Cryptanalysis of the Tiger Hash Function[⋆]

Florian Mendel and Vincent Rijmen

Institute for Applied Information Processing and Communications (IAIK),
Graz University of Technology, Inffeldgasse 16a, A-8010 Graz, Austria

{Florian.Mendel,Vincent.Rijmen}@iaik.tugraz.at

Abstract. Tiger is a cryptographic hash function with a 192-bit hash value. It was proposed by Anderson and Biham in 1996. Recently, weaknesses have been shown in round-reduced variants of the Tiger hash function. First, at FSE 2006, Kelsey and Lucks presented a collision attack on Tiger reduced to 16 and 17 (out of 24) rounds with a complexity of about 2^{44} and a pseudo-near-collision for Tiger reduced to 20 rounds. Later, Mendel *et al.* extended this attack to a collision attack on Tiger reduced to 19 rounds with a complexity of about 2^{62}. Furthermore, they show a pseudo-near-collision for Tiger reduced to 22 rounds with a complexity of about 2^{44}. No attack is known for the full Tiger hash function. In this article, we show a pseudo-near-collision for the full Tiger hash function with a complexity of about 2^{47} hash computations and a pseudo-collision (free-start-collision) for Tiger reduced to 23 rounds with the same complexity.

Keywords: cryptanalysis, hash functions, differential attack, collision, near-collision, pseudo-collision, pseudo-near-collision

1 Introduction

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 results in the cryptanalysis of Tiger show weaknesses in round-reduced variants of the hash function. At FSE 2006, Kelsey and Lucks presented a collision attack on 16 and 17 (out of 24) rounds of Tiger. The attack has a complexity of about 2^{44} evaluations of the compression function. Furthermore, they present a pseudo-near-collision for a variant of Tiger reduced to 20 rounds with a complexity of about 2^{48}. These results were later improved by Mendel *et al.* in [3]. They show that a collision can be found for Tiger reduced to 19 rounds with a complexity of about 2^{62} evaluations of the compression function. Furthermore, they present a pseudo-near-collision for Tiger reduced to 22 rounds with a complexity of about 2^{44}. However, so far no attack is known for the full Tiger hash function.

[⋆] The work in this paper has been supported by the Austrian Science Fund (FWF), project P18138.

In this article, we present a 1-bit circular pseudo-near-collision for the full Tiger hash function with a complexity of about 2^{47} hash computations and a pseudo-collision (free-start-collision) for a variant of Tiger reduced to 23 rounds with the same complexity. The attack is based on previous attacks presented in [2] and [3]. Note that in the attacks of Kelsey and Lucks and Mendel *et al.* on round-reduced variants of Tiger, the S-boxes of the hash function are addressed wrongly (big endian instead of little endian). However, this error can be fixed easily, because there is really a large amount of freedom in these attacks on round-reduced variants of Tiger.

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 describe the basic attack strategy on Tiger based on the work of Kelsey and Lucks on round-reduced Tiger. We follow this attack strategy in Section 4 to construct a 1-bit circular pseudo-near-collision for Tiger with a complexity of about 2^{47}. In Section 5, we show a pseudo-collision for Tiger reduced to 23 rounds with the same complexity. Finally, we present conclusions in Section 6.

2 Description of the Hash Function Tiger

Tiger is a cryptographic hash function that was designed by Anderson and Biham in 1996 [1]. It is an iterative hash function that processes 512-bit input message blocks 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 transformation. 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
X_i	message word i (64 bits)
$X_i[\mathbf{even}]$	the even bytes of message word X_i (32 bits)
$X_i[\mathbf{odd}]$	the odd bytes of message word X_i (32 bits)

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 is used to update the three state variables A, B and C as follows:

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

The results are then shifted such that A, B, C become B, C, A. Fig. 1 shows one round of the state update transformation of Tiger.

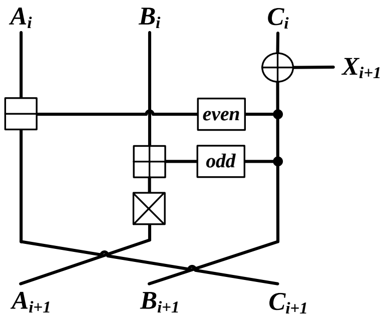

Fig. 1. The round function of Tiger.

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 not c_0). 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 $\mathtt{mult} \in \{5, 7, 9\}$ at the end of each round. The value of the constant is different in each pass of the Tiger hash function.

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 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:

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

The key schedule modifies the inputs $(Y_0, \ldots, Y_7)$ in two steps:

first step	**second step**
$Y_0 = Y_0 \boxminus (Y_7 \oplus \texttt{A5A5A5A5A5A5A5A5})$	$Y_0 = Y_0 \boxplus Y_7$
$Y_1 = Y_1 \oplus Y_0$	$Y_1 = Y_1 \boxminus (Y_0 \oplus ((\neg Y_7) \ll 19))$
$Y_2 = Y_2 \boxplus Y_1$	$Y_2 = Y_2 \oplus Y_1$
$Y_3 = Y_3 \boxminus (Y_2 \oplus ((\neg Y_1) \ll 19))$	$Y_3 = Y_3 \boxplus Y_2$
$Y_4 = Y_4 \oplus Y_3$	$Y_4 = Y_4 \boxminus (Y_3 \oplus ((\neg Y_2) \gg 23))$
$Y_5 = Y_5 \boxplus Y_4$	$Y_5 = Y_5 \oplus Y_4$
$Y_6 = Y_6 \boxminus (Y_5 \oplus ((\neg Y_4) \gg 23))$	$Y_6 = Y_6 \boxplus Y_5$
$Y_7 = Y_7 \oplus Y_6$	$Y_7 = Y_7 \boxminus (Y_6 \oplus \texttt{0123456789ABCDEF})$

The final values $(Y_0, \ldots, Y_7)$ are the output of the key schedule and the message words for the next pass.

3 Basic Attack Strategy

In this section, we briefly describe the attack strategy of Kelsey and Lucks to attack round-reduced variants of the Tiger hash function. A detailed description of the attack is given in [2]. For a good understanding of our attack it is recommended to study it carefully. The attack can be summarized as follows.

1. Find a characteristic for the key schedule of Tiger which holds with high probability. In the ideal case this probability is 1.
2. Use a kind of message modification technique developed for Tiger to construct certain differences in the state variables, which can then be canceled by the differences of the message words in the following rounds.

These two steps of the attack are described in detail in the following sections.

3.1 Finding a good Characteristic for the Key Schedule of Tiger

To find a good characteristic for the key schedule of Tiger, we use a linearized model of the key schedule. Therefore, we replace all modular additions and subtractions by an XOR operation resulting in a linear code over $GF(2)$. Finding a characteristic in the linear code is not difficult, since it depends only on the differences in the message words. The probability that the characteristic holds in the original key schedule of Tiger is related to the Hamming weight of the characteristic. In general, a characteristic with low Hamming weight has a higher probability than one with a high Hamming weight.

For finding a characteristic with high probability (low Hamming weight), we use probabilistic algorithms from coding theory. It has been shown in the past (cryptanalysis of SHA-1 [4]) that these algorithms work quite well. Furthermore, we can impose additional restrictions on the characteristic by forcing certain bits/words to zero. Note that this is needed to find suitable characteristics for the key schedule of Tiger. For an attack on the Tiger hash function we need many zeros in the first and last rounds of the hash function.

3.2 Message Modification by Meet-in-the-Middle

In order to construct a collision in Tiger reduced to 16 rounds, Kelsey and Lucks use a message modification technique developed for Tiger. The idea of message modification in general is to use the degree of freedom one has in the choice of the message words to fulfill conditions on the state variables. In the attack on Tiger this method is used to construct a certain differential pattern in the state variables, which can then be canceled by the differences of the message words in the following rounds. This leads to a collision in a round reduced variant of Tiger. In the following we will briefly describe this message modification technique according to Fig. 2.

Assume, we are given A_{i-1}, B_{i-1}, C_{i-1} and A^*_{i-1}, B^*_{i-1}, C^*_{i-1} as well as $\Delta^{\oplus}(X_i)$ and $\Delta^{\oplus}(X_{i+1})$. Then the modular difference $\Delta^{\boxplus}(C_{i+1})$ can be forced to be any difference δ with a probability of 2^{-1} by using a birthday attack. We try out all 2^{32} possibilities for $X_{i-1}[\mathbf{odd}]$ to generate 2^{32} candidates for $\Delta^{\boxplus}(\mathbf{odd}(B_i))$. Similarly, we try out all $X_i[\mathbf{even}]$ to generate 2^{32} candidates for $\Delta^{\boxplus}(\mathbf{even}(B_{i+1}))$. Subsequently, we use a meet-in-the-middle approach to solve the following equation:

$$\Delta^{\boxplus}(C_{i+1}) = \texttt{mult} \boxtimes [\Delta^{\boxplus}(B_{i-1}) \boxplus \Delta^{\boxplus}(\mathbf{odd}(B_i))] \boxminus \Delta^{\boxplus}(\mathbf{even}(B_{i+1})) = \delta \; . \quad (1)$$

The method can be summarized as follows:

1. Store the 2^{32} candidates for $\Delta^{\boxplus}(\mathbf{odd}(B_i))$ in a table.
2. For all 2^{32} candidates for $\Delta^{\boxplus}(\mathbf{even}(B_{i+1}))$, test if some $\Delta^{\boxplus}(\mathbf{odd}(B_i))$ exists in the table with

$$\Delta^{\boxplus}(\mathbf{odd}(B_i)) = (\Delta^{\boxplus}(\mathbf{even}(B_{i+1})) \boxplus \delta) \boxtimes \texttt{mult}^{-1} \boxminus \Delta^{\boxplus}(B_{i-1}) \; .$$

This technique needs about 2^{36} bytes of storage and takes 2^{33} evaluations of each of the functions **odd** and **even**. This is equivalent to about 2^{29} evaluations of the compression function of Tiger.

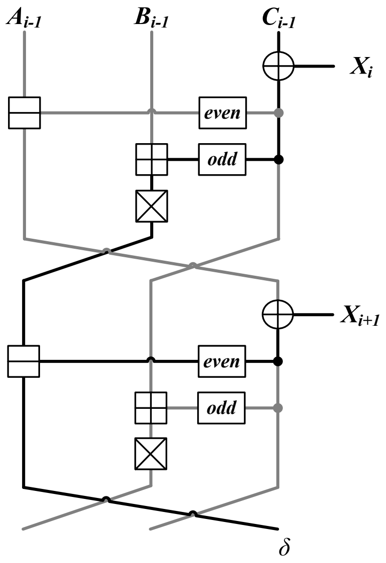

Fig. 2. Message Modification by Meet-in-the-Middle.

4 A Pseudo-Near-Collision for Tiger

In this section, we will present a 1-bit circular pseudo-near-collision for the Tiger hash function. Note that the difference in the final hash value is the same as in the initial value. In other words, we have a pseudo-collision in the compression function of Tiger after 24 rounds, but due to the feed forward the collision after 24 rounds is destroyed, resulting in a 1-bit pseudo-near-collision for the Tiger hash function. The attack has a complexity of about 2^{47} evaluations of the compression function. In the attack, we extend techniques invented by Kelsey and Lucks in the attack on round-reduced variants of Tiger.

We use the characteristic given below for the key schedule of Tiger to construct the pseudo-near-collision in the hash function. This characteristic holds with a probability of 2^{-1} which facilitates the attack.

$$(0, I, 0, 0, 0, I, I', 0) \rightarrow (0, I, 0, I, 0, 0, 0, 0) \rightarrow (0, I, 0, 0, 0, 0, 0, 0) \qquad (2)$$

I denotes a difference in the MSB of the message word and $I' := I \gg 23$. Note that the XOR-difference (denoted by $\Delta^{\oplus}$) equals I if and only if the modular difference (denoted by $\Delta^{\boxplus}$) equals I.

In order to have a pseudo-collision in the compression function of Tiger after 24 rounds, it is required that there is a pseudo-collision after round 17. Hence, the following differences are needed in the state variables for round 14 of Tiger (see Table 2).

$$\Delta^{\oplus}(A_{14}) = 0, \quad \Delta^{\oplus}(B_{14}) = I, \quad \Delta^{\oplus}(C_{14}) = 0 \qquad (3)$$

Constructing these differences in the state variables for round 14 is the most difficult part of the attack. We use the message modification technique described in Section 3.2 for this. In the following sections, we will describe all steps of the attack in detail.

Table 2. Characteristic for a 1-bit pseudo-near-collision in the Tiger hash function.

	i	ΔA_i	ΔB_i	ΔC_i	ΔX_i
initial value	-1	I	0	0	
Pass 1	0	0	0	I	0
	1	0	0	0	I
	2	0	0	0	0
	3	0	0	0	0
	4	0	0	0	0
	5	$*$	I	0	I
	6	$*$	I'	$*$	I'
	7	$*$	$*$	$*$	0
Pass 2	8	$*$	$*$	$*$	0
	9	$*$	$*$	$*$	I
	10	$*$	$*$	$*$	0
	11	$*$	$*$	$K^{\oplus}$	I
	12	$*$	K^{+}	$L^{\oplus}$	0
	13	0	L^{+}	I	0
	14	0	I	0	0
	15	I	0	0	0
Pass 3	16	0	0	I	0
	17	0	0	0	I
	18	0	0	0	0
	19	0	0	0	0
	20	0	0	0	0
	21	0	0	0	0
	22	0	0	0	0
	23	0	0	0	0
feed forward	24	I	0	0	

4.1 Precomputation

The precomputation step basically consists of 2 parts. First, we have to find a set $\mathcal{L}$ of *possible* modular differences L^{+} which are consistent to a low weight XOR-difference $L^{\oplus}$. A modular difference L^{+} is consistent to $L^{\oplus}$ if there exist p and p^{*} such that $p^{*} \oplus p = L^{\oplus}$ and $p^{*} \boxminus p = L^{+}$. Let $\mathcal{L}'$ be the set of modular differences L^{+} which are consistent to the XOR-difference $L^{\oplus}$ then we define the set $\mathcal{L}$ of *possible* modular differences as follows:

$$\mathcal{L} = \{L^{+} \in \mathcal{L}' : \; L^{+} = \mathbf{odd}(B_{14} \oplus I) \boxminus \mathbf{odd}(B_{14})\}$$

Note that the size of the set $\mathcal{L}'$ is related to the Hamming weight of $L^{\oplus}$, namely $|\mathcal{L}'| = 2^{\mathrm{HW}(L^{\oplus})}$. In order to optimize the complexity of the meet-in-the-middle step used in the attack, we need an $L^{\oplus}$ with low Hamming weight. In [2], the authors assume that an $L^{\oplus}$ with Hamming weight of 8 exists. However, the best Hamming weight we found for $L^{\oplus}$ is 10.

$$L^{\oplus} = \texttt{02201080A4020104} \tag{4}$$

In total we found $502 = |\mathcal{L}|$ *possible* modular differences (out of $1024 = |\mathcal{L}'|$) which are consistent to the XOR-difference $L^{\oplus}$ given above. This facilitates the attack in the following steps.

Second, we need a set $\mathcal{K}$ of *possible* modular differences K^+ which are consistent to a low weight XOR-difference $K^{\oplus}$.

$$\mathcal{K} = \{K^+ \in \mathcal{K}' : \; K^+ = \mathbf{odd}(B_{13} \oplus L^{\oplus}) \boxminus \mathbf{odd}(B_{13})\}$$

where $\mathcal{K}'$ is the set of modular differences K^+ which are consistent to the XOR-difference $K^{\oplus}$. Of course, the choice of $L^{\oplus}$ and the number of possible modular differences $L^+ \in \mathcal{L}$ restricts our choices for $B_{13}[\mathbf{odd}]$. Nevertheless, we found $2 = |\mathcal{K}|$ possible modular differences K^+ (out of $256 = |\mathcal{K}'|$) which are consistent to the XOR-difference $K^{\oplus}$ given below.

$$K^{\oplus} = \texttt{0880020019000900} \tag{5}$$

Note that the precomputation step of the attack has to be done only once. It has a complexity of about $2 \cdot 2^{32}$ round computations of Tiger. This is approximately about $2^{28.5}$ evaluations of the compression function of Tiger.

4.2 Compute B_9, C_9, and C_{10}

In this step of the attack, we have to compute B_9, C_9 and C_{10}. Therefore, we first choose random values for B_4 and B_5 and compute $A_5 = (B_4 \boxplus \mathbf{odd}(B_5)) \boxtimes \texttt{mult}$. Since there is a difference in the MSB of X_5 and no differences in B_4 and C_4, we also get $\Delta^{\boxplus}(B_5) = I$ and $\Delta^{\boxplus}(A_5) = A_5^* \boxminus A_5$. Note that there is no difference in C_5, since there are no differences in A_4 and $B_5[\mathbf{even}]$.

Second, we choose a random value for B_6. Since there is a difference in $\Delta^{\oplus}(X_6) = I'$ and no difference in C_5, we also know the modular difference of $\Delta^{\boxplus}(B_6) = (B_6 \oplus I') \boxminus B_6$. Once we know B_6 and $B_6^* = B_6 \boxplus \Delta^{\boxplus}(B_6)$, we can calculate B_9, C_9, C_{10} (and B_9^*, C_9^*, C_{10}^*) by choosing random values for $X_7, \ldots, X_9$ and $X_{10}[\mathbf{even}]$. This step of the attack has a complexity of about 12 round computations of Tiger and fixes the message words $X_7, \ldots, X_9$ and $X_{10}[\mathbf{even}]$.

4.3 Constructing the XOR-difference $\Delta^{\oplus}(C_{11}) = K^{\oplus}$

To construct the XOR-difference $K^{\oplus}$ in round 11, we use the message modification technique described in Section 3.2. For all modular differences $K^+ \in \mathcal{K}'$, we do a message modification step and check if $\Delta^{\oplus}(C_{11}) = K^{\oplus}$. Since the Hamming weight of $K^{\oplus}$ is 8, this holds with a probability of 2^{-8}. Furthermore, the message modification step has a probability of 2^{-1}. Hence, this step of the attack succeeds with a probability of $2^{-8} \cdot 2^{-1} \cdot |\mathcal{K}'| = 2^{-1}$ and determines the message words $X_{10}[\mathbf{odd}]$ and $X_{11}[\mathbf{even}]$.

Finishing this step of the attack has a complexity of about $(12 + 2^{32} + 2^{8} \cdot 2^{32}) \cdot 2 \approx 2^{41}$ round computations of Tiger. This is approximately about $2^{36.5}$ evaluations of the compression function of Tiger.

4.4 Constructing the XOR-difference $\boldsymbol{\Delta^{\oplus}(C_{12}) = L^{\oplus}}$

Once we have fixed $X_{10}[\mathbf{odd}]$ and $X_{11}[\mathbf{even}]$, we can calculate the state variables B_{10}, C_{10}, C_{11} (and B^{*}_{10}, C^{*}_{10}, C^{*}_{11}). To construct $L^{\oplus}$ in round 12, we use the same method as described before. For all modular differences $L^{+} \in \mathcal{L}'$, we do a message modification step and check if $\Delta^{\oplus}(C_{12}) = L^{\oplus}$. Since the Hamming weight of $L^{\oplus}$ is 10, this equation holds with a probability of 2^{-10}. Hence, this step of the attack has a probability of $2^{-10} \cdot 2^{-1} \cdot |\mathcal{L}'| = 2^{-1}$ and determines the message words $X_{11}[\mathbf{odd}]$ and $X_{12}[\mathbf{even}]$. Finishing this step of the attack has a complexity of about $(2^{41} + (2^{32} + 2^{32} \cdot 2^{10})) \cdot 2 \approx 2^{43.6}$ round computations of Tiger. This is approximately about 2^{39} evaluations of the compression function of Tiger.

4.5 Constructing the XOR-difference $\boldsymbol{\Delta^{\oplus}(C_{13}) = I}$

Once we have fixed $X_{11}[\mathbf{odd}]$ and $X_{12}[\mathbf{even}]$, we can compute B_{11}, C_{11} and C_{12} as well as the according modular differences. In order to construct the needed difference $\Delta^{\oplus}(A_{13}) = I$ in round 13, we apply again a message modification step. Since the XOR-difference and the modular difference is the same for differences in the MSB, we do not need to compute the list of modular differences that are consistent to the XOR-difference I for the message modification step. This step of the attack succeeds with a probability of 2^{-1} and determines the message words $X_{12}[\mathbf{odd}]$ and $X_{13}[\mathbf{even}]$.

Once we have fixed the message words, we can compute B_{12}, C_{12} and C_{13} as well as the according modular differences. In order to guarantee that $\Delta^{\boxplus}(B_{12})$ can be canceled by $\Delta^{\boxplus}(\mathbf{odd}(B_{13}))$, we need that $\Delta^{\boxplus}(B_{12}) \in \mathcal{K}$. Since the number of modular differences $\Delta^{\boxplus}(B_{12}) = K^{+}$ consistent to $K^{\oplus}$ is $|\mathcal{K}'| = 2^{8}$ and $|\mathcal{K}| = 2$, the probability that $\Delta^{\boxplus}(B_{12}) \in \mathcal{K}$ is 2^{-7}. Hence, we have to repeat the attack about $2 \cdot 2^{7}$ times to finish this step of the attack. This determines the message words $X_{12}[\mathbf{odd}]$, $X_{13}[\mathbf{even}]$ and $X_{13}[\mathbf{odd}]$ and has a complexity of about $(2^{43.6} + (2^{32} + 2^{32})) \cdot 2^{8} \approx 2^{51.6}$ round computations of Tiger. This is about 2^{47} evaluations of the compression function of Tiger.

Once we have fixed $X_{13}[\mathbf{odd}]$ and X_{13}, we can compute A_{13}, B_{13} and C_{13} as well as the according modular differences. In order to guarantee that $\Delta^{\boxplus}(B_{13})$ can be canceled in round 14 by $\Delta^{\boxplus}(\mathbf{odd}(B_{14}))$, we need that $\Delta^{\boxplus}(B_{13}) \in \mathcal{L}$. Due to the choice of $L^{\oplus}$ and $K^{\oplus}$ in the precomputation step this holds with probability 1.

Hence, we can construct a pseudo-collision in the compression function of Tiger after 17 rounds, respectively after 24 rounds with a complexity close to 2^{47} evaluations of the compression function of Tiger.

4.6 Computing the message words $X_0, \ldots, X_7$

The attack fixes the message words $X_7, \ldots, X_{13}$ and $X_{14}[\textbf{odd}]$. To compute the message words $X_0, \ldots, X_7$ we use the inverse key schedule of Tiger. Therefore, we choose a random value for $X_{14}[\textbf{even}]$ and compute X_{15} as follows:

$$X_{15} = (X_7 \oplus (X_{14} \boxminus X_{13})) \boxminus (X_{14} \oplus \texttt{0123456789ABCDEF})$$

This guarantees that X_7 is correct after computing the key schedule backward.

Since the characteristic we use for the key schedule of Tiger has a probability 2^{-1} to hold, we expect that we have to repeat this step of the attack (for a different value of $X_{14}[\textbf{even}]$) about two times such that the characteristic holds in the key schedule of Tiger. This adds negligible cost to the attack complexity.

4.7 Computing the initial value IV

Once we have computed the message words $X_0, \ldots, X_7$, we can run the rounds $6, 5, \ldots, 0$ backwards to get the initial value IV. Since there is a difference I induced in round 1 by X_1, we have to inject the same difference in the initial value to cancel it out, namely

$$\Delta^{\oplus}(A_{-1}) = I \ .$$

Since the difference is in the MSB, this happens with probability 1. Of course, the feed forward destroys the pseudo-collision. After the feed forward we get the same output differences as in the initial values.

$$\Delta^{\oplus}(A_{24}) = \Delta^{\oplus}(A_{-1} \oplus A_{23}) = I$$

Hence, we get a 1-bit circular pseudo-near-collision for the Tiger hash function with a complexity of about 2^{47} evaluations of the compression function of Tiger. Note that for an ideal hash function with a hash value of 192-bit one would expect a complexity of about 2^{90} to construct a pseudo-near-collision with a 1-bit difference.

5 A pseudo-collision for 23 rounds of Tiger

In a similar way as we construct the pseudo-near-collision for the full Tiger hash function, we can also construct a pseudo-collision (free-start-collision) for Tiger reduced to 23 rounds by using another characteristic for the key schedule. For the attack we use the key schedule differences given below. It holds with probability 1.

$$(0,0,0,I,0,0,0,I) \rightarrow (0,I,0,0,0,0,0,I) \rightarrow (0,0,0,0,0,0,0,I) \qquad (6)$$

This characteristic for the key schedule of Tiger can be used in a similar way (as in the pseudo-near-collision for the full Tiger hash function) to construct a pseudo-collision in Tiger reduced to 23 rounds. The attack has a complexity of about 2^{47} evaluations of the compression of Tiger. It can be summarized as follows:

0. Precomputation: First, find a set of possible modular differences L^+ with a low Hamming weight XOR-difference $L^\oplus$ which can be canceled by a suitable choice for B_{12}. Second, we have to find a set of possible modular differences K^+ with a low Hamming weight XOR-difference $K^\oplus$ which can be canceled out by a suitable choice for B_{11}. Note that we use in the attack the same value for $L^\oplus$ and $K^\oplus$ as in the pseudo-near-collision attack on the full Tiger hash function. This step of the attack has a complexity of about $2^{28.5}$ evaluations of the compression function of Tiger.
1. Choose random values for A_2, B_2, C_2 and $X_3, \ldots, X_7$ and $X_8[\textbf{even}]$ to compute B_7, C_7 and C_8. This step of the attack has a complexity of about 12 round computations of Tiger.
2. Apply a message modification step to construct the XOR-difference $K^\oplus$ in round 9. This has a complexity of about $2^{36.5}$ and determines the message words $X_8[\textbf{odd}]$ and $X_9[\textbf{even}]$.
3. Apply another message modification step to construct the XOR-difference $L^\oplus$ in round 10. Finishing this step of the attack has a complexity of about 2^{39} and determines the message words $X_9[\textbf{odd}]$ and $X_{10}[\textbf{even}]$.
4. To construct the XOR-difference I in round 11, we apply again a message modification step. This step has a complexity of about 2^{40} and determines the message words $X_{10}[\textbf{odd}]$ and $X_{11}[\textbf{even}]$.
5. Once we have fixed the message words, we can compute B_{10}, C_{10} and C_{11} as well as the according modular differences. Since the difference in B_{10} can be cancel out with a probability close to 2^{-7} (*cf.* Section 4.5), we have to repeat the attack about 2^7 times. Hence, finishing this step of the attack has a complexity of about 2^{47} hash computations.
6. Determine $X_{11}[\textbf{odd}]$ and $X_{12}[\textbf{odd}]$ according to the result of the precomputation step. This adds no additional cost to the attack complexity.
7. To compute the message words $X_0, \ldots, X_7$, we have to choose suitable values for $X_{12}[\textbf{even}]$ and $X_{13}, \ldots, X_{15}$ such that X_5, X_6 and X_7 are correct after computing the key schedule backward. Note that X_3 and X_4 can be chosen freely, because we can modify C_2 and C_3 such that $C_2 \oplus X_3$ and $C_3 \oplus X_4$ stay constant. In detail, we choose arbitrary values for X_{13} ,X_{14}, X_{15} and calculate $X_{13}, \ldots, X_{15}$ as follows.

$$\begin{aligned}
X_{13} &= (X_5 + (X_{12} + (X_{11} \oplus (\neg X_{10} \gg 23)))) \oplus X_{12} \\
X_{14} &= (X_6 - (X_{13} \oplus X_{12} \oplus (\neg(X_{12} + (X_{11} \oplus (\neg X_{10} \gg 23))) \gg 23))) + X_{13} \\
X_{15} &= (X_7 \oplus (X_{14} - X_{13})) - (X_{14} \oplus \texttt{0123456789ABCDEF})
\end{aligned}$$

This adds negligible cost to the attack complexity and guarantees that X_5, X_6 and X_7 are always correct after computing the key schedule backward.

8. To compute the initial chaining values A_{-1}, B_{-1} and C_{-1} run the rounds 2, 1, and 0 backwards.

Hence, we can construct a pseudo-collision (free-start-collision) for Tiger reduced to 23 rounds with a complexity of about 2^{47} applications of the compression function.

6 Conclusion

In this article, we have shown a 1-bit circular pseudo-near-collision for the full Tiger hash function with a complexity of about 2^{47} evaluations of the compression function of Tiger. This is the first attack on the full Tiger hash function. Furthermore, we show a pseudo-collision for Tiger reduced to 23 (out of 24) rounds with the same complexity. Our attack is based on the attack of Kelsey and Lucks on round-reduced variants of the Tiger hash function. This work shows that the security margins of the Tiger hash function are not as good as one would expect.

Acknowledgement

The authors wish to thank Sebastiaan Indesteege, and the anonymous referees for useful comments and discussions.

References

1. Ross J. Anderson and Eli Biham. TIGER: A Fast New Hash Function. In Dieter Gollmann, editor, *FSE*, volume 1039 of *Lecture Notes in Computer Science*, pages 89–97. Springer, 1996.
2. John Kelsey and Stefan Lucks. Collisions and Near-Collisions for Reduced-Round Tiger. In Matthew J. B. Robshaw, editor, *FSE*, volume 4047 of *Lecture Notes in Computer Science*, pages 111–125. Springer, 2006.
3. 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 *Lecture Notes in Computer Science*, pages 63–79. Springer, 2006.
4. Norbert Pramstaller, Christian Rechberger, and Vincent Rijmen. Exploiting Coding Theory for Collision Attacks on SHA-1. In Nigel P. Smart, editor, *IMA Int. Conf.*, volume 3796 of *Lecture Notes in Computer Science*, pages 78–95. Springer, 2005.

A Collision Attack on Tiger reduced to 16 rounds

In this section, we briefly describe the attack of Kelsey and Lucks on Tiger reduced to 16 rounds. Note that in the original description of the attack the wrong S-boxes are addressed. However, the attack can be easily modified to work with the correct S-boxes as well. Note that the modified attack has a slightly worse complexity, namely about 2^{47} instead of 2^{44} hash computations. For the attack the same characteristic is used for the key schedule of Tiger as in the original attack. The characteristic is shown below.

$$(I, I, I, I, 0, 0, 0, 0) \rightarrow (I, I, 0, 0, 0, 0, 0, 0) \qquad (7)$$

It has a probability of 1 to hold in the key schedule of Tiger, which facilitates the attack. The attack can be summarized as follows.

0. Precomputation: Like in the pseudo-near-collision attack on Tiger described before, we have to find a set of possible modular differences L^+ with a low Hamming weight XOR-difference $L^\oplus$ which can be canceled out by a suitable choice for B_6.

$$\mathcal{L} = \{L^+ \in \mathcal{L}' : \; L^+ = \mathbf{odd}(B_6 \oplus I) \boxminus \mathbf{odd}(B_6)\}$$

Second, we have to find a set of possible modular differences K^+ with a low Hamming weight XOR-difference $K^\oplus$ which can be canceled out by a suitable choice for B_7.

$$\mathcal{K} = \{K^+ \in \mathcal{K}' : \; K^+ = \mathbf{odd}(B_5 \oplus L^\oplus) \boxminus \mathbf{odd}(B_5)\}$$

Note that we assume in the attack that we can find a XOR-difference $L^\oplus$ with Hamming weight of 10 and a XOR-difference $K^\oplus$ with Hamming weight of 8 (as in the pseudo-near-collision attack on the full Tiger hash function). The precomputation step of the attack has a complexity of about $2^{28.5}$ evaluations of the compression function of Tiger.
1. Choose random values for $X_0, \ldots, X_1$ and $X_2[\mathbf{even}]$ to compute B_1, C_1 and C_2. This step of the attack has a complexity of about 6 round computations of Tiger.
2. Apply a message modification step to construct the XOR-difference $K^\oplus$ in round 3. This step has a complexity of about $2^{36.5}$ hash computations and determines the message words $X_2[\mathbf{odd}]$ and $X_3[\mathbf{even}]$.
3. Apply a second message modification step to construct the XOR-difference $L^\oplus$ in round 4. Finishing this step of the attack has a complexity of about 2^{39} and determines the message words $X_3[\mathbf{odd}]$ and $X_4[\mathbf{even}]$.
4. To construct the XOR-difference I in round 5, we apply again a message modification step. Finishing this step has a complexity of about 2^{40} and determines the message words $X_4[\mathbf{odd}]$ and $X_5[\mathbf{even}]$.
5. Once we have fixed the message words, we can compute B_4, C_4 and C_5 as well as the according modular differences. To cancel the difference in B_4 we need that $\Delta^\boxplus(B_4) \in \mathcal{K}$. Since we assume that the Hamming weight of $K^\oplus$ is 8, this has (in the worst case) a probability of 2^{-7}.
In order to guarantee that the difference in B_5 is canceled, we need that $\Delta^\boxplus(B_5) \in \mathcal{L}$. Since $L^\oplus$ has a Hamming weight of 10, this has a probability (in the worst case) of 2^{-9}. Hence, we expect that we have to repeat the attack about 2^{16} to finish this step. However, by choosing $L^\oplus$ and $K^\oplus$ carefully this can be improved. Form our analysis (for the pseudo-near-collision for the full Tiger hash function), we expect that this probability can be improved by a factor of 2^9, resulting in an attack complexity of about 2^{47} hash computations.

6. Determine $X_5[\mathbf{odd}]$ and $X_6[\mathbf{odd}]$ according to the results of the precomputation step. This adds no additional cost to the attack complexity.

Hence, a collision can be constructed in Tiger reduced to 16 rounds with a complexity close to 2^{47} evaluations of the compression function. Note that the other attacks on round-reduced variants of Tiger can be adjusted in a similar way.

B Collision Attack on Tiger reduced to 19 rounds

In this section, we show how the collision attack on Tiger-19 presented in [3] has to be modified to work with the correct S-boxes. The complexity of the attack is close to 2^{62} evaluations of the compression function of Tiger. To construct a collision in Tiger-19 the key schedule difference given in (8) is used. It has probability 1 to hold in the key schedule of Tiger which facilitates the attack.

$$(0,0,0,I,I,I,I,0) \rightarrow (0,0,0,I,I,0,0,0) \rightarrow (0,0,0,I,I,I,I,I) \tag{8}$$

Since the key schedule difference from round 3 to 18 is the 16-round difference used in the attack on Tiger-16, the same attack strategy can be used for the collision attack on Tiger-19 as well. The attack can be summarized as follows:

1. Choose arbitrary values for $X_0, \ldots, X_4$ and compute the state variables A_3, B_3, and B_4.
2. Employ the attack on 16 rounds of Tiger, to find the message words $X_5, \ldots, X_7$ and $X_8, X_9[\mathbf{odd}]$ such that the output after round 18 collides.
3. To guarantee that $X_8, X_9[\mathbf{odd}]$ are correct after applying the key schedule, we use the degrees of freedom we have in the choice of $X_0, \ldots, X_4$. Note that for any difference injected in X_0 and X_1 one can adjust X_2, X_3, X_4 accordingly such that A_3, $B_3 = C_2 \oplus X_3$ and $B_4 = C_3 \oplus X_4$ stay constant. Furthermore, we get the following equations for X_8 and X_9 from the key schedule of Tiger.

$$\begin{aligned} X_8 &= Y_0 \boxplus Y_7 \\ X_9 &= Y_1 \boxminus (X_8 \oplus (\neg Y_7 \ll 19)) \end{aligned}$$

where

$$\begin{aligned} Y_0 &= X_0 \boxminus (X_7 \oplus \texttt{A5A5A5A5A5A5A5A5}) \\ Y_1 &= X_1 \oplus Y_0 \\ Y_2 &= X_2 \boxplus Y_1 \\ Y_3 &= X_3 \boxminus (Y_2 \oplus (\neg Y_1 \ll 19)) \\ Y_4 &= X_4 \oplus Y_3 \\ Y_5 &= X_5 \boxplus Y_4 \\ Y_6 &= X_6 \boxminus (Y_5 \oplus (\neg Y_4 \gg 23)) \\ Y_7 &= X_7 \oplus Y_6 \end{aligned}$$

To solve these equations the following method is used:

(a) Choose a random value for Y_0. This determines Y_7 and X_0.
(b) Choose a random value for $X_9[\mathbf{even}]$. This determines X_1.
(c) Adjust X_2, X_3, X_4 accordingly such that A_3, $B_3 = C_2 \oplus X_3$ and $B_4 = C_3 \oplus X_4$ stay constant.
(d) Once we have fixed X_2, X_3, and X_4, we have to check if Y_7 is correct (this holds with a probability of 2^{-64}). After repeating the method about 2^{64} times for different values of Y_0, we expect to find a match.

Hence, this step of the attack has a complexity of at about 2^{64} key schedule computations and $4 \cdot 2^{64}$ round computations of Tiger. This is equivalent to about 2^{62} evaluations of the compression function of Tiger.

Thus, we can construct a collision in Tiger reduced to 19 rounds with a complexity of about $2^{62} + 2^{47} \approx 2^{62}$ evaluations of the compression function of Tiger.