Two Passes of Tiger Are Not One-Way

Florian Mendel

Institute for Applied Information Processing and Communications (IAIK),
Graz University of Technology, Inffeldgasse 16a, A-8010 Graz, Austria
Florian.Mendel@iaik.tugraz.at

Abstract. Tiger is a cryptographic hash function proposed by Anderson and Biham in 1996 and produces a 192-bit hash value. Recently, weaknesses have been shown in round-reduced variants of the Tiger hash function. Collision attacks have been presented for Tiger reduced to 16 and 19 (out of 24) rounds at FSE 2006 and Indocrypt 2006. Furthermore, Mendel and Rijmen presented a 1-bit pseudo-near-collision for the full Tiger hash function at ASIACRYPT 2007. The attack has a complexity of about 2^{47} compression function evaluations. While there exist several collision-style attacks for Tiger, the picture is different for preimage attacks. At WEWoRC 2007, Indesteege and Preneel presented a preimage attack on Tiger reduced to 12 and 13 rounds with a complexity of $2^{64.5}$ and $2^{128.5}$, respectively.
In this article, we show a preimage attack on Tiger with two passes (16 rounds) with a complexity of about 2^{174} compression function evaluations. Furthermore, we show how the attack can be extended to 17 rounds with a complexity of about 2^{185}. Even though the attacks are only slightly faster than brute force search, they present a step forward in the cryptanalysis of Tiger.

1 Introduction

A cryptographic hash function H maps a message M of arbitrary length to a fixed-length hash value h. A cryptographic hash function has to fulfill the following security requirements:

- *Collision resistance:* it is practically infeasible to find two messages M and M^*, with $M^* \neq M$, such that $H(M) = H(M^*)$.
- *Second preimage resistance:* for a given message M, it is practically infeasible to find a second message $M^* \neq M$ such that $H(M) = H(M^*)$.
- *Preimage resistance:* for a given hash value h, it is practically infeasible to find a message M such that $H(M) = h$.

The resistance of a hash function to collision and (second) preimage attacks depends in the first place on the length n of the hash value. Regardless of how a hash function is designed, an adversary will always be able to find preimages or second preimages after trying out about 2^n different messages. Finding collisions requires a much smaller number of trials: about $2^{n/2}$ due to the birthday paradox. A function is said to achieve *ideal security* if these bounds are guaranteed.

B. Preneel (Ed.): AFRICACRYPT 2009, LNCS 5580, pp. 29–40, 2009.
http://dx.doi.org/10.1007/978-3-642-02384-2_3

Tiger is a cryptographic iterated hash function that processes 512-bit blocks and produces a 192-bit hash value. It was proposed by Anderson and Biham in 1996. Recent cryptanalytic results on the hash function Tiger mainly focus on collision attacks. At FSE 2006, Kelsey and Lucks presented a collision attack on 16 and 17 (out of 24) rounds of Tiger [6]. Both attacks have a complexity of about 2^{44} evaluations of the compression function. These results were later improved by Mendel *et al.* in [9]. They showed that a collision can be found for Tiger reduced to 19 rounds with a complexity of about 2^{62} evaluations of the compression function. At Asiacrypt 2007, Mendel and Rijmen presented the first attack on the full Tiger hash function [10]. They showed that a 1-bit pseudo-near-collision for Tiger can be constructed with a complexity of about 2^{47} compression function evaluations.

While several results have been published regarding the collision-resistance of Tiger, this picture is different for preimage attacks. At WEWoRC 2007, Indesteege and Preneel [4] presented a preimage attack on Tiger reduced to 12 and 13 rounds with a complexity of $2^{64.5}$ and $2^{128.5}$, respectively.

In this article, we will present a security analysis with respect to preimage resistance for the hash function Tiger. We show a preimage attack on Tiger reduced to 2 passes (16 rounds). It has a complexity of about 2^{174} compression function evaluations and memory requirements of 2^{39}. Very recently Isobe and Shibutani presented a preimage attack on 2 passes of Tiger with complexity of about 2^{161} and memory requirements of 2^{32} [5]. This is slightly more efficient than the attack presented in this paper. However, their attack method seems to be limited to 2 passes, while our attack can be extended to 17 rounds. In detail, we show how the attack can be extended to 17 rounds with a complexity of about 2^{185} and memory requirements of 2^{160}

In the preimage attack on Tiger, we combine weaknesses in the key schedule of Tiger with a generic meet-in-the-middle approach to construct a preimage for the compression function faster than brute force search. A similar attack strategy was use to construct preimages for the compression function of round-reduced MD5 in [2,12]. Once we have found a preimage for the compression function of round-reduced Tiger, we use a meet-in-the-middle attack respectively a tree based approach, to turn it into a preimage attack for the hash function.

The remainder of this article is structured as follows. A description of the Tiger hash function is given in Section 2. In Section 3, we present preimages for the compression function of Tiger reduced to 16 rounds (2 passes) and 17 rounds. We show how to extend these attacks for the compression function to the hash function in Section 4. Finally, we present conclusions in Section 5.

2 Description of the Hash Function Tiger

Tiger is an iterated hash function based on the Merkle-Damgård construction. It processes 512-bit input message blocks, maintains a 192-bit state and produces a 192-bit hash value. In the following, we briefly describe the hash function. It basically consists of two parts: the key schedule and the state update trans-

formation. A detailed description of the hash function is given in [1]. For the remainder of this article, we will follow the notation given in Table 1.

Table 1. Notation

Notation	Meaning
$A \boxplus B$	addition of A and B modulo 2^{64}
$A \boxminus B$	subtraction of A and B modulo 2^{64}
$A \boxtimes B$	multiplication of A and B modulo 2^{64}
$A \oplus B$	bit-wise XOR-operation of A and B
$\neg A$	bit-wise NOT-operation of A
$A \ll n$	bit-shift of A by n positions to the left
$A \gg n$	bit-shift of A by n positions to the right
$A[i]$	the i-th bit of the word A (64 bits)
X_i	message word i (64 bits)
round	single execution of the round function
pass	set of consecutive *round*, has a size of 8 (1 *pass* = 8 *rounds*)

2.1 State Update Transformation

The state update transformation of Tiger starts from a (fixed) initial value IV of three 64-bit words and updates them in three passes of eight rounds each. In each round one 64-bit word X introduced by the key schedule is used to update the three state variables A, B and C as follows:

$$\begin{aligned} C &= C \oplus X \\ A &= A \boxminus \mathbf{even}(C) \\ B &= B \boxplus \mathbf{odd}(C) \\ B &= B \boxtimes \texttt{mult} \end{aligned}$$

The results are then shifted such that A, B, C will be B, C, A in the next iteration. Fig. 1 shows one round of the state update transformation of Tiger.
The non-linear functions **even** and **odd** used in each round are defined as follows:

$$\begin{aligned} \mathbf{even}(C) &= T_1[c_0] \oplus T_2[c_2] \oplus T_3[c_4] \oplus T_4[c_6] \\ \mathbf{odd}(C) &= T_4[c_1] \oplus T_3[c_3] \oplus T_2[c_5] \oplus T_1[c_7] \end{aligned}$$

where state variable C is split into eight bytes $c_7, \ldots, c_0$ with c_7 is the most significant byte and c_0 is the least significant byte. Four S-boxes $T_1, \ldots, T_4 : \{0,1\}^8 \rightarrow \{0,1\}^{64}$ are used to compute the output of the non-linear functions **even** and **odd**. For the definition of the S-boxes we refer to [1]. Note that state variable B is multiplied with the constant $\texttt{mult} \in \{5, 7, 9\}$ at the end of each round. The value of the constant is different in each pass of the Tiger hash function.

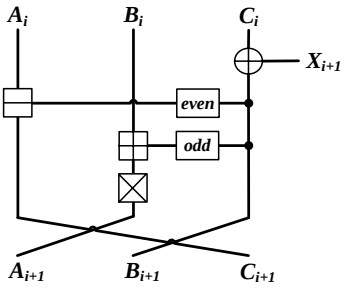

Fig. 1. The round function of Tiger.

After the last round of the state update transformation, the initial values A_{-1}, B_{-1}, C_{-1} and the output values of the last round A_{23}, B_{23}, C_{23} are combined, resulting in the final value of one iteration (feed forward). The result is the final hash value or the initial value for the next message block.

$$\begin{aligned} A_{24} &= A_{-1} \oplus A_{23} \\ B_{24} &= B_{-1} \boxminus B_{23} \\ C_{24} &= C_{-1} \boxplus C_{23} \end{aligned}$$

2.2 Key Schedule

The key schedule takes a 512-bit message block $X_0, \ldots, X_7$ and produces 24 64-bit words $X_0, \ldots, X_{23}$. It is an invertible function which ensures that changing a small number of bits in the message will affect a lot of bits in the next pass. While the message words $X_0, \ldots, X_7$ are used in the first pass to update the state variables, the remaining 16 message words, 8 for the second pass and 8 for the third pass, are generated by applying the key schedule as follows:

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

The key schedule modifies the inputs $(I_0, \ldots, I_7)$ in two steps:

first step	**second step**
$T_0 = I_0 \boxminus (I_7 \oplus \texttt{A5A5A5A5A5A5A5A5})$	$O_0 = T_0 \boxplus T_7$
$T_1 = I_1 \oplus T_0$	$O_1 = T_1 \boxminus (O_0 \oplus ((\neg T_7) \lll 19))$
$T_2 = I_2 \boxplus T_1$	$O_2 = T_2 \oplus O_1$
$T_3 = I_3 \boxminus (T_2 \oplus ((\neg T_1) \lll 19))$	$O_3 = T_3 \boxplus O_2$
$T_4 = I_4 \oplus T_3$	$O_4 = T_4 \boxminus (O_3 \oplus ((\neg O_2) \ggg 23))$
$T_5 = I_5 \boxplus T_4$	$O_5 = T_5 \oplus O_4$
$T_6 = I_6 \boxminus (T_5 \oplus ((\neg T_4) \ggg 23))$	$O_6 = T_6 \boxplus O_5$
$T_7 = I_7 \oplus T_6$	$O_7 = T_7 \boxminus (O_6 \oplus \texttt{0123456789ABCDEF})$

The final values $(O_0, \ldots, O_7)$ are the output of the key schedule.

3 Preimage Attacks on the Compression Function

In this section, we will present two preimage attacks on the compression function of Tiger – one for Tiger with 2 passes (16 rounds) and one for 17 rounds. Both attacks are based on structural weaknesses in the key schedule of Tiger. By combining these weaknesses with a generic meet-in-the-middle approach we can construct a preimage for the compression function of Tiger reduced to 16 rounds (2 passes) with a complexity of about 2^{173} compression function evaluations and memory requirements of 2^{38}. The attack can be extended to 17 rounds of Tiger at the cost of about 2^{184} compression function evaluations and memory requirements of 2^{159}. In the following, we will describe both attacks in more detail.

3.1 Preimage Attack on Two Passes of Tiger

Before describing the preimage attack on the compression function of Tiger reduced to 2 passes (16 rounds), we first have a closer look at the key schedule of Tiger. In the following, we present a differential characteristic for the key schedule of Tiger which we can use to construct preimages for the compression function faster than brute force search. Consider the differential

$$(\delta_1, 0, \delta_2, 0, 0, 0, 0, 0) \rightarrow (\delta_1, 0, 0, 0, 0, 0, 0, 0) \ , \tag{1}$$

with $\delta_1 \boxplus \delta_2 = 0$, where δ_1 and δ_2 denote modular differences in the 19 most significant bits of the message words X_0, X_2 and X_8. In order to guarantee that this characteristic holds in the key schedule of Tiger, several conditions have to be fulfilled.

Due to the design of the key schedule of Tiger, the difference δ_1 in X_0 will lead to the same difference δ_1 in $T_0 = X_0 \boxminus (X_7 \oplus \texttt{A5A5A5A5A5A5A5A5})$. Furthermore, by choosing $X_1 = 0$, we get $T_1 = T_0$ and hence $\Delta T_1 = \Delta T_0 = \delta_1$. Since $\Delta T_1 = \delta_1$, $\Delta X_2 = \delta_2$ and $\delta_1 \boxplus \delta_2 = 0$, there will be no difference in $T_2 = X_2 \boxplus T_1$. Note that by restricting the choice of δ_1 and hence δ_2 to differences in the 19 most significant bits we can ensure that there will be no differences in $T_3 = X_3 \boxminus (T_2 \oplus ((\neg T_1) \ll 19))$. It is easy to see, that due to the left shift of T_1 by 19 bits these differences will be canceled. Since there are no difference in T_2 and T_3, there will be no differences in $T_4, \ldots, T_7$. To ensure that there will be only a difference in $X_8 = T_0 \boxplus T_7$, namely δ_1 after the second step of the key schedule of Tiger, we need that $T_7 = 0$. This can be achieved by adjusting X_6 accordingly, such that $T_6 \oplus X_7 = 0$. It is easy to see that if $T_7 = 0$ then $X_8 = T_0$ and hence $\Delta X_8 = \Delta T_0 = \delta_1$. Furthermore, $X_9 = T_1 \boxminus X_8$ and hence $\Delta X_9 = \delta_1 \boxminus \delta_1 = 0$. Since $\Delta X_9 = 0$ and there are no differences in $T_2, \ldots, T_7$ there will be no differences in $X_{10}, \ldots, X_{15}$. By fulfilling all these conditions on the message words and restricting the differences of δ_1 and hence δ_2 to the 19 most significant bits, this characteristic for the key schedule of Tiger will always hold.

We will use this characteristic for the key schedule of Tiger to show a preimage attack on Tiger reduced to 16 rounds (2 passes). We combine the characteristic

for the key schedule of Tiger with a generic meet-in-the-middle approach, to construct a preimage for the compression function of Tiger with 2 passes. The attack has a complexity of about 2^{173} compression function evaluations and memory requirements of 2^{38}. It can be summarized as follows.

1. Suppose we seek a preimage of $h = AA \| BB \| CC$, then we chose $A_{-1} = AA$, $B_{-1} = BB$, and $C_{-1} = CC$. To guarantee that the output after the feed forward is correct, we need that $A_{15} = 0$, $B_{15} = 0$, and $C_{15} = 0$.
2. In order to guarantee that the characteristic for the key schedule of Tiger holds, we choose random values for the message words $X_0, X_2, \ldots, X_7$ and set $X_1 = 0$. Furthermore, we adjust X_6 accordingly, such that $T_7 = 0$.
3. Next we compute A_7, B_7, and C_7 for all 2^{38} choices of $B_{-1}[63-45]$ and $C_{-1}[63-45]$ and save the result in a list L. In other words, we get 2^{38} entries in the list L by modifying the 19 most significant bits of B_{-1} and the 19 most significant bits of C_{-1}.
4. For all 2^{38} choices of the 19 most significant bits of B_{15} and the 19 most significant bits of C_{15} we compute A'_7, B'_7, C'_7 (by going backward) and check if there is an entry in the list L such that the following conditions are fulfilled:

$$\begin{aligned} A_7[i] &= A'_7[i] \quad \text{for } 0 \le i \le 63 \\ B_7[i] &= B'_7[i] \quad \text{for } 0 \le i \le 63 \\ C_7[i] &= C'_7[i] \quad \text{for } 0 \le i \le 44 \end{aligned}$$

   These conditions will hold with probability of 2^{-173}. Note that we can always adjust the 19 most significant bits of X_8 such that the 19 most significant bits of C_7 and C'_7 match.
   Since there are 2^{38} entries in the list L and we test 2^{38} candidates, we expect to find a matching entry with probability of $2^{-173} \cdot 2^{76} = 2^{-97}$. Hence, finishing this step of the attack has a complexity of about $2^{38} \cdot 2^{97} = 2^{135}$ evaluations of the compression function of Tiger and memory requirements of 2^{38}.
5. Once we have found a solution, we have to modify the 19 most significant bits of X_0 and X_2 such that the characteristic in the key schedule of Tiger holds. To cancel the differences in X_0 and X_2, we have to adjust the 19 most significant bits of B_{-1} and C_{-1} accordingly. Thus, after applying the feed-forward we get a partial pseudo preimage for 154 (out of 192) bits of the compression function of Tiger reduced to 16 rounds.

Hence, we will find a partial pseudo preimage (154 out of 192 bits) with a complexity of 2^{135} and memory requirements of 2^{38}. By repeating the attack 2^{38} times we will find a preimage for the compression function with a complexity of about 2^{173} instead of the expected 2^{192} compression function evaluations. Note that the partial pseudo preimage (154 out of 192 bits) is also a fixed-point in 154 bits for the compression function f. We will need this later to turn the attack on the compression function into an attack on the hash function.

3.2 Going Beyond Two Passes

In a similar way as we can construct a preimage for the compression function of Tiger reduced to 16 rounds, we can also construct a preimage for the compression function of Tiger reduced to 17 rounds. The attack has a complexity of about 2^{184} compression function evaluations and has memory requirements of 2^{159}.

For the attack on 17 rounds we use a slightly different characteristic for the key schedule of Tiger. It is shown below.

$$(0, \delta_1, 0, 0, 0, 0, 0, \delta_2) \rightarrow (0, 0, 0, 0, 0, 0, 0, \delta_3) \rightarrow (\delta_4, ?, ?, ?, ?, ?, ?, ?) \qquad (2)$$

where δ_4 denotes modular difference in the 31 most significant bits of the message word X_{16} and δ_1, δ_2, δ_3 denote modular difference in the 8 most significant bits of the message words X_1, X_7, X_{15}. Note that while in the attack on 2 passes we have only differences in the 19 most significant bits, we have now differences in the 8 (respectively 31) most significant bits of the message words.

In order to guarantee that this characteristic holds in the key schedule of Tiger, several conditions have to be fulfilled. In detail, a difference δ_2 in X_7 will lead to a difference in $T_0 = X_0 \boxminus (X_7 \oplus \texttt{A5A5A5A5A5A5A5A5})$ after the first step of the key schedule. By adjusting X_1 accordingly (choosing the difference δ_1 carefully), we can prevent that the difference in T_0 propagates to $T_1 = X_1 \oplus T_0$ and hence, there will be no differences in $T_1, \ldots, T_6$. However, due to the design of the key schedule of Tiger there will be a difference in $T_7 = X_7 \oplus T_6$. In order to prevent the propagation of the differences in T_7 to X_8 we need that $T_6 = \texttt{A5A5A5A5A5A5A5A5}$. Thus, we have that

$$\begin{aligned} X_8 &= T_0 \boxplus T_7 \\ &= X_0 \boxminus (X_7 \oplus \texttt{A5A5A5A5A5A5A5A5}) \boxplus (X_7 \oplus \texttt{A5A5A5A5A5A5A5A5}) \\ &= X_0. \end{aligned}$$

We can guarantee that $T_6 = \texttt{A5A5A5A5A5A5A5A5}$ by adjusting X_6 accordingly. Note that by restricting the differences of δ_2 and hence also δ_1 to the 8 most significant there will be only differences in the 8 most significant bits of $T_7 = X_7 \oplus T_6$ and therefore no differences in $X_9 = T_1 \boxminus (X_8 \oplus ((\neg T_7) \lll 19))$ and $X_{10}, \ldots, X_{14}$, only in $X_{15} = T_7 \boxminus (X_{14} \oplus \texttt{0123456789ABCDEF})$ there will be a difference δ_3 in the 8 most significant bits.

However, in the third pass there will be differences in the 31 most significant bits of X_{16} (denoted by δ_4) due to the design of the key schedule of Tiger. It is easy to see that a difference in the 8 most significant bits in X_{15} will result in differences in the 8 most significant bits of $T_0, \ldots, T_5$. Furthermore, since $T_6 = X_{14} \boxminus (T_5 \oplus \neg T_4 \ggg 23)$ we will get differences in the 31 most significant bits of T_6 and hence also in T_7 as well as in $X_{16} = T_0 \boxplus T_7$.

Again, by combining this characteristic for the key schedule of Tiger with a generic meet-in-the-middle approach, we can construct preimages for the compression function of Tiger for more than 2 passes (17 rounds) with a complexity of about 2^{184} compression function evaluations. The attack can be summarized as follows.

1. Suppose we seek a preimage of $h = AA \| BB \| CC$, then we chose $A_{-1} = AA$, $B_{-1} = BB$, and $C_{-1} = CC$. To guarantee that the output after the feed forward is correct, we need that $A_{16} = 0$, $B_{16} = 0$, and $C_{16} = 0$.
2. Choose random values for the message words $X_0, X_1, \ldots, X_7$ such that $T_6 =$ `A5A5A5A5A5A5A5A5` after the first step of the key schedule of Tiger. Note that this can be easily done by adjusting X_6 accordingly, *i.e.* $X_6 = T_6 \boxplus (T_5 \oplus (\neg T_4 \gg 23))$. This is needed to ensure that differences in T_7 will be canceled in the key schedule – leading to the correct value of X_8 after the second step of the key schedule.
3. Next we compute A_6, B_6, C_6 for all 2^{159} choices of A_{-1}, C_{-1} and $B_{-1}[63-33]$ and save the result in a list L. In other words, we get 2^{159} entries in the list L by modifying A_{-1}, C_{-1} and the 31 most significant bits of B_{-1}.
4. For all 2^{159} choices of A_{16}, C_{16} and the 31 most significant bits of B_{16} we compute A'_6, B'_6, C'_6 (by going backward) and check if there is an entry in the list L such that the following conditions are fulfilled:

$$\begin{aligned} A_6[i] &= A'_6[i] \quad \text{for } 0 \le i \le 63 \\ B_6[i] &= B'_6[i] \quad \text{for } 0 \le i \le 63 \\ C_6[i] &= C'_6[i] \quad \text{for } 0 \le i \le 55 \end{aligned}$$

   These conditions will hold with probability of 2^{-184}. Note that we can always adjust the 8 most significant bits of X_7 such that $C_6 = C'_6$ will match. Since there are 2^{159} entries in the list L and we test 2^{159} candidates, we will find $2^{-184} \cdot 2^{318} = 2^{134}$ solutions. In other words, we get 2^{134} solutions with a complexity of about 2^{159} evaluations of the compression function of Tiger and memory requirements of 2^{159}.
5. For each solution, we have to modify the 8 most significant bits of X_1 such that $T_1 = X_1 \oplus T_0$ is correct in the first step of the key schedule for the new value of X_7. Note that by ensuring that T_1 is correct, we will get the same values for $X_8, \ldots, X_{14}$ after applying the key schedule of Tiger, since $T_6 =$ `A5A5A5A5A5A5A5A5` due to step 2 of the attack. In order to cancel the differences in the 8 most significant bits of X_1, we have to adjust the 8 most significant bits of A_{-1} accordingly. Furthermore, the 8 most significant bits of X_{15} and the 31 most significant bits of X_{16} will change as well. This results in new values for A_{16}, C_{16} and the 31 most significant bits of B_{16}.
   Since, we modify A_{-1}, C_{-1} and the 31 most significant bits of B_{-1} in the attack we get after the feed-forward 2^{134} partial pseudo preimage (partial meaning 33 out of 192 bits) for the compression function of Tiger reduced to 17 rounds.

Hence, we will find 2^{134} partial pseudo preimage (33 out of 192 bits) with a complexity of 2^{159}. By repeating the attack 2^{25} times we will find a preimage for the compression function of Tiger reduced to 17 rounds with a complexity of about $2^{159} \cdot 2^{25} = 2^{184}$ instead of the expected 2^{192} compression function evaluations.

4 Extending the Attacks to the Hash Function

If we want to extend the preimage attack on the compression function of Tiger to the hash function, we encounter two obstacles. In contrast to an attack on the compression function, where the chaining value (or initial value) can be chosen freely, the initial value IV is fixed for the hash function. In other words, for a preimage attack on the hash function we have to find a message m such that $H(IV, m) = h$. Furthermore, we have to ensure that the padding of the message leading to the preimage of h is correct.

First, we choose the message length such that only a single bit of padding will be set in X_6 of the last message block. The last bit of X_6 has to be 1 as specified by the padding rule. Since we use in both attacks characteristics for the key schedule of Tiger where no difference appears in X_6, we can easily guarantee that the last bit of X_6 is 1. However, X_7 of the last message block will contain the message length as a 64-bit integer. While we can choose X_7 free in the attack on 2 passes (16 rounds), this is not the case for the attack on 17 rounds. The 8 most significant bits of X_7 are determined during the attack (*cf.* Section 3.2). However, the remaining bits of X_7 can be chosen freely. Therefore, we can always guarantee that we will have a message length such that the padding of the last block is correct. For the sake of simplicity let us assume for the following discussion that the message (after padding) consists of $\ell + 1$ message blocks.

We show how to construct a preimage for Tiger reduced to 16 rounds consisting of $\ell + 1$ message blocks, *i.e.* $m = M_1 \| M_2 \| \cdots \| M_{\ell+1}$. Note that the attack for Tiger reduced to 17 rounds works similar. It can be summarized as follows.

1. First, we invert the last iteration of the compression function $f(H_\ell, M_{\ell+1}) = h$ to get H_ℓ and $M_{\ell+1}$. Note that this determines the length of our preimage. This step of the attack has a complexity of about 2^{173} compression function evaluations.
2. Once we have fixed the last message block $M_{\ell+1}$ and hence the length of the message m, we have to find a message $m^* = M_1 \| M_2 \| \cdots \| M_\ell$ consisting of ℓ message blocks such that $H(IV, m^* \| M_{\ell+1}) = h$. This can be done once more by using a meet-in-the-middle approach.
   (a) Use the preimage attack on the compression function to generate 2^{10} pairs $(H^j_{\ell-1}, M^j_\ell)$ leading to the chaining value H_ℓ and save them in a list L. This has a complexity of about $2^{10} \cdot 2^{173} = 2^{183}$ compression function evaluations.
   (b) Compute $H_{\ell-1}$ by choosing random values for the message blocks M_i for $1 \leq i < \ell$ and check for a match in L. After testing about 2^{182} candidates, we expect to find a match in the list L. Once, we have found a matching entry, we have found a preimage for the hash function Tiger reduced to 16 rounds consisting of $\ell + 1$ message blocks.

Hence, we can construct a preimage for the Tiger hash function reduced to 16 rounds with a complexity of about 2^{183} compression function evaluations. In a similar way we can find a preimage for Tiger reduced to 17 rounds with a complexity of about 2^{188}.

However, due to the special structure of the partial-pseudo-preimages for the compression function of Tiger reduced to 16 and 17 rounds, this complexity can be reduced by using a tree-based approach. This was first used by Mendel and Rijmen in the cryptanalysis of HAS-V [11]. Later variants and extensions of this method were presented in [3,7,8]. With this method, we can construct a preimage for the Tiger hash function reduced to 16 and 17 rounds with a complexity of about 2^{174} and 2^{185} compression function evaluations, respectively. In the following, we will describe this in more detail for Tiger reduced to 16 rounds. Note that the attack for Tiger reduced to 17 rounds works similar.

1. Assume we want to construct a preimage for Tiger reduced to 16 rounds consisting of $\ell + 1$ message blocks.
2. First, compute H_ℓ and $M_{\ell+1}$ by inverting the last iteration of the compression function. Note that this determines the length of our preimage m. This step of the attack has a complexity of about 2^{173} compression function evaluations.
3. Next, we construct a list L containing 2^{39} partial-pseudo-preimages for the compression function of Tiger. Note that all partial-pseudo-preimages will have the following form: $H_i = f(H_{i-1}, M_i)$, where $H_i \wedge \texttt{mask} = H_{i-1} \wedge \texttt{mask}$ and $hw(\texttt{mask}) = 154$, where $hw(x)$ denotes the bit Hamming weight of x. In other words, each preimage for the compression function is also a fixed-point for $192 - 38 = 154$ bits. Note that this is important for the attack to work. Constructing the list L has a complexity of about $2^{39} \cdot 2^{135} = 2^{174}$ compression function evaluations.
4. Next, by using the entries in the list L we build a backward tree starting from H_ℓ. For each node in the tree we expect to get two new nodes on the next level. It is easy to see that since we have 2^{39} entries in the list L, where 154 bits are equal for each entry, we will always have two entries, where H_i is equal. Therefore, we will have about 2^{20} nodes at level 20. In other words, we have about 2^{20} candidates for $H_{\ell-20}$.
5. To find a message consisting of $\ell - 20$ message blocks leading to one of the 2^{20} candidates for $H_{\ell-20}$ we use a meet-in-the-middle approach. First, we choose an arbitrary message (of $\ell-21$ message blocks) leading to some $H_{\ell-21}$. Second, we have to find a message block $M_{\ell-20}$ such that $f(H_{\ell-21}, M_{\ell-20}) = H_{\ell-20}$ for one of the 2^{20} candidates for $H_{\ell-20}$ in the list L. After testing about 2^{172} message blocks $M_{\ell-20}$ we expect to find a matching entry in the tree and hence, a preimage for Tiger reduced to 16 rounds. Thus, this step of the attack has a complexity of about 2^{172} compression function evaluations of Tiger.

Hence, with this method we can find a preimage for the Tiger hash function reduced to 16 rounds with a complexity of about 2^{174} compression function evaluations and memory requirement of 2^{39}. Note that the same method can be used to construct preimages for the Tiger hash function reduced to 17 rounds with a complexity of about 2^{185} compression function evaluations and memory requirements of 2^{160}.

5 Conclusion

In this article, we presented a preimage attack on the compression function of Tiger reduced to 16 and 17 rounds with a complexity of about 2^{173} and 2^{184} compression function evaluations and memory requirements of 2^{38} and 2^{159}, respectively. In the attack, we combined weaknesses in the key schedule of Tiger with a generic meet-in-the-middle approach. Furthermore, we used a tree-based approach to extend the attacks for the compression function to the hash function with a complexity of about 2^{174} and 2^{185} compression function evaluations and memory requirements of 2^{39} and 2^{160}, respectively. Even though the complexities of the presented attacks are only slightly faster than brute force search, they show that the security margins of the Tiger hash function with respect to preimage attacks are not as good as expected.

Acknowledgements

The author wishes to thank Mario Lamberger, Vincent Rijmen, and the anonymous referees for useful comments and discussions. The work in this paper has been supported in part by the European Commission under contract ICT-2007-216646 (ECRYPT II).

References

1. Ross J. Anderson and Eli Biham. TIGER: A Fast New Hash Function. In Dieter Gollmann, editor, *FSE*, volume 1039 of *LNCS*, pages 89–97. Springer, 1996.
2. Jean-Philippe Aumasson, Willi Meier, and Florian Mendel. Preimage Attacks on 3-Pass HAVAL and Step-Reduced MD5. In Roberto Avanzi, Liam Keliher, and Francesco Sica, editors, *SAC*, LNCS. Springer, 2008. To appear.
3. Christophe De Cannière and Christian Rechberger. Preimages for Reduced SHA-0 and SHA-1. In David Wagner, editor, *CRYPTO*, volume 5157 of *LNCS*, pages 179–202. Springer, 2008.
4. Sebastiaan Indesteege and Bart Preneel. Preimages for Reduced-Round Tiger. In Stefan Lucks, Ahmad-Reza Sadeghi, and Christopher Wolf, editors, *WEWoRC*, volume 4945 of *LNCS*, pages 90–99. Springer, 2007.
5. Takanori Isobe and Kyoji Shibutani. Preimage Attacks on Reduced Tiger and SHA-2. In Orr Dunkelman, editor, *FSE*, LNCS. Springer, 2009. To appear.
6. John Kelsey and Stefan Lucks. Collisions and Near-Collisions for Reduced-Round Tiger. In Matthew J. B. Robshaw, editor, *FSE*, volume 4047 of *LNCS*, pages 111–125. Springer, 2006.
7. Lars R. Knudsen, Florian Mendel, Christian Rechberger, and Søren S. Thomsen. Cryptanalysis of MDC-2. In Antoine Joux, editor, *EUROCRYPT*, volume 5479 of *LNCS*, pages 106–120. Springer, 2009.
8. Gaëtan Leurent. MD4 is Not One-Way. In Kaisa Nyberg, editor, *FSE*, volume 5086 of *LNCS*, pages 412–428. Springer, 2008.
9. Florian Mendel, Bart Preneel, Vincent Rijmen, Hirotaka Yoshida, and Dai Watanabe. Update on Tiger. In Rana Barua and Tanja Lange, editors, *INDOCRYPT*, volume 4329 of *LNCS*, pages 63–79. Springer, 2006.

10. Florian Mendel and Vincent Rijmen. Cryptanalysis of the Tiger Hash Function. In Kaoru Kurosawa, editor, *ASIACRYPT*, volume 4833 of *LNCS*, pages 536–550. Springer, 2007.
11. Florian Mendel and Vincent Rijmen. Weaknesses in the HAS-V Compression Function. In Kil-Hyun Nam and Gwangsoo Rhee, editors, *ICISC*, volume 4817 of *LNCS*, pages 335–345. Springer, 2007.
12. Yu Sasaki and Kazumaro Aoki. Preimage attacks on one-block MD4, 63-step MD5 and more. In Roberto Avanzi, Liam Keliher, and Francesco Sica, editors, *SAC*, LNCS. Springer, 2008. To appear.