Converting Meet-In-The-Middle Preimage Attack into Pseudo Collision Attack: Application to SHA-2

Ji Li[1], Takanori Isobe[2], and Kyoji Shibutani[2]

[1] Sony China Research Laboratory, China
Ji.Li@sony.com.cn
[2] Sony Corporation, Japan
{Takanori.Isobe,Kyoji.Shibutani}@jp.sony.com

Abstract. In this paper, we present a new technique to construct a collision attack from a particular preimage attack which is called a partial target preimage attack. Since most of the recent meet-in-the-middle preimage attacks can be regarded as the partial target preimage attack, a collision attack is derived from the meet-in-the-middle preimage attack. By using our technique, pseudo collisions of the 43-step reduced SHA-256 and the 46-step reduced SHA-512 can be obtained with complexities of 2^{126} and $2^{254.5}$, respectively. As far as we know, our results are the best pseudo collision attacks on both SHA-256 and SHA-512 in literature. Moreover, we show that our pseudo collision attacks can be extended to 52 and 57 steps of SHA-256 and SHA-512, respectively, by combined with the recent preimage attacks on SHA-2 by bicliques. Furthermore, since the proposed technique is quite simple, it can be directly applied to other hash functions. We apply our algorithm to several hash functions including Skein and BLAKE, which are the SHA-3 finalists. We present not only the best pseudo collision attacks on SHA-2 family, but also a new insight of relation between a meet-in-the-middle preimage attack and a pseudo collision attack.

Keywords: hash function, narrow-pipe, SHA-2, Skein, BLAKE, meet-in-the-middle attack, preimage attack, pseudo collision attack.

1 Introduction

Cryptographic hash functions play a central role in the modern cryptography. A secure hash function, which produces a fixed length hash value from an arbitrary length message, is required to satisfy at least three security properties: preimage resistance, second preimage resistance and collision resistance.

While there has not been a generic method to convert a collision attack into a preimage attack, it has been known that the preimage attack that can find at least two distinct preimages from the same target can be directly converted into a collision attack. However, the converted collision attack is often not efficient due to that the birthday bound of a collision attack ($2^{n/2}$) is far lower than

A. Canteaut (Ed.): FSE 2012, LNCS 7549, pp. 264–286, 2012.

the generic bound of the preimage attack (2^n), where n is the bit size of the hash value. Thus, it is left as open question that how to convert an efficient preimage attack into an efficient collision attack. In the case of the reduced SHA-256 regarding the number of attacked rounds, a preimage attack, covering 43 steps [4], is much better than the best known collision attack, with only 27 steps [17]. Moreover, basically, a collision attack and a preimage attack require quite different techniques. In other words, in general, the techniques used for the collision attack do not work well for a preimage attack, and vice versa. In fact, most of the recent collision attacks are based on a differential attack [32,31], in contrast to that most of the recent preimage attacks are based on a meet-in-the-middle (MITM) attack [2]. Though converting the differential collision attack to a (pseudo) preimage attack was discussed in [8], there is no generic way to construct a collision attack from a MITM preimage attack.

In this paper, we give a generic method to convert a particular preimage attack into a collision attack. By using our technique, an efficient collision attack which works faster than a generic collision attack can be constructed from a partial target preimage attack even if the complexity of the preimage attack is more than the birthday bound ($2^{n/2}$). Our method is especially fit for converting a MITM preimage attack into a pseudo collision attack, since most of the recent MITM preimage attacks can be considered as the partial target preimage attack as long as its matching point is located in the end of the compression function. We first apply our algorithm to SHA-256 and SHA-512 and show the best pseudo collision attacks on them in literature. Specifically, pseudo collisions of the 43-step (out of 64-step) reduced SHA-256 and the 46-step (out of 80-step) reduced SHA-512 can be derived faster than a generic attack. Combined with the recent preimage attacks on SHA-2 [14], these attacks are extended to the 52-step and 57-step reduced SHA-256 and SHA-512, respectively. Then we show some other applications of our conversion techniques including a pseudo collision attack on the 37-round reduced Skein-512 and pseudo collision attacks on the 4-round reduced BLAKE-256/512 without the initialization function. While it seems hard to extend our pseudo collision attacks to collision attacks, the proposed conversion technique is a generic, and thus it is expected to be widely used for security evaluations of hash functions.

This paper is organized as follows. Some security notions and a meet-in-the-middle preimage attack are introduced in Section 2. Section 3 introduces our approach for constructing a pseudo collision attack. Then, applications of our technique to SHA-256 and SHA-512 are presented in Section 4. The result on Skein is described in Section 5. Finally, we conclude in Section 6.

2 Preliminaries

In this section, we first give security notions used throughout this paper, then briefly refer a meet-in-the-middle (MITM) preimage attack.

2.1 Security Notions

Let f be a compression function which outputs an n-bit chaining variable h_i from an n-bit input chaining variable h_{i-1} and a k-bit input message m_i, i.e., $h_i = f(h_{i-1}, m_i)$. Similarly, let H be an iterated hash function consisting of f, which produces an n-bit hash value d from an initial value $IV(= h_0)$ and an arbitrary length message M, i.e., $d = H(IV, M) = f(\cdots f(f(IV, m_1), m_2), \cdots, m_t)$, where $pad(M) = (m_1|m_2|\cdots|m_t)$ and pad denotes a padding function. This type of hash function, in which the size of an intermediate chaining variable is the same as that of a hash value, is called a *narrow-pipe* hash function. On the other hand, a hash function having a larger internal state size is called a *wide-pipe* hash function, i.e., the size of a final hash value is smaller than that of a chaining variable. We use the terminology introduced in [15] for a collision attack and a pseudo (or free-start) collision attack on hash functions as follows.

Definition 1 (Collision attack). *Given IV, find (M, M') such that $M \neq M'$ and $H(IV, M) = H(IV, M')$.*

Definition 2 (Free-start or pseudo collision attack). *Find (IV, IV', M, M') such that $H(IV, M) = H(IV', M')$ and $(IV, M) \neq (IV', M')$.*

Additionally, we give several definitions for (pseudo) preimage attacks on hash functions and (pseudo) preimage attacks on compression functions.

Definition 3 (Preimage attack). *Given IV and $d(= H(IV, M))$, find M' such that $H(IV, M') = d$.*

Definition 4 (Pseudo preimage attack). *Given $d(= H(IV, M))$, find (IV', M') such that $H(IV', M') = d$.*

Definition 5 ($(t$-bit) partial target preimage attack). *Given IV and t-bit partial target of $d(= H(IV, M))$, find M' such that t-bit of $d'(= H(IV, M'))$ is the same as the t-bit of d at the same position, and the other part of d' is randomly obtained.*

Definition 6 (Preimage attack on compression function). *Given h_{i-1} and $h_i(= f(h_{i-1}, m_i))$, find m'_i such that $f(h_{i-1}, m'_i) = h_i$.*

Definition 7 (Pseudo preimage attack on compression function). *Given $h_i(= f(h_{i-1}, m_i))$, find (h'_{i-1}, m'_i) such that $f(h'_{i-1}, m'_i) = h_i$.*

Definition 8 ($(t$-bit) partial target preimage attack on compression function). *Given h_{i-1} and t-bit partial target of $h_i(= f(h_{i-1}, m_i))$, find m'_i such that t-bit of $h'_i(= f(h_{i-1}, m'_i))$ is the same as the t-bit of h_i at the same position, and the other part of h'_i is randomly obtained.*

Definition 9 ($(t$-bit) pseudo partial target preimage attack on compression function). *Given t-bit partial target of $h_i(= f(h_{i-1}, m_i))$, find (h'_{i-1}, m'_i) such that t-bit of $h'_i(= f(h'_{i-1}, m'_i))$ is the same as the t-bit of h_i at the same position, and the other part of h'_i is randomly obtained.*

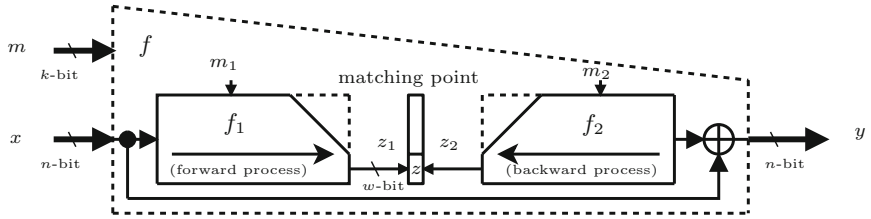

Fig. 1. Meet-in-the-middle preimage attack

2.2 Meet-In-The-Middle Preimage Attack

The basic concept of the MITM preimage attack was introduced in [22,16]. Since then, the MITM preimage attacks have been drastically improved and applied to several hash functions [2,28,27,3,13,4,10]. Also, the techniques for the MITM preimage attacks on hash functions have been extended to the attacks on several block ciphers [7,12].

As shown in Fig. 1,[1] in the MITM preimage attack on a compression function, the compression function f is assumed to be divided into two sub-functions: f_1 (forward process) and f_2 (backward process) so that the w-bit matching point z calculated by f_1 does not depend on m_2 which is some message bits of m, and z calculated by f_2 does not depend on m_1 which is other message bits of m. Such m_1 and m_2 are called neutral bits of f_2 and f_1, respectively. Then, the MITM preimage attack finds a preimage m' such that $f(x, m') = y$ from a given x and $y(= f(x, m))$ as follows.

Step 1. Choose a random m except for m_1 and m_2.
Step 2. For all possible m_1, calculate w-bit $z_1(= f_1(x, m_1))$, and add a pair of $(z_1^{(i)}, m_1^{(i)})$ to a list, where $(1 \leq i \leq 2^{|m_1|})$, and $|*|$ denotes the bit size of $*$.
Step 3. For all possible m_2, calculate w-bit $z_2(= f_2^{-1}(x \oplus y, m_2))$, and add a pair of $(z_2^{(j)}, m_2^{(j)})$ to a list, where $(1 \leq j \leq 2^{|m_2|})$.
Step 4. Compare two lists to find pairs satisfying $z_1^{(p)} = z_2^{(q)}$. If such pair is found, then check if the other bits of the matching point derived from $m_1^{(p)}$ and $m_2^{(q)}$ are the same value.
Step 5. If the other parts are also the same, then outputs such m including $m_1^{(p)}$ and $m_2^{(q)}$. Otherwise, go back to Step 1 and repeat the computation.

From Steps 2 and 3, we have $2^{|m_1|}$ and $2^{|m_2|}$ values of w-bit z_1 and z_2, i.e., we have $2^{|m_1|+|m_2|}$ values of $(z_1 \oplus z_2)$. Since the probability of $(z_1 \oplus z_2 = 0)$ is 2^{-w}, we have $2^{|m_1|+|m_2|} \cdot 2^{-w}$ pairs such that $z_1 = z_2$ in Step 4. Thus, by repeating this algorithm about $2^{n-w} \cdot 2^{-(|m_1|+|m_2|)} \cdot 2^w$ times, we expect to obtain a desired preimage. The required computation for the one process from

[1] Here, we show the MITM preimage attack on Davies-Meyer mode as an example. MITM preimage attacks on other modes like Matyas-Meyer-Oseas mode can be performed in a similar way.

Step 1 to 4 is at most $\max(2^{|m_1|}, 2^{|m_2|})$ calls of the compression function. Thus, the total computation to find a preimage of the compression function is about $2^n \cdot 2^{-(|m_1|+|m_2|)} \cdot \max(2^{|m_1|}, 2^{|m_2|})$.[2]

For a narrow-pipe hash function, by replacing x and y by IV and d, this MITM preimage attack on a compression function can be directly converted into a preimage attack on a hash function. However, for an attack on a hash function, some of the message bits related to the padding bits are required to be controlled by the attacker to set appropriate padding data.

3 Method to Convert Preimage Attack into Collision Attack

In this section, we present how to efficiently convert a particular preimage attack into a pseudo collision attack. First, we introduce a generic technique to construct a pseudo collision attack from a partial target preimage attack. Then, we introduce the MITM preimage attack whose matching point is located at the end of the compression function. We show that such class of the MITM preimage attack is regarded as the partial target preimage attack. Finally, we show that a pseudo collision attack can be efficiently constructed from the MITM preimage attack whose matching point is at the end by showing how to efficiently obtain many partial target preimages.

3.1 Generic Conversion of Partial Target Preimage Attack into Collision Attack

We consider the oracle $\mathcal{A}$ that can find a t-bit partial target preimage with a complexity of 2^s. Also, $\mathcal{A}$ is assumed to return different M' for each call. Obviously, we can construct a collision attack with a complexity of $2^s \cdot 2^{(n-t)/2}$ by iteratively calling $\mathcal{A}$ as follows.

- Set t-bit random data as d'
- Call $\mathcal{A}$ with the parameter IV and d' in $2^{(n-t)/2}$ times

After this procedure, we have $2^{(n-t)/2}$ of $(n-t)$-bit random data, and thus there exists a colliding data with a high probability. Once the colliding data are found, we have a collision of the hash function since the rest of the hash value d' is fixed. The total complexity is $2^{(n-t)/2} \cdot 2^s$. The memory requirement can be reduced to the memory requirement of finding a partial target preimage by using memory free birthday attack [29,21]. This conversion itself can be applied to not only a narrow-pipe hash function but also a wide-pipe hash function, since the required complexity depends only on the size of the digest. The basic

[2] The estimated complexity does not depend on the size of the matching point w. However, as discussed in [10], if w is extremely small like $w = 1$, the total complexity is dominated by the recomputations in Step 4 which is ignored in our estimation. Thus, in our evaluation, we assume that w is sufficiently large.

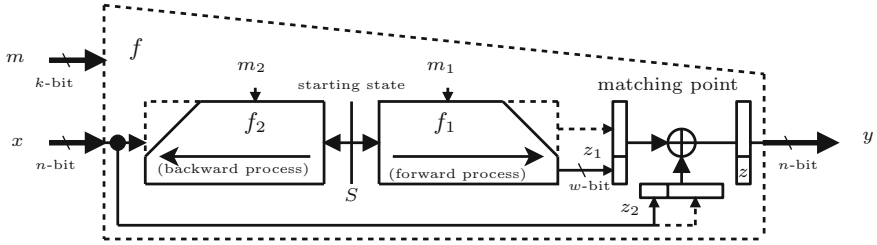

Fig. 2. MITM preimage attack with the matching point in the last step

concept of this attack that fixes t-bit of the target with the complexity of 2^s has been used to find a collision of (new) FORK-256 in [22] and a collision and a second preimage of LUX in [33]. However, the method does not work if the partial target preimage attack is not efficient, i.e., $(s \geq t/2)$. In this case, the required complexity in total will be higher than $2^{n/2}$.

3.2 Meet-In-The-Middle Attack with Matching Point in Last Step

We consider a similar model explained in Section 2.2. The difference from the model shown in Fig. 1 is that the matching point is restricted to be in the last step as shown in Fig. 2. In this scenario, the MITM pseudo preimage attack on a compression function finds a preimage m' and a random x' such that $f(x', m') = y$ from a given $y(= f(x, m))$ as follows.

Step 1. Choose a random m except for m_1 and m_2, and a random starting state S.

Step 2. For all possible m_1, calculate w-bit $z_1(= f_1(S, m_1))$, and add a pair of $(z_1^{(i)}, m_1^{(i)})$ to a list, where $(1 \leq i \leq 2^{|m_1|})$.

Step 3. For all possible m_2, calculate w-bit $z_2(= f_2^{-1}(S, m_2))$, and add a pair of $(z_2^{(j)}, m_2^{(j)})$ to a list, where $(1 \leq j \leq 2^{|m_2|})$.

Step 4. Compare two lists to find pairs satisfying that $z_1^{(p)} \oplus z_2^{(q)}$ equals the t-bit of y. If such pair is found, then check if the XORed other bits of the matching point derived from $m_1^{(p)}$ and $m_2^{(q)}$ is the same as the rest of y.

Step 5. If the XORed other bits are also the same as y, then output such m including $m_1^{(p)}$ and $m_2^{(q)}$, and x' calculated from the data of the matching point. Otherwise, go back to Step 1 and repeat the computation.

Note that, this attack basically cannot obtain a preimage from the given x unlike the attack described in Section 2.2, since x' will be randomly derived. Thus, this attack is considered as a pseudo preimage attack on a compression function. However, for a narrow-pipe hash, it has been known that a pseudo preimage attack on a compression function can be converted into a preimage attack on a hash function assuming that the attacker can set valid padding bits [19,10]. The estimated complexity to find a desired pseudo preimage is the same as that presented in Section 2.2, i.e., $2^n \cdot 2^{-(|m_1|+|m_2|)} \cdot \max(2^{|m_1|}, 2^{|m_2|})$.

3.3 Conversion of MITM Preimage Attack into Pseudo Collision Attack

If we can construct the MITM pseudo preimage attack whose matching point is located at the end of the compression function, we can control part of the output variables as explained in the previous subsection. In other words, the MITM pseudo preimage attack described in the previous subsection can be regarded as the pseudo partial target preimage attack on a compression function. For the MITM preimage attack, at least $2^{t/2}$ computations are required to derive a preimage of an t-bit partial target. Thus, the directly converted pseudo collision attack will at least have the complexity of $2^{(n-t)/2+t/2} = 2^{n/2}$, that is not an efficient pseudo collision attack.

In order to overcome this problem, we exploit extra freedom of a neutral word after finding a partial target preimage. For example, in the case of $t = 10$ and $|m_1| = |m_2| = 8$ ($> t/2$), we can find $2^6 (= 2^{8+8}/2^{10})$ 10-bit partial target preimages with the complexity of 2^8. It essentially means that a 10-bit partial target preimage is found with the complexity of $2^2 (= 2^8/2^6) < 2^5 (= 2^{10/2})$. When $t \leq w$, the required complexity to find a partial target preimage from a given t-bit partial target is estimated as

$$2^{t-(|m_1|+|m_2|)} \cdot \max(2^{|m_1|}, 2^{|m_2|}),$$

where recall that w denotes the bit size of the matching point. In particular, $s < t/2$, which is the condition for a successful attack as mentioned in Section 3.1, holds when $\min(|m_1|, |m_2|) > t/2$, where recall that 2^s represents the required complexity to find a t-bit partial target preimage. Therefore, if we can move the matching point of the MITM attack to the end of the compression function and there is enough freedom in neutral words, we can construct an efficient pseudo collision attack on a compression function.

Moreover, for a narrow-pipe hash function, it has been known that a (pseudo) collision attack on a compression function can be directly converted to a (pseudo) collision attack on a hash function by appending another message block illustrated in Fig. 3, which is called multi-block message technique. By using the multi-block message technique, an attacker can append arbitrary messages. Thus, unlike the conversion to a (pseudo) preimage attack on a hash function, for the conversion to a pseudo collision attack on a hash function, there is no restriction on controllability of message bits for a MITM pseudo preimage attack on a compression function. This will relax conditions on the position of the matching point for the MITM pseudo preimage attack on a compression function, and thus may allow us to attack larger number of steps. Note that, for a wide-pipe hash function, even though a (pseudo) collision attack on a compression function can not be directly converted to a (pseudo) collision attack on a hash function by using multi-block message, we still can convert a MITM pseudo preimage attack on a hash function to a pseudo collision attack on a hash function since the conversion of a partial target preimage attack into a collision attack is generic. Furthermore, in our attack, t-bit of the colliding digest can be determined by the attacker unlike the usual collision attack that derives a completely random

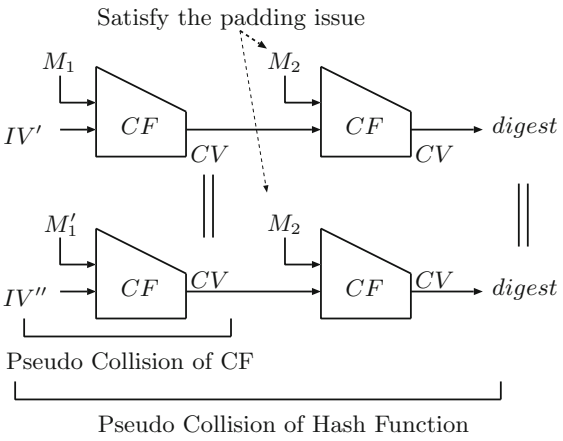

Fig. 3. Multi-block pseudo collision

digest. This is another feature of our approach. On the other hand, the required complexity of our converted (pseudo) collision attack is likely to be high due to a few gains from the MITM procedure, though it is still more efficient than the generic attack. This is considered as one of the limitations of our approach.

4 Pseudo Collision Attacks on SHA-2

In this section, we apply our conversion technique to SHA-2. At first, we briefly describe the algorithm of SHA-2. Then, we review the previous collision attacks on SHA-2. After that, we introduce the known MITM preimage attack on the 43-step SHA-256 presented in [4]. After we modify these results in order to fit our conversion technique, i.e., moving the matching point to the end of the compression function, we show the pseudo collision attack on the 43-step SHA-256. Moreover, we present the pseudo collision attack on the 46-step SHA-512 based on the MITM preimage attack on the 46-step SHA-512 [4]. Furthermore, pseudo collision attacks on the 40-step reduced SHA-224 and SHA-384 are demonstrated as well. Finally, we discuss pseudo collision attacks based on the recent MITM preimage attacks [14], which significantly improve the results of [4] in terms of the number of attacked steps by using *bicliques*. These results on SHA-2 are summarized in Table 1.

4.1 Description of SHA-2

While our target is both SHA-256 and SHA-512, we only explain the structure of SHA-256, since SHA-512 is structurally equivalent to SHA-256 except for the number of steps, the amount of rotations and the word size. The compression function of SHA-256 consists of a message expansion function and a state update function. The message expansion function expands a 512-bit message block into

Table 1. Summary of collision attacks on the reduced SHA-2

algorithm	type of attack	steps	complexity	based attack	paper
SHA-256	collision	24	$2^{28.5}$	-	[11]
	collision	27	(practical)	-	[17]
	semi-free-start-collision^{*1}	24	2^{17}	-	[11]
	semi-free-start-collision^{*1}	32	(practical)	-	[17]
	pseudo-near-collision	31	2^{32}	-	[11]
	pseudo collision	42	2^{123}	[4]	Our (Section 4.7)
	pseudo collision	43	2^{126}	[4]	Our (Section 4.4)
	pseudo collision	45	$2^{126.5}$	[14]	Our (Section 4.9)
	pseudo collision	52	$2^{127.5}$	[14]	Our (Section 4.9)
SHA-224	pseudo collision	40	2^{110}	[4]	Our (Section 4.8)
SHA-512	collision	24	$2^{28.5}$	-	[11]
	pseudo collision	42	2^{244}	[4]	Our (Section 4.7)
	pseudo collision	46	$2^{254.5}$	[4]	Our (Section 4.6)
	pseudo collision	50	$2^{254.5}$	[14]	Our (Section 4.9)
	pseudo collision	57	$2^{255.5}$	[14]	Our (Section 4.9)
SHA-384	pseudo collision	40	2^{183}	[4]	Our (Section 4.8)

$*1$: semi-free-start-collision attack finds (IV', M, M') such that $H(IV', M) = H(IV', M')$ and $M \neq M'$.

64 32-bit message words $(W_0, \cdots, W_{63})$ as follows:

$$W_i = \begin{cases} M_i & (0 \leq i < 16), \\ \sigma_1(W_{i-2}) + W_{i-7} + \sigma_0(W_{i-15}) + W_{i-16} & (16 \leq i < 64), \end{cases}$$

where the functions $\sigma_0(X)$ and $\sigma_1(X)$ are defined by

$$\sigma_0(X) = (X \ggg 7) \oplus (X \ggg 18) \oplus (X \gg 3),$$
$$\sigma_1(X) = (X \ggg 17) \oplus (X \ggg 19) \oplus (X \gg 10).$$

The state update function updates eight 32-bit chaining variables, A, B, $\cdots$, G, H in 64 steps as follows:

$$T_1 = H_i + \Sigma_1(E_i) + Ch(E_i, F_i, G_i) + K_i + W_i,$$
$$T_2 = \Sigma_0(A_i) + Maj(A_i, B_i, C_i),$$
$$A_{i+1} = T_1 + T_2,\ B_{i+1} = A_i,\ C_{i+1} = B_i,\ D_{i+1} = C_i,$$
$$E_{i+1} = D_i + T_1,\ F_{i+1} = E_i,\ G_{i+1} = F_i,\ H_{i+1} = G_i,$$

where K_i is the i-th step constant and the functions Ch, Maj, Σ_0 and Σ_1 are given as follows:

$$Ch(X, Y, Z) = XY \oplus \overline{X}Z,$$
$$Maj(X, Y, Z) = XY \oplus YZ \oplus XZ,$$
$$\Sigma_0(X) = (X \ggg 2) \oplus (X \ggg 13) \oplus (X \ggg 22),$$
$$\Sigma_1(X) = (X \ggg 6) \oplus (X \ggg 11) \oplus (X \ggg 25).$$

After 64 steps, a feed-forward process is executed with initial state variables by using word-wise addition modulo 2^{32}.

4.2 Known Collision Attacks on SHA-2

The first collision attack on reduced SHA-256 was presented in [18] which is a 19-step near collision attack. Since then, the collision attacks on SHA-2 have been improved [20,23,25,24,26,11,17]. The previously published best collision attacks in terms of the number of attacked steps are the 27 steps on SHA-256 [17] and the 24 steps on SHA-512 [11,25]. A non-random property, which is a second-order differential collision, of the 47-step reduced SHA-256 compression function was reported in [6].

4.3 Known MITM Preimage Attack on 43-Step SHA-256 [4]

The MITM preimage attack on the 43-step SHA-256 presented in [4] uses the 33-step two chunks $W_j, \ldots, W_{j+32}$ including the 4-step initial structure (IS), the 2-step partial fixing (PF), the 7-step partial matching (PM) and the 1-step indirect partial matching (IPM). In the following, we review the details of these techniques.

33-step Two Chunks with the 4-Step IS. The message words of length 33 is divided into two chunks as $\{W_j, \ldots, W_{j+14}, W_{j+18}\}$ and $\{W_{j+15}, W_{j+16}, W_{j+17}, W_{j+19}, \ldots, W_{j+32}\}$. Using message compensation technique [4], the first chunk and the second chunk are independent from W_{j+15} and W_{j+18}, respectively. In particular, the following constraints ensure the above message words to be neutral words with respect to each chunk;

$$\begin{array}{lll} W_{j+17} = \sigma_1(W_{j+15}), & W_{j+19} = \sigma_1^2(W_{j+15}), & W_{j+21} = \sigma_1^3(W_{j+15}), \\ W_{j+22} = W_{z+5}, & W_{j+23} = \sigma_1^4(W_{j+15}), & W_{j+24} = 2\sigma_1(W_{j+15}), \\ W_{j+25} = \sigma_1^5(W_{j+15}), & & \end{array} \quad (1)$$

where $\sigma_1^2(X)$ means $\sigma_1 \circ \sigma_1(X)$.

These two chunks include the 4-step IS, which essentially exchanges the order of the words W_i and W_{i+3} by exploiting the absorption property of the function Ch. After the swapping, the final output after the step $(i+3)$ still keeps unchanged. Here, W_{j+18} is moved to the first chunk and W_{j+15}, W_{j+16} and W_{j+17} are moved to the second chunk.

In the forward direction, a state value of $p_{j+33} = A_{j+33}|| \ldots ||H_{j+33}$ can be computed independently of the first chunk. In the backward direction, a state value of $p_j = A_j|| \ldots ||H_j$ can be computed independently of the second chunk. Note that the 33-step two-chunk is valid regardless of the choice of j for $j > 0$.

7-step PM. In the backward computation, A_j can be computed from p_{j+7} without knowing $\{W_j, \cdots, W_{j+6}\}$ for any j as used in [13].

2-step PF. PF is a technique to enhance PM by fixing a part of a neutral word. The equation for H_{j-1} is as follows:

$$\begin{cases} H_{j-1} = A_j - \Sigma_0(B_j) - Maj(B_j, C_j, D_j) - \Sigma_1(F_j) \\ \qquad\quad -Ch(F_j, G_j, H_j) - K_{j-1} - W_{j-1}, \\ W_{j-1} = W_{j+15} - \sigma_1(W_{j+13}) - W_{j+8} + \sigma_0(W_j). \end{cases}$$

If we fix the lower ℓ bits of W_{j+15}, which is assumed to be a neutral word for the other chunk, the lower ℓ bits of H_{j-1} can be computed without using the value of the higher $(32-\ell)$ bits of W_{j+15}. Furthermore, the equation for H_{j-2} is expressed as follows:

$$\begin{cases} H_{j-2} = A_{j-1} - \Sigma_0(B_{j-1}) - Maj(B_{j-1}, C_{j-1}, D_{j-1}) - \Sigma_1(F_{j-1}) \\ \qquad\quad -Ch(F_{j-1}, G_{j-1}, H_{j-1}) - K_{j-2} - W_{j-2}, \\ W_{j-2} = W_{j+14} - \sigma_1(W_{j+12}) - W_{j+7} + \sigma_0(W_{j-1}). \end{cases}$$

The lower $(\ell - 18)$ bits of H_{j-2} can be computed if we can obtain the lower ℓ bits of $Ch(F_{j-1}, G_{j-1}, H_{j-1})$ and the lower $(\ell - 18)$ bits of $\sigma_0(W_{j-1})$. Note that these values can be computed by using only the lower ℓ bits of W_{j+15}. Thus, when we fix the lower ℓ bits of W_{j+15}, the lower $(\ell - 18)$ bits of H_{j-2} can be computed without knowing the higher $(32-\ell)$ bits of W_{j+15}. Therefore, by combining the 7-step PM with the 2-step PF, 9 steps can be skipped in the backward computation.

1-step IPM. For the forward computation, A_{j+34} can be expressed as a sum of two independent functions ψ_F, ξ_F of each neutral word as follows;

$$\begin{cases} A_{j+34} = \Sigma_0(A_{j+33}) + Maj(A_{j+33}, B_{j+33}, C_{j+33}) + H_{j+33} + \Sigma_1(A_{j+33}) \\ \qquad\quad +Ch(A_{j+33}, B_{j+33}, C_{j+33}) + K_{j+33} + W_{j+33}, \\ W_{j+33} = \sigma_1(W_{j+31}) + W_{j+26} + \sigma_0(W_{j+18}) + W_{j+17}, \\ \qquad\qquad\qquad\qquad \Rightarrow A_{j+34} = \psi_F(W_{j+15}) + \xi_F(W_{j+18}). \end{cases}$$

Then, we can compute $\psi_F(W_{j+15})$ and $\xi_F(W_{j+18})$ independently. It is equivalent to move the computation of $\xi_F(W_{j+18})$ to the backward chunk. In this case, $\xi_F(W_{j+18}) = \sigma_0(W_{j+18})$.

Attack Overview. These techniques enable us to construct the 43 (= 33 + 7 + 2 + 1)-step attack on SHA-256. Here, we have the freedom of choice of j as long as 36 steps (W_{j-2} to W_{j+34}) is located sequentially.

For the actual attack in [4], j is chosen as $j = 3$, because W_{13}, W_{14} and W_{15} can be freely chosen to satisfy the message padding rule. The matching state is the lower 4 bits of A_{37}. In addition, the number of fixed bits ℓ for PF is chosen as $\ell = 23$. Then, neutral words of W_{18} and W_{21} have 5- and 4-bit freedom degrees, respectively. As a result, a pseudo preimage is found with the complexity of $2^{251.9}$. After that, pseudo preimages are converted into a preimage with the complexity of $2^{254.9}$. See [4] for more details about this attack.

4.4 Pseudo Collision Attack on 43-Step SHA-256

As discussed in Section 3.3, to convert a MITM preimage attack into a pseudo collision attack, the matching point is located into the end of the compression function, i.e., the addition of the feed-forward. As mentioned in section 4.3, the

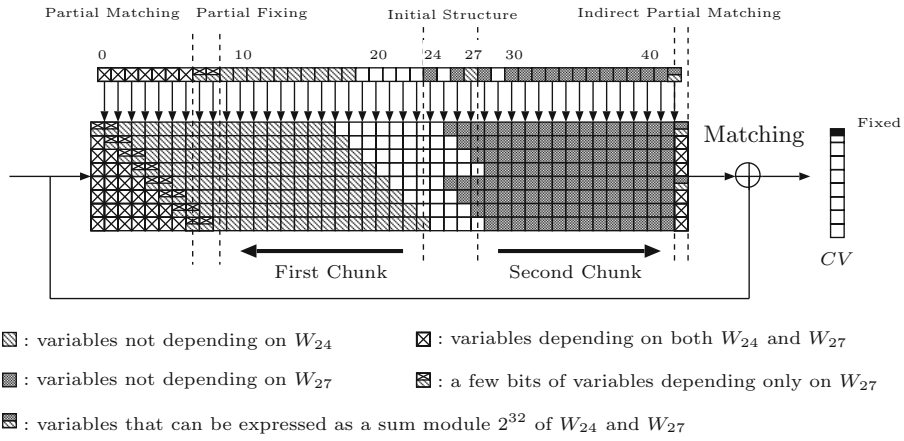

Fig. 4. 43-step pseudo collision attack on SHA-256

matching point of the 43-step MITM preimage attack is selected at the state after the step 37 ($j = 3$) due to the padding bits.

However, for a (pseudo) collision attack, we do not need to control message words for satisfying the padding rules, since we can generate correct padding by simply adding another message block as discussed in Section 3.3. It means that the last block of a compression function is used only for satisfying the padding condition in the collision attack when pseudo collision can be found before the last compression function as shown in Fig. 3. As a result, for a (pseudo) collision attack, we can move the matching point to the state after the step 43 ($j = 9$) that is the end of the compression function.[3]

Let a 256-bit output of the compression function be $CV = \{Z_A||\cdots||Z_H\}$, where each word is 32 bits. For $j = 9$, W_{24} and W_{27} are neutral words, and the matching point is the lower 4 bits of $A_{43}(= A_0 \oplus Z_A)$.

In order to construct the pseudo collision attack, we give the efficient method to obtain 4-bit partial target preimages by using the MITM technique [4]. Figure 4 shows the overview of the 43-step pseudo collision attack.

Attack Procedure

1. Choose the lower 4 bits of Z_A, which are target values.
2. Randomly choose the value of p_{25} and message W_{25}. Randomly fix the lower 23 bits of W_{24}. Then we can find 2^5 values of W_{24} on average from 9 free bits that correctly construct the 4-step initial structure and store them in the table T_W.
3. Randomly choose message words not related to the initial structure and the neutral words, i.e., W_{19}, W_{20}, W_{21}, W_{22}, W_{23} and W_{29} (called an initial configuration).

[3] It is also pointed out in [10] as the matching point can be rotated to the end of the compression function.

4. For all 2^5 possible W_{24} in T_W, compute W_{26}, W_{28}, W_{30}, W_{31}, W_{32}, W_{33} and W_{34} following Eq. (1). Compute forward and find $\psi_F(W_{24})$. Then, store the pairs $(W_{24}, \psi_F(W_{24}))$ in a list L_F.
5. For all 2^4 possible values (the lower 4 bits) of W_{27}, compute backward and find $\xi_F(W_{27})$ and the lower 4 bits of A_0. Then, store the pairs $(W_{27}, Z_A \oplus A_0 - \sigma_0(W_{27}))$ in a list L_B.
6. If a match is found, i.e., $\psi_F(W_{24}) = Z_A \oplus A_0 - \sigma_0(W_{27})$, then compute two group of states $A_{43}, B_{43}, \cdots, H_{43}$ and $A_0, B_0, \cdots, H_0$ with corresponding W_{24} and W_{27}, respectively. Then obtain 2^5 $(= 2^9/2^4)$ CV whose 4-bit are fixed, i.e., the lower 4 bits of Z_A, and store these in a List L_1.
7. Repeat (3)-(6) 2^{121} times with different values of the initial configuration.

After the above procedures, we obtain 2^{126} $(= 2^5 \times 2^{121})$ pairs whose 4 bits are fixed.[4] Thus, there exists a colliding pair with a high probability, because of the equation of $(2^{126} = 2^{(256-4)/2})$.

Evaluation. We assume that the complexity for the 1-step function and the 1-step message expansion is 1/43 compression function operation of the 43-step SHA-256. As estimated in [10], the complexity of Step 2 in the presented attack is 2^9, and that of Steps 3-6 is $2^{4.878}$, which is the complexity for finding 2^5 4-bit partial target preimages. Thus, whole complexity of the pseudo collision attack on the 43-step SHA-256 is estimated as $2^{126} \approx 2^9 + (2^{121} \times 2^{4.878})$.

4.5 Known MITM Preimage Attack on 46-Step SHA-512 [4]

The MITM preimage attack on the 46-step SHA-512 presented in [4] uses the 31-step two chunks $W_j, \ldots, W_{j+30}$ including the 2-step IS, the 8-step PF for $W_{j-1}, \ldots, W_{j-6}$ and W_{j+31},W_{j+32} and the 7-step PM. In this attack, we can choose j as long as 39 step (W_{j-6} to W_{j+32}) are located sequentially. For the actual attack in [10], j is chosen as $j = 6$ to satisfy the padding rule. Then, the neutral words W_{21} and W_{22} have 4 and 3-bit freedom degrees, respectively, and the bit size of the matching point is 3. Thus, a preimage of the 46-step SHA-512 is found with the complexity of $2^{511.5}$. See [4] for more details about this attack.

4.6 Pseudo Collision Attack on 46-Step SHA-512

Similarly to the attack on the reduced SHA-256, we can move the matching point to the end of the compression function, because the padding issue can be avoided by using multi-block message technique in the pseudo collision attack. In the case of SHA-512, since the bit size of the matching point is 3, we utilize the 3-bit partial target preimages for the attack. Then, the complexity of the attack is estimated as $2^{254.5} = (2^{(512-3)/2})$.

[4] It is noted that we need a slightly more than 2^{121} times repeated experiments to get 2^{126} pairs that will achieve a probability higher than 2^{-1}. However the difference is so small that we ignore it here.

4.7 Pseudo Collision Attacks on 42-step SHA-256 and 42-step SHA-512

We consider pseudo collision attacks on smaller number of rounds of SHA-2 in order to save the time complexity. For the 42-step reduced SHA-256, we can use 10 bits of freedom in both directions to find a 10-bit partial target preimage as discussed in Section 5.4 of [4]. This implies that a 10-bit partial target preimage is obtained with the complexity 1 ($< 2^5$). Thus, a pseudo collision is found with the complexity of $2^{123} (= 2^{(256-10)/2} \times 2^{10}/2^{10})$. Similarly to this, for the 42-step reduced SHA-512, we can use 24 bits of freedom in both directions to find a 24-bit partial target preimage as discussed in Section 6.5 of [4]. Therefore, a pseudo collision of the 42-step reduced SHA-512 is found with the complexity of $2^{244} (= 2^{(512-24)/2} \times 2^{24}/2^{24})$.

4.8 Pseudo Collision Attacks on Reduced SHA-224 and SHA-384

The pseudo collision attack on the 43-step SHA-256 described in Section 4.4 is applicable to the 43-step SHA-224 in the similar manner. However, we can not use the multi-block message technique straightforwardly, because the pseudo collision attack on SHA-224 needs to be done in the last compression function whose output Z_H is disregarded. Thus, due to the padding issue, we can mount only pseudo collision attack on a compression function of 43-step, not a hash function. The estimated complexity is 2^{110} for this attack.

However, the smaller number of rounds of SHA-224 hash function can be attacked by using another MITM attack. The 40-step SHA-224 hash function can be attacked by using the same two chunks for the 43-step preimage attack on SHA-256 in [4], i.e., the case of $j = 3$. The 7-step partial matching for backward computation are replaced by the 4-step one. Then the message words W_{13}, W_{14} and W_{15} are left as free message words to satisfy the padding rule. Instead of the lower 4 bits of Z_A, we use the lower 4 bits of Z_D as the target value. Here, we need additional one step: when finding matches at the lower 4 bits of A_{37}, we compute forward from the matching point to the end of the compression function (40-th step) by using these values that are computed forward from the starting point. Since $A_{37} = D_{40} = D_0 \oplus Z_D$ for the 40-step SHA-224, the lower 4 bits of Z_D will keep unaffected by the additional step. Thus, we can still get a partial target preimage. It can be converted into a pseudo collision attack on a hash function, because we can set W_{13}, W_{14} and W_{15} to follow the padding rule.

The detail of the attack procedure is as follows.

1. Choose the lower 4 bits of Z_D, which are target values.
2. Randomly choose the value of p_{19} and message W_{19}. Randomly fix the lower 23 bits of W_{18}. Then we can find 2^5 values of W_{18} on average from 9 free bits that correctly construct the 4-step initial structure and store them in the table T_W.
3. Randomly choose message words not related to the initial structure and the neutral words, i.e., W_{13}, W_{14}, W_{15}, W_{16}, W_{17}, W_{23} (called an initial configuration [4]).

4. For all 2^5 possible W_{18} in T_W, compute W_{20}, W_{22}, W_{24}, W_{25}, W_{26}, W_{27}, W_{28} following Eq, (1). Compute forward and find $\psi_F(W_{18})$. Store the pairs $(W_{18}, \psi_F(W_{18}))$ in a list L_F.
5. For all 2^4 possible values (the lower 4 bits) of W_{21}, compute backward and find $\xi_F(W_{21})$ and the lower 4 bits of A_{37} $(= D_{40} = Z_D \oplus D_0)$. Store the pairs $(W_{21}, Z_D \oplus D_0 - \sigma_0(W_{27}))$ in a list L_B.
6. If a match is found, i.e., $\psi_F(W_{24}) = Z_D \oplus D_0 - \sigma_0(W_{27})$, then compute forward to get the states $A_{40}, B_{40}, \cdots, H_{40}$ with corresponding W_{24} and W_{27}, respectively. D_{40} will keep unaffected in this step. Then obtain 2^5 $(= 2^9/2^4)$ CV whose 4 bits are fixed, i.e., the lower 4 bits of Z_D, and store these in a List.
7. Repeat (3)-(6) 2^{105} times with different values of the initial configuration.

The complexity of the attack is estimated as 2^{110}.

Similarly, the pseudo collision attack on the 46-step SHA-512 hash function described in 4.6 can also be applied to the 46-step SHA-384 compression function with the complexity of $2^{190.5} = (2^{(384-3)/2})$. For a pseudo collision attack on the reduced SHA-384 hash function, we use the 43-step preimage attack on SHA-384 [4]. Combining the result in [4] with our conversion technique, a pseudo collision attack on the 40-step SHA-384 hash function can be constructed. The matching bit is 18 when chosen parameter of partial matching as $\ell = 27$. The complexity of the pseudo collision attack on the 40-step SHA-384 is estimated as $2^{(384-18)/2} = 2^{183}$. These 40-step pseudo collision attacks give examples that the matching point is not at but near the end of compression function. That is compatible to solve padding problem.

4.9 Application to Other Results of SHA-2

Recently, the MITM preimage attacks on the reduced SHA-2 are improved by using "bicliques" technique which is considered as generalized initial structure [14]. This technique enables us to construct longer initial structures than those of the attacks [4]. In the following, let us consider pseudo collision attacks based on [14].

For SHA-256, the 36-step two independent chunks including the 6-step IS based on bicliques are constructed. Combining the 2-step PM with the 7-step PM and the 1-step IPM, the MITM preimage attack on the 45-step SHA-2 is derived. In this attack, both neutral words have 3-bit freedom degrees, and the matching point is 4-bit. Since our conversion technique does not need to consider the padding issue, the matching point can be moved to the end of the compression function similar to the 43-step attack. Then, we can convert it into the 45-step pseudo collision attack on SHA-256 with the complexity of $2^{126.5}$ $(= 2^{(256-3)/2})$[5]. Similarly, we can construct the 50-step pseudo collision attack on SHA-512 based on the 50-step MITM preimage attack [14]. In this attack, both neutral words have 3-bit freedom degrees, and the bit size of the matching point is 3. Thus, the complexity of the attack is estimated as $2^{254.5}$ $(= 2^{(512-3)/2})$.

[5] Our attack uses only 3 bits for the matching and find 3-bit partial target preimages, because this setting is optimal with respect to the time complexity.

In addition, [14] showed pseudo preimage attacks on the 52-step SHA-256 and the 57-step SHA-512. For the setting of a pseudo preimage attack, the cost of converting a pseudo preimage to a preimage is omitted. Thus, larger number of rounds can be attacked. Note that in these attacks, the amount of freedom degrees for both neutral words are only 1-bit, and the bit size of the matching point is 1. In order to construct a pseudo collision attack by using our conversion technique, it is sufficient to obtain a pseudo preimage on a compression function, i.e., a preimage on a hash function is not needed. Therefore, the above explained pseudo preimage attacks can also be converted into pseudo collision attacks in a similar way. The complexities of the pseudo collision attacks on the 52-step SHA-256 and the 57-step SHA-512 are estimated as $2^{127.5}$ $(= 2^{(256-1)/2})$ and $2^{255.5}$ $(= 2^{(512-1)/2})$, respectively.

5 Application to Skein

In this section, we show pseudo collision attacks on the reduced Skein-512 [9] based on the preimage attacks presented in [14].

5.1 Description of Skein

Skein is built from the tweakable block cipher Threefish $E_{K,T}(P)$, where K, T and P denote a key, a tweak and a plaintext message, respectively. The compression function $F(CV, T, M)$ of Skein outputs the next chaining variable as $F(CV, T, M) = E_{CV,T}(M) \oplus M$, where CV is the previous chaining variable and M is an input message block.

Threefish-512 supports a 512-bit block and a 512-bit key, and operates on 64-bit words. The subkey $K^s = (K_0^s, K_1^s, \ldots, K_7^s)$ injected every four rounds is generated from the secret key $K = K[0], K[1], \ldots, K[7]$ as follows:

$$K_j^s = K[(s+j) \bmod 9], (0 \le j \le 4); \qquad K_5^s = K[(s+5) \bmod 9] + T[s \bmod 3];$$
$$K_6^s = K[(s+6) \bmod 9] + T[(s+1) \bmod 3]; \quad K_7^s = K[(s+7) \bmod 9] + s,$$

where s denotes a round counter, $T[0]$ and $T[1]$ denote tweak words, $T[2] = T[0] + T[1]$, and $K[8] = C_{240} \oplus \bigoplus_{j=0}^{7} K[j]$ with a constant C_{240}. Each Threefish-512 round consists of four MIX functions followed by a permutation of the eight 64-bit words. The 128-bit function MIX processes the pairs of eight words of internal state $I^0, I^1, \ldots, I^7$ after key addition.

5.2 Known Pseudo Preimage Attacks on Skein [14].

We briefly review two MITM preimage attacks on Skein-512 presented in [14]: one is a preimage attack on the 22-round reduced Skein-512 hash function starting from the 3rd round, and the other is a preimage attack on the 37-round reduced Skein-512 compression function starting from the 2nd round.

For the 22-round attack, the 3-dimension biclique at rounds 12-15 is obtained with the complexity of 2^{200}. Since many bicliques can be produced out of one,

the cost of constructing the bicliques is negligible in the total complexity of the attack. In this attack, we can obtain 2^3 pairs matched in 3 bits by $2^{2.3}$ calls of the 22-round Skein-512 compression function. As a result, a preimage of the 22-round reduced Skein is found with the complexity of $2^{511.2}$.

Table 2. Parameters of the (pseudo) preimage attacks on the reduced Skein-512 [14]

Parameters of the preimage attack on the 22-round Skein-512 hash function

Chunks			Matching			
Forward	Backward	Biclique	Partial matching	Matching bits	Total matching pairs	Complexity
8-11	16-19	12-15	$20 \rightarrow 24 = 3 \leftarrow 7$	$I^1_{30,31,53}$	2^3	$2^{2.3}$

Parameters of the pseudo preimage attack on the 37-round Skein-512 compression function

Chunks			Matching			
Forward	Backward	Biclique	Partial matching	Matching bits	Total matching pairs	Complexity
8-15	24-31	16-23	$32 \rightarrow 38 = 2 \leftarrow 7$	I^3_{25}	2	$2^{1.2}$

Considering a pseudo preimage attack on the compression function, it is natural to assume that tweak bits T can also be controlled by the attacker. Due to additional freedom, the pseudo preimage attack on the 37-round reduced Skein-512 is feasible by using the 1-dimension biclique at rounds 16-23. In this attack, we can obtain 2 pairs matched in 1 bit by $2^{1.2}$ calls of the 37-round Skein-512 compression function. Consequently, a pseudo preimage of the 37-round reduced Skein is found with the complexity of $2^{511.2}$.

The parameters for the preimage attacks on the 22-round and the 37-round reduced Skein-512 hash function and compression function are summarized in Table 2. See [14] for more details about this attack.

5.3 Pseudo Collision Attacks on Skein

Since the matching point used in the MITM preimage attack on the 22-round reduced Skein-512 hash function [14] is located in the end of the compression function, our conversion technique can directly convert it to the pseudo collision attack on the 22-round reduced Skein-512. In this attack, the neutral words have 3-bit freedom degrees, and the bit size of the matching point is 3. As reported in [14], a 3-bit matching candidate can be found with the complexity of $2^{2.3}/2^3$. Thus, the complexity of the pseudo collision attack on the 22-round reduced Skein-512 hash function is estimated as $2^{253.8}$ $(= 2^{(512-3)/2} \times 2^{2.3}/2^3)$.

The pseudo preimage attack on the 37-round reduced Skein compression function can be converted into a pseudo collision attack on a hash function in a similar way. The required complexity for the pseudo collision attack on the 37-round reduced Skein hash function is estimated as $2^{255.7}$ $(= 2^{(512-1)/2} \times 2^{1.2}/2)$.

6 Conclusion

In this paper, we gave a generic method to convert preimage attacks to pseudo collision attacks. It provides a new insight to evaluate the security of hash

functions. The essence of the method is converting a partial target preimage attack to a pseudo collision attack. That is especially compatible to meet-in-the-middle preimage attacks since it can be converted into a partial target preimage attack if the matching point can be moved to the end of a hash function or a compression function and enough freedom on neutral bits are left.

Using the proposed approach, we presented the best pseudo collision attacks on SHA-2 based on the known preimage attacks, which has been left as open question. We showed pseudo collision attacks on the 43- and 46-step reduced SHA-256 and SHA-512 based on the MITM preimage attacks presented in [4]. Also, pseudo collision attacks on the 52- and 57-step reduced SHA-256 and SHA-512 based on the more advanced MITM preimage attacks in [14] were demonstrated. We also applied the conversion technique to other hash functions including Skein and BLAKE with the meet-in-the-middle preimage attacks, which showed the widely usage of this method. The pseudo collision attacks on the 22- and 37-round reduced Skein-512 were presented. The 4-round reduced BLAKE-256/512 without the initialization function can be attacked by the converted pseudo collision attack (see Appendix A). Our technique may also apply to other hash functions, such as Tiger [1]. Based on the MITM preimage attack on the full Tiger [10], we might construct the pseudo collision attack on the full Tiger. We believe that the technique can be used for more hash algorithms once their preimage or pseudo preimage attacks are found.

By this method, now we only can get pseudo collision attacks. It is left as future works that how to construct collision attacks from known preimage attacks.

Acknowledgments. The author would like to thank the anonymous reviewers for their helpful comments.

References

1. Anderson, R.J., Biham, E.: Tiger: A Fast New Hash Function. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 89–97. Springer, Heidelberg (1996)
2. Aoki, K., Sasaki, Y.: Preimage Attacks on One-Block MD4, 63-Step MD5 and More. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 103–119. Springer, Heidelberg (2009)
3. Aoki, K., Sasaki, Y.: Meet-in-the-Middle Preimage Attacks Against Reduced SHA-0 and SHA-1. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 70–89. Springer, Heidelberg (2009)
4. Aoki, K., Guo, J., Matusiewicz, K., Sasaki, Y., Wang, L.: Preimages for Step-Reduced SHA-2. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 578–597. Springer, Heidelberg (2009)
5. Aumasson, J.-P., Henzen, L., Meier, W., Phan, R.C.-W.: SHA-3 proposal BLAKE (version 1.3). Submission to NIST (December 2010), `http://131002.net/blake/blake.pdf`
6. Biryukov, A., Lamberger, M., Mendel, F., Nikolić, I.: Second-Order Differential Collisions for Reduced SHA-256. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 270–287. Springer, Heidelberg (2011)

7. Bogdanov, A., Rechberger, C.: A 3-Subset Meet-in-the-Middle Attack: Cryptanalysis of the Lightweight Block Cipher KTANTAN. In: Biryukov, A., Gong, G., Stinson, D.R. (eds.) SAC 2010. LNCS, vol. 6544, pp. 229–240. Springer, Heidelberg (2011)
8. De Cannière, C., Rechberger, C.: Preimages for Reduced SHA-0 and SHA-1. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 179–202. Springer, Heidelberg (2008)
9. Ferguson, N., Lucks, S., Schneier, B., Whiting, D., Bellare, M., Kohno, T., Callas, J., Walker, J.: The Skein hash function family (version 1.3, October 1, 2010), http://www.schneier.com/skein1.3.pdf
10. Guo, J., Ling, S., Rechberger, C., Wang, H.: Advanced Meet-in-the-Middle Preimage Attacks: First Results on Full Tiger, and Improved Results on MD4 and SHA-2. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 56–75. Springer, Heidelberg (2010)
11. Indesteege, S., Mendel, F., Preneel, B., Rechberger, C.: Collisions and Other Nonrandom Properties for Step-Reduced SHA-256. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 276–293. Springer, Heidelberg (2009)
12. Isobe, T.: A Single-Key Attack on the Full GOST Block Cipher. In: Joux, A. (ed.) FSE 2011. LNCS, vol. 6733, pp. 290–305. Springer, Heidelberg (2011)
13. Isobe, T., Shibutani, K.: Preimage Attacks on Reduced Tiger and SHA-2. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 139–155. Springer, Heidelberg (2009)
14. Khovratovich, D., Rechberger, C., Savelieva, A.: Bicliques for Preimages: Attacks on Skein-512 and the SHA-2 Family. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 244–263. Springer, Heidelberg (2012)
15. Lai, X., Massey, J.L.: Hash Functions Based on Block Ciphers. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 55–70. Springer, Heidelberg (1993)
16. Leurent, G.: MD4 is Not One-Way. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 412–428. Springer, Heidelberg (2008)
17. Mendel, F., Nad, T., Schläffer, M.: Finding SHA-2 characteristics: Searching through a minefield of contradictions. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 288–307. Springer, Heidelberg (2011)
18. Mendel, F., Pramstaller, N., Rechberger, C., Rijmen, V.: Analysis of Step-Reduced SHA-256. In: Robshaw, M.J.B. (ed.) FSE 2006. LNCS, vol. 4047, pp. 126–143. Springer, Heidelberg (2006)
19. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press (1997)
20. Nikolić, I., Biryukov, A.: Collisions for Step-Reduced SHA-256. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 1–15. Springer, Heidelberg (2008)
21. Quisquater, J.-J., Delescaille, J.-P.: How Easy Is Collision Search? Application to DES (Extended Summary). In: Quisquater, J.-J., Vandewalle, J. (eds.) EUROCRYPT 1989. LNCS, vol. 434, pp. 429–434. Springer, Heidelberg (1990)
22. Saarinen, M.-J.O.: A Meet-in-the-Middle Collision Attack Against the New FORK-256. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 10–17. Springer, Heidelberg (2007)
23. Sanadhya, S.K., Sarkar, P.: 22-step collisions for SHA-2. CoRR, abs/0803.1220 (2008)
24. Sanadhya, S.K., Sarkar, P.: Attacking Reduced Round SHA-256. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 130–143. Springer, Heidelberg (2008)

25. Sanadhya, S.K., Sarkar, P.: New Collision Attacks against Up to 24-Step SHA-2. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 91–103. Springer, Heidelberg (2008)
26. Sanadhya, S.K., Sarkar, P.: Non-linear Reduced Round Attacks against SHA-2 Hash Family. In: Mu, Y., Susilo, W., Seberry, J. (eds.) ACISP 2008. LNCS, vol. 5107, pp. 254–266. Springer, Heidelberg (2008)
27. Sasaki, Y., Aoki, K.: Finding Preimages in Full MD5 Faster Than Exhaustive Search. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 134–152. Springer, Heidelberg (2009)
28. Sasaki, Y., Aoki, K.: Preimage Attacks on 3, 4, and 5-Pass HAVAL. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 253–271. Springer, Heidelberg (2008)
29. Sedgewick, R., Szymanski, T.G., Yao, A.C.-C.: The complexity of finding cycles in periodic functions. SIAM J. Comput. 11(2), 376–390 (1982)
30. Wang, L., Ohta, K., Sakiyama, K.: Free-start preimages of round-reduced Blake compression function. Rump session at ASIACRYPT 2009 (2009)
31. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)
32. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)
33. Watanabe, D.: OFFICIAL COMMENT: LUX. NIST mailing list (2009), `http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/documents/LUX_Comments.pdf`

Appendix

A Application to BLAKE

We apply our technique to BLAKE hash function family consisting of BLAKE-224, BLAKE-256, BLAKE-384 and BLAKE-512 [5]. We utilize the result presented in [30] which showed a pseudo preimage attack on the 4-round reduced BLAKE compression function without the initialization function. While the practical impact on the attack for this reduced BLAKE compression function is debatable, a pseudo collision on the reduced BLAKE can be directly derived by using our conversion technique. As a result, we can find a pseudo collision of the 4-round reduced BLAKE-256 compression function without the initialization with the complexity of 2^{112}. Similarly, a pseudo collision of the 4-round reduced BLAKE-512 compression function without the initialization can be found with the complexity of 2^{224}.

A.1 Description of BLAKE

The compression function of BLAKE-256 consists of *initialization*, *round function* and *finalization*.

Table 3. Message and Constants Permutation

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
σ_0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
σ_1	14	10	4	8	9	15	13	6	1	12	0	2	11	7	5	3
σ_2	11	8	12	0	5	2	15	13	10	14	3	6	7	1	9	4
σ_3	7	9	3	1	13	12	11	14	2	6	5	10	4	0	15	8

Initialization. 8 words of chaining variables $h_0, \ldots, h_7$ are transformed into 16 words of an initial state $v_0, \ldots, v_{15}$ as $v_i = h_i$ for $0 \leq i < 8$, where h_i, $v_j \in \{0,1\}^{32}$. The other 8 words of the initial state v_i $(8 \leq i < 16)$ are determined from the given salts $s_0, ..., s_3$ and counter t_0, t_1, where s_i, $t_j \in \{0,1\}^{32}$.

Round Function. An initial state v is updated by 14 round functions with message words $m_0, ..., m_{15}$ and constants $c_0, ..., c_7$, where m_i, $c_j \in \{0,1\}^{32}$. Each round function includes the following steps, $G_0(v_0, v_4, v_8, v_{12})$, $G_1(v_1, v_5, v_9, v_{13})$, $G_2(v_2, v_6, v_{10}, v_{14})$, $G_3(v_3, v_7, v_{11}, v_{15})$, $G_4(v_0, v_5, v_{10}, v_{15})$, $G_5(v_1, v_6, v_{11}, v_{12})$, $G_6(v_2, v_7, v_8, v_{13})$, $G_7(v_3, v_4, v_7, v_{14})$. The function $G_i(a, b, c, d)$ is defined as:

$$
\begin{aligned}
&a \leftarrow a + b + (m_{\sigma_r(2i)} \oplus c_{\sigma_r(2i+1)}), && d \leftarrow (d \oplus a) \ggg 16,\\
&c \leftarrow c + d, && b \leftarrow (b \oplus c) \ggg 12,\\
&a \leftarrow a + b + (m_{\sigma_r(2i+1)} \oplus c_{\sigma_r(2i)}), && d \leftarrow (d \oplus a) \ggg 8,\\
&c \leftarrow c + d, && b \leftarrow (b \oplus c) \ggg 7,
\end{aligned}
$$

where permutations $\sigma_r(j)$ $(0 \leq j < 16)$ of the first 4 rounds refer to Table 3. The functions G_0 to G_3 and G_4 to G_7 denote the column transforms and the diagonal transforms, respectively.

Finalization. After the round functions, the new chaining variables are extracted with the updated state, the salts and the feed-forward of the initial chaining variables as follows.

$$
\begin{aligned}
&h_0^{'} \leftarrow h_0 \oplus s_0 \oplus v_0 \oplus v_8 && h_1^{'} \leftarrow h_1 \oplus s_1 \oplus v_1 \oplus v_9\\
&h_2^{'} \leftarrow h_2 \oplus s_2 \oplus v_2 \oplus v_{10} && h_3^{'} \leftarrow h_3 \oplus s_3 \oplus v_3 \oplus v_{11}\\
&h_4^{'} \leftarrow h_4 \oplus s_0 \oplus v_4 \oplus v_{12} && h_5^{'} \leftarrow h_5 \oplus s_1 \oplus v_5 \oplus v_{13}\\
&h_6^{'} \leftarrow h_6 \oplus s_2 \oplus v_6 \oplus v_{14} && h_7^{'} \leftarrow h_7 \oplus s_3 \oplus v_7 \oplus v_{15}
\end{aligned}
$$

BLAKE-512 operates on 64-bit words and outputs 512 bits. The compression function of BLAKE-512 is similar to that of BLAKE-256 except for the number of rounds (16 instead of 14), and the constants and the amount of rotation used in G functions.

A.2 Known MITM Preimage Attacks on 4-Round Compression Function of BLAKE [30]

In the setting of the pseudo preimage attack on the reduced BLAKE compression function presented in [30], the initialization step is disregarded, and an attacker

can select a random start value from the start of round functions (the end of initialization step) as shown in Fig. 5.

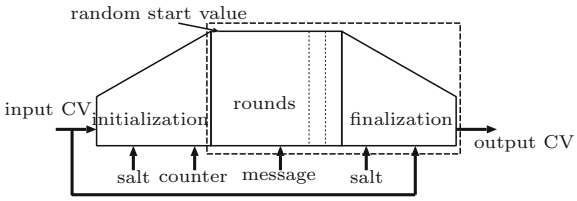

Fig. 5. MITM preimage attack for finalization

Figure 6 shows the overview of the pseudo preimage attack on the 4-round reduced BLAKE compression function without the initialization. Let an input state of the round i be $v^{i-1} = \{v_0^{i-1}, \ldots, v_{15}^{i-1}\}$, where $v_j^i \in \{0,1\}^{32}$. In this attack, message words m_4 and m_6 are used as the neutral words, and the starting point of the attack is the state after the column transformation of the round 3. In the forward computation from the starting point, v_6^4, v_{14}^4 can be computed without using m_6. Similarly, in the backward computation, v_6^0 can be computed without using m_4. Therefore, storing m_4, v_6^4, v_{14}^4 in a list L_F, and m_6, v_6^0 in a list L_B, we expect to find matching pairs satisfying $h'_6 = v_6^0 \oplus v_6^4 \oplus v_{14}^4$. As a result, a pseudo preimage of the 4-round reduced BLAKE without the initialization is found with the complexity of 2^{224}.

A.3 Pseudo Collision Attacks on BLAKE Compression Function

Since the matching point of the known pseudo preimage attack is at the end of the compression function, a pseudo collision attack can be directly constructed from it.

Attack Procedure

1. Randomly choose the 7-th word of the output value h'_6, which is the target value.
2. Randomly choose the values of state words and message words except for m_4 and m_6.
3. For all 2^{32} possible m_4, compute forward and find v_6^4 and v_{14}^4. Store the pairs (m_4, $v_6^4 \oplus v_{14}^4$) in a list L_F
4. For all 2^{32} possible m_6, compute forward and find v_6^0. Store the pairs (m_4, $h'_6 \oplus v_6^0$) in a list L_B.
5. Compare the value $v_6^4 \oplus v_{14}^4$ and $h'_6 \oplus v_6^0$ in two lists L_F and L_B.
6. Once matching, compute states $v_0^0, v_1^0, \cdots, v_{15}^0$ and $v_0^4, v_1^4, \cdots, v_{15}^4$. Compute output values $h'_0, h'_1, \ldots, h'_{15}$ according to finalization steps and store with message words together. Then obtain 2^{32} items in which the value of h'_6 are fixed.

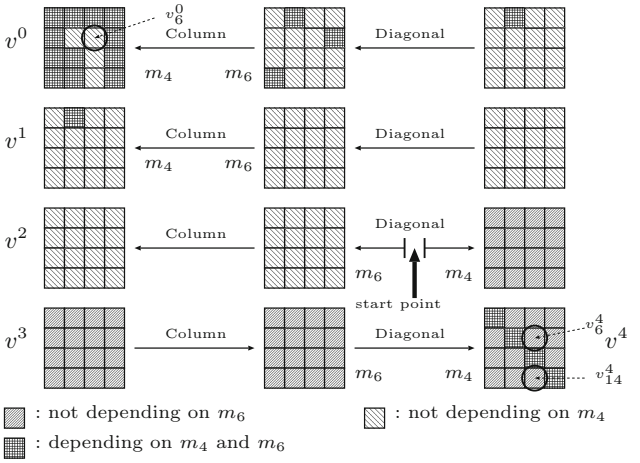

Fig. 6. Pseudo preimage attacks on reduced BLAKE compression function

7. Repeat steps (2) - (6) 2^{80} times.

We can obtain 2^{112} items in which the value of h'_6 are fixed. A colliding pair exists with a high probability that the other 224 bits of output values are also same. Finally, we can find a pseudo collision of the 4-round reduced BLAKE-256 compression function with the complexity of $2^{112} = 2^{80} \cdot 2^{32}$.

The attack is applicable to the reduced BLAKE-512 in a similar way, since the components of BLAKE-512 are similar to those of BLAKE-256. In BLAKE-224, the variable h'_7 is truncated and discarded. However, the truncation does not affect our conversion, since we use h'_6 as a partial target preimage. Thus, a pseudo collision attack on the 4-round reduced BLAKE-224 without the initialization can be constructed with the complexity of $2^{96} (= 2^{(224-32)/2})$. For BLAKE-384, in contrast to the other variants, the variable h'_6 is discarded by the truncation as well. Therefore, it is hard to straightforwardly apply our conversion to the reduced BLAKE-384, since h'_6 cannot be used as a partial target preimage.