cdot (\frac{14}{35} + \frac{2}{35})$ CF before comparison. So $2^{16} \bullet 2^{17}$ pairs take part in comparison. $(^{31-17}m, {}^{31-17}A_3)$ and $(^{17-16}n, {}^{17-16}E_3)$ match 16 bits in total. Therefore 2^{17} $(= 2^{16} \bullet 2^{17} \bullet 2^{-16})$ pairs remain. The complexity for comparison is negligible compared to other steps. In step 3d, the complexity for computing W_3 is $2^{17} \cdot \frac{1}{35}$ ME. The complexity for computing p_3 is $2^{17} \cdot \frac{1}{35}$ CF. $(^{15-2}n, {}^{15-2}E_3)$ matches 14 bits in total. So $2^3 (= 2^{17} \bullet 2^{-14})$ pairs remain. Moreover, all of bit values of A_3 and E_3 are known, and thus the carry from the 15th bit to the 16th bit for $(^{31-16(2^l)}A_3, {}^{31-16(2^l)}E_3)$ is known. Therefore, $2^2 (= 2^3 \bullet 2^{-1})$ pairs remain.

The cost in step 3e~3h is negligible compared to other steps. Hereto, the complexity is $2^{16} \bullet \frac{19}{35} (= 2^{16} \bullet (\frac{7}{35} + \frac{8}{35} + \frac{2}{35}) + 2^{17} \bullet \frac{1}{35})$ ME and $2^{16} \bullet \frac{27}{35} (= 2^{16} \bullet (\frac{1}{35} + \frac{7}{35} + \frac{1}{35} + \frac{14}{35} + \frac{2}{35}) + 2^{17} \bullet \frac{1}{35})$ CF. It is about $2^{16} \bullet \frac{27}{35} \approx 2^{15.63}$ 35-step DHA-256 operations. It produces $2^{-224} (= 2^{16} \bullet 2^{16} \bullet 2^{-256})$ matched pair. Thus, it needs to repeat 2^{224} times in step 3i to produce one matched pair.

Hence, the complexity of pseudo-preimage is $2^{239.63} (= 2^{224+15.63})$ 35-step DHA-256 operations. The complexity of preimage is $2^{248.82}$ 35-step DHA-256 operations. The memory complexity is $2^{16} \bullet 11$ words in step 2c.

5. CONCLUSION

This paper proposes two attacks on reduced DHA-256. The first attack is one-block second preimage and preimage attack on 26-step DHA-256, and the second one is preimage attack on 35-step DHA-256. We propose a 2-word initial structure which is a variant of initial structure. This first attack is based upon the strategy which is proposed by Isobe and Shibutani and uses two-way message expansion technique. The second attack is based on the general framework proposed by Sasaki and Aoki and combines two-way message expansion and message compensation techniques with 2-word initial structure. We also point out the incorrect matrix of the inverse function σ_1^{-1} of SHA-2 showed in [11] and show a correct one. This is the first analysis of second preimage resistance and preimage resistance of DHA-256, as far as we know.

REFERENCES

1. National Institute of Standards and Technology (NIST), "FIPS-180-2: Secure Hash Standard," August 2002, http://www.itl.nist.gov/fipspubs/.

2. K. Aoki and Y. Sasaki, "Meet-in-the-Middle Preimage Attacks Against Reduced SHA-0 and SHA-1," *CRYPTO 2009,* Springer-Verlag.

3. Kazumaro Aoki, Jian Guo, Krystian Matusiewicz, Yu Sasaki and Lei Wang, "Preimages for Step-Reduced SHA-2," *ASIACRYPT 2009,* Springer-Verlag.

4. Kazumaro Aoki and Yu Sasaki, "Preimage attacks on one-block MD4, 63-step MD5 and more," Liam Keliher Roberto Avanzi, and Francesco Sica, *SAC 2008,* pp.82-98.

5. Jean-Philippe Aumasson, Willi Meier and Florian Mendel, "Preimage attacks on 3-pass HAVAL and step-reduced MD5," Liam Keliher Roberto Avanzi, and Francesco Sica, *SAC 2008,* pp.99 - 144.

6. C. De Canniere and C. Rechberger, "Preimages for reduced SHA-0 and SHA-1," David Wagner, *CRYPTO 2008,* Springer-Verlag, pp.179-202.

7. W. Diffie and M. Hellman (1977). "Exhaustive cryptanalysis of the NBS Data Encryption Standard." Computer(10(6)): 74 - 84.

8. IAIK Krypto Group, "Preliminary Analysis of DHA-256," http://eprint.iacr.org/2005/398.pdf.

9. Jian Guo and Krysitan Matusiewicz, "Preimages for Step-Reduced SHA-2,"

Submission version for ASIACRYPT2009, http://www1.spms.ntu.edu.sg/~guojian/doc/SHA2.pdf.

10. S Indesteege, F Mendel, B Preneel and C Rechberger, "Collisions and other non-random properties for step-reduced SHA-256," *SAC 2008.*

11. T Isobe and K Shibutani, "Preimage Attacks on Reduced Tiger and SHA-2," *FSE 2009,* Springer-Verlag.

12. Jesang Lee, Donghoon Chang, Eunjin Lee Hyun Kim, Deukjo Hong, Jaechul Sung, Seokhie Hong and Sangjin Lee, "A New 256-bit Hash Function DHA-256: Enhancing the Security of SHA-256," NIST, *2005 Cryptographic Hash Workshop.*

13. Xuejia Lai and James L. Massey, "Hash Function Based on Block Ciphers," *Eurocrypt 1992,* Springer-Verlag, pp.55-70.

14. G. Leurent, "MD4 is Not One-Way," K. Nyberg, *FSE 2008,* Springer-Verlag, pp.412-428.

15. SK Sanadhya and P Sarkar, "New collision attacks against up to 24-step SHA-2," D.R. Chowdhury, Rijmen, V., Das, A., *INDOCRYPT 2008,* Springer-Verlag, pp.91-103.

16. Y. Sasaki and K. Aoki, "Preimage Attacks on 3, 4, and 5-Pass HAVAL," Josef Pieprzyk, *Advances in Cryptology - Asiacrypt 2008,* Springer-Verlag, pp.253-271.

17. Y. Sasaki and K. Aoki, "Preimage attacks on step-reduced MD5," Yi Mu and Willy Susilo, *ACISP2008,* Springer-Verlag, pp.282-296.

18. Y. Sasaki and K. Aoki, "Finding Preimages in Full MD5 Faster Than Exhaustive Search," Ronald Cramer, *Advances in Cryptology - Eurocrypt 2009,* Springer-Verlag, pp.134-152.

19. Y. Sasaki and K. Aoki, "Meet-in-the-Middle Preimage Attacks on Double-Branch Hash Functions: Application to RIPEMD and Others," *ACISP 2009,* pp.214-231.

20. Y. Sasaki and K. Aoki, "A Preimage Attack for 52-Step HAS-160," Pil Joong Lee and Jung Hee Cheon, *Information Security and Cryptology - ICISC 2008,* Springer-Verlag, pp.302-317.

21. Yu Sasaki, Lei Wang and Kazumaro Aoki, "Preimage Attacks on 41-Step SHA-256 and 46-Step SHA-512 ", http://eprint.iacr.org/2009/479.

22. National Institute of Standards and Technology (NIST) U.S. Department of Commerce, "Secure Hash Standard (SHS) (Federal Information Processing Standards Publication 180-3)," 2008,

http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf.

23. G. L. Wang and S. H. Wang, "Preimage Attack on Hash Function RIPEMD," *ISPEC 2009,* Springer-Verlag, pp.274-284.

24. X Wang, X Lai, D Feng, H Chen and X Yu, "Cryptanalysis of the Hash Functions MD4 and RIPEMD," Ronald Cramer, *Advances in Cryptology - CEUROCRYPT 2005,* Springer-Verlag, pp.1-18.

25. X Wang, YL Yin and H Yu, "Finding collisions in the full SHA-1," *CRYPTO 2005,* Springer-Verlag, pp.17–36.

26. X Wang and H Yu, "How to Break MD5 and Other Hash Functions," Ronald Cramer, *Advances in Cryptology - EUROCRYPT 2005,* Springer-Verlag, pp.19-35.

27. X Wang, H Yu and YL Yin, "Efficient Collision Search Attacks on SHA-0," *CRYPTO 2005,* Springer-Verlag, pp.1-16.

28. H. B. Yu, G. L. Wang, G. Y. Zhang and X. Y. Wang, "The second-preimage attack on MD4," Y.G. Desmedt, Wang, H., Mu, Y., Li, Y, *CANS 2005,* Springer-Verlag, pp.1-12.

Appendix A

The program codes for computing the inversion of the matrix of Sigma function of SHA-256 and DHA-256

```
% The codes are written by MATLAB 7.1
%*******The Inversion of the matrix of Sigma-1 function of SHA-256*********
% The codes written using matlab are used to computer the matrix for the function
σ_1^{-1} % of sha-256.
%Y(X)=(X>>>17)^(X>>>19)^(X>>10)
% gf() is used to operate the matrix in GF(2).
a=gf(zeros(32, 32),1);
for i=1:32 %Note that the subscript in matrix is from 1 to 32.
  x= mod( i+17, 32);
  if x==0
       x=32;
  end
  a(i,x)=1;

  y=mod( i+19, 32);
  if y==0
       y=32;
  end
  a(i, y)=1;

  z=i+10;
  if z <= 32
       a(i, z)=1;
  end
end
b=gf(zeros(32, 32),1);
% Let the first line in matrix is the 32th data line and left-msb.
for i=1:32
     for j=1:32
          b(i,j)=a(32-i+1,32-j+1);
     end
end

c=gf(zeros(32, 32),1);
c=inv(b) % Get the inversion and c is the matrix which we seek.


%**********The Inversion of the matrix of Sigma-1 function of DHA-256********
%Y(X)=X^(X<<<7)^(X<<<22)
```

```
a=gf(zeros(32, 32),1);
for i=1:32
   a(i,i)=1;                % Deal with the operation X
    x= mod( i-7, 32);    % Deal with the operation (X<<<7)
    if x==0
        x=32;
    end
    a(i,x)=1;
    y=mod( i-22, 32);    % Deal with the operation (X<<<22)
    if y==0
        y=32;
    end
    a(i, y)=1;
end
b=gf(zeros(32, 32),1);
% Let the first line in matrix is the 32th data line and left-msb.
for i=1:32
     for j=1:32
          b(i,j)=a(32-i+1,32-j+1);
     end
end
c=gf(zeros(32, 32),1);
c=inv(b) % Get the inversion and c is the matrix which we seek.
```

%*****The Inversion of the matrix of Sigma-2 function of DHA-256********

%Y(X)=X^(X<<<13)^(X<<<27)

```
a=gf(zeros(32, 32),1);
for i=1:32
   a(i,i)=1;                 % Deal with the operation X
    x= mod( i-13, 32);    % Deal with the operation (X<<<13)
```

```
    if x==0
        x=32;
    end
    a(i,x)=1;

    y=mod( i-27, 32);    % Deal with the operation (X<<<27)
    if y==0
        y=32;
    end
    a(i, y)=1;
end
b=gf(zeros(32, 32),1);
% Let the first line in matrix is the 32th data line and left-msb.
for i=1:32
     for j=1:32
          b(i,j)=a(32-i+1,32-j+1);
     end
end
c=gf(zeros(32, 32),1);
c=inv(b) % Get the inversion and c is the matrix which we seek.
```

Appendix B

The Inversion of the matrix of Sigma-1 function of SHA-256

$$M_{\sigma_1^{-1}} = \begin{bmatrix}
1&0&1&1&0&1&1&1&1&1&1&1&1&1&1&1&1&1&0&0&1&1&0&0&1&1&0&1&0&0\\
1&1&1&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&1&0&1&0&1&0&1&1&1&0\\
0&1&1&1&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&1&0&1&0&1&0&1&1&1&1\\
0&0&0&0&1&1&1&0&1&0&1&0&1&0&1&0&1&1&1&0&0&1&0&0&0&1&1&0&1&0&&\\
0&0&0&0&0&1&1&1&0&1&0&1&0&1&0&1&0&1&1&1&0&0&1&0&0&0&1&1&0&1&&\\
1&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&0&0&1&1&0&1&1&0&0&0&0&1&1\\
1&1&0&0&0&0&1&0&1&0&1&0&1&0&1&0&1&1&1&0&1&1&1&1&1&0&0&1&0&0&&\\
0&1&1&0&0&0&0&1&0&1&0&1&0&1&0&1&0&1&1&1&0&1&1&1&1&1&0&0&1&0&&\\
1&0&0&0&0&1&1&1&0&1&0&1&0&1&0&1&0&1&1&1&0&1&1&1&0&0&1&1&0&1&&\\
1&1&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&0&0&1&1&0&0&1&1&0&0&0&1&1&\\
1&1&1&0&0&0&1&0&1&0&1&0&1&0&1&0&1&1&1&0&1&1&1&0&1&1&0&1&0&0&&\\
0&1&1&1&0&0&0&1&0&1&0&1&0&1&0&1&0&1&1&1&0&1&1&1&0&1&1&0&1&0&&\\
1&0&0&0&1&1&1&1&0&1&0&1&0&1&0&1&0&1&1&1&0&1&1&1&0&1&1&0&0&1&&\\
1&1&0&0&0&1&0&1&1&1&1&1&1&1&1&1&1&1&0&0&1&1&0&0&1&1&0&1&0&0&1&\\
1&1&1&0&0&0&0&0&1&0&1&0&1&0&1&0&1&1&1&0&1&1&1&0&1&1&0&1&1&0&&\\
0&1&0&0&0&1&0&1&1&1&1&1&1&1&1&1&1&1&0&0&1&1&0&0&1&1&0&1&0&0&1&\\
1&0&1&0&0&0&0&0&1&0&1&0&1&0&1&0&1&1&1&0&1&1&1&0&1&1&0&1&1&0&&\\
0&1&1&0&0&1&0&1&1&1&1&1&1&1&1&1&1&1&0&0&1&1&0&0&1&1&0&1&0&0&1&\\
1&0&1&1&0&0&0&0&1&0&1&0&1&0&1&0&1&1&1&0&1&1&1&0&1&1&0&1&1&0&&\\
0&1&1&0&1&1&1&1&0&1&0&1&0&1&0&1&0&1&1&1&0&1&1&1&0&1&1&0&1&0&&\\
1&0&1&1&0&1&0&0&1&0&1&0&1&0&1&0&1&1&1&0&1&1&1&0&1&1&0&1&1&0&&\\
1&1&0&1&1&0&0&0&1&0&1&0&0&0&0&0&0&0&1&0&1&0&1&0&1&0&1&1&1&0&1&\\
0&1&0&1&1&0&0&1&1&1&1&1&1&0&1&0&1&0&1&1&0&0&0&1&0&0&0&1&1&1&1&\\
1&0&1&0&1&1&1&0&1&0&1&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&1&0&1&0\\
0&1&0&1&0&1&1&1&0&1&0&1&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&1&0&1&\\
1&0&1&0&1&0&0&1&1&1&1&1&1&1&1&1&0&1&0&0&1&1&0&0&1&0&0&0&0&1&1&1\\
1&1&0&1&0&1&1&0&1&0&1&0&1&0&1&1&1&0&1&1&1&0&1&1&0&1&1&0&0&0&1&1&0\\
0&1&1&0&1&0&1&1&0&1&0&1&0&1&0&1&1&1&0&1&1&1&0&1&1&0&1&1&0&0&0&1&1
\end{bmatrix}$$

Appendix C

The Inversion of the matrix of Sigma-1 function of DHA-256

$$M_{\sigma_1^{-1}} = \begin{bmatrix}
1&1&1&0&0&1&1&0&0&0&1&0&1&1&0&1&1&1&1&1&1&0&0&0&0&1&1&0&1&0&0&0\\
0&1&1&1&0&0&1&1&0&0&0&1&0&1&1&0&1&1&1&1&1&1&0&0&0&0&1&1&0&1&0&0\\
0&0&1&1&1&0&0&1&1&0&0&0&1&0&1&1&0&1&1&1&1&1&1&0&0&0&0&1&1&0&1&0\\
0&0&0&1&1&1&0&0&1&1&0&0&0&1&0&1&1&0&1&1&1&1&1&1&0&0&0&0&1&1&0&1\\
1&0&0&0&1&1&1&0&0&1&1&0&0&0&1&0&1&1&0&1&1&1&1&1&1&0&0&0&0&1&1&0\\
0&1&0&0&0&1&1&1&0&0&1&1&0&0&0&1&0&1&1&0&1&1&1&1&1&1&0&0&0&0&1&1\\
1&0&1&0&0&0&1&1&1&0&0&1&1&0&0&0&1&0&1&1&0&1&1&1&1&1&1&0&0&0&0&1\\
1&1&0&1&0&0&0&1&1&1&0&0&1&1&0&0&0&1&0&1&1&0&1&1&1&1&1&1&0&0&0&0\\
0&1&1&0&1&0&0&0&1&1&1&0&0&1&1&0&0&0&1&0&1&1&0&1&1&1&1&1&1&0&0&0\\
0&0&1&1&0&1&0&0&0&1&1&1&0&0&1&1&0&0&0&1&0&1&1&0&1&1&1&1&1&1&0&0\\
0&0&0&1&1&0&1&0&0&0&1&1&1&0&0&1&1&0&0&0&1&0&1&1&0&1&1&1&1&1&1&0\\
0&0&0&0&1&1&0&1&0&0&0&1&1&1&0&0&1&1&0&0&0&1&0&1&1&0&1&1&1&1&1&1\\
1&0&0&0&0&1&1&0&1&0&0&0&1&1&1&0&0&1&1&0&0&0&1&0&1&1&0&1&1&1&1&1\\
1&1&0&0&0&0&1&1&0&1&0&0&0&1&1&1&0&0&1&1&0&0&0&1&0&1&1&0&1&1&1&1\\
1&1&1&0&0&0&0&1&1&0&1&0&0&0&1&1&1&0&0&1&1&0&0&0&1&0&1&1&0&1&1&1\\
1&1&1&1&0&0&0&0&1&1&0&1&0&0&0&1&1&1&0&0&1&1&0&0&0&1&0&1&1&0&1&1\\
1&1&1&1&1&0&0&0&0&1&1&0&1&0&0&0&1&1&1&0&0&1&1&0&0&0&1&0&1&1&0&1\\
1&1&1&1&1&1&0&0&0&0&1&1&0&1&0&0&0&1&1&1&0&0&1&1&0&0&0&1&0&1&1&0\\
0&1&1&1&1&1&1&0&0&0&0&1&1&0&1&0&0&0&1&1&1&0&0&1&1&0&0&0&1&0&1&1\\
1&0&1&1&1&1&1&1&0&0&0&0&1&1&0&1&0&0&0&1&1&1&0&0&1&1&0&0&0&1&0&1\\
1&1&0&1&1&1&1&1&1&0&0&0&0&1&1&0&1&0&0&0&1&1&1&0&0&1&1&0&0&0&1&0\\
0&1&1&0&1&1&1&1&1&1&0&0&0&0&1&1&0&1&0&0&0&1&1&1&0&0&1&1&0&0&0&1\\
1&0&1&1&0&1&1&1&1&1&1&0&0&0&0&1&1&0&1&0&0&0&1&1&1&0&0&1&1&0&0&0\\
0&1&0&1&1&0&1&1&1&1&1&1&0&0&0&0&1&1&0&1&0&0&0&1&1&1&0&0&1&1&0&0\\
0&0&1&0&1&1&0&1&1&1&1&1&1&0&0&0&0&1&1&0&1&0&0&0&1&1&1&0&0&1&1&0\\
0&0&0&1&0&1&1&0&1&1&1&1&1&1&0&0&0&0&1&1&0&1&0&0&0&1&1&1&0&0&1&1\\
1&0&0&0&1&0&1&1&0&1&1&1&1&1&1&0&0&0&0&1&1&0&1&0&0&0&1&1&1&0&0&1\\
1&1&0&0&0&1&0&1&1&0&1&1&1&1&1&1&0&0&0&0&1&1&0&1&0&0&0&1&1&1&0&0\\
0&1&1&0&0&0&1&0&1&1&0&1&1&1&1&1&1&0&0&0&0&1&1&0&1&0&0&0&1&1&1&0\\
0&0&1&1&0&0&0&1&0&1&1&0&1&1&1&1&1&1&0&0&0&0&1&1&0&1&0&0&0&1&1&1\\
1&0&0&1&1&0&0&0&1&0&1&1&0&1&1&1&1&1&1&0&0&0&0&1&1&0&1&0&0&0&1&1\\
1&1&0&0&1&1&0&0&0&1&0&1&1&0&1&1&1&1&1&1&0&0&0&0&1&1&0&1&0&0&0&1
\end{bmatrix}$$

Appendix D

The Inversion of the matrix of Sigma-2 function of DHA-256

$$M_{\sigma_2^{-1}} = \begin{bmatrix}
1&0&1&1&1&1&0&0&1&0&0&0&0&1&1&1&0&1&0&1&0&1&0&1&1&0&0&0&1&1&0&1\\
1&1&0&1&1&1&1&0&0&1&0&0&0&0&1&1&1&0&1&0&1&0&1&0&1&1&0&0&0&1&1&0\\
0&1&1&0&1&1&1&1&0&0&1&0&0&0&0&1&1&1&0&1&0&1&0&1&0&1&1&0&0&0&1&1\\
1&0&1&1&0&1&1&1&1&0&0&1&0&0&0&0&1&1&1&0&1&0&1&0&1&0&1&1&0&0&0&1\\
1&1&0&1&1&0&1&1&1&1&0&0&1&0&0&0&0&1&1&1&0&1&0&1&0&1&0&1&1&0&0&0\\
0&1&1&0&1&1&0&1&1&1&1&0&0&1&0&0&0&0&1&1&1&0&1&0&1&0&1&0&1&1&0&0\\
0&0&1&1&0&1&1&0&1&1&1&1&0&0&1&0&0&0&0&1&1&1&0&1&0&1&0&1&0&1&1&0\\
0&0&0&1&1&0&1&1&0&1&1&1&1&0&0&1&0&0&0&0&1&1&1&0&1&0&1&0&1&0&1&1\\
1&0&0&0&1&1&0&1&1&0&1&1&1&1&0&0&1&0&0&0&0&1&1&1&0&1&0&1&0&1&0&1\\
1&1&0&0&0&1&1&0&1&1&0&1&1&1&1&0&0&1&0&0&0&0&1&1&1&0&1&0&1&0&1&0\\
0&1&1&0&0&0&1&1&0&1&1&0&1&1&1&1&0&0&1&0&0&0&0&1&1&1&0&1&0&1&0&1\\
1&0&1&1&0&0&0&1&1&0&1&1&0&1&1&1&1&0&0&1&0&0&0&0&1&1&1&0&1&0&1&0\\
0&1&0&1&1&0&0&0&1&1&0&1&1&0&1&1&1&1&0&0&1&0&0&0&0&1&1&1&0&1&0&1\\
1&0&1&0&1&1&0&0&0&1&1&0&1&1&0&1&1&1&1&0&0&1&0&0&0&0&1&1&1&0&1&0\\
0&1&0&1&0&1&1&0&0&0&1&1&0&1&1&0&1&1&1&1&0&0&1&0&0&0&0&1&1&1&0&1\\
1&0&1&0&1&0&1&1&0&0&0&1&1&0&1&1&0&1&1&1&1&0&0&1&0&0&0&0&1&1&1&0\\
0&1&0&1&0&1&0&1&1&0&0&0&1&1&0&1&1&0&1&1&1&1&0&0&1&0&0&0&0&1&1&1\\
1&0&1&0&1&0&1&0&1&1&0&0&0&1&1&0&1&1&0&1&1&1&1&0&0&1&0&0&0&0&1&1\\
1&1&0&1&0&1&0&1&0&1&1&0&0&0&1&1&0&1&1&0&1&1&1&1&0&0&1&0&0&0&0&1\\
1&1&1&0&1&0&1&0&1&0&1&1&0&0&0&1&1&0&1&1&0&1&1&1&1&0&0&1&0&0&0&0\\
0&1&1&1&0&1&0&1&0&1&0&1&1&0&0&0&1&1&0&1&1&0&1&1&1&1&0&0&1&0&0&0\\
0&0&1&1&1&0&1&0&1&0&1&0&1&1&0&0&0&1&1&0&1&1&0&1&1&1&1&0&0&1&0&0\\
0&0&0&1&1&1&0&1&0&1&0&1&0&1&1&0&0&0&1&1&0&1&1&0&1&1&1&1&0&0&1&0\\
0&0&0&0&1&1&1&0&1&0&1&0&1&0&1&1&0&0&0&1&1&0&1&1&0&1&1&1&1&0&0&1\\
1&0&0&0&0&1&1&1&0&1&0&1&0&1&0&1&1&0&0&0&1&1&0&1&1&0&1&1&1&1&0&0\\
0&1&0&0&0&0&1&1&1&0&1&0&1&0&1&0&1&1&0&0&0&1&1&0&1&1&0&1&1&1&1&0\\
0&0&1&0&0&0&0&1&1&1&0&1&0&1&0&1&0&1&1&0&0&0&1&1&0&1&1&0&1&1&1&1\\
1&0&0&1&0&0&0&0&1&1&1&0&1&0&1&0&1&0&1&1&0&0&0&1&1&0&1&1&0&1&1&1\\
1&1&0&0&1&0&0&0&0&1&1&1&0&1&0&1&0&1&0&1&1&0&0&0&1&1&0&1&1&0&1&1\\
1&1&1&0&0&1&0&0&0&0&1&1&1&0&1&0&1&0&1&0&1&1&0&0&0&1&1&0&1&1&0&1\\
1&1&1&1&0&0&1&0&0&0&0&1&1&1&0&1&0&1&0&1&0&1&1&0&0&0&1&1&0&1&1&0\\
0&1&1&1&1&0&0&1&0&0&0&0&1&1&1&0&1&0&1&0&1&0&1&1&0&0&0&1&1&0&1&1
\end{bmatrix}$$