Cryptanalysis of the Tiger Hash Function

No Author Given

No Institute Given

Abstract. Tiger is a cryptographic hash function with a 192-bit hash value. It was proposed by Anderson an Biham in 1996. Recently, weaknesses has 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 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 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}. So far 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}. Furthermore, we present 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 Ross Anderson and Eli 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 rounds of Tiger. The attack has a complexity of about 2^{44} evaluations of the compression function of Tiger. Furthermore, they present a pseudo-near-collision for a variant of Tiger reduced to 20 rounds with complexity of about 2^{48}. These results were later improved by Mendel *et al.* in [3]. They show that a collision can be found in Tiger reduced to 19 rounds with a complexity of about 2^{62} evaluations of the Tiger compression function. In the paper they also 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.

In this article, we will show how these attacks can be extended to the full Tiger hash function. We will present a pseudo-near-collision for the full Tiger hash function with a complexity of about 2^{47} evaluations of the compression function of Tiger. Furthermore, we will present a pseudo-collision for a variant of Tiger reduced to 23 rounds with the same complexity.

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 pseudo-near-collision on Tiger with complexity of about 2^{47}. In Section 5, we show a pseudo-collision for the Tiger hash function reduced to 23 rounds. Finally, we present conclusions in Section 6.

2 Description of the Hash Function Tiger

Tiger is a cryptographic hash function that was designed by Ross Anderson and Eli 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 use 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 registers 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:

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

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.

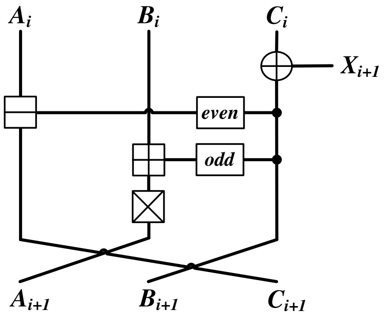

Fig. 1. The round function of Tiger.

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 C is split into eight bytes $c_0, \ldots, c_7$ with c_7 is the most significant byte. The 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 four S-boxes we refer to [1]. Note that chaining variable B is multiplied with the constant `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.

c_0 is LSB that was wrong in previous paper

After the last round of the state update transformation, the chaining variables 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 chaining 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 shown below.

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

The key schedule modifies the inputs $(Y_0, \ldots, Y_7)$ in two steps, as shown below.

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 a certain difference in the state variables, which can then be canceled out by the differences in the message words.

In the following sections of this article we will describe these two steps of the attack in more detail.

3.1 Finding a good characteristic in the Key Schedule of Tiger

To find a good characteristic in 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 in $GF(2)$. 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 chaining variables, which can then be canceled out by the differences in 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.

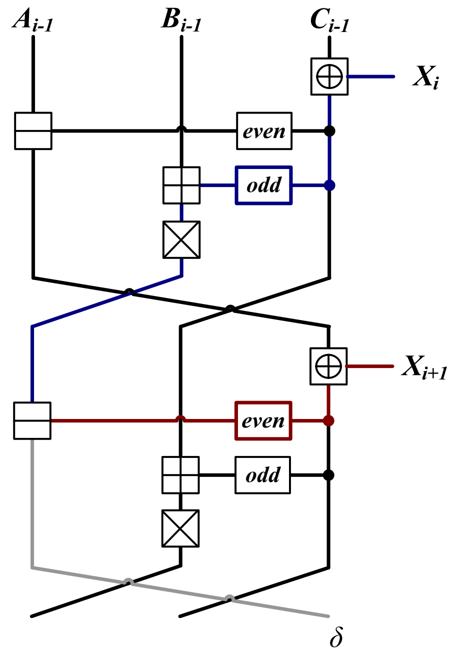

Fig. 2. Message Modification by Meet-in-the-Middle.

Assume, we have 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 probability of 2^{-1} by using a birthday attack. The idea is to use the degree of freedom we have in the choice of the message words ($X_{i-1}[\mathbf{odd}]$ and $X_i[\mathbf{even}]$) to generate 2^{32} candidates for $\Delta^{\boxplus}(\mathbf{odd}(B_i))$ and $\Delta^{\boxplus}(\mathbf{even}(B_{i+1}))$ and then to 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

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

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.

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 destroying again, resulting in an 1-bit pseudo-near-collision for the Tiger hash function. The attack has a complexity of about 2^{47} evaluations of the compression function of Tiger. In the attack we extend techniques invented by Kelsey and Lucks in the attack on Tiger-16.

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 $\Delta^{\oplus}$) and the modular difference (denoted by $\Delta^{\boxplus}$) is the same for a difference in the MSB.

In order to have a pseudo-collision in the compression function of Tiger after 24 rounds, there has to be a pseudo-collision after round 17 as well. Hence, the following differences are needed in the chaining 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 after 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.

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

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	

XOR-difference $L^{\oplus}$. A modular difference L^{+} is consistent to $L^{\oplus}$ if there exist p, 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:

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

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 in the attack we need a $L^{\oplus}$ 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.

mention: this is not so surprising since the other sbox (odd) is addressed

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

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 $K^\oplus$.

$$\mathcal{K} = \{K^+ \in \mathcal{K}' : \; K^+ = \mathbf{odd}(B_{12} \oplus L^\oplus) \boxminus \mathbf{odd}(B_{12})\}$$

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 $B_{12}[\mathbf{odd}]$. 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.

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

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 a random value for B_4 and B_5 and compute $A_5 = (B_4 \boxplus \mathbf{odd}(B_5)) \boxtimes \mathtt{mult}$. 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}, respectively 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.

4.3 Constructing the XOR difference $K^\oplus$

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 probability of 2^{-8}. Furthermore, the message modification step has a probability of 2^{-1}. Hence, this step of the attack succeeds with 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 $(12 + 2^{32} + 2^8 \cdot 2^{32}) \cdot 2 = 2^{41}$ round computations of Tiger. This is approximately about $2^{36.5}$ evaluations of the compression function of Tiger.

4.4 Constructing the XOR difference $L^\oplus$

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 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 $(2^{41} + (2^{32} + 2^{32} \cdot 2^{10})) \cdot 2 = 2^{43.6}$ round computations of Tiger. This is approximately about 2^{39} evaluations of the compression function of Tiger.

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

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 which are consistent to the XOR difference I for the message modification step. This step of the attack succeeds with 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 $(2^{43.6} + (2^{32} + 2^{32})) \cdot 2^8 = 2^{51.6}$ 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 of 1.

explain probability of 1 in detail

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 real message words $X_0, \ldots, X_7$

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

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

Hence, we get a pseudo-near-collision for the Tiger hash function with complexity of about 2^{47} evaluations of the compression function of Tiger. 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 an 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 for Tiger reduced to 23 rounds by using an other characteristic for the key schedule. For the attack we use the key schedule differences given below. It holds with probability 1 in the key schedule of Tiger which facilitates the attack.

$$(0, 0, 0, I, 0, 0, 0, I) \rightarrow (0, I, 0, 0, 0, 0, 0, I) \rightarrow (0, 0, 0, 0, 0, 0, 0, I) \tag{6}$$

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.

6 Conclusion

In this article, we have shown a 1-bit 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. Our work shows that the security margins of Tiger are not as good as one would expect.

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.