Finding Preimages of Tiger Up to 23 Steps

Lei Wang[1] and Yu Sasaki[2]

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

Abstract. This paper evaluates the preimage resistance of the Tiger hash function. We will propose a pseudo-preimage attack on its compression function up to 23 steps with a complexity of 2^{181}, which can be converted to a preimage attack on 23-step Tiger hash function with a complexity of $2^{187.5}$. The memory requirement of these attacks is 2^{22} words. Our pseudo-preimage attack on the Tiger compression function adopts the meet-in-the-middle approach. We will divide the computation of the Tiger compression function into two independent parts. This enables us to transform the target of finding a pseudo-preimage to another target of finding a collision between two independent sets of some internal state, which will reduce the complexity. In order to maximize the number of the attacked steps, we derived several properties or weaknesses in both the key schedule function and the step function of the Tiger compression function, which gives us more freedom to separate the Tiger compression function.

Keywords: Tiger, hash function, meet-in-the-middle, preimage attack, independent chunks

1 Introduction

Tiger is a cryptographic hash function designed by Anderson and Biham [1]. It adopts the well-known Merkle-Damgård structure, and produces 192-bit hash digests. Throughout this paper, "Tiger" and "tiger" are referred to as the Tiger hash function and the Tiger compression function respectively.

This paper will evaluate the preimage resistance of Tiger. If Tiger is secure, it should take no less than 2^{192} tiger computations to find a preimage of a given hash digest. At WEWoRC 2007, Indesteege *et al.* proposed a preimage attack on Tiger reduced to 13 steps with a complexity of $2^{128.5}$ [2], where the full version of Tiger consists of 24 steps. At FSE 2009, Isobe *et al.* published another preimage attack on Tiger, which extended the number of the attacked steps to 16 with a complexity of 2^{161} and a memory requirement of 2^{32} words [3]. At AFRICACRYPT 2009, Mendel published his preimage attack on Tiger up to 17 steps with a complexity of 2^{185} and a memory requirement of 2^{160} words [4].

Besides preimage resistance, cryptographers also pay attention to the collision resistance of Tiger. Several papers have been published to evaluate the collision resistance of Tiger [5] [6] [7]. Here we only point out that the maximum number of the attacked steps of Tiger in the sense of collision resistance is 19 [6].

Our contributions. This paper will propose a preimage attack on Tiger up to 23 steps with a complexity of $2^{187.5}$ tiger computations, which are lower than the exhaustive search complexity. This attack is based on a meet-in-the-middle pseudo-preimage attack on tiger with a complexity of 2^{181}. The memory requirement of the above attacks is 2^{22} words. A comparison with previous related works is shown in Table 1.

Table 1. Comparison with previous preimage attacks

Reference	#steps	Complexity	Memory requirement
[2]	13	$2^{128.5}$	Negligible
[3]	16	2^{161}	2^{32}
[4]	17	2^{185}	2^{160}
This paper	23	$2^{187.5}$	2^{22}

The applicability of the meet-in-the-middle pseudo-preimage attack on tiger essentially depends on the existence of two sets of message words independent from each other and suitable for applying the attack. This paper denotes the independent sets of message words as *independent chunks*. If such independent chunks do exist as a matter of fact (maybe cryptographers have not found them yet), the preimage resistance of Tiger will surely be broken by the meet-in-the-middle attack. In order to evaluate the maximum number of the attacked steps, we exploit all the properties we found on tiger. From its key schedule function, we derive several properties which can be adopted to make message words independent from each other. In specific, we use the following properties. 1) Bit-shift operations eliminate some information. This gives more freedom to search for independent chunks. 2) In our attack, we add several least significant bits of two variables and several most significant bits of the same two variables. The large word-size (64 bits) helps us to make these two additions independent because the carry from the lower bits is hard to transmit to the upper bits due to the large number of intermediate bits. 3) Even if Tiger uses addition, subtraction, and XOR as its operations, they can be linearized by setting conditions, and it is possible to cancel two different operations. From its step function, we find several properties that enable us to make the related techniques work for more steps. Finally we find the independent chunks that can be applied for a preimage attack on Tiger reduced to 23 out of 24 steps.

Organization of the paper. Section 2 describes the specification of Tiger. Section 3 introduces the meet-in-the-middle preimage attack procedure on Tiger. Section 4 shows our independent chunks feasible up to 23 steps. Section 5 illustrates the preimage attack procedure. Section 6 gives a conclusion.

2 Specification of Tiger

An input message M of Tiger will be padded and then divided into 512-bit message blocks $\{M_0, M_1, \ldots, M_{l-1}\}$. The padding rule is simple: first add a single '1', then add a minimum number of '0's to make the bit length become 448 modulo 512, and finally add the bit length of the original M to the last 64 bits. Message blocks will be fed into tiger sequentially from M_0 until M_{l-1} as follows:

$$h_{i+1} \leftarrow \text{tiger}(h_i, M_i), \text{ for } i = 0, 1, \ldots, l-1,$$

where h_0 is a public constant and each h_i from $\{h_0, \ldots, h_l\}$ has 192 bits. h_l is the hash digest of M.

Specification of tiger. The inputs h_i and M_i are divided into 64-bit variables, denoted as (A_0, B_0, C_0) and $(X_0, X_1, \ldots, X_7)$ respectively. Correspondingly addition, subtraction, and multiplication are carried out with modulo 2^{64}. Hereafter we will omit the description "modulo 2^{64}" for simplicity. tiger consists of 24 step functions, regrouped into three 8-step passes. The step function at step t $(1 \le t \le 24)$ is as follows, which is also shown in Fig. 1:

$$\begin{aligned} A_t &= (B_{t-1} + odd(C_{t-1} \oplus X_{t-1})) \times s_{t-1}, \\ B_t &= C_{t-1} \oplus X_{t-1}, \\ C_t &= A_{t-1} - even(C_{t-1} \oplus X_{t-1}), \end{aligned}$$

where s_{t-1} is a constant, $even(\cdot)$ and $odd(\cdot)$ are two non-linear functions based on S-boxes, and X_{t-1} $(9 \le t \le 24)$ is derived from $\{X_0, X_1, \ldots, X_7\}$. The constant s_{t-1} differs for each pass, which is 5, 7, and 9 for the first, second, and third passes respectively. Details of $even(\cdot)$ and $odd(\cdot)$ are as follows:

$$\begin{aligned} even(C) &= T_0(c[0]) \oplus T_1(c[2]) \oplus T_2(c[4]) \oplus T_3(c[6]), \\ odd(C) &= T_3(c[1]) \oplus T_2(c[3]) \oplus T_1(c[5]) \oplus T_0(c[7]), \end{aligned}$$

where each T from $\{T_0, T_1, T_2, T_3\}$ is a S-box mapping 8-bit values to 64-bit values, and the input C is divided to 8 bytes $(c_7, c_6, \cdots, c_0)$ with c_7 as the most significant byte and c_0 as the least significant byte.

The variables of $\{X_8, \ldots, X_{23}\}$ are derived from $\{X_0, \ldots, X_7\}$ by computing a Key Schedule Function (KSF):

$$\begin{aligned} (X_8, \ldots, X_{15}) &= KSF(X_0, \ldots, X_7), \\ (X_{16}, \ldots, X_{23}) &= KSF(X_8, \ldots, X_{15}). \end{aligned}$$

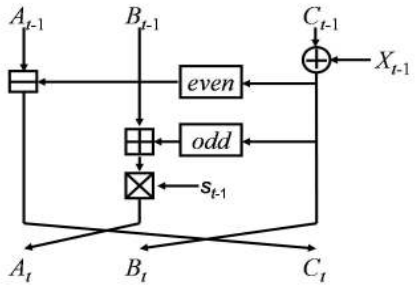

Fig. 1. Step function

Here we will pick $(X_8, \cdots, X_{15})$ as an example to describe the details of KSF.

$$
\begin{aligned}
Y_0 &= X_0 - X_7 \oplus const_1, & X_8 &= Y_0 + Y_7,\\
Y_1 &= X_1 \oplus Y_0, & X_9 &= Y_1 - (X_8 \oplus ((\neg Y_7) \ll 19)),\\
Y_2 &= X_2 + Y_1, & X_{10} &= Y_2 \oplus X_9,\\
Y_3 &= X_3 - (Y_2 \oplus ((\neg Y_1) \ll 19)), & X_{11} &= Y_3 + X_{10},\\
Y_4 &= X_4 \oplus Y_3, & X_{12} &= Y_4 - (X_{11} \oplus ((\neg X_{10}) \gg 23)),\\
Y_5 &= X_5 + Y_4, & X_{13} &= Y_5 \oplus X_{12},\\
Y_6 &= X_6 - (Y_5 \oplus ((\neg Y_4) \gg 23)), & X_{14} &= Y_6 + X_{13},\\
Y_7 &= X_7 \oplus Y_6, & X_{15} &= Y_7 - X_{14} \oplus const_2,
\end{aligned}
$$

where $const_1$ and $const_2$ are `0xA5A5A5A5A5A5A5A5` and `0x0123456789ABCDEF` respectively, and $\neg$ means bitwise complement. KSF is invertible. We will denote by KSF^{-1} the inverse computation of KSF in this paper.

Finally the output h_{i+1} is computed as follows:

$$h_{i+1} = (A_{24} \oplus A_0)||(B_{24} - B_0)||(C_{24} + C_0).$$

3 Meet-in-the-middle preimage attack on Tiger

This section introduces the application of a meet-in-the-middle attack procedure, which was proposed by Aoki *et al.* [8], to preimage attacks on Tiger. Isobe *et al.*'s preimage attack on Tiger up to 16 steps adopted this meet-in-the-middle attack approach [3].

3.1 Notations

The notations in Table 2 are used to explain the meet-in-the-middle preimage attack procedure. We will describe $(X_i^{\square}, X_j^{\triangle})$ as *independent words*, and $(\mathcal{X}^{\square}, \mathcal{X}^{\triangle})$ as *independent chunks*. $X_t \in \mathcal{X}^{\square}$ sometimes is denoted as $X_t^{\square}$ for simplicity.

Table 2. Notations for our meet-in-the-middle preimage attack

$X_i^{\square}$, $X_j^{\triangle}$:	Two message words whose values change independently.
$\mathcal{X}^{\square}$:	A set of message words which change with only $X_i^{\square}$.
$\mathcal{X}^{\triangle}$:	A set of message words which change with only $X_j^{\triangle}$.
$\mathcal{X}^{\triangle,\square}$:	A set of message words which change with both $X_i^{\square}$ and $X_j^{\triangle}$.
$\mathcal{X}^*$:	A set of message words which are fixed as constant values.
$\mathcal{E}^{\square}$:	Consecutive step functions with input message words from only $\mathcal{X}^{\square} \bigcup \mathcal{X}^*$.
$\mathcal{E}^{\triangle}$:	Consecutive step functions with input message words from only $\mathcal{X}^{\triangle} \bigcup \mathcal{X}^*$.

Similarly $X_t^{\triangle}$, $X_t^{\triangle,\square}$ and X_t^* denote $X_t \in \mathcal{X}^{\triangle}$, $X_t \in \mathcal{X}^{\triangle,\square}$ and $X_t \in \mathcal{X}^*$ respectively. During the independent computations of $\mathcal{E}^{\square}$ and $\mathcal{E}^{\triangle}$, the internal states will be denoted as $(A^{\square}, B^{\square}, C^{\square})$ and $(A^{\triangle}, B^{\triangle}, C^{\triangle})$ correspondingly.

3.2 Meet-in-the-middle preimage attacks on Tiger

A preimage attack on Tiger is constructed by combining a meet-in-the-middle pseudo-preimage attack on tiger and a meet-in-the-middle attack on Merkle-Damgård structure.

Pseudo-preimage attacks on tiger. tiger is designed following the Davies-Meyer scheme. Recall the structure of Davies-Meyer: $h' = \mathcal{E}(M, h) \oplus h$, where $\mathcal{E}$ is a block cipher, M is a message block, h is the current intermediate hash value, and h' is the next intermediate hash value. More precisely, M is expanded to $X_0||\cdots||X_{23}$. Note that h' is not calculated by $h \oplus \mathcal{E}(M, h)$ in tiger. But in this section, we regard h' as $h \oplus \mathcal{E}(M, h)$ for simplicity. The main novelty of the pseudo-preimage attacks on tiger is dividing $X_0||\cdots||X_{23}$ into suitable independent chunks. The simplest case is $X_0||\cdots||X_{23} \longrightarrow \mathcal{X}^{\triangle}||\mathcal{X}^{\square}$, which is also shown in Fig. 2. The high-level description of finding a pseudo-preimage (h, M) for a given value h' is as follows.

1. Set a random value to h, which also fixes the output of $\mathcal{E}$ as $h' \oplus h$.
2. For all the values of $\mathcal{X}^{\triangle}$, calculate $\mathcal{E}^{\triangle}(h, \mathcal{X}^{\triangle})$, and store them in a table $\mathcal{T}$.
3. For each value of $\mathcal{X}^{\square}$, calculate $\mathcal{E}^{\square}(h \oplus h', \mathcal{X}^{\square})$, and compare it with all the elements in $\mathcal{T}$. If it is equal to one element in $\mathcal{T}$, a pseudo-preimage of h' is found.
4. If no pseudo-preimage is found after trying all the values of $\mathcal{X}^{\square}$, change the value of h at step 1, and repeat steps $2 - 4$.

Suppose there is enough degree of freedom for the independent chunks. The above meet-in-the-middle attack procedure only takes 2^{96} tiger computations and 2^{96} memories to find a pseudo-preimage with a good probability. Moreover, the above attack procedure can be transformed to a memoryless meet-in-the-middle attack [9], where the complexity becomes 2^{97} tiger computations.

Meet-in-the-middle attacks on Merkle-Damgård [10]. Suppose finding a pseudo-preimage on tiger takes 2^s tiger computations. Denote by h' the given hash digest. First generate $2^{\frac{192-s}{2}}$ pseudo-preimages of h': $\{(h^1, m^1), \ldots, (h^{2^{\frac{192-s}{2}}}, m^{2^{\frac{192-s}{2}}})\}$, where $\text{tiger}(h^i, m^i) = h'$. Then randomly select a message m, calculate $\text{tiger}(h_0, m)$, and compare it with all the values of $\{h^1, \ldots, h^{2^{\frac{n-s}{2}}}\}$. If it is equal to h^i for some i, $m||m^i$ is a preimage of h'. After $2^{\frac{192+s}{2}}$ m are tried, one preimage will be found with a good probability. The total complexity is $2^{\frac{192+s}{2}+1}$ tiger computations and $2^{\frac{192-s}{2}}$ memories, which will be lower than the exhaustive search complexity as long as $s < 190$.

3.3 Related techniques

The applicability of the meet-in-the-middle pseudo-preimage attack on tiger depends on whether suitable independent chunks exist in $X_0||\cdots||X_{23}$. The example in Section 3.2 is the simplest case. Usually the attacker has to deal with more complicated cases. Cryptographers have developed several techniques for more complicated cases. Aoki *et al.* proposed *splice-and-cut*, *partial-matching* and *partial-fixing* [8]. Sasaki *et al.* proposed *initial-structure* [11].

Splice-and-cut. This technique is based on the fact that once the value h is determined, the output of $\mathcal{E}$ will be fixed as $h' \oplus h$. Therefore, the first step and the last step of $\mathcal{E}$ can be regarded to be consecutive. For example, the attacker obtains the independent chunks as follows: $X_0||\cdots||X_{23} \longrightarrow \mathcal{X}^{\triangle}||\mathcal{X}^{\square}||\mathcal{X}^{\triangle}$, which is also shown in Fig. 3. Obviously, the procedure in Section 3.2 cannot be applied directly. However, by adopting the splice-and-cut technique, the attacker will randomly determine the internal state IS, where $\mathcal{X}^{\triangle}$ and $\mathcal{X}^{\square}$ separate from each other, and then compute $\mathcal{E}^{\triangle}$ and $\mathcal{E}^{\square}$ independently.

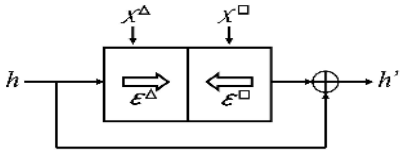

Fig. 2. Simplest meet-in-the-middle attack

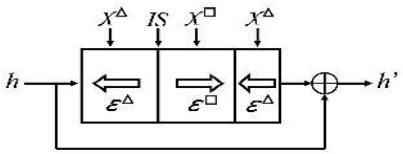

Fig. 3. Splice-and-cut

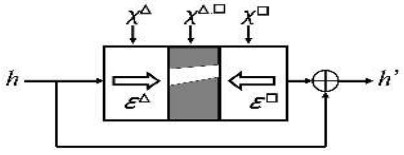

Fig. 4. Partial-matching and partial-fixing

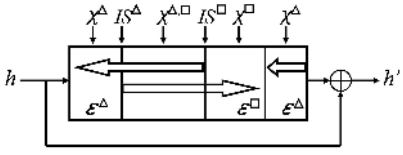

Fig. 5. Initial-structure

Partial-matching and partial-fixing. These two techniques are based on the fact that the output of one step function can be partially determined with the knowledge of only part of the input. Therefore internal states at different step positions can be partially compared if their step distance is reasonable. For instance, the attacker obtains the independent chunks as follows: $X_0||\cdots||X_{23} \longrightarrow \mathcal{X}^{\triangle}||\mathcal{X}^{\triangle,\square}||\mathcal{X}^{\square}$, which is also shown in Fig. 4. In such a case, during applying the attack procedure in Section 3.2, the internal state $\mathcal{E}^{\triangle}(h, \mathcal{X}^{\triangle})$ is not at the same step position as the internal state $\mathcal{E}^{\square}(h \oplus h', \mathcal{X}^{\square})$, but with a several-step distance. By adopting the partial-matching and partial-fixing techniques, $\mathcal{E}^{\triangle}(h, \mathcal{X}^{\triangle})$ and $\mathcal{E}^{\square}(h \oplus h', \mathcal{X}^{\square})$ will be partially compared.

Initial-structure. We will pick an example to illustrate this technique. Suppose the attacker obtains two independent chunks as follows: $X_0||\cdots||X_{23} \longrightarrow \mathcal{X}^{\triangle}||\mathcal{X}^{\triangle,\square}||\mathcal{X}^{\square}||\mathcal{X}^{\triangle}$, which is also shown in Fig. 5. By adopting the initial-structure technique, for each value of $\mathcal{X}^{\triangle}$, the attacker generates a corresponding $IS^{\triangle}$. For each value of $\mathcal{X}^{\square}$, the attacker generates a corresponding $IS^{\square}$. Moreover, for any pair of $(IS^{\triangle}, \mathcal{X}^{\triangle})$ and any pair of $(IS^{\square}, \mathcal{X}^{\square})$, $IS^{\triangle}$ always matches with $IS^{\square}$ using $\mathcal{X}^{\triangle,\square}$. Therefore, the attacker can carry out the independent calculations $\mathcal{E}^{\triangle}$ and $\mathcal{E}^{\square}$ using $(IS^{\triangle}, \mathcal{X}^{\triangle})$ and $(IS^{\square}, \mathcal{X}^{\square})$ respectively.

4 Our independent chunks

As we discussed in Section 3, one most important part of the meet-in-the-middle pseudo-preimage attack on tiger is how to separate the message words into two independent chunks $(\mathcal{X}^{\square}, \mathcal{X}^{\triangle})$, which is hard because the key schedule function of tiger is complicated. We implemented an automated independent chunk search program based on several properties of the key schedule function and the step function of tiger that we found. For the details of our program, refer to the full version of this paper [12].

This section will describe the independent chunks, which can be used to attack 23-step tiger. The independent words are X_{15} and X_{23}, which will be denoted as $X_{15}^{\square}$ and $X_{23}^{\triangle}$ respectively. The overview of the two independent chunks is detailed in Table 3, following the notations in Table 2.

Table 3. Our independent chunks

$X_0^{\square}$	$X_1^{\square}$	$X_2^{\square}$	$X_3^{\triangle,\square}$	X_4^*	X_5^*	X_6^*	$X_7^{\triangle,\square}$
$Y_0^{\triangle,\square}$	$Y_1^{\triangle,\square}$	$Y_2^{\triangle}$	Y_3^*	Y_4^*	Y_5^*	Y_6^*	$Y_7^{\triangle,\square}$
X_8^*	$X_9^{\triangle}$	X_{10}^*	X_{11}^*	X_{12}^*	X_{13}^*	X_{14}^*	$X_{15}^{\triangle,\square}$
$Y_8^{\triangle,\square}$	$Y_9^{\square}$	$Y_{10}^{\square}$	$Y_{11}^{\square}$	$Y_{12}^{\square}$	$Y_{13}^{\square}$	$Y_{14}^{\square}$	$Y_{15}^{\triangle,\square}$
$X_{16}^{\square}$	$X_{17}^{\square}$	$X_{18}^{\square}$	$X_{19}^{\square}$	$X_{20}^{\square}$	$X_{21}^{\square}$	$X_{22}^{\square}$	$X_{23}^{\triangle,\square}$

4.1 The independent chunk $\mathcal{X}^{\triangle}$

This section will explain the independence/dependence from $X_{23}^{\triangle}$ for each message word in detail. In this section the notation X_i^* (resp. Y_i^*) means that X_i (resp. Y_i) is independent from $X_{23}^{\triangle}$. Roughly speaking, we will first regard the message words $X_{16}, \ldots, X_{22}$ as independence from $X_{23}^{\triangle}$, and then determine the relation between the other message words and $X_{23}^{\triangle}$ backwards utilizing the properties of KSF^{-1}.

Before explaining the details for each message word, we point out that several conditions are set on the message words in order to make this chunk work, which are listed in Table 4. We can only change the 19 MSBs of $X_{23}^{\triangle}$ in order to make Y_9 be independent from it. More details are given below.

The message words $(Y_8, \ldots, Y_{15})$. $Y_{10}, \ldots, Y_{14}$ are independent from $X_{23}^{\triangle}$ because they are computed by KSF^{-1} using $X_{17}^*, \ldots, X_{22}^*$.

- Y_{15}: $Y_{15} = X_{23}^{\triangle} + (X_{22}^* \oplus const_2)$
  Obviously the 19 MSBs of Y_{15} will change with $X_{23}^{\triangle}$.

- Y_9: $Y_9 = X_{17}^* + (X_{16}^* \oplus ((\neg Y_{15}^{\triangle}) \ll 19))$
  Since only the 19 MSBs of Y_{15} change with $X_{23}^{\triangle}$ and these bits disappear after the bit-shift operation, Y_9 is independent from $X_{23}^{\triangle}$. This is also the reason why we can only change the 19 MSBs of $X_{23}^{\triangle}$.

- Y_8: $Y_8 = X_{16}^* - Y_{15}^{\triangle}$
  The 19 MSBs of Y_8 will change with $X_{23}^{\triangle}$. Moreover, from two conditions in Table 4: (1) $X_{16,63-45} = 1 \cdots 1$; and (2) no carry occurs from bits 44 to 45 during $X_{16} - Y_{15}$, we can get that the 19 MSBs of $Y_8^{\triangle}$ are always the bitwise complement of the 19 MSBs of $Y_{15}^{\triangle}$, which is denoted as $Y_{8,63-45}^{\triangle} = \neg Y_{15,63-45}^{\triangle}$.

Table 4. The conditions on the message words

X_{i,j_2-j_1} (resp. Y_{i,j_2-j_1}) is the consecutive bits from j_1 to j_2 of X_i (resp. Y_i).

$X_{0,63-45} = 1 \cdots 1$;	$X_{1,63-45} = const_{1,63-45}$;	$X_{2,63-45} = 0 \cdots 0$;
$Y_{6,63-45} = const_{1,63-45}$;	$Y_{7,44-26} = 0 \cdots 0$	$X_{8,63-45} = 1 \cdots 1$;
$X_{10,63-45} = 0 \cdots 0$;	$X_{14,63-45} = const_{2,63-45}$;	$Y_{9,63-45} = 0 \cdots 0$;
$Y_{14,63-45} = const_{1,63-45}$;	$X_{16,63-45} = 1 \cdots 1$;	
No carry occurs from bits 44 to 45 during the following computations:		
$X_{16} - Y_{15}$;	$Y_8 + (X_{15} \oplus const_1)$;	$X_{15} + (X_{14} \oplus const_2)$;
$X_9 + (X_8 \oplus ((\neg Y_7) \ll 19))$;	$X_8 - Y_7$;	$Y_0 + (X_7 \oplus const_1)$;
$Y_2 - Y_1$;		

Hereafter we will denote all the message words, which change with $X_{23}^{\triangle}$, as equations on $Y_{15,63-45}^{\triangle}$.

The message words $(X_8, \ldots, X_{15})$. $X_{10}, \ldots, X_{14}$ are independent from $X_{23}^{\triangle}$ because they are computed by KSF^{-1} using $Y_9^*, \ldots, Y_{14}^*$.

- X_{15}: $X_{15} = Y_{15}^{\triangle} \oplus Y_{14}^*$
  The 19 MSBs of X_{15} will change with $X_{23}^{\triangle}$. From one condition in Table 4: $Y_{14,63-45} = const_{1,63-45}$, we can get that $X_{15,63-45}^{\triangle} = Y_{15,63-45}^{\triangle} \oplus const_{1,63-45}$.

- X_9: $X_9 = Y_9^* \oplus Y_8^{\triangle}$
  The 19 MSBs of X_9 will change with $X_{23}^{\triangle}$. From one condition in Table 4: $Y_{9,63-45} = 0 \cdots 0$, we can get that $X_{9,63-45}^{\triangle} = Y_{8,63-45}^{\triangle} = \neg Y_{15,63-45}^{\triangle}$.

- X_8: $X_8 = Y_8^{\triangle} + (X_{15}^{\triangle} \oplus const_1)$
  Since Y_8 and X_{15} will only change their 19 MSBs with $X_{23}^{\triangle}$, the 45 LSBs of X_8, namely $X_{8,44-0}$, are independent from $X_{23}^{\triangle}$. Moreover, from one condition in Table 4: no carry occurs from bits 44 to 45 during $Y_8 + (X_{15} \oplus const_1)$, we can get that $X_{8,63-45} = Y_{8,63-45}^{\triangle} + (X_{15,63-45}^{\triangle} \oplus const_{1,63-45}) = (\neg Y_{15,63-45}^{\triangle}) + (Y_{15,63-45}^{\triangle} \oplus const_{1,63-45} \oplus const_{1,63-45}) = 1 \cdots 1$. Note that $X_{8,63-45}$ is predetermined to be $1 \cdots 1$ as a condition in Table 4. Therefore $X_{8,63-45}$ does not change with $X_{23}^{\triangle}$. Finally we get that X_8 is independent from $X_{23}^{\triangle}$.

The messag words $(Y_0, \ldots, Y_7)$. $Y_3, \ldots, Y_6$ are independent from $X_{23}^{\triangle}$ because they are computed by KSF^{-1} using $X_{10}^*, \ldots, X_{14}^*$.

- Y_7: $Y_7 = X_{15}^{\triangle} + (X_{14}^* \oplus const_2)$
  The 19 MSBs of Y_7 will change with $X_{23}^{\triangle}$. From two conditions in Table 4: (1) $X_{14,63-45} = const_{2,63-45}$; and (2) no carry occurs from bits 44 to 45 during $X_{15} + (X_{14} \oplus const_2)$, we can get that $Y_{7,63-45}^{\triangle} = X_{15,63-45}^{\triangle} = Y_{15,63-45}^{\triangle} \oplus const_{1,63-45}$.

- Y_2: $Y_2 = X_{10}^* \oplus X_9^{\triangle}$
  The 19 MSBs of Y_2 will change with $X_{23}^{\triangle}$. From one condition in Table 4: $X_{10,63-45} = 0 \cdots 0$, we can get that $Y_{2,63-45}^{\triangle} = X_{9,63-45}^{\triangle} = \neg Y_{15,63-45}^{\triangle}$.

- Y_1: $Y_1 = X_9^{\triangle} + (X_8^* \oplus ((\neg Y_7^{\triangle}) \lll 19))$
  Because $Y_7^{\triangle}$ will only change its 19 MSBs with $X_{23}^{\triangle}$, which disappear after the bit-shift operation, $X_8^* \oplus ((\neg Y_7^{\triangle}) \lll 19)$ is independent from $X_{23}^{\triangle}$. Therefore the 19 MSBs of Y_1 will change with $X_{23}^{\triangle}$. From three conditions in Table 4: (1) $X_{8,63-45} = 1 \cdots 1$; (2) $Y_{7,44-26} = 0 \cdots 0$; and (3) no carry

occurs from bits 44 to 45 during $X_9 + (X_8 \oplus ((\neg Y_7) \ll 19))$, we can get $Y^{\triangle}_{1,63-45} = X^{\triangle}_{9,63-45} = \neg Y^{\triangle}_{15,63-45}$.

– Y_0: $Y_0 = X^*_8 - Y^{\triangle}_7$
The 19 MSBs of Y_0 will change with $X^{\triangle}_{23}$. From two conditions in Table 4: (1) $X_{8,63-45} = 1 \cdots 1$; and (2) no carry occurs from bits 44 to 45 during $X_8 - Y_7$, we can get that $Y^{\triangle}_{0,63-45} = \neg Y^{\triangle}_{7,63-45} = \neg(Y^{\triangle}_{15,63-45} \oplus const_{1,63-45})$.

The message words $(X_0, \ldots, X_7)$. X_4, X_5 and X_6 are independent from $X^{\triangle}_{23}$, because they are computed by KSF^{-1} using Y^*_3, Y^*_4, Y^*_5 and Y^*_6.

– X_7: $X_7 = Y^{\triangle}_7 \oplus Y^*_6$
The 19 MSBs of X_7 will change with $X^{\triangle}_{23}$. From one condition in Table 4: $Y_{6,63-45} = const_{1,63-45}$, we can get that $X^{\triangle}_{7,63-45} = Y^{\triangle}_{7,63-45} \oplus const_{1,63-45}$ $= (Y^{\triangle}_{15,63-45} \oplus const_{1,63-45}) \oplus const_{1,63-45} = Y^{\triangle}_{15,63-45}$.

– X_3: $X_3 = Y^*_3 + (Y^{\triangle}_2 \oplus (\neg Y^{\triangle}_1) \ll 19)$
Y_1 only changes its 19 MSBs with $X^{\triangle}_{23}$, which will disappear after the bit-shift operation. So $(\neg Y^{\triangle}_1) \ll 19$ is independent from $X^{\triangle}_{23}$. The 19 MSBs of X_3 will change with $X^{\triangle}_{23}$. Here we cannot determine the relation between $X^{\triangle}_{3,63-45}$ and $Y^{\triangle}_{15,63-45}$, but it is actually not necessary for this chunk. The reason is that step 4, where X_3 is used, will be skipped by the partial-matching and partial-fixing techniques. More details are shown in Section 5.3.

– X_2: $X_2 = Y^{\triangle}_2 - Y^{\triangle}_1$
Since Y_2 and Y_1 will only change their 19 MSBs with $X^{\triangle}_{23}$, the 45 LSBs of X_2, namely $X_{2,44-0}$, will be independent from $X^{\triangle}_{23}$. From one condition in Table 4: no carry occurs from bits 44 to 45 during $Y_2 - Y_1$, we can get that $X_{2,63-45} = Y^{\triangle}_{2,63-45} - Y^{\triangle}_{1,63-45} = (\neg Y^{\triangle}_{15,63-45}) - (\neg Y^{\triangle}_{15,63-45}) = 0 \cdots 0$. Note that $X_{2,63-45}$ is predetermined to be $0 \cdots 0$ as a condition in Table 4. $X_{2,63-45}$ does not change with $X^{\triangle}_{23}$. Therefore X_2 is independent from $X^{\triangle}_{23}$.

– X_1: $X_1 = Y^{\triangle}_1 \oplus Y^{\triangle}_0$
Similarly the 45 LSBs of X_1, namely $X_{1,44-0}$, are independent from $X^{\triangle}_{23}$. $X_{1,63-45} = Y^{\triangle}_{1,63-45} \oplus Y^{\triangle}_{0,63-45} = (\neg Y^{\triangle}_{15,63-45}) \oplus (\neg(Y^{\triangle}_{15,63-45} \oplus const_{1,63-45})) =$ $const_{1,63-45}$. Note that $X_{1,63-45}$ is predetermined to be $const_{1,63-45}$ as a condition in Table 4, so $X_{1,63-45}$ does not change with $X^{\triangle}_{23}$. Therefore X_1 is independent from $X^{\triangle}_{23}$.

– X_0: $X_0 = Y^{\triangle}_0 + (X^{\triangle}_7 \oplus const_1)$
Similarly $X_{0,44-0}$ is independent from $X^{\triangle}_{23}$. From one condition in Table 4: no carry occurs from bits 44 to 45 during $Y_0 + (X_7 \oplus const_1)$, we can get that $X_{0,63-45} = Y^{\triangle}_{0,63-45} + (X^{\triangle}_{7,63-45} \oplus const_{1,63-45}) = (\neg(Y^{\triangle}_{15,63-45} \oplus$

$const_{1,63-45})) + (Y^{\triangle}_{15,63-45} \oplus const_{1,63-45}) = 1 \cdots 1$. Note that $X_{0,63-45}$ is predetermined to be $1 \cdots 1$ as a condition in Table 4, so $X_{0,63-45}$ is also independent from $X^{\triangle}_{23}$. Therefore X_0 is independent from $X^{\triangle}_{23}$.

4.2 The independent chunk $\mathcal{X}^{\square}$

This section will explain the independence/dependence from $X^{\square}_{15}$ for each message word in detail. In this section, the notations X^*_i (resp. Y^*_i) means that X_i (resp. Y_i) is independent from $X^{\square}_{15}$. Roughly speaking, we will first define the message words $X_8, \ldots, X_{14}$ are independent from $X^{\square}_{15}$, and then determine the relation between the other message words and $X^{\square}_{15}$ backwards and forwards utilizing the properties of KSF^{-1} and KSF respectively.

We point out that the independence/dependence of the other message words from $X^{\square}_{15}$ is determined just following the specifications of KSF and KSF^{-1}. We only need to pay attention to make sure that this chunk does not influence the bit positions, where the conditions in Table 4 are set, in order to guarantee the two chunks are *really* independent. Note that all the conditions in Table 4 locate at upper bits. We decide to change several lower bits of $X^{\square}_{15}$ in order to avoid bit overlap at some message word. Finally we will change bits $19 - 9$ of $X^{\square}_{15}$, namely $X_{15,19-9}$. [3] Moreover, in order to clearly make sure that this chunk will not influence the conditions in Table 4, we set several conditions on the message words to control bit-carry propagations, which are listed in Table 5.

Table 5. Conditions on message words to control carry propagation

$X_{7,22} = const_{1,22}$	$Y_{0,22} = 0;$	$Y_{1,41} = 0;$	$Y_{2,41} = 1;$	$Y_{7,21} = 0;$	$X_{8,20} = 1;$
$X_{8,21} = 1;$	$X_{8,40} = 1;$	$X_{9,40} = 0$	$X_{10,21} = 0;$	$X_{11,40} = 1;$	$X_{13,41} = 0;$
$X_{14,20} = const_{2,20};$	$X_{14,42} = 1;$	$X_{15,20} = 0;$	$Y_{8,43} = 0;$	$Y_{9,21} = 0;$	$Y_{10,40} = 1;$
$Y_{12,41} = 0;$	$Y_{13,42} = 0;$	$Y_{15,43} = 0;$			

In the following discussion, we will mainly pay attention to which bit positions of the message words will change with $X^{\square}_{15}$.

The message words $(Y_0, \ldots, Y_7)$. $Y_2, \ldots, Y_6$ will not change with $X^{\square}_{15}$ since they are computed by KSF^{-1} using $X^*_9, \ldots, X^*_{14}$.

- Y_7: $Y_7 = X^{\square}_{15} + (X^*_{14} \oplus const_2)$
  From two conditions in Table 5: $X_{15,20} = 0$ and $X_{14,20} = const_{2,20}$, no carry will occur from bits 20 to 21 no matter how X_{15} change its bits $19 - 9$. Therefore, only $Y_{7,20-9}$ will change with $X^{\square}_{15}$.

[3] The reason why we choose 11 lower bits is because of our attack procedure in Section 5.4.

– Y_1: $Y_1 = X_9^* + (X_8^* \oplus ((\neg Y_7^{\square}) \ll 19))$
From two conditions in Table 5: $X_{8,40} = 1$ and $Y_{7,21} = 0$, bit 40 of $X_8^* \oplus ((\neg Y_7^{\square}) \ll 19)$ is 0. From another condition in Table 5: $X_{9,40} = 0$, no carry will occur from bits 40 to 41 during $Y_7^{\square}$ changes. Therefore, only $Y_{1,40-28}$ will change with $X_{15}^{\square}$.

– Y_0: $Y_0 = X_8^* - Y_7^{\square}$
From two conditions in Table 5: $X_{8,21} = 1$ and $Y_{7,21} = 0$, no carry will happen from bits 21 to 22 during $Y_7^{\square}$ changes. Therefore, only $Y_{0,21-9}$ will change with $X_{15}^{\square}$.

The message words $(X_0, \ldots, X_7)$. X_4, X_5 and X_6 will not change with $X_{15}^{\square}$ because they are computed by KSF^{-1} using Y_3^*, Y_4^*, Y_5^* and Y_6^*.

– X_7: $X_7 = Y_7^{\square} \oplus Y_6^*$
$X_{7,20-9}$ will change with $X_{15}^{\square}$.

– X_3: $X_3 = Y_3^* + (Y_2^* \oplus ((\neg Y_1^{\square}) \ll 19))$
Since no condition has been set on X_3 in Table 4, we do not need to pay attention to which bit positions of X_3 will change with $X_{15}^{\square}$, but only the fact that it will change with $X_{15}^{\square}$.

– X_2: $X_2 = Y_2^* - Y_1^{\square}$
From two conditions in Table 5: $Y_{1,41} = 0$ and $Y_{2,41} = 1$, no carry will occur from bits 41 to 42 during $Y_1^{\square}$ changes. Therefore $X_{2,41-28}$ will change with $X_{15}^{\square}$.

– X_1: $X_1 = Y_1^{\square} \oplus Y_0^{\square}$
$X_{1,40-9}$ will change with $X_{15}^{\square}$.

– X_0: $X_0 = Y_0^{\square} + (X_7^{\square} \oplus const_1)$
From two conditions in Table 5: $Y_{0,22} = 0$ and $X_{7,22} = const_{1,22}$, no carry will occur from bits 22 to 23 during $Y_0^{\square}$ and $X_7^{\square}$ change. Therefore $X_{0,22-9}$ will change with $X_{15}^{\square}$.

The message words $(Y_8, \ldots, Y_{15})$.

– Y_8: $Y_8 = X_8^* - (X_{15}^{\square} \oplus const_1)$
From two conditions in Table 5: $X_{8,20} = 1$ and $X_{15,20} = 0$ ($const_{1,20} = 0$), no carry will occur from bits 20 to 21 during $X_{15}^{\square}$ changes. So $Y_{8,20-9}$ will change with $X_{15}^{\square}$.

– Y_9: $Y_9 = X_9^* \oplus Y_8^{\square}$
$Y_{9,20-9}$ will change with $X_{15}^{\square}$.

– Y_{10}: $Y_{10} = X^*_{10} + Y^{\square}_9$
From two conditions in Table 5: $X_{10,21} = 0$ and $Y_{9,21} = 0$, no carry will occur from bits 21 to 22 during $Y^{\square}_9$ changes. So $Y_{10,21-9}$ will change with $X^{\square}_{15}$.

– Y_{11}: $Y_{11} = X^*_{11} - (((\neg Y^{\square}_9) \lll 19) \oplus Y^{\square}_{10})$
From two conditions in Table 5: $Y_{9,21} = 0$ and $Y_{10,40} = 1$, bit 40 of $((\neg Y^{\square}_9) \lll 19) \oplus Y^{\square}_{10}$ is 0. From another condition in Table 5: $X_{11,40} = 1$, no carry will occur from bits 40 to 41 during $Y^{\square}_9$ and $Y^{\square}_{10}$ change. Therefore $Y_{11,40-9}$, will change with $X^{\square}_{15}$.

– Y_{12}: $Y_{12} = X^*_{12} \oplus Y^{\square}_{11}$
$Y_{12,40-9}$ will change with $X^{\square}_{15}$.

– Y_{13}: $Y_{13} = X^*_{13} + Y^{\square}_{12}$
From two conditions in Table 5: $X_{13,41} = 0$ and $Y_{12,41} = 0$, no carry will occur from bits 41 to 42 during $Y^{\square}_{12}$ changes. So $Y_{13,41-9}$ will change with $X^{\square}_{15}$.

– Y_{14}: $Y_{14} = X^*_{14} - (Y^{\square}_{13} \oplus ((\neg Y^{\square}_{12}) \ggg 23))$
From one condition in Table 5: $Y_{13,42} = 0$, bit 42 of $Y^{\square}_{13} \oplus ((\neg Y^{\square}_{12}) \ggg 23)$ is 0. From another condition in Table 5: $X_{14,42} = 1$, no carry will occur from bits 42 to 43 during $Y^{\square}_{12}$ and $Y^{\square}_{13}$ change. Therefore $Y_{14,42-0}$ will change with $X^{\square}_{15}$.

– Y_{15}: $Y_{15} = X^{\square}_{15} \oplus Y^{\square}_{14}$
$Y_{15,42-0}$ will change with $X^{\square}_{15}$.

The message words $(X_{16}, \ldots, X_{23})$. We do not need to pay attention to which bit positions of the message words $X_{17}, \ldots, X_{23}$ will change with $X^{\square}_{15}$, but only the fact that these message words will change with $X^{\square}_{15}$.

– X_{16}: $X_{16} = Y^{\square}_8 + Y^{\square}_{15}$
From two conditions in Table 5: $Y_{8,43} = 0$ and $Y_{15,43} = 0$, no carry will occur from bits 43 to 44 during $Y^{\square}_8$ and $Y^{\square}_{15}$ change. Therefore $X_{16,43-0}$ will change with $X^{\square}_{15}$.

4.3 Summary of our independent chunks

We will first find a message block that can satisfy all the conditions in Tables 4 and 5. Then we will change the 19 MSBs of $X^{\triangle}_{23}$ and bits $19 - 9$ of $X^{\square}_{15}$ independently to apply a pseudo-preimage attack on tiger.

5 Preimage attack on 23-step Tiger

This section will propose a pseudo-preimage attack on tiger up to 23 steps, which will be converted to a preimage attack on 23-step Tiger. The overview of the

attack has been shown in Table 6. The attack target is the first 23 steps. Hence, X_{23} is erased from Table 6.

Table 6. Overview of our pseudo-preimage attack on tiger

$X_0^{\square}$	$X_1^{\square}$	$X_2^{\square}$	$X_3^{\triangle,\square}$	X_4^{*}	X_5^{*}	X_6^{*}	$X_7^{\triangle,\square}$
$\longrightarrow$	$\longrightarrow$	$\longrightarrow \vert$	partial-matching				
X_8^{*}	$X_9^{\triangle}$	X_{10}^{*}	X_{11}^{*}	X_{12}^{*}	X_{13}^{*}	X_{14}^{*}	$X_{15}^{\triangle,\square}$
$\vert \longleftarrow$	$\longleftarrow$	$\mathcal{E}^{\triangle}$ calculations for $\mathcal{X}^{\triangle}$ chunk				$\longleftarrow \vert$	initial-structure
$X_{16}^{\square}$	$X_{17}^{\square}$	$X_{18}^{\square}$	$X_{19}^{\square}$	$X_{20}^{\square}$	$X_{21}^{\square}$	$X_{22}^{\square}$	
$\vert \longrightarrow$	$\mathcal{E}^{\square}$ calculations for $\mathcal{X}^{\square}$ chunk				$\longrightarrow$	$\longrightarrow$	

5.1 Precomputation

Before starting the pseudo-preimage attack on tiger, we need to find a message block $X_0||\cdots||X_7$, which can satisfy all the conditions in Tables 4 and 5. The total number of the conditions in these two tables is 237. But the complexity of searching such a message block will be greatly reduced by the message modification technique. Moreover, we stress that the precomputation will be executed only *once* during the pseudo-preimage attack on tiger. The search procedure is as follows.

1. Randomly choose a message block and modify X_0, X_1, X_2 and X_7 to satisfy the conditions on them.
2. Modify Y_0 to satisfy the condition $Y_{0,22} = 0$, and then inversely calculate X_0 without changing the other message words. Due to the long bit distance from bits 22 to 45, the conditions on X_0 will not be influenced with an overwhelming probability.
3. Similarly make the conditions on Y_1 and Y_2 be satisfied by modifying X_1, X_2 and X_3.
4. Modify Y_6 to satisfy the conditions, and then inversely compute X_6.
5. Make the conditions on Y_7 be satisfied by modifying Y_6 and X_6.
6. Note that the conditions on X_8 in Table 4 will be automatically satisfied if both X_0 and Y_6 satisfy the conditions on them.
7. The remaining conditions will be satisfied by the exhaustive search.

In total there are 115 conditions, which will be satisfied by the exhaustive search at step 7. Although more conditions can be satisfied by applying message modification, we will not discuss more about the precomputation due to limited space and the fact $2^{115} \ll 2^{192}$.

5.2 Apply the initial-structure technique at step 16

We will illustrate how the initial-structure technique works at step 16 of tiger, which is also shown in Fig. 6. Recall that the 19 MSBs of X_{15} will change with the $\mathcal{X}^{\triangle}$ chunk, its bits $19-9$ will change with the $\mathcal{X}^{\square}$ chunk, and the other bits wll be constant. Let the 19 MSBs, the 20 LSBs and the intermediate 25 bits of $X_{15}^{\triangle,\square}$ be $X_{15}^{\triangle}$, $X_{15}^{\square}$ and α respectively. Then $X_{15}^{\triangle,\square}$ is written as $(X_{15}^{\triangle}||0_{(45)}) \oplus (0_{(19)}||\alpha||X_{15}^{\square})$, where $0_{(b)}$ represents b-bit sequential '0's. We can analyze the impact to step 16 from $\mathcal{X}^{\triangle}$ and from $\mathcal{X}^{\square}$ independently.

We first fix the constant numbers $const, const'$, and $const''$ marked in Fig. 6 to randomly chosen values. Then, every time we obtain the value of $X_{15}^{\triangle}$, we compute

$$(A_{15}^{\triangle}, B_{15}^{\triangle}, C_{15}^{\triangle}) \leftarrow (const, const', (const'' \oplus (X_{15}^{\triangle} \| 0_{(45)}))).$$

Similarly, every time we obtain the value of $X_{15}^{\square}$, we compute

$$temp \leftarrow const'' \oplus (0_{(19)} \| \alpha \| X_{15}^{\square}),$$
$$(A_{16}^{\square}, B_{16}^{\square}, C_{16}^{\square}) \leftarrow ((const' + odd(temp)) \times 7,\ temp,\ const - even(temp)).$$

Finally, we can compute $(A_{15}^{\triangle}, B_{15}^{\triangle}, C_{15}^{\triangle})$ and $(A_{16}^{\square}, B_{16}^{\square}, C_{16}^{\square})$ independently even though X_{15} are affected by both chunks.

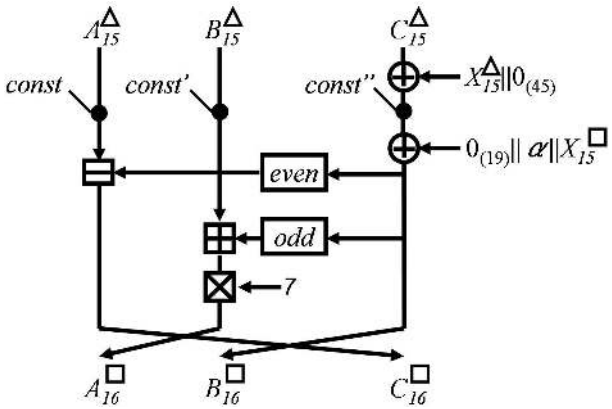

Fig. 6. Initial-structure at step 16

5.3 Apply the partial-matching and partial-fixing techniques at steps 8−4

We will illustrate how to partially compare $(A_3^{\square}, B_3^{\square}, C_3^{\square})$ with $(A_8^{\triangle}, B_8^{\triangle}, C_8^{\triangle})$ using the partial-matching and partial-fixing techniques, which is also shown

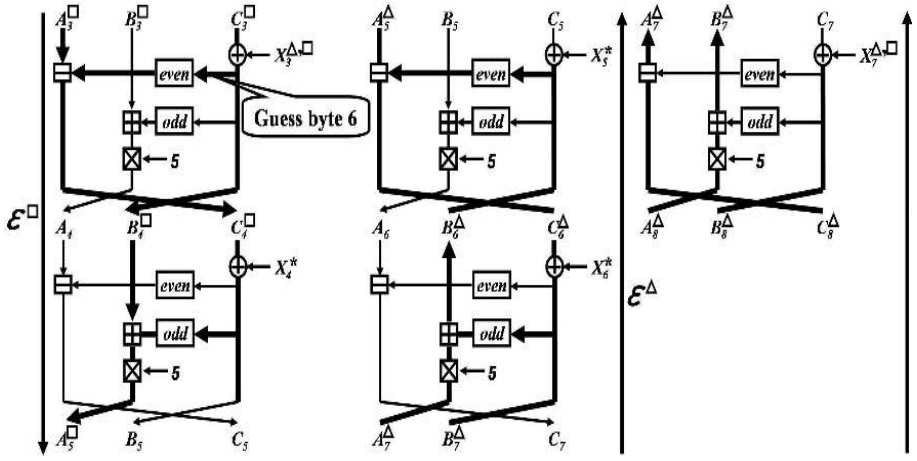

Fig. 7. Partial-matching and partial-fixing for steps 8-4

in Fig. 7. The main idea is that for both $\mathcal{E}^{\triangle}$ and $\mathcal{E}^{\square}$, the value of A_5 will be partially computed. With this idea, we can compare the 45 LSBs of $A_5^{\square}$ and of $A_5^{\triangle}$.

- For the $\mathcal{E}^{\square}$ computation, since only the 19 MSBs of X_3 change with $\mathcal{X}^{\triangle}$, bits $44-0$ of $X_3^{\triangle,\square}$ are known. At step 4, we compute bits $44-0$ of $B_4^{\square}$. Then we guess byte 6 of $X_3 \oplus C_3$, namely bits $55-48$, and compute $C_4^{\square}$. At step 5, we compute bits $44-0$ of $A_5^{\square}$.
- For the $\mathcal{E}^{\triangle}$ computation, we can compute $A_5^{\triangle}$ easily from step 8.

5.4 Pseudo-preimage attack on tiger

1. Generate a message block satisfying all the conditions in Tables 4 and 5. The details have been shown in Section 5.1.
2. Set *const*, *const*′ and *const*″ in Fig. 6 to random values.
3. For all the values of the 19 MSBs of $X_{23}^{\triangle}$,
   (a) Compute the value of all $X_i \in \mathcal{X}^{\triangle}$ and partial value of all $X_i \in \mathcal{X}^{\triangle,\square}$, namely, all bits of $X_9^{\triangle}$ and partially-known bits of $X_{15}^{\triangle,\square}, X_7^{\triangle,\square}$ and $X_3^{\triangle,\square}$. Then, compute the corresponding $IS^{\triangle}$, that is $(A_{15}^{\triangle}, B_{15}^{\triangle}, C_{15}^{\triangle})$. The details have been explained in Section 5.2.
   (b) From $(A_{15}^{\triangle}, B_{15}^{\triangle}, C_{15}^{\triangle})$ and $X_{14}, X_{13}, \ldots, X_8$, compute $\mathcal{E}^{\triangle}$ to obtain the value of $(A_8^{\triangle}, B_8^{\triangle}, C_8^{\triangle})$.
   (c) By following the backward computation of the partial-matching and partial-fixing techniques explained in Section 5.3, compute the values of $A_5^{\triangle}$.
   (d) Store $(\mathcal{X}^{\triangle}, A_5^{\triangle}, A_8^{\triangle}, B_8^{\triangle}, C_8^{\triangle})$ in a table $\mathcal{T}$.

4. For all the values of bits $19-9$ of $X_{15}^{\square}$,
   (a) Compute the value of all $X_i \in \mathcal{X}^{\square}$ and partial value of all $X_i \in \mathcal{X}^{\triangle,\square}$, namely, all bits of $X_{16}^{\square}, X_{17}^{\square}, \ldots, X_{22}^{\square}, X_0^{\square}, X_1^{\square}, X_2^{\square}$ and partially-known bits of $X_3^{\triangle,\square}, X_7^{\triangle,\square}$, and $X_{15}^{\triangle,\square}$. Then, compute the corresponding $IS^{\square}$, that is $(A_{16}^{\square}, B_{16}^{\square}, C_{16}^{\square})$. The details have been explained in Section 5.2.
   (b) From $(A_{16}^{\square}, B_{16}^{\square}, C_{16}^{\square})$ and $X_{16}, X_{17}, \ldots, X_{22}, X_0, X_1, X_2$, compute $\mathcal{E}^{\square}$ to obtain the value of $(A_3^{\square}, B_3^{\square}, C_3^{\square})$.
   (c) At Step 4, we know all bits of $(A_3^{\square}, B_3^{\square}, C_3^{\square})$ and the 45 LSBs of $X_3^{\triangle,\square}$. We compute the 45 LSBs of B_4 by $C_3^{\square} \oplus X_3^{\triangle,\square}$. Then, we exhaustively guess byte 6 of $X_3^{\triangle,\square}$, which are 8 bits (bits 48−55) of $X_3^{\triangle,\square}$. Based on each guessed value, we compute $even(\cdot)$ function and obtain all bits of $C_4^{\square}$.
   (d) At Step 5, we compute the 45 LSBs of $A_5^{\square}$ by using the 45 LSBs of $B_4^{\square}$ and all bits of $C_4^{\square}$.
   (e) Check whether or not the obtained bit-values of $A_5^{\square}$ will match one $A_5^{\triangle}$ in $\mathcal{T}$.
   (f) If it matches, recover the value of $X_3^{\triangle,\square}$. Then, obtain the values of $(A_8^{\square}, B_8^{\square}, C_8^{\square})$ by updating $(A_3^{\square}, B_3^{\square}, C_3^{\square})$ with recovered X_3 and already fixed $X_4, X_5, \ldots, X_7$, and check whether or not the remaining 147 bits match or not.
   (g) If all bits match, the corresponding M and A_0, B_0, C_0 is a valid pseudo-preimage with a probability 2^{-8} (the success probability of guess at Step 4c). If a matched pair does not exist for all the degree of freedom, we will change the value at step 2, and repeats steps 3 and 4.

5.5 The complexity of our pseudo-preimage attack on tiger

We regard one tiger computation as a unit.

Step 1. The complexity is 2^{115}. This step will be executed only once. Since $2^{115} \ll 2^{192}$, we will ignore the complexity of step 1.
Step 2. Negligible.
Step 3a. The complexity is 2^{19} computations of KSF.
Step 3b. The complexity is $2^{19} \times \frac{7}{23}$.
Step 3c. The complexity is $2^{19} \times \frac{3}{23}$.
Step 3d. The memory requirements is 2^{22} message words. ($\mathcal{X}^{\triangle}$ consists of 4 words $X_9^{\triangle}, X_{15}^{\triangle,\square}, X_7^{\triangle,\square}, X_3^{\triangle,\square}$).
Step 4a. The complexity is 2^{11} computations of KSF.
Step 4b. The complexity is $2^{11} \times \frac{10}{23}$.
Step 4c. The complexity is $2^{19} \times \frac{1}{23}$.
Step 4d. The complexity is $2^{19} \times \frac{1}{23}$.
Step 4e. Negligible.

With the complexity less than 2^{19}, we can compare 2^{38} pairs and will find 2^{-7} pairs that match the 45 LSBs. Note that each guess at Step 4c has a success probability 2^{-8}. Therefore, by repeating the steps 2−4 of the above procedure

$2^{162}(= 2^{192-45+7+8})$ times, we expect to obtain a pseudo-preimage. Finally the complexity of finding a pseudo-preimage for Tiger compression function is $2^{181}(= 2^{19} \cdot 2^{162})$. The dominant memory use is 2^{22} words at Step 3d.

5.6 Preimage attack on Tiger

Our pseudo-preimage attack on tiger can be converted to a preimage attack on Tiger adopting the meet-in-the-middle attack on Merkle-Damgård structure detailed in Section 3.2. The complexity is $2^{187.5}$ and the memory requirement is 2^{22} words. Note that we have to fix bit 56 of X_6 to be '1' and the 9 LSBs of X_7 to be binary-encoding of 447 in order to make the bit length matched.

6 Open discussion and conclusion

Compared with the MD4-family, Tiger uses a stronger key schedule function, a stronger step function, but a smaller number of steps. However, based on our analyses, we found several properties of both the key schedule function and the step function, which can be used for the preimage attack.

For the key schedule function, we found the following properties.

- Bit-shift is easily used to introduce independence of computations.
- The large word size is suitable to make upper and lower bits independent with respect to carry.
- Mixing the use of addition, subtraction and XOR does not introduce enough non-linearity. They can be linearized by setting conditions.

For the step function, we found the following properties.

- Even though the whole internal state is updated at each step function, a part of internal state (A_t, B_t) are updated by using independent values (only *even* bytes and only *odd* bytes of $C_{t-1} \oplus X_{t-1}$).
- Tiger's S-boxes are so-called target heavy, which map 8-bit values to 64-bit values. This enables us to obtain the knowledge of a large number of bits by only guessing the values of a small number of bits, and later efficiently find out the correct guesses by matching the large bits.

So far, we have not found a preimage attack on full-step Tiger yet.[4] However, by considering the future attack improvement, the number of steps seems a little bit small with respect to the preimage resistance. For the confidence of security, we suggest that the number of steps should be increased.

Conclusion

This paper presented a meet-in-the-middle pseudo-preimage attack on tiger up to 23 steps with a complexity of 2^{181}. This was converted to a preimage attack on 23-step Tiger with a complexity of $2^{187.5}$. The memory requirement of our attacks is 2^{22} words.

[4] We notice that recently Guo *et al.* announced that they found a preimage attack on full-step Tiger [13].

Acknowledgments The authors would like to thank Kazuo Ohta, Kazuo Sakiyama and anonymous reviewers for their valuable comments.

References

1. Anderson, R., Biham, E.: Tiger: A Fast New Hash Function. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 89-97. Springer, Heidelberg (1996)
2. Indesteege, S., Preneel, B.: Preimages for Reduced-Round Tiger. In: Lucks, S., Sadeghi, A.-R., Wolf, C. (eds.) WEWoRC 2007. LNCS, vol. 4945, pp. 90-99. Springer, Heidelberg (2008)
3. 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)
4. Mendel, F.: Two Passes of Tiger Are Not One-Way. In: Preneel, B. (ed.) AFRICACRYPT 2009. LNCS, vol. 5580, pp. 29-40, Springer, Heidelberg (2009)
5. Kelsey, J., Lucks, S.: Collisions and Near-Collisions for Reduced-Round Tiger. In: Robshaw, M.J.B. (ed.) FSE 2006. LNCS, vol. 4047, pp. 111-125. Springer, Heidelberg (2006)
6. Mendel, F., Preneel, B., Rijmen, V., Yoshida, H., Watanabe, D.: Update on Tiger. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 63-79. Springer, Heidelberg (2006)
7. Mendel, F., Rijmen, V.: Cryptanalysis of the Tiger Hash Function. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 536-550. Springer, Heidelberg (2007)
8. Aoki, K., Sasaki, Y.: Preimage Attacks on One-Block MD4, 63-Step MD5 and More. In: Avanzi, R., Keliher, L., Sica. F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 103-119. Springer, Heidelberg (2008)
9. Morita, H., Ohta, K., Miyaguchi, S.: A Switching Closure Test to Analyze Cryptosystems. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 183-193. Springer, Heidelberg (1991)
10. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1997)
11. 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)
12. Wang, L., Sasaki, Y., Finding Preimages of Tiger Up to 23 Steps (full version of this paper), `http://www.oslab.ice.uec.ac.jp/member/wang/`
13. 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. `http://eprint.iacr.org/2010/016.pdf`