Preimage Attacks on 41-Step SHA-256 and 46-Step SHA-512

Yu Sasaki[1], Lei Wang[2], and Kazumaro Aoki[1]

[1] NTT Information Sharing Platform Laboratories, NTT Corporation
3-9-11 Midori-cho, Musashino-shi, Tokyo, 180-8585 Japan
sasaki.yu@lab.ntt.co.jp
[2] The University of Electro-Communications
1-5-1 Choufugaoka, Choufu-shi, Tokyo, 182-8585 Japan

Abstract. In this paper, we propose preimage attacks on 41-step SHA-256 and 46-step SHA-512, which drastically increase the number of attacked steps compared to the best previous preimage attack working for only 24 steps. The time complexity for 41-step SHA-256 is $2^{253.5}$ compression function operations and the memory requirement is $2^{16} \times 10$ words. The time complexity for 46-step SHA-512 is $2^{511.5}$ compression function operations and the memory requirement is $2^3 \times 10$ words. Our attack is a meet-in-the-middle attack. We first consider the application of previous meet-in-the-middle attack techniques to SHA-2. We then analyze the message expansion of SHA-2 by considering all previous techniques to find a new independent message-word partition. We first explain the attack on 40-step SHA-256 whose complexity is 2^{249} to describe the ideas. We then explain how to extend the attack.

keywords: SHA-256, SHA-512, hash, preimage attack, meet-in-the-middle

1 Introduction

Cryptographic hash functions are important to build secure systems. SHA-1 and SHA-2 (SHA-224, SHA-256, SHA-384, and SHA-512) [14] are hash functions standardized by the National Institute of Standards and Technology (NIST), and widely used all over the world. However, regarding SHA-1, a collision attack has already been discovered by Wang *et al.* [16]. Because the structure of SHA-2 is similar to SHA-1, some attack might be discovered on SHA-2 in the future. To avoid such a situation, NIST is currently conducting a competition to determine the new hash function standard called SHA-3 [13]. From engineering viewpoint, migration from SHA-1 to SHA-3 will take a long time. SHA-2 will take an important role during that period. Hence, rigorous security evaluation of SHA-2 using the latest analytic techniques is important.

In the SHA-3 competition, 51 algorithms were accepted as candidates. Currently, researches on SHA-3 candidates are very active, in particular, security evaluation on SHA-3 candidates. NIST requires SHA-3 candidates of n-bit hash length to satisfy the several security properties [13], *e.g.*,

- Preimage resistance of n bits,
- Second-preimage resistance of $n - k$ bits for any message shorter than 2^k blocks,
- Collision resistance of $n/2$ bits.

NIST claims that security of each candidate is evaluated in the environment where they are tuned so that it runs as fast as SHA-2 [15]. It is obvious that NIST tries to evaluate each candidate by comparing with SHA-2. However, the security of SHA-2 is not well understood yet. Hence, evaluating the security of SHA-2 with respect to the security requirements for SHA-3 candidates is also important.

SHA-256 and SHA-512 consist of 64 steps and 80 steps, respectively. There exist some analyses on the reduced steps of SHA-2. The first analysis on SHA-2 with respect to SHA-3 requirements was done by Mendel *et al.* [6], which presented the collision attack on SHA-2 reduced to 19 steps. After

that, several researches have improved the result. To the best of our knowledge, the best collision attacks so far are the one proposed by Indesteege *et al.* [3] and the one proposed by Sanadhya and Sarkar [10], which present collision attacks on 24 steps. Apart from the collision attack, the only analysis we know is the one proposed by Isobe and Shibutani [4], which presented preimage attacks on SHA-2 reduced to 24 steps. One may note the work announced at the rump session by Yu and Wang [17], which claimed to have found a non-randomness property of SHA-256 reduced to 39 steps. Since the non-randomness property is not included in the security requirements for SHA-3, we do not discuss it in this paper. In summary, the current best attacks on SHA-2 with respect to the security requirements for SHA-3 work for only 24 steps.

After Saarinen [9] and Leurent [5] showed preimage attacks, the techniques for preimage attack have been developed very rapidly. These attacks are based on the framework of the meet-in-the-middle and have been reported for various hash functions, for example MD5 [8], SHA-1, HAVAL [18], and so on [1, 12, 2, 11]. Due to the complex message schedule in SHA-2, these recently developed techniques are not fully applied for SHA-2. Thus, it is interesting to evaluate the security of SHA-2 against these techniques.

Our contribution. We propose preimage attacks on 41-step SHA-256 and 46-step SHA-512 which drastically increase the number of attacked steps compared to the previous preimage attack on 24 steps. We first explain the attack on 40-step SHA-256 to simply describe ideas of our attack. This attack requires 2^{249} SHA-256 computation and $2^{16} \cdot 10$ words of memory. Because SHA-256 and SHA-512 have the similar structure, this attack is also applied to 40-step SHA-512. Due to the double word size of SHA-512 compared to SHA-256, the attack becomes much faster than the brute force attack in SHA-512; 2^{497} SHA-512 operations, but requires much more memory; $2^{32} \cdot 10$ words. We then explain how to extend the attack to work for more steps. This is achieved with so-called partial-fixing technique proposed by Aoki and Sasaki [1]. It is interesting that SHA-512 can be attacked more steps than SHA-256. This is due to the difference of word size. σ and Σ functions in SHA-2 mix the data X by applying the XOR of three different rotations/shifts of X. This mixes 32-bit variables in SHA-256 rapidly, but cannot mix double-size variables in SHA-512 rapidly.

Our attacks are based on the meet-in-the-middle attack. We first consider the application of the previous meet-in-the-middle techniques to SHA-2. We then analyze the message expansion of SHA-2 by considering all previous techniques and construct the attack by finding new independent message-word partition, which is a very fundamental part of this attack.

Results of our attacks and others are summarized in Table 1.

Outline. In Section 2, we describe the specification of SHA-2 and previous meet-in-the-middle preimage attacks. In Section 3, we consider an application of previous techniques to SHA-2. In Section 4, we analyze the message expansion to identify a new message-word partition. In Section 5, we explain the attack procedure and complexity evaluation. In Section 6, we conclude this paper.

2 SHA-2 Specification and Related Works

2.1 Description of SHA-256 and SHA-512

In this section, we first describes the specification of SHA-256, then we explain the differences of SHA-256 and SHA-512. Please refer to the original specifications [14] for details.

Table 1. Comparison of preimage attacks on reduced SHA-2

Reference		Target	Steps	Complexity		Memory
				Pseudo-preimage	Preimage	
[4]		SHA-256	24	2^{240}	2^{240}	$2^{16} \cdot 64$ bits
Ours	Section 5.1	SHA-256	40	2^{240}	2^{249}	$2^{16} \cdot 10$ words
Ours	Section 5.2	SHA-256	41	2^{249}	$2^{253.3}$	$2^{16} \cdot 10$ words
[4]		SHA-512	24	2^{480}	2^{480}	not given
Ours	Section 5.3	SHA-512	40	2^{480}	2^{497}	$2^{32} \cdot 10$ words
Ours	Section 5.3	SHA-512	41	2^{484}	2^{499}	$2^{28} \cdot 10$ words
Ours	Section 5.3	SHA-512	42	2^{488}	2^{501}	$2^{24} \cdot 10$ words
Ours	Section 5.3	SHA-512	43	2^{501}	$2^{507.5}$	$2^{11} \cdot 10$ words
Ours	Section 5.3	SHA-512	44	2^{504}	2^{509}	$2^{8} \cdot 10$ words
Ours	Section 5.3	SHA-512	45	2^{505}	$2^{509.5}$	$2^{7} \cdot 10$ words
Ours	Section 5.3	SHA-512	46	2^{509}	$2^{511.5}$	$2^{3} \cdot 10$ words

Description of SHA-256. SHA-256 adopts the Merkle-Damgård structure [7, Algorithm 9.25]. The message string is first padded to be a 512-bit multiple, and divided into 512-bit blocks,

$$(M_0, M_1, \ldots, M_{N-1}) \ (M_i \in \{0,1\}^{512}).$$

The hash value h_N is computed by iteratively using the compression function CF, which takes a 512-bit message string and a 256-bit chaining variable as input and an updated 256-bit chaining variable as output.

$$\begin{cases} h_0 \leftarrow \mathrm{IV}, \\ h_{i+1} \leftarrow \mathrm{CF}(h_i, M_i) \ (i = 0, 1, \ldots, N-1), \end{cases} \tag{1}$$

where IV is the constant number defined in the specification.

The compression function is based on the Davies-Meyer mode [7, Algorithm 9.42]. It consists of a message expansion and a data processing. Let $\gg x$ and $\ggg x$ denote the x-bit right shift and x-bit right rotation, respectively. First, the message block is expanded using the message expansion function.

$$\begin{cases} W_j \leftarrow m_j, & (\ 0 \le j < 16) \\ W_j \leftarrow \sigma_1(W_{j-2}) + W_{j-7} + \sigma_0(W_{j-15}) + W_{j-16}, & (16 \le j < 80) \end{cases} \tag{2}$$

where $(m_0, m_1, \ldots, m_{15}) \leftarrow M_i$ $(m_j \in \{0,1\}^{32})$ and "+" denotes the wordwise addition. In SHA-256 and SHA-512, the word-size is 32 bits and 64 bits respectively. $\sigma_0(X)$ and $\sigma_1(X)$ are defined as follows:

$$\begin{cases} \sigma_0(X) \leftarrow (X^{\ggg 7}) \oplus (X^{\ggg 18}) \oplus (X^{\gg 3}), \\ \sigma_1(X) \leftarrow (X^{\ggg 17}) \oplus (X^{\ggg 19}) \oplus (X^{\gg 10}). \end{cases} \tag{3}$$

where "$\oplus$" denotes bitwise XOR operation.

The data processing computes h_{i+1} as follows. Here, we use p_j to denote a 256-bit value consisting of the concatenation of eight words $A_j, B_j, C_j, D_j, E_j, F_j, G_j$ and H_j.

$$\begin{cases} p_0 \leftarrow h_i, \\ p_{j+1} \leftarrow R_j(p_j, W_j), \quad (j = 0, 1, \ldots, 63) \\ h_{i+1} \leftarrow h_i + p_{64}, \end{cases} \tag{4}$$

Step function R_j is defined as given hereafter:

$$\begin{cases} T_1^{(j)} \leftarrow H_j + \Sigma_1(E_j) + Ch(E_j, F_j, G_j) + K_j + W_j, \\ T_2^{(j)} \leftarrow \Sigma_0(A_j) + Maj(A_j, B_j, C_j), \\ A_{j+1} \leftarrow T_1^{(j)} + T_2^{(j)}, \quad B_{j+1} \leftarrow A_j, \quad C_{j+1} \leftarrow B_j, \quad D_{j+1} \leftarrow C_j, \\ E_{j+1} \leftarrow D_j + T_1^{(j)}, \quad F_{j+1} \leftarrow E_j, \quad G_{j+1} \leftarrow F_j, \quad H_{j+1} \leftarrow G_j. \end{cases} \tag{5}$$

where K_j is a constant number for each step and $Ch(X,Y,Z), Maj(X,Y,Z), \Sigma_0(X)$, and $\Sigma_1(X)$ are defined as follows.

$$\begin{cases} Ch(X,Y,Z) \leftarrow (X \vee Y) \oplus (\neg X \vee Z), \\ Maj(X,Y,Z) \leftarrow (X \vee Y) \oplus (X \vee Z) \oplus (Y \vee Z), \\ \Sigma_0(X) \leftarrow (X^{\ggg 2}) \oplus (X^{\ggg 13}) \oplus (X^{\ggg 22}), \\ \Sigma_1(X) \leftarrow (X^{\ggg 6}) \oplus (X^{\ggg 11}) \oplus (X^{\ggg 25}). \end{cases} \tag{6}$$

Description of SHA-512. The structure of SHA-512 is basically the same as SHA-256. In SHA-512, the word size is double of SHA-256, hence, the message-block size is 1024 bits and the size of chaining variable p_j is 512 bits. The compression function consists of 80 steps. Rotation numbers in $\sigma_0, \sigma_1, \Sigma_0$, and Σ_1 are different from SHA-256 so that double-size variables can be mixed rapidly.

2.2 Previous meet-in-the-middle preimage attacks

This section gives a high-level description of the previous meet-in-the-middle preimage attacks on hash functions. Usually a preimage attack on a Merkle-Damgård hash function is based on a pseudo-preimage attack on its underlying compression function, where a pseudo-preimage is a preimage of the compression function with an appropriate padding. Many compression functions adopt Davis-Meyer mode, which is designed based on a block cipher E as follows: $E_A(B) \oplus B$, where A and B are either intermediate hash values or messages. First we recall the attack strategy on a compression function, which has been explained in Fig. 1. Denote by h the given target hash value. The high-level description of the attack for the simplest case is as follows.

1. Divide key A of block cipher E into two *independent* parts: A_1 and A_2. Hereafter, independent parts are called "chunks," and independent inputs A_1 and A_2 are called "neutral words."
2. Randomly determine the other input value B of the block cipher E.
3. Carry out the forward calculation utilizing B and all possible values of A_1, and store all the obtained intermediate values in a table denoted as T_F.
4. Carry out the backward calculation utilizing $h \oplus B$ and all possible values of A_2, and store all the obtained intermediate values in a table denoted as T_B.
5. Check whether there exists a collision between T_F and T_B. If a collision exists, a pseudo-preimage of h has been generated. Otherwise, go to Step 2.

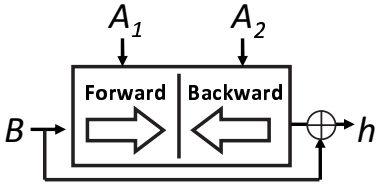

Fig. 1. Attack strategy on compression function $E_A(B) \oplus B$

The main novelty of the meet-in-the-middle preimage attacks is, by utilizing independence of A_1 and A_2 of the key input, transforming the problem of finding a preimage of h to the problem of finding a collision on the intermediate values, which has a much lower complexity than the former one. Suppose there are in total 2^t possible values for each of A_1 and A_2. Then with a complexity of 2^t compression function computations, the attacker obtain 2^t elements in each of T_F and T_B. The collision probability is roughly 2^{2t-n}, where n denotes the bit length of h. This is better than a probability of finding a preimage with a complexity of 2^t using brute force attack.

Various techniques. This section describes the techniques used in the previous meet-in-the-middle attacks [1, 11, 12]. An applications of these techniques on SHA-2 is described in Section 3.

Splice-and-cut: the meet-in-the-middle attack in Section 2.2 starts with dividing the key input into two independent parts. Aoki and Sasaki [1] point out that the last and first steps of E can be regarded as *consecutive*, by considering the feedforward addition. This technique was named *splice-and-cut*. Following this technique, the attacker can regard any step as a starting step of the meet-in-the-middle, which helps the attacker to find more suitable independent chunks. This technique can find only pseudo-preimages of given hash value instead of preimages. However, pseudo-preimages can be converted to preimages with a conversion algorithm explained in the following paragraph.

Conversion from pseudo-preimages to preimages: In x-bit iterated hash functions, a pseudo-preimage attack whose complexity is $2^y, y < x - 2$ can be converted to a preimage attack with a complexity of $2^{\frac{x+y}{2}+1}$ [7, Fact9.99]. The idea is applying the unbalanced meet-in-the-middle attack with generating $2^{(x-y)/2}$ pseudo-preimages and generating $2^{(x+y)/2}$ 1-block chaining variables starting from IV.

Partial-matching: the example in Fig. 1 is the simplest and optimistic case. In fact, in the previous attacks, the key input cannot be divided into just two independent chunks. Usually besides the two independent chunks A_1 and A_2, there is another part, which depends on both A_1 and A_2. Hence, the stored intermediate values in T_F and T_B are ones at different steps. This arise a problem: how the values in T_F and T_B can be compared. In many hash functions including SHA-2, the intermediate values is only updated partially in each step. In other words, a part of intermediate values do not change during several steps. Therefore, the attacker can check the match of two values partially. This technique is denoted as *partial-matching* technique.

Partial-fixing: this is an extension of the partial-matching technique. It increases the number of steps that can exist between two independent chunks. Assume that the attacker is carrying out the computation using A_1 and he is facing a step whose key input depends on both A_1 and A_2. Because the computation cannot go ahead without the knowledge of A_2, the chunk for A_1 must stop at this step. The *partial-fixing* technique is partially fixing the values of A_1 and A_2 so that we can obtain partial knowledge even if the computation depends on both A_1 and A_2.

Initial structure: in some case, the two independent chunks A_1 and A_2 will overlap with each other. The typical example is that the order of the input key of E is $A_1A_2A_1A_2$. This arises a problem: how should the attacker carry out the forward and backward computations independently. The *Initial Structure* technique was proposed by [12] to solve such a problem. Previous attacks usually set a certain step as the starting step, then randomly determine the intermediate value at that step, and carry out the independent computations. However, the initial structure technique

sets all the steps of A_2A_1 in the middle of $A_1A_2A_1A_2$ together as the starting point. Denote the intermediate values at the beginning and last step of A_2A_1 as I_1 and I_2 respectively. For each possible value of A_1, the attacker can derive a corresponding value I_1. Similarly, for each possible value of A_2, the attacker can derive a corresponding value I_2. Moreover, any pair (I_1, A_1) and (I_2, A_2) can be matched at the steps of A_2A_1 of $A_1A_2A_1A_2$. Thus, the attacker can carry out independent computations utilizing (I_1, A_1) and (I_2, A_2).

2.3 Revisiting the previous attack on SHA-2 reduced to 24 steps

Isobe and Shibutani [4] presented the first preimage attack on SHA-2 reduced to 24 steps. Their attack was meet-in-the-middle attack. The partial-matching technique was used in the attack. Following the previous attack procedure, the attacker found two independent chunks A_1 and A_2. We emphasize that in their attack procedure the starting step of forward and backward computations was exactly the beginning step of the block cipher E. Besides the two independent chunks, there were another chunk depending on both A_1 and A_2. As a result, the stored values in the two tables T_F and T_B are ones at different steps. The attacker adopted the partial-matching technique, so the distance between the steps for values in T_F and the steps for values in T_B can be at most 7. Finally the attack works on SHA-2 reduced to 24 steps.

3 Applying Previous Techniques to SHA-2

Our attack basically takes the same approach as Isobe and Shibutani [4], *i.e.* use the meet-in-the-middle attack. They could attack only 24 steps of SHA-2, hence the application to SHA-2 seems to be very limited. However, we found that the number of attacked steps could drastically increases by searching for a better neutral-word partition with considering various techniques on the meet-in-the-middle preimage attack. This section identifies how those techniques work on SHA-2 so that we can consider them when we search for a neutral-word partition.

3.1 Splice-and-cut technique.

The attack by Isobe and Shibutani [4] found 1-block preimages. In contrast, the splice-and-cut technique enables attackers to find 1-block pseudo-preimages. Then, they are converted to preimages with the conversion algorithm [7, Fact 9.99] by adding another message block. Therefore, by considering the splice-and-cut technique, the number of attacked steps may be extended because finding pseudo-preimages is usually easier than finding preimages.

3.2 Partial-matching technique

The partial-matching technique on SHA-2 was implied (but not explicitly mentioned) by Isobe and Shibutani [4]. Assume one chunk produces the value of p_j and the other chunk produces the value of p_{j+s}. The attacker wants to efficiently check whether or not p_j and p_{j+s} match without the knowledge of $W_j, W_{j+1}, \ldots, W_{j+s-1}$. In SHA-2, the maximum number of s is 7.

Assume the value of $p_{j+7} = A_{j+7} \| B_{j+7} \| \cdots \| H_{j+7}$ is known and W_{j+6} is unknown. By backward computation, we can obtain the values of $A_{j+6}, B_{j+6}, \ldots, G_{j+6}$. This is because $A_{j+6}, B_{j+6}, C_{j+6}$, E_{j+6}, F_{j+6}, and G_{j+6} are just copies of corresponding values in p_{j+7} and D_{j+6} is computed as follows.

$$D_{j+6} \leftarrow E_{j+7} - (A_{j+7} - (\Sigma_0(B_{j+7}) + Maj(B_{j+7}, C_{j+7}, D_{j+7}))). \tag{7}$$

By repeating the similar computation, in the end, A_j is computed from p_{j+7} without the knowledge of $W_j, W_{j+1}, \ldots, W_{j+6}$.

3.3 Partial-fixing technique

The partial-fixing technique on SHA-2 has not been considered in the previous work. Assume we can fix the lower x bits of the message word in each step. Under this assumption, 8 steps can be partially computed. Let us consider the step function of SHA-2 in the forward direction. Equations using W_j is as follows.

$$\begin{cases} T_1^{(j)} \leftarrow H_j + \Sigma_1(E_j) + Ch(E_j, F_j, G_j) + K_j + W_j, \\ A_{j+1} \leftarrow T_1^{(j)} + T_2^{(j)}, \quad E_{j+1} \leftarrow D_j + T_1^{(j)}. \end{cases} \tag{8}$$

If the lower x bits of W_j are fixed, the lower x bits of A_{j+1} (and E_{j+1}) can be computed independently of the upper $32 - x$ bits of W_j. Let us consider to skip another step in forward direction. The equation for A_{j+2} is as follows:

$$A_{j+2} \leftarrow T_1^{(j+1)} + \Sigma_0(A_{j+1}) + Maj(A_{j+1}, B_{j+1}, C_{j+1}). \tag{9}$$

We know only the lower x bits on A_{j+1}. Hence, we can compute Maj function for only the lower x bits. How about the Σ_0 function? We analyzed the relationship of the number of consecutive fixed bits from LSB in the input and output of $\sigma_0, \sigma_1, \Sigma_0$, and Σ_1. This is summarized in Table 2.

Table 2. Relationship of number of consecutive fixed bits from LSB in input and output of σ and Σ

	SHA-256				SHA-512			
	Σ_0	Σ_1	σ_0	σ_1	Σ_0	Σ_1	σ_0	σ_1
Input	x	x	x	x	x	x	x	x
output	$x-22$	$x-25$	$x-18$	$x-19$	$x-39$	$x-41$	$x-8$	$x-61$

When x agrees with the word size, the output is x. When the number described in the output is minus, the output is 0.

From Table 2, if x is large enough, we can compute the lower $x - 22$ bits of A_{j+2} in SHA-256 and the lower $x - 39$ bits in SHA-512, though the number of fixed bits greatly reduced after the Σ_0 function. This fact also implies that we cannot obtain the value of A_{j+3} since the number of fixed bits will be 0. In the end, we can conclude that the partial-fixing technique can be applied up to 2 steps in forward direction. Similarly, we considered the partial-fixing technique in backward direction, and found that it can be applied up to 6 steps. Hence the maximum number of steps that we can partially compute is 8 for both SHA-256 and SHA-512.

However we have another problem in the first assumption; the lower x bits of each message word can be fixed. This is difficult to achieve because the fixed bits in message words are mixed by the σ function in the message expansion. In fact, we could apply the partial-fixing technique for computing 1 step in forward, and 2 steps in backward for SHA-256. However, in SHA-512, the bit-mixing speed of σ is relatively slow due to the double word size. In fact, we could compute 2 steps in forward, and 6 steps in backward. Finally, 10 steps in total can be skipped by the partial-matching and partial-fixing techniques for SHA-256, and 15 steps for SHA-512. (These numbers of steps are explained in Section 5.)

3.4 Initial structure

We apply the initial structure technique proposed by Sasaki and Aoki [12] to SHA-2. They introduced the concept of the initial structure but did not give the details of how to construct it. We

manually found initial structures for skipping 2 steps and 3 steps. In Section 4, we search for a new neutral-word partition by considering both initial structures. As a result, we use the initial structure for skipping 2 steps in our attack. We show the initial structure for 2 steps in Fig. 2 and explain how it works in the below. We show the initial structure for 3 steps in Fig. 4 in Appendix A. In these figures, the order of additions is changed by equivalent transformation, addition of K_j is omitted for simplicity, and NW is an abbreviation of neutral word for a chunk.

Initial structure for 2 steps. The purpose of the initial structure shown in Fig. 2 is to guarantee that the change of a neutral word W_j does not impact the value of p_j, and the change of another neutral word W_{j+1} does not impact the value of p_{j+2}. This is achieved as follows.

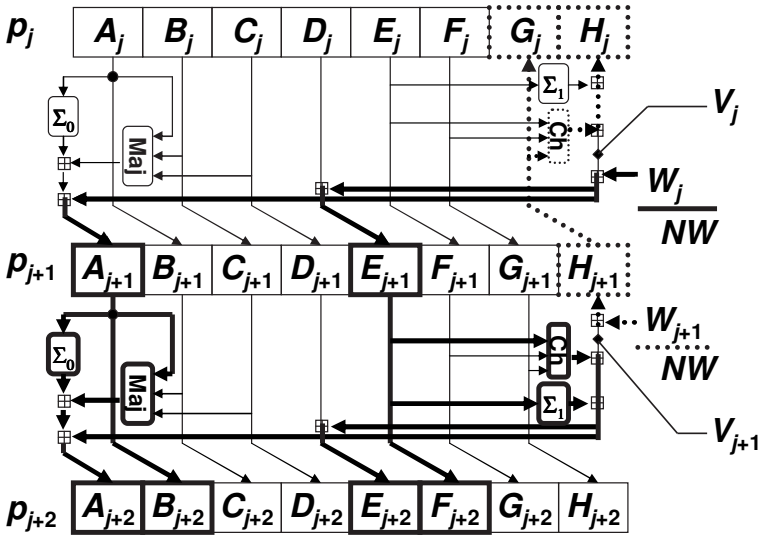

Fig. 2. Initial structure for 2 steps of SHA-2

Absorb W_{j+1}: Fix the values of V_j and V_{j+1} indicated in Fig. 2 to randomly chosen values. Every time we change the value of W_{j+1}, we change the value of H_{j+1} so that V_{j+1} will not change. Namely, we compute $H_{j+1} \leftarrow V_{j+1} - W_{j+1}$. The change of H_{j+1} will propagate through Ch function in Step j. We change the value of H_j to absorb the change of output of Ch function. Namely, we compute $H_j \leftarrow V_j - Ch(E_j, F_j, G_j) - \Sigma_1(E_j)$.

Absorb W_j: Every time we change the value of W_j, $T_1^{(j)}$ and $T_1^{(j+1)}$ in the step function are computed using the value of V_j and V_{j+1}, respectively. Namely, $T_1^{(j)} \leftarrow V_j + K_j + W_j$ or $T_1^{(j+1)} \leftarrow \Sigma_1(E_{j+1}) + Ch(E_{j+1}, F_{j+1}, G_{j+1}) + V_{j+1} + K_{j+1}$. In two steps, the change of W_j will propagate to $A_{j+2}, B_{j+2}, E_{j+2}$, and F_{j+2}.

Finally, we can make p_j independent of W_j and p_{j+2} independent of W_{j+1}.

As used in the above, fixing the value between two additions in different chunks, such as V_j and V_{j+1}, is useful to separate the computation into two independent parts. In fact, we use the similar method to analyze the independence of message words in Section 4.

4 Neutral-Word Partition

We search for neutral words that can attack as many steps as possible by considering techniques explained in Section 3. We consider the following properties when we search for the neutral words.

- The first and last steps can be regarded as consecutive by the splice-and-cut technique.
- 2 or 3 steps can be skipped at the begging of the two independent chunks by the initial structure.
- Up to 15 steps can be skipped at the last of the two independent chunks by the partial-matching and partial-fixing techniques.

The message expansion for SHA-2 is invertible. In other words, if consecutive intermediate 16 message words are fixed, all other message words are uniquely fixed. Hence, we focus on consecutive intermediate 16 message words $W_j, \ldots, W_{j+15}$ and next 16 expanded words $W_{j+16}, \ldots, W_{j+31}$. We separate these 32 steps into two independent chunks as shown in Fig. 3.

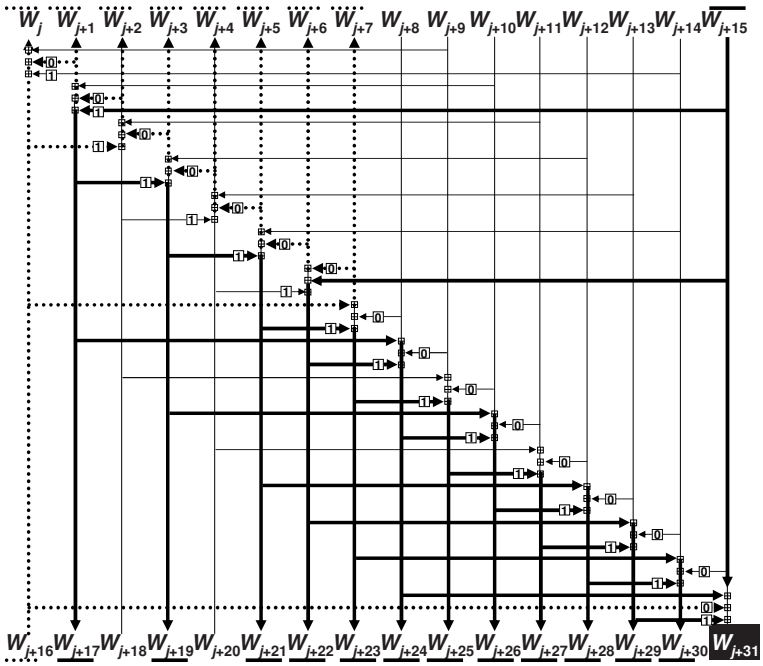

Fig. 3. Neutral word partition of SHA-2

In Fig. 3, message words for 31 steps ranging from W_j to W_{j+30} make two independent parts. We choose W_{j+15} and W_{j+16} as neutral words. The analysis of the message expansion is similar to that of the initial structure. When we change the value of W_{j+16}, we change the value of W_{j+7} so that the change of W_{j+16} does not impact W_{j+23}. This can be done by fixing the value of $W_{j+7} + W_{j+16}$ in advance as explained in Section 3.4. Then, the change of $\Sigma_0(W_{j+7})$ is absorbed by changing W_{j+6} accordingly. Similarly, as shown in Fig. 3, the change of W_{j+16} can be absorbed by changing $(W_j, W_{j+1}, \ldots, W_{j+7})$. Analysis on the neutral word W_{j+15} is the same. See Fig. 3 for details.

We applied the initial structure shown in Fig. 2 for Steps $j + 15$ and $j + 16$. Therefore, when we update the chaining variables using the step function, the change of W_{j+15} only gives influence to Step 17 and latter steps but never to Step 14 and the previous steps. Similarly, the change of W_{j+16} only gives influence to Step 14 and previous steps but never to Step 17 and the latter steps.

In the end, we can conclude that backward computation for Steps $j+16, j+14, j+13, \ldots, j$ and forward computation for steps $j+15, j+17, j+18, \ldots, j+30$, in total 31 steps can be computed independently.

As explained in Section 3, partial-matching and partial-fixing techniques can skip 10 steps in SHA-256 and 15 steps in SHA-512. Hence, 41 steps of SHA-256 and 46 steps of SHA-512 can be attacked. To apply the partial-fixing technique for SHA-256, we need to guarantee that several lower n bits of W_{j+31}, W_{j-1}, and W_{j-2} must be fixed. This can be achieved as follows:

- Due to the message expansion, as shown in Fig. 3, the change of W_{j+16} impacts W_{j+31}. We need to fix the lower n bits of W_{j+31} regardless of the value of W_{j+16}. To achieve this, we fix the lower n bits of $X(=\sigma_0(W_{j+16}))$, and choose all possible values for higher $32-n$ bits. Then, compute $\sigma_0^{-1}(X)$, and use them as values of W_{j+16}[3].
- Equations for W_{j-1} and W_{j-2} can be obtained by inverse of message expansion as follows.

$$W_{j-1} \leftarrow W_{j+15} - \sigma_1(W_{j+13}) - W_{j+8} - \sigma_0(W_j), \tag{10}$$
$$W_{j-2} \leftarrow W_{j+14} - \sigma_1(W_{j+12}) - W_{j+7} - \sigma_0(W_{j-1}). \tag{11}$$

  We need to fix the lower several bits of W_{j-1} and W_{j-2} regardless of the value of W_{j+15}. If we fix the lower n bits of W_{j+15}, the lower n bits of W_{j-1} are fixed. Furthermore, from Table 2, the lower $n-18$ bits of W_{j-2} are fixed.

5 Preimage Attack on SHA-2

For simplicity, we first explain the attack procedure for 40 steps of SHA-256. Then, we explain how to extend the attack to 41 steps and how to attack SHA-512. Due to the space limitation, we omit the details on SHA-512.

5.1 Attack procedure for 40-step SHA-256

The attack target is Steps 0 to 39 of SHA-256. We apply the message analysis shown in Fig. 3 to W_1 to W_{31}. In other words, we choose $j=1$ in Fig. 3. This attack generates 2-block preimages. The procedure for a given hash value *Hash* is as follows.

Set up

1. Set up the initial structure shown in Fig. 2 to Steps 16 to 17, *i.e.* fix the values of $A_{16}, B_{16}, C_{16}, D_{16}, E_{16}, F_{16}, V_{16}$, and V_{17} to randomly chosen values.
2. Set up the neutral word partition shown in Fig. 3, *i.e.* fix the values of $W_9, W_{10}, W_{11}, W_{12}$, and values between two additions for different chunks to randomly chosen values. Fix the values of W_{13}, W_{14}, and W_{15} to satisfy the padding string for a 2-block message.
3. For the partial-fixing technique, fix the lower 16 bits of W_{16} and $X(=\sigma_0(W_{17}))$ to randomly chosen values.

Forward computation

4. For all the higher 16 bits of W_{16},
   (a) Compute the values of $W_{18}, W_{20}, W_{22}, W_{23}, \ldots W_{31}$ by following the neutral word partition in Fig. 3. Then, compute the lower 16 bits of W_{32} by using the fixed lower 16 bits of X.
   (b) Compute p_{17} and p_{18} by following Fig. 2.

[3] How to compute σ^{-1} was described by Isobe and Shibutani [4, Appendix A].

(c) For j=18 to 31, compute $p_{j+1} \leftarrow R_j(p_j, W_j)$.
(d) Partially compute $p_{33} \leftarrow R_{32}(p_{32}, W_{32})$ by partial-fixing technique, *i.e.* compute the lower 16 bits of A_{33}.
(e) Store the obtained (p_{32}, A_{33}, W_{16}) in a list L.

Backward computation

5. For all the higher 16 bits of X,
(a) Compute $W_{17} \leftarrow \sigma_0^{-1}(X)$.
(b) Compute the values of $W_8, W_7, \ldots W_1$ by following the neutral word partition in Fig. 3. Then, compute the lower 16 bits of W_0 by using the fixed lower 16 bits of W_{16}.
(c) Compute p_{17} and p_{16} by following Fig. 2.
(d) For j=15 to 1, compute $p_j \leftarrow R_j^{-1}(p_{j+1}, W_j)$.
(e) Partially compute $p_0 \leftarrow R_0^{-1}(p_1, W_0)$ by partial-fixing technique, *i.e.* compute the lower 16 bits of H_0.
(f) Use the splice-and-cut technique, *i.e.* compute $p_{40} \leftarrow \mathit{Hash} - p_0$ for partially known p_0. Then, we obtain the values of $A_{40}, B_{40}, \ldots, G_{40}$ and the lower 16 bits of H_{40}.
(g) From the partially known p_{40}, compute the lower 16 bits of A_{33} by the partial-matching technique and check whether or not the values match the items in the list L.

Matching test

(h) If A_{33} is matched, compute W_{32} to W_{39} and W_0 following the message expansion and compute all bits of $H_{40}(= \mathit{Hash} - H_0)$ with the corresponding message.
(i) Compute $p_j \leftarrow R_j^{-1}(p_{j+1}, W_j)$ for $j = 39, 38, \ldots, 32$. Check whether or not the obtained p_{32} match those stored in the list L.
(j) If a match is found, the corresponding message and p_0 is a pseudo-preimage of *Hash*. Otherwise, repeat the attack until a pseudo-preimage is found.

Complexity evaluation. We assume that the speed of memory access is negligible compared to computation time of compression function and message expansion. The cost for set up is negligible. Step 4a costs roughly $2^{16} \cdot \frac{13}{40}$ message expansion operations. In Step 4b, p_{17} is obtained by increment of two variables, which requires negligible cost. Hence, this step costs $2^{16} \cdot \frac{1}{40}$ compression function operation to update p_{18}. Steps 4c and 4d cost $2^{16} \cdot \frac{15}{40} (= 2^{16} \cdot (\frac{14}{40} + \frac{1}{40}))$ SHA-256 compression function operations. Step 4e requires a memory of $2^{16} \cdot 10$ words. Steps 5a and 5b cost roughly $2^{16} \cdot \frac{9}{40}$ message expansion operations. Step 5c costs $2^{16} \cdot \frac{1}{40}$ compression function operation as similar to Step 4b. Steps 5d, 5e, and 5f cost $2^{16} \cdot \frac{17}{40} (= 2^{16} \cdot (\frac{15}{40} + \frac{1}{40} + \frac{1}{40}))$ compression function operations. In Step 5g, because the value of A_{33} is a copy of D_{36}, the cost for this step is $2^{16} \cdot \frac{4}{40}$ compression function operation.

In Step 5g, $2^{32} (= 2^{16} \cdot 2^{16})$ pairs are compared and we obtain $2^{16} (= 2^{32} \cdot 2^{-16})$ pairs that match 16 bits out of 256 bits. Step 5h costs roughly $2^{16} \cdot \frac{9}{40}$ message expansion operations. Step 5i can be performed step by step. Namely, we first compute unknown 16 bits of p_{32} for 2^{16} pairs and check the match of these 16 bits. This costs $2^{16} \cdot \frac{1}{40}$ compression function operation and we obtain $1 (= 2^{16} \cdot 2^{-16})$ matched pair. Since the number of remaining pair is enough reduced, the complexity for following steps is negligible. In the end, in Step 5i, we check whether the remaining 240 bits match or not. Hence, we obtain $2^{-224} (= 2^{16} \cdot 2^{-240})$ matched pair. Therefore, by repeating the above procedure 2^{224} times, we expect to obtain a pseudo-preimage. The time complexity for one iteration

is $2^{16} \cdot \frac{39}{40} (= 2^{16} \cdot (\frac{1}{40} + \frac{15}{40} + \frac{1}{40} + \frac{17}{40} + \frac{4}{40} + \frac{1}{40}))$ and $2^{16} \cdot \frac{31}{40} (= 2^{16} \cdot (\frac{13}{40} + \frac{9}{40} + \frac{9}{40}))$ message expansion operations, which is approximately 2^{16} SHA-256 operations. This is repeated 2^{224} times, hence, the total time complexity of this pseudo-preimage attack is 2^{240} SHA-256 operations. The dominant memory complexity is $2^{16} \cdot 10$ words for Step 4e. Finally, this attack is converted to a preimage attack with a complexity of 2^{249} by using the conversion algorithm explained in Section 2.2. Since $2^{16} \cdot 10$ words of memory is quite practical, we do not discuss the memoryless attack in this paper.

5.2 Attack on SHA-256 reduced to 41 steps

In the attack on 40 steps, 2 steps are skipped by the partial-fixing technique. To attack 41 steps, we skip one more step in backward direction. The attack target is steps 0 to 40, in total 41 steps of SHA-256. We choose $j = 2$ for neutral-word partition in Fig. 3. The initial structure and partial-fixing in forward are exactly the same. Hence, we apply initial structure to Steps 17 and 18.

All we have to do now is fixing the lower several bits of W_1 and W_0 regardless of the values of the neutral words for the other chunk. Equations for W_1 and W_0 are as follows. Here, underlined values are neutral word for other chunk and variables impacted by it. The small numbers besides the underline in parentheses represent the number of the lower fixed bits.

$$\underline{W_1}_{(25)} \leftarrow \underline{W_{17}}_{(25)} - \sigma_1(W_{15}) - W_{10} - \sigma_0(W_2), \tag{12}$$

$$\underline{W_0}_{(7)} \leftarrow W_{16} - \sigma_1(W_{14}) - W_9 - \underline{\sigma_0(\underline{W_1}_{(25)})}_{(7)}. \tag{13}$$

W_{17} is the neutral word for the other chunk. Fixing the lower 25 bits of W_{17}, fix the lower 25 bits of W_1. From Table 2, this fixes the lower 7 bits of $\sigma_0(W_1)$, hence the lower 7 bits of W_0 are fixed.

Whole attack procedure for 41 steps is almost the same as that for 40 steps. Because we fix 25 bits of the neutral word W_{17}, the free bits in the chunk is only 7. Hence, the required memory is reduced to $2^7 \cdot 10$ words, instead, the time complexity can be reduced from the brute force attack only by a factor of 2^7. Hence, the time complexity for pseudo-preimage attack is 2^{249}. This can be converted to preimage attack with a complexity of $2^{253.5}$ with the conversion algorithm in Section 2.2. Note that the number of fixed bits 25 was chosen so that the number of free bits in each neutral word and the number of match bits in the partial-fixing technique would be balanced.

5.3 Attacks on SHA-512

The attack on SHA-256 can be applied to SHA-512 as it is. Since the word size of SHA-512 is double of SHA-256, free bits in each chunk is also doubled. Hence each chunk has 32 free bits. This changes the required memory of the pseudo-preimage attack to $2^{32} \cdot 10$ words, and the attack becomes faster than the brute force attack by a factor of 2^{32}. In the end, time and memory complexity for the pseudo-preimage attack are 2^{480} and 2^{32}, respectively. This can be converted to a preimage attack with a complexity of 2^{497} by the conversion algorithm in Section 2.2.

We can extend the attack for 46 steps of SHA-512 by the partial-fixing technique, which partially compute 2 steps in forward and 6 steps in backward. For message-word partition, we choose $j = 6$ in Fig. 3. In forward direction, we want to fix the lower several bits of W_{j+31} and W_{j+32} regardless of the value of W_{j+16}. We found that by fixing the lower 56 bits of W_{j+16}, we can fix the lower 48 bits of W_{j+31} and the lower 56 bits of W_{j+32}. These message words enable us to compute the lower 48 bits of A_{j+32} and the lower 9 bits of A_{j+33}. In backward direction, we want to fix the lower several bits of $W_{j-1}, W_{j-2}, W_{j-3}, W_{j-4}, W_{j-5}$, and W_{j-6} regardless of the free bits of W_{j+15}. We found that by fixing the lower 61 bits of W_{j+15}, we can fix the lower 61 bits of W_{j-1}, the lower

53 bits of W_{j-2}, the lower 45 bits of W_{j-3}, the lower 37 bits of W_{j-4}, the lower 29 bits of W_{j-5}, and the lower 21 bits of W_{j-6}. These message words enable us to compute the lower 61 bits of H_{j-1}, the lower 53 bits of H_{j-2}, the lower 45 bits of H_{j-3}, the lower 20 bits of H_{j-4}, the lower 12 bits of H_{j-5}, and the lower 4 bits of H_{j-6}. Finally, we confirmed that these partially computed values can yield the lower 4 bits of A_{j-13} by the partial-matching technique. The number of free bits in the neutral word W_{j+15} is 3 bits and we can check the match of 4 bits by the partial-fixing technique. The time complexity for a pseudo-preimage attack on 46 steps SHA-512 is 2^{509}, and this is converted to a preimage attack with a complexity of $2^{511.5}$. The memory requirement is $2^3 \cdot 10$ words. Satisfying message padding is also possible. Since j is 6, W_{14} and W_{15} can be fixed to any value. To satisfy the 1 bit of W_{13} we need to fix the MSB of W_{13}. This is achieved by fixing the higher 2 bits of W_{22} and constant values in neutral-word partition appropriately.

We also considered the attack complexity on SHA-512 reduced to $45, 44, \ldots, 41$ steps. These results are listed in Table 1.

6 Conclusions

In this paper, we presented preimage attacks on 41 steps SHA-256 and 46 steps SHA-512. The time complexity of the attack for 41-step SHA-256 is $2^{253.5}$ and it requires $2^{16} \cdot 10$ words of memory. The time complexity of the attack for 46-step SHA-512 is $2^{511.5}$ and it requires $2^3 \cdot 10$ words of memory. The number of attacked steps is greatly improved from the best previous attack, in other words, the security margin of SHA-256 and SHA-512 is greatly reduced. Because SHA-256 and SHA-512 have 64 and 80 steps, respectively, they are currently secure. However, considering that the attack may be improved more in the future, we should pay attention to the security of SHA-2.

References

1. Kazumaro Aoki and Yu Sasaki. Preimage attacks on one-block MD4, 63-step MD5 and more. In *Workshop Records of SAC 2008*, pages 82–98, Sackville, Canada, 2008.
2. Kazumaro Aoki and Yu Sasaki. Meet-in-the-middle preimage attacks against reduced SHA-0 and SHA-1. In Shai Halevi, editor, *Advances in Cryptology — CRYPTO 2009*, Lecture Notes in Computer Science, Berlin, Heidelberg, New York, 2009. Springer-Verlag. to appear.
3. Sebastiaan Indesteege, Florian Mendel, Bart Preneel, and Christian Rechberger. Collisions and other non-random properties for step-reduced SHA-256. In *Workshop Records of SAC 2008*, pages 257–274, 2008.
4. Takanori Isobe and Kyoji Shibutani. Preimage attacks on reduced Tiger and SHA-2. In *Fast Software Encryption 2009 Preproceedings*, pages 141–158, 2009.
5. Gaëtan Leurent. MD4 is not one-way. In Kaisa Nyberg, editor, *Fast Software Encryption (FSE 2008)*, volume 5086 of *Lecture Notes in Computer Science*, pages 412–428, Berlin, Heidelberg, New York, 2008.
6. Florian Mendel, Norbert Pramstaller, Christian Rechberger, and Vincent Rijmen. Analysis of step-reduced SHA-256. In Matt Robshaw, editor, *Fast Software Encryption — 13th International Workshop, FSE 2006*, volume 4047 of *Lecture Notes in Computer Science*, pages 126–143, Berlin, Heidelberg, New York, 2006. Springer-Verlag.
7. Alfred John Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of applied cryptography*. CRC Press, 1997.
8. Ronald L. Rivest. *Request for Comments 1321: The MD5 Message Digest Algorithm*. The Internet Engineering Task Force, 1992. (`http://www.ietf.org/rfc/rfc1321.txt`).
9. Markku-Juhani O. Saarinen. A meet-in-the-middle collision attack against the new FORK-256. In Kannan Srinathan, Chanrasekharan Pandu Rangan, and Moti Yung, editors, *Progress in Cryptology – INDOCRYPT 2007*, volume 4859 of *Lecture Notes in Computer Science*, pages 10–17, Berlin, Heidelberg, New York, 2007. Springer-Verlag.
10. Somitra Kumar Sanadhya and Palash Sarkar. New collision attacks against up to 24-step SHA-2 (extended abstract). In Vincent Rijmen, Abhijit Das, and Dipanwita Roy Chowdhury, editors, *Progress in Cryptology – INDOCRYPT 2008*, volume 5365 of *Lecture Notes in Computer Science*, pages 91–103, Berlin, Heidelberg, New York, 2008. Springer-Verlag.

11. Yu Sasaki and Kazumaro Aoki. Preimage attacks on 3, 4, and 5-pass HAVAL. In Josef Pawel Pieprzyk, editor, *Advances in Cryptology - ASIACRYPT 2008*, volume 5350 of *Lecture Notes in Computer Science*, pages 253–271, Berlin, Heidelberg, New York, 2008. Springer-Verlag.
12. Yu Sasaki and Kazumaro Aoki. Finding preimages in full MD5 faster than exhaustive search. In Antoine Joux, editor, *Advances in Cryptology — EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 134–152, Berlin, Heidelberg, New York, 2009. Springer-Verlag.
13. U.S. Department of Commerce, National Institute of Standards and Technology. *Federal Register /Vol. 72, No. 212/Friday, November 2, 2007/Notices*, 2007. (`http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf`).
14. U.S. Department of Commerce, National Institute of Standards and Technology. *Secure Hash Standard (SHS) (Federal Information Processing Standards Publication 180-3)*, 2008. (`http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf`).
15. U.S. Department of Commerce, National Institute of Standards and Technology. *NISTfs Plan for Handling Tunable Parameters*, February 2009. Presentation by Souradyuti Paul at The First SHA-3 Candidate Conference. (`http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/Feb2009/`).
16. Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full SHA-1. In Victor Shoup, editor, *Advances in Cryptology — CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 17–36, Berlin, Heidelberg, New York, 2005. Springer-Verlag.
17. Hongbo Yu and Xiaoyun Wang. Non-randomness of 39-step SHA-256, 2008. `http://www.iacr.org/conferences/eurocrypt2008v/index.html`.
18. Yuliang Zheng, Josef Pieprzyk, and Jennifer Seberry. HAVAL — one-way hashing algorithm with variable length of output. In Jennifer Seberry and Yuliang Zheng, editors, *Advances in Cryptology — AUSCRYPT'92*, volume 718 of *Lecture Notes in Computer Science*, pages 83–104. Springer-Verlag, Berlin, Heidelberg, New York, 1993.

A Other initial structures for SHA-2

A.1 Initial structure for 3 steps.

Construction of the initial structure for 3 steps shown in Fig. 4 is similar to that for 2 steps. However, we need additional effort in Step $j+1$. One of the input variables to Ch function in Step $j+1$, *i.e.* G_{j+1}, changes depending on the value of W_{j+2} and another input variable E_{j+1} changes depending on the value of W_j. Hence, in the straightforward manner, the output of Ch function depends on both W_j and W_{j+2}. To make independence on this Ch function, we change W_j so that the lower n bits of E_{j+1} is always fixed. Similarly, we change W_{j+2} so that the higher $32-n$ ($64-n$ for SHA-512) bits of G_{j+1} is always fixed. Due to this, the lower n bits of the output of the Ch function depend on only W_{j+2} and the higher $32-n$ bits depend on only W_j. In details, we compute H_{j+1} and $T_1^{(j+1)}$ as shown below. Let $Ch^H \| Ch^L$ be the output of $Ch(E_{j+1}, F_{j+1}, G_{j+1})$ and Ch^L is n-bit long.

$$T_1^{(j+1)} \leftarrow V_{j+1} + \Sigma_1(E_{j+1}) + 0 \| Ch^L, \tag{14}$$

$$H_{j+1} \leftarrow V_{j+1} - W_{j+1} - K_{j+1} - Ch^H \| 0. \tag{15}$$

Finally, we can separate Step $j+1$ into two independent parts.

A.2 Initial structure for 4 steps or more.

We found skipping 4 steps or more seems to be difficult. To construct the initial structure for 4 steps, we have to make independent part in Ch function in both Steps $j+1$ and $j+2$. Let us fix the lower n bits of W_j, and this fixes the lower n bits of E_{j+1}. This can fix the lower n bits of Ch function in Step $j+1$, however, fixed positions are mixed when it goes through Σ_1 function. Hence, we cannot efficiently control the fixed bit positions of E_{j+2} and Ch in Step $j+2$. As a result, we cannot construct initial structure for 4 steps or more.

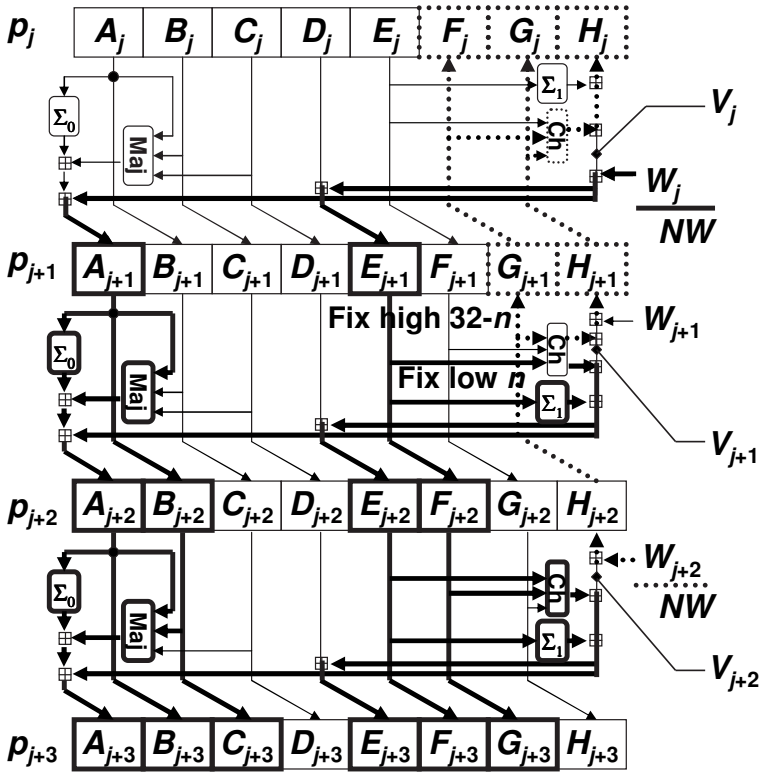

Fig. 4. Initial structure for 3 steps of SHA-2