Update on Tiger [⋆]

Florian Mendel[1], Bart Preneel[2], Vincent Rijmen[1],
Hirotaka Yoshida[3], and Dai Watanabe[3]

[1] Graz University of Technology
Institute for Applied Information Processing and Communications
Inffeldgasse 16a, A–8010 Graz, Austria
{Florian.Mendel,Vincent.Rijmen}@iaik.tugraz.at
[2] Katholieke Universiteit Leuven, Dept. ESAT/SCD-COSIC,
Kasteelpark Arenberg 10, B–3001 Heverlee, Belgium
Bart.Preneel@esat.kuleuven.be
[3] Systems Development Laboratory, Hitachi, Ltd.,
1099 Ohzenji, Asao-ku, Kawasaki-shi, Kanagawa-ken, 215-0013 Japan
{hirotaka.yoshida.qv,dai.watanabe.td}@hitachi.com

Abstract. Tiger is a cryptographic hash function with a 192-bit hash value which was proposed by Anderson and Biham in 1996. At FSE 2006, Kelsey and Lucks presented a collision attack on Tiger reduced to 16 (out of 24) rounds with complexity of about 2^{44}. Furthermore, they showed that a pseudo-near-collision can be found for a variant of Tiger with 20 rounds with complexity of about 2^{48}.
In this article, we show how their attack method can be extended to construct a collision in the Tiger hash function reduced to 19 rounds. We present two different attack strategies for constructing collisions in Tiger-19 with complexity of about 2^{62} and 2^{69}. Furthermore, we present a pseudo-near-collision for a variant of Tiger with 22 rounds with complexity of about 2^{44}.

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

1 Introduction

Recent results in cryptanalysis of hash function show weaknesses in many commonly used hash functions, such as SHA-1 and MD5 [4,5]. Therefore, the cryptanalysis of alternative hash functions, such as Tiger, is of great interest.

In [2], Kelsey and Lucks presented a collision attack on Tiger-16, a round reduced variant of Tiger (only 16 out of 24 rounds), with complexity of about 2^{44}. In the attack they used a kind of message modification technique developed

⋆ This work was supported in part by the Austrian Science Fund (FWF), project P18138. This work was supported in part by a consignment research from the National Institute on Information and Communications Technology (NiCT), Japan. This work was supported in part by the Concerted Research Action (GOA) Ambiorics 2005/11 of the Flemish Government.

for Tiger to force a differential pattern in the chaining variables after round 7, which can then be canceled by the differences in the expanded message words in the following rounds. This led to a collision in the Tiger hash function after 16 rounds. Furthermore, they showed that a pseudo-near-collision can be found in a variant of Tiger with 20 rounds in about 2^{48} applications of the compression function.

In this article, we extend the attack to construct a collision in Tiger-19. We present two different collision attacks on Tiger-19 with complexity of 2^{62} and 2^{69}. Furthermore, we present a pseudo-near-collision attack for a variant of Tiger with 22 rounds with complexity of about 2^{44} and a pseudo-collision attack for Tiger-23/128, a version of Tiger reduced to 23 rounds with truncated output, with complexity 2^{44}. A summary of our results is given in Table 1.

Table 1. Overview of attacks on the Tiger hash function.

number of rounds	type	complexity	
Tiger-16	collision	2^{44}	in [2]
Tiger-19	collision	2^{62} and 2^{69}	in this article
Tiger-19	pseudo-collision	2^{44}	in this article
Tiger-21	pseudo-collision	2^{66}	in this article
Tiger-23/128	pseudo-collision	2^{44}	in this article
Tiger-20 [4]	pseudo-near-collision	2^{48}	in [2]
Tiger-21	pseudo-near-collision	2^{44}	in this article
Tiger-22	pseudo-near-collision	2^{44}	in this article

The remainder of this article is structured as follows. A description of the Tiger hash function is given in Section 2. The attack of Kelsey and Lucks on Tiger-16 is described in Section 3. In Section 4, we describe a method to construct collisions in Tiger-19. Another method for construction collisions in Tiger-19 is described in Section 5. Furthermore, we present a pseudo-near-collision for Tiger-22 in Section 6 and a pseudo-collision for Tiger-23/128 in Section 7. Finally, we present conclusions in Section 8.

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 use the same notation as is used in [2]. The notation is given in Table 2.

[4] Kelsey and Lucks show a pseudo-near-collision for the last 20 rounds of Tiger.

Table 2. Notation

Notation	Meaning
$A + B$	addition of A and B modulo 2^{64}
$A - B$	subtraction of A and B modulo 2^{64}
$A * B$	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 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 chaining variables A, B and C as follows.

$$\begin{aligned} C &= C \oplus X \\ A &= A - \mathbf{even}(C) \\ B &= B + \mathbf{odd}(C) \\ B &= B \times \texttt{mult} \end{aligned}$$

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.

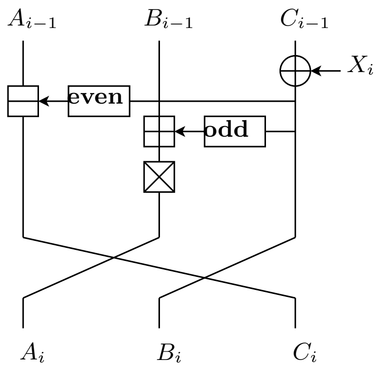

Fig. 1. The round function of Tiger.

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 $c_0, \ldots, c_7$ where c_0 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.

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 pass 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.

$$\begin{aligned}A'_{23} &= A_{-1} \oplus A_{23}\\ B'_{23} &= B_{-1} - B_{23}\\ C'_{23} &= C_{-1} + C_{23}\end{aligned}$$

2.2 Key Schedule

Between two passes of Tiger, there is one 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**
$Y_0 = Y_0 - (Y_7 \oplus \texttt{A5A5A5A5A5A5A5A5})$	$Y_0 = Y_0 + Y_7$
$Y_1 = Y_1 \oplus Y_0$	$Y_1 = Y_1 - (Y_0 \oplus ((\neg Y_7) \ll 19))$
$Y_2 = Y_2 + Y_1$	$Y_2 = Y_2 \oplus Y_1$
$Y_3 = Y_3 - (Y_2 \oplus ((\neg Y_1) \ll 19))$	$Y_3 = Y_3 + Y_2$
$Y_4 = Y_4 \oplus Y_3$	$Y_4 = Y_4 - (Y_3 \oplus ((\neg Y_2) \gg 23))$
$Y_5 = Y_5 + Y_4$	$Y_5 = Y_5 \oplus Y_4$
$Y_6 = Y_6 - (Y_5 \oplus ((\neg Y_4) \gg 23))$	$Y_6 = Y_6 + Y_5$
$Y_7 = Y_7 \oplus Y_6$	$Y_7 = Y_7 - (Y_6 \oplus \texttt{0123456789ABCDEF})$

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

3 Previous Attack on Tiger

In this section, we will briefly describe the attack of Kelsey and Lucks on Tiger-16. A detailed description of the attack is given in [2]. For a good understanding of our results, it is recommended to study it very carefully. Space restrictions do not permit us to copy all the important details of the original attack. The attack on Tiger-16 can be summarized as follows.

1. Choose a characteristic for the key schedule of Tiger that holds with high probability (ideally with probability 1).
2. Use a kind of message modification technique [5] developed for Tiger to construct certain differences in the chaining variables for round 7, which can then be canceled by the differences in the message words in the following rounds. This leads to a collision in the Tiger hash function after 16 rounds.

In the following we will describe both parts of the attack in detail.

3.1 High Probability Characteristic for the Key Schedule of Tiger

For the attack Kelsey and Lucks used the key schedule difference given in (1). It has probability 1 to hold in the key schedule of Tiger. This facilitates the attack.

$$(I, I, I, I, 0, 0, 0, 0) \rightarrow (I, I, 0, 0, 0, 0, 0, 0) \tag{1}$$

Note that I denotes a difference in the MSB of the message word. Hence, the XOR difference (denoted by $\Delta^{\oplus}$) and the additive difference (denoted by Δ^{+}) is the same in this particular case.

To have a collision after 16 rounds, there has to be a collision after round 9 as well. Hence, the following differences are needed in the chaining variables for round 7 of Tiger.

$$\Delta^{\oplus}(A_6) = I, \quad \Delta^{\oplus}(B_6) = I, \quad \Delta^{\oplus}(C_6) = 0 \tag{2}$$

Constructing these differences in the chaining variables after round 6 is the most difficult part of the attack. Therefore, Kelsey and Lucks adapted the idea of message modification from the MD-family to Tiger. The main idea of message modification is to use the degrees of freedom we have in the choice of the message words to control the differences in the chaining variables. In the case of Tiger, the differential pattern given in (2) has to be met in order to have a collision after 16 rounds of Tiger.

3.2 Message modification by Meeting in the Middle

In this section, we explain the idea of message modification in Tiger according to Fig. 2. Assume that the values of $(A_{i-1}, B_{i-1}, C_{i-1})$ and the additive differences $\Delta^{+}(A_{i-1})$, $\Delta^{+}(B_{i-1})$, $\Delta^{+}(C_{i-1})$ are known as well as the additive differences in the message words X_i and X_{i+1}. Then the additive difference $\Delta^{+}(C_{i+1})$ can

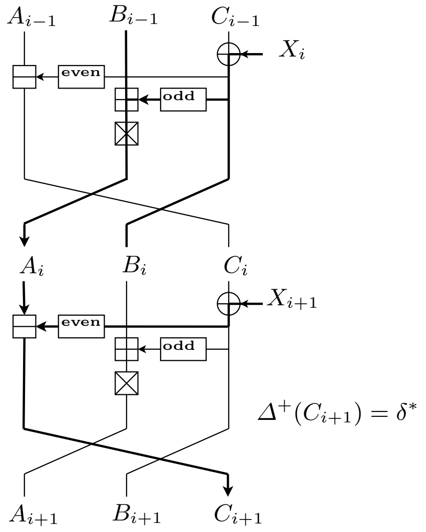

Fig. 2. Outline of the message modification step in Tiger.

be forced to be any difference δ^* with probability 1/2 by applying the birthday attack. As depicted in Fig. 2, the additive difference $\Delta^+(C_{i+1})$ depends on the additive differences $\Delta^+(B_{i-1})$, $\Delta^+(\mathbf{odd}(B_i))$, and $\Delta^+(\mathbf{even}(B_{i+1}))$.

For any nonzero XOR difference $\Delta^\oplus(B_{i+1}[\mathbf{even}])$, one expect about 2^{32} *different* corresponding additive output differences $\Delta^+(\mathbf{even}(B_{i+1}))$. Similarly, for any nonzero XOR difference $\Delta^\oplus(\mathbf{odd}(B_i))$, one expect close to 2^{32} corresponding *different* additive output differences $\Delta^+(\mathbf{odd}(B_i))$.

Thus, if the XOR differences $\Delta^\oplus(B_{i+1}[\mathbf{even}])$ and $\Delta^\oplus(B_i[\mathbf{odd}])$ both are nonzero, a meet-in-the-middle (MITM) approach can be applied to solve the following equation:

$$\texttt{mult} \times (\Delta^+(B_{i-1}) + \Delta^+(\mathbf{odd}(B_i))) - \Delta^+(\mathbf{even}(B_{i+1})) = \delta^* \ .$$

This is done by performing the following two steps:

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

This technique takes 2^{33} evaluations of each of the functions **odd** and **even**, which is equivalent to about 2^{29} evaluations of the compression function of Tiger reduced to 16 rounds and some 2^{33} 64-bit word units of storage space.

Note that if the choice of the values of the message words X_i and X_{i+1} is constrained by k-bits then the success probability of the message modification step is reduced by a factor of 2^k. This is referred to as a constrained message modification step.

3.3 The collision attack on Tiger-16

With the key schedule difference given in Section 3.1 and the new developed message modification technique for Tiger described in Section 3.2, Kelsey and Lucks show a collision attack on Tiger reduced to 16 rounds. The method can be summarized as follows (see [2]).

0. Precomputation: Find an additive difference L^{+} with a low Hamming weight XOR difference $L^{\oplus}$ which can be canceled out by a suitable choice for $X_6[\textbf{even}]$. In the analysis Kelsey and Lucks assume, that an additive difference L can be found which is consistent to an 8-bit XOR difference L^{xor}. This step of the attack has a complexity of about 2^{27}.
1. Choose suitable values for $X_0, X_1, X_2[\textbf{even}]$ such that $\Delta^{\oplus}(A_2)$, $\Delta^{\oplus}(B_2)$, $\Delta^{\oplus}(C_2)$ are *useful*. A difference is called *useful* if there are differences in the even and odd bytes of the word. This step adds negligible cost to the attack complexity.
2. Do a message modification step to get a suitable XOR-difference L^{xor} in C_3 which is consistent with the additive difference L of the precomputation step. This step has complexity of about 2^{36} and determines the message words $X_2[\textbf{odd}]$ and $X_3[\textbf{even}]$.
3. Do a constrained message modification step to get $\Delta^{\oplus}(C_4) = I$. This determines $X_3[\textbf{odd}]$ and $X_4[\textbf{even}]$. Completing this step has complexity of about 2^{40}. This is due to the fact that 8 bits of X_4 (4 bits in $X_4[\textbf{even}]$ and 4 bits in $X_4[\textbf{odd}]$) are constrained by the transition of the XOR difference L^{xor} in C_3 to the additive difference L in B_4.
4. Do a constrained message modification step to get $\Delta^{\oplus}(C_5) = I$. This determines $X_4[\textbf{odd}]$ and $X_5[\textbf{even}]$. Completing this step has complexity of about 2^{44}.
5. Determine $X_6[\textbf{even}]$ by using C_5 and the results of the precomputation step. This adds no additional cost to the attack complexity.

Hence, a collision in Tiger-16 can be found with a complexity of about 2^{44} applications of the compression function. In the attack a characteristic for the key schedule differences is used which has probability 1 as well as a message modification technique developed for Tiger to force certain differences in the chaining variables after round 6 which can then be canceled by the differences in the expanded message words X_8 and X_9. For a detailed description of the attack we refer to [2].

4 A Collision Attack on Tiger-19 – Method 1

In this section we present a collision attack on Tiger-19 with complexity of about 2^{62} hash computations. First, we show how the attack of Kelsey and Lucks can be extended to construct a pseudo-collision in Tiger-19 with complexity of about 2^{44} hash computations. Second, we show how this pseudo-collision can be turned into a collision for Tiger-19 by using a kind of neutral bit technique. The collision attack on Tiger-19 has a complexity of 2^{62} hash computations.

4.1 A Pseudo-Collision for Tiger-19

In this section we will show how to construct a pseudo-collision for Tiger-19 with a complexity of about 2^{44}. The attack is an extension of the attack of Kelsey and Lucks on Tiger-16.

To construct a pseudo-collision in Tiger-19 we use the key schedule difference given in (3). It has probability 1 to hold in the key schedule of Tiger which facilitates the attack.

$$(0,0,0,I,I,I,I,0) \rightarrow (0,0,0,I,I,0,0,0) \rightarrow (0,0,0,I,I,I,I,I) \qquad (3)$$

Note that the key schedule difference from round 3 to 18 is the 16-round difference used by Kelsey and Lucks in the attack on Tiger-16. Hence, we can use the same attack strategy which was used to break Tiger-16 in the attack on Tiger-19 as well. The attack work as follows:

1. Choose arbitrary values for the chaining variables A_2, B_2, C_2 for round 3.
2. Employ the attack on 16 rounds, to find message words $X_3, \ldots, X_7$ and $X_8[\textbf{even}], X_9[\textbf{even}]$ such that the output after round 18 collides.
3. To compute the real message words $X_0, \ldots, X_7$, we have to choose suitable values for $X_8[\textbf{odd}], X_9[\textbf{odd}]$ and $X_{10}, \ldots, X_{15}$ such that X_4, X_5, X_6 and X_7 are correct after computing the key schedule backward. Note that X_3 can be chosen freely, because we can modify C_2 such that $C_2 \oplus X_3$ stay constant. In detail, we choose arbitrary values for $X_8[\textbf{odd}], X_9[\textbf{odd}]$, X_{10}, X_{11} and calculate $X_{12}, \ldots, X_{15}$ as follows.

$$\begin{aligned}
X_{12} &= (X_4 \oplus (X_{11} - X_{10})) - (X_{11} \oplus (\neg X_{10} \gg 23)) \\
X_{13} &= (X_5 + (X_{12} + (X_{11} \oplus (\neg X_{10} \gg 23)))) \oplus X_{12} \\
X_{14} &= (X_6 - (X_{13} \oplus X_{12} \oplus (\neg(X_{12} + (X_{11} \oplus (\neg X_{10} \gg 23))) \gg 23))) + X_{13} \\
X_{15} &= (X_7 \oplus (X_{14} - X_{13})) - (X_{14} \oplus \texttt{0123456789ABCDEF})
\end{aligned}$$

   This adds negligible cost to the attack complexity and guarantees that X_4, X_5, X_6 and X_7 are always correct after computing the key schedule backward.
4. 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 for Tiger-19 with a complexity of about 2^{44} applications of the compression function. We can turn this pseudo-collision into a collision for Tiger-19. This is described in detail in the next section.

4.2 From a Pseudo-Collision to a Collision in Tiger-19

Constructing a collision in Tiger-19 works quite similar as constructing the pseudo-collision. Again we use the key schedule difference given in (3) and employ the attack on 16 rounds of Tiger. The attack can be summarized as follows.

1. Choose arbitrary values for X_0,X_1 and X_2 and compute the chaining variables A_2, B_2, C_2 for round 3.
2. Employ the attack on 16 rounds, to find the message words $X_3, \ldots, X_7$ and $X_8[\mathbf{even}], X_9[\mathbf{even}]$ such that the output after round 18 collides.
3. To guarantee the $X_8[\mathbf{even}], X_9[\mathbf{even}]$ are correct after applying the key schedule, we use the degrees of freedom we have in the choice of X_0, X_1, X_2, X_3. Note that for any difference we introduce into X_0, you can introduce canceling differences into X_1, X_2, X_3 such that A_2, B_2 and $B_3 = C_2 \oplus X_3$ stay constant. This is a kind of local collision for the first 4 rounds of Tiger.

$$\begin{aligned} X_0^{\mathbf{new}} &= \text{arbitrary} \\ X_1^{\mathbf{new}} &= C_0^{\mathbf{new}} \oplus C_0 \oplus X_1 \\ X_2^{\mathbf{new}} &= C_1^{\mathbf{new}} \oplus C_1 \oplus X_2 \\ X_3^{\mathbf{new}} &= C_2^{\mathbf{new}} \oplus C_2 \oplus X_3 \end{aligned}$$

   After testing all 2^{64} possible choices for X_0 and changing X_1, X_2, and X_3 accordingly such that A_2, B_2 and B_3 stay constant, we expect to get the correct values for $X_8[\mathbf{even}], X_9[\mathbf{even}]$ after applying the key schedule of Tiger.
   Hence, this step of the attack has a complexity of at about 2^{64} key schedule computations and 3×2^{64} round computations. This is equivalent to about 2^{62} applications of the compression function of Tiger-19.

Thus, we can construct a collision in Tiger-19 with complexity of about $2^{62} + 2^{44} \approx 2^{62}$ applications of the compression function. We are not aware of any other collision attack on Tiger which works for so many rounds. The best collision attack on Tiger so far was for 16 rounds by Kelsey and Lucks described in [2].

5 Collision Attack on Tiger-19 – Method 2

We now present another method to find collisions for the 19-round Tiger. The attack complexity of this attack method is slightly higher than the one in the previous attack method. One difference from the previous method is that the first method uses larger space of message than the second one. This can been seen where X_0 is used in each attack. The first method uses whole 64 bits of X_0 and the second one uses less bits of X_0.

The attack described here is also an extension of the attack by Kelsey and Lucks. However, our attack is in a different situation from their attack. Their attack precomputes the additive difference L and then use X_6 to cancel it out in the main phase. Similarly, our attack precomputes the additive difference α and then use X_9 to cancel it out in the main phase. The key difference is that their attack controls X_6 in a deterministic way but our attack has to do in a probabilistic way due to the key schedule. This causes the main difficulty we have to solve here.

The outline of the attack is as follows:

1. Search for a good differential characteristic of the message words for 19 rounds.
2. Construct a good differential characteristic for 19 rounds by considering the message word differences expected from the characteristic in Step 1.
3. Divide this characteristic for round 3-9 into two consecutive characteristics (characteristic for round 3-7 and characteristic for round 8-9) so that we work on them *independently.*
4. Do the MITM step for the characteristic for round 3-7. Determine the chaining values A_3, B_3, B_3 and the message words X_4, X_5, X_6, $X_7[\mathbf{even}]$.
5. Do the MITM step for the characteristic for round 8-9 by varying the message words $X_0, X_1, X_2, X_3, X_7[\mathbf{odd}]$ while keeping the previously determined values unchanged. Determine all of the values.

In the attack, we use the same characteristic for the key schedule as in Section 4 and then construct a differential characteristic as shown in Table 3, where α and γ are some useful values in our attack. We will explain how these value are chosen in the next section.

Table 3. A collision-producing differential characteristic

i	$\Delta(A_i)$	$\Delta(B_i)$	$\Delta(C_i)$	$\Delta(X_i)$
3	0	I	$*$	I
4	$*$	$*$	$*$	I
5	$*$	$*$	$*$	I
6	$*$	$*$	γ	I
7	$*$	γ	I	0
8	α	$*$	I	0
9	I	I	0	0
10	I	0	I	0
11	0	0	I	I
12	0	0	0	I

5.1 The Precomputation Phase of the Attack

Before performing our attack, we need an algorithm to find a good differential characteristic starting with $\Delta^+(C_6)$ and ending with $\Delta^+(C_9)$ as shown in Table 3. We need the additive difference $\Delta^+(\mathbf{even}(B_9))$ to be equal to $\Delta^+(A_8)$. The question we have here is what difference we want in C_6 for obtaining a high probability. A solution to this is to compute the differences backward starting from the additive difference $\Delta^+(B_9) = I$. By performing experiments, we searched for α and γ such that the corresponding differential probabilities p_1, p_2 are high[5]. As a result, we found a high probability differential characteristic

[5] We have searched some sub space for the values α and γ so far. Searching the whole space could give us the better values for both of two.

which is shown in the following:

$$\begin{aligned}\Delta^+(B_9) = I &\overset{\textbf{even}}{\to} \Delta^+(\textbf{even}(B_9)) = \Delta^+(A_8) = \alpha \text{ with probability } p_2 \ ,\\ \Delta^+(A_8) = \alpha &\overset{\div,+}{\to} \Delta(B_7)^+ = \gamma \text{ with probability } 1 \ ,\\ \Delta^+(B_7) = \gamma &\overset{\oplus}{\to} \Delta(C_6)^+ = \Delta^+(B_7 \oplus X_7) = \gamma \text{ with probability } p_1 \ .\end{aligned}$$

Here the additive differences are

$$\begin{aligned}\alpha &= \texttt{0x80c02103d43214d6} \ ,\\ \gamma &= \alpha/7 \mod 2^{64} = \texttt{0xedd24ddbf9be02fa} \ ,\end{aligned}$$

and probabilities are $p_1 = 2^{-26}$ and $p_2 = 2^{-28}$. We here study the above characteristic with probability p_1 in detail.

In general, for a pair of data (J, J') and some constant value Q, if we assume the Hamming weight of $\Delta^{\oplus}(J, J')$ to be k, then the probability that $\Delta^+(J, J') = \Delta^+(J \oplus Q, J' \oplus Q)$ is 2^{-k}. This means that k bits of Q are constrained[6] to hold the above equation. Therefore, in the case of the characteristic with probability $p_1 = 2^{-26}$, we expect α to have 26 active bits as a XOR difference, which imposes a 26-bit condition on X_7.

Because of the large number of active bits, it seems plausible to assume that there is a 13-bit condition on $X_7[\textbf{even}]$ and a 13-bit condition on $X_7[\textbf{odd}]$. We denote the probabilities that these two conditions hold by $p_{1,even} = p_{1,odd} = 2^{-13}$ respectively.

5.2 The Main Phase of the Attack

We here describe how the main attack phase is performed. For a preparation we present the following lemma explaining the generic birthday attack which will be used for the MITM technique to work.

Lemma 1. *Consider two functions f and g having the same output space of n bit length. If we assume that f and g are random and we have r_1 inputs for f and r_2 inputs for g, the probability of having a pair of inputs (x, y) producing a collision $f(x) = g(y)$ is given by $p = 1 - \exp(-r_1 r_2/2^n)$ [3].*

This tells us that the MITM step works with some probability even if the number of output differences of the **odd** or **even** is less than 2^{32}. The main attack phase is performed as follows.

1. Arbitrarily choose the chaining values A_3, B_3, C_3 for round 4.
2. Choose $X_4[\textbf{even}]$ and ensure that the difference $\Delta^{\oplus}C_4$ is useful. By useful we mean that the corresponding XOR difference has at least 1 active bit in each odd byte for having the 2^{32} values for the additive difference $\Delta^+\textbf{odd}(B_5)$. The work here is negligible.

[6] For example, an XOR difference of 1 is consistent with an additive difference of either -1 or $+1$. If the low bit in J is 0, the low bit in J' will be 1, and reaching an additive difference of -1 will require fixing the low bit of Q to 1

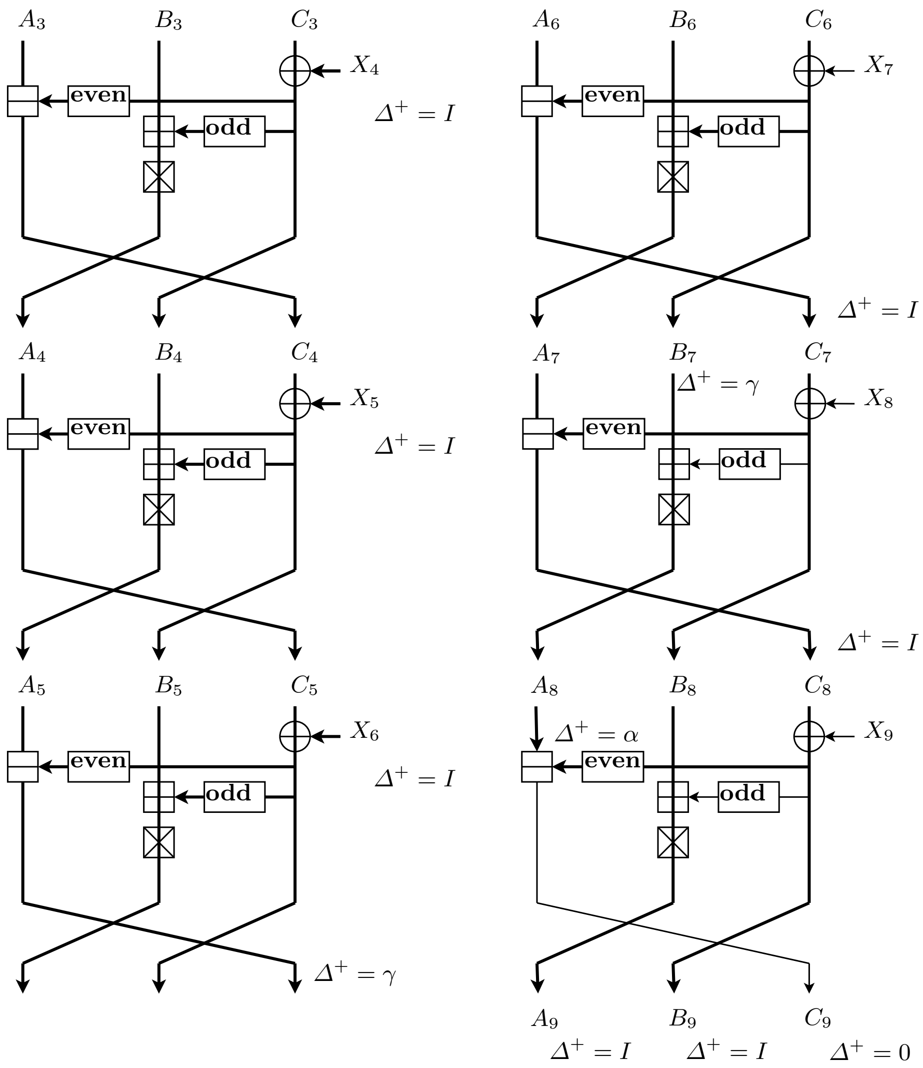

Fig. 3. The information flow from C_6 to C_9

3. Choose $X_4[\mathbf{odd}]$ and $X_5[\mathbf{even}]$ to ensure that the difference is $\Delta^{\oplus}C_5$ useful.
4. Perform a MITM step by choosing $X_5[\mathbf{odd}]$ and $X_6[\mathbf{even}]$ to get an additive difference γ in C_6. The expected work here is approximately 2^{33} evaluations of both of the **odd** function and the **even** function, and we determine $X_5[\mathbf{odd}]$ and $X_6[\mathbf{even}]$. Each failure requires that we go back to Step 3.
5. Set 13 bits of $X_7[\mathbf{even}]$ to hold the 13-bit condition on $X_7[\mathbf{even}]$ derived in the precomputation phase in Sect. 5.1 and then perform a MITM step using the generic birthday attack of Lemma 1. This is performed by choosing $X_6[\mathbf{odd}]$ and the rest bits of $X_7[\mathbf{even}]$ to get additive difference I in C_7.
   Each failure requires that we go back to Step 3. The expected work here is about 2^{13} computations, each of which consists of two kinds of evaluations: 2^{33} evaluations of the **odd** function and 2^{19} evaluations of the **even** function. We determine $X_6[\mathbf{odd}]$ and $X_7[\mathbf{even}]$ at the end of this step.
6. Set 13 bits of $X_7[\mathbf{odd}]$ to hold the 13-bit condition on $X_7[\mathbf{odd}]$ derived in the precomputation phase in Sect. 5.1 and then perform a MITM step to get the additive difference I in C_8. This is done by randomly choosing the rest bits of $X_7[\mathbf{odd}]$ and randomly generating $X_8[\mathbf{even}]$.
   The message word $X_8[\mathbf{even}]$ is generated in the following way: Randomly choose the message word X_0 and determine X_1, X_2, and X_3 so that the resulting A_3, B_3, C_3 are consistent with A_3, B_3, C_3 chosen in Step 1. We then determine $X_8[\mathbf{even}]$ from the key schedule.
   The above MITM step is performed with 2^{19} values for $\Delta^{+}(A_7)$ and 2^{28} values for $\Delta^{+}(\mathbf{even}(B_8))$ [7]. According to Lemma 1, the success probability of this attack is 2^{-17}. Therefore the expected work here is about 2^{18} computations, each of which consists of two kinds of evaluations: 2^{19} evaluations of the **odd** function and 2^{28} evaluations of the **even** function.
7. Compute $X_9[\mathbf{even}]$ by processing the key schedule and check if $\Delta^{+}\mathbf{even}(B_9) = \Delta^{+}(A_8)$, which means $\Delta^{+}(C_9) = 0$. Each failure requires that we go back to Step 6.

5.3 Complexity Analysis

We discuss the attack complexity in the attack in Sect. 5.2. The important thing to consider when we estimate the complexity is that the task of Steps 1-5 can be performed *independently* of the task of Steps 6-7. We first perform Steps 1-5 and then perform Steps 6-7 without changing the values which have been determined in Steps 1-5.

In order to determine $X_4, X_5, X_6, X_7[\mathbf{even}]$ by performing from Step 1 to Step 5, the required time complexity is equivalent to $p_{1,even}^{-1} \cdot 2^{33}$ evaluations of the **odd** function. In order to determine $X_0, X_1, X_2, X_3, X_7[\mathbf{odd}]$ by performing from Step 6 to Step 7, the required time complexity is equivalent to $p_2^{-1} \cdot 2^{18} \cdot 2^{28} = 2^{28} \cdot 2^{46} = 2^{74}$ evaluations of the **odd** function.

[7] Because of the XOR difference $\Delta^{\oplus}(B_8) = I$, there is only one active S-box at the input of the $\mathbf{even}(B_8)$. This makes the number of the additive difference smaller than 2^{32}.

The time complexity required by this attack is dominated by the latter part, which is equivalent to 2^{69} computations of the compression function of Tiger reduced to 19 rounds.

6 A Pseudo-Near-Collision for Tiger-22

In this section we present a pseudo-near-collision for Tiger-22 with complexity of about 2^{44}. Similar as we construct a pseudo-collision in Tiger-19, we can construct a pseudo-near-collision in Tiger-22. Again we use a key schedule difference that holds with probability 1 in the key schedule of Tiger and employ the attack on 16 rounds of Tiger. The key-schedule difference used in the attack is given in (4).

$$(0,0,I,0,0,0,I,I) \rightarrow (I,0,0,0,0,0,I,I) \rightarrow (0,0,0,0,0,0,I,I) \tag{4}$$

The attack work as follows:

1. Choose arbitrary values for the chaining variables A_5, B_5, C_5 for round 6.
2. Employ the attack on 16 rounds, to find message words $X_6, \ldots, X_{10}$ and $X_{11}[\mathbf{even}], X_{12}[\mathbf{even}]$ such that the output after round 21 collides.
3. To compute the real message words $X_0, \ldots, X_7$, we have to choose suitable values for $X_{11}[\mathbf{odd}], X_{12}[\mathbf{odd}]$ and $X_{13}, \ldots, X_{15}$ to guarantee that X_7 is correct after computing the key schedule backward. Therefore, we choose arbitrary values for $X_{11}[\mathbf{odd}], X_{12}[\mathbf{odd}], X_{13}, X_{14}$ and calculate X_{15} as follows:
$$X_{15} = (X_7 \oplus (X_{14} - X_{13})) - (X_{14} \oplus \texttt{0123456789ABCDEF})$$
This adds negligible cost to the attack complexity and guarantees that X_7 is correct after computing the key schedule backward. Note that X_6 can be chosen freely, because we can modify C_5 such that $C_5 \oplus X_6$ stays constant.
4. Run the rounds 5, 4, 3, 2, 1 and 0 backwards to compute the initial values A_{-1}, B_{-1} and C_{-1}. Since there is a difference in the message word X_2 in the MSB, we have to introduce the same difference in the initial value to cancel it out, namely
$$\Delta^{\oplus}(B_{-1}) = I \ .$$
Since the difference is in the MSB this happens with probability 1.
5. Of course, the feed forward destroys the pseudo-collision. After the feed forward we get the same output differences as in the initial values. Since the difference is in the MSB this has probability 1.
$$\Delta^{\oplus}(B'_{21}) = \Delta^{\oplus}(B_{-1} - B_{21}) = I$$

Hence, we can construct a pseudo-near-collision for Tiger-22 with complexity of about 2^{44}. For an ideal hash function with a hash value of 192-bit we would expect a complexity of about 2^{90} to construct a pseudo-near-collision with a one bit difference. Note that a pseudo-near-collision for Tiger-21 with a one bit difference can be found with the same complexity. A detailed description of the attack is given in the appendix.

7 A Pseudo-Collision for Tiger-23/128

Tiger/128 is a variant of Tiger, where the final hash value is truncated to 128 bit. This variant was specified in [1] to make Tiger compatible to MD5. In this section, we present a pseudo-collision for 23 rounds of Tiger/128. In detail, we can turn the pseudo-near-collision for Tiger-22 into a pseudo-collision for Tiger-23/128 by adding one additional round. If we add one round then the output after 23 rounds has the following differences in the chaining variables:

$$\Delta^{\oplus}(A_{22}) = 0, \quad \Delta^{\oplus}(B_{22}) = I, \quad \Delta^{\oplus}(C_{22}) \neq 0\,(\text{arbitrary})\ .$$

Due to the feed-forward the difference in B_{22} cancels out with probability 1. Hence, we have a pseudo-collision in Tiger-23/128, since only register A and B are used for the final hash value of Tiger-128. The attack has a complexity of about 2^{44} applications of the compression function.

8 Conclusion

In [2], Kelsey and Lucks discussed the possibility of extending their attack to more rounds of Tiger and the applicability of their attack techniques to the full hash function.

In this article, we presented two strategies for constructing collision in the Tiger-19 hash function. The first has a complexity of about 2^{62} hash computations and the second has a slightly higher complexity of about 2^{69} hash computations.

The best attack on a reduced variant of Tiger so far was proposed by Kelsey and Lucks in [2]. They showed a collision attack on Tiger-16 with a complexity of about 2^{44} and a pseudo-near-collision for a variant of Tiger with 20 rounds with a complexity of about 2^{48}. We have extended their approach to show collision attacks on Tiger-19 and presented a pseudo-near-collision for Tiger-22 and a pseudo-collision for Tiger-23/128. Based on this we conclude that the security margin of Tiger is not as large as one could hope for. It remains a topic of further research to determine whether the attacks can be extended to Tiger variants with more than 23 rounds.

Acknowledgement

The authors wish to thank Antoine Joux, Elisabeth Oswald, 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, *Fast Software Encryption, Third International Workshop, Cambridge, UK, February 21-23, 1996, Proceedings*, 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 Matt Robshaw, editor, *Fast Software Encryption, 13th International Workshop, FSE 2006, Graz, Austria, March 15-17, 2006*, volume 4047 of *LNCS*, pages 111–125, 2006.
3. Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1997. Available online at `http://www.cacr.math.uwaterloo.ca/hac/`.
4. Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding Collisions in the Full SHA-1. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005, 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *LNCS*, pages 17–36. Springer, 2005.
5. Xiaoyun Wang and Hongbo Yu. How to Break MD5 and Other Hash Functions. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings*, volume 3494 of *LNCS*, pages 19–35. Springer, 2005.

A A Pseudo-Near-Collision for Tiger-21

In a similar way as we construct a pseudo-near-collision in Tiger-22, we can construct a pseudo-near-collision for Tiger-21. For the attack we use the key-schedule difference given in (5). It has probability 1 to hold in the key-schedule of Tiger.

$$(0,I,0,0,0,I,I,I) \to (0,0,0,0,0,I,I,0) \to (0,0,0,0,0,I,I,I) \tag{5}$$

Again we use the attack on 16 rounds of Tiger (described in Section 3) to construct a pseudo-near-collision in Tiger-21. The attack work as follows:

1. Choose arbitrary values for the chaining variables A_4, B_4, C_4 for round 5.
2. Employ the attack on 16 rounds, to find message words $X_5, \dots, X_9$ and $X_{10}[\textbf{even}], X_{11}[\textbf{even}]$ such that the output after round 20 collides.
3. To compute the real message words $X_0, \dots, X_7$, we have to choose suitable values for $X_{10}[\textbf{odd}], X_{11}[\textbf{odd}]$ and $X_{12}, \dots, X_{15}$ such that X_6 and X_7 is correct after computing the key schedule backward. Therefore, we choose arbitrary values for $X_{10}[\textbf{odd}], X_{11}[\textbf{odd}]$ and X_{12}, X_{13} and calculate X_{14}, X_{15} as follows:

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

   This adds negligible cost to the attack complexity and X_6, X_7 are always correct after computing the key schedule backward. Note that X_5 can be chosen freely, because we can modify C_4 such that $C_4 \oplus X_5$ stay constant.
4. Run the rounds 4, 3, 2, 1 and 0 backwards to compute the initial values A_{-1}, B_{-1} and C_{-1}. Since there is a difference in the message word X_1 in the MSB, we introduce the same difference in the initial value to cancel it out. Since the difference is in the MSB, this happens with probability 1.

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

5. Of course, the feed forward destroys the pseudo-collision. After the feed forward we get the same output differences as in the initial values:

$$\Delta^{\oplus}(A'_{20}) = \Delta^{\oplus}(A_{-1} \oplus A_{20}) = I \ .$$

Hence, we can construct a pseudo-near-collision for Tiger-21 with complexity of about 2^{44} applications of the compression function. For an ideal hash function with a hash value of 192-bit we would expect a complexity of about 2^{90} applications of the compression function instead of 2^{44}.

B A Pseudo-Collision for Tiger-21

In a similar way as we construct a pseudo-near-collision in Tiger-21, we can construct a pseudo-collision in Tiger-21. For the attack we use again the key-schedule difference given in (5). The attack can be summarized as follows:

1. Choose arbitrary values for the chaining variables A_0, B_0, C_0 for round 1.
2. Choose random values for X_1, X_2, X_3, X_4 and calculate A_4, B_4, C_4.
3. Employ the attack on 16 rounds of Tiger, to find message words $X_5, \ldots, X_9$ and $X_{10}[\mathbf{even}], X_{11}[\mathbf{even}]$ such that the output after round 20 collides.
4. To compute the real message words $X_0, \ldots, X_7$, we have to choose suitable values for X_0, X_1 and X_2 such that X_8, X_9 and $X_{10}[\mathbf{even}], X_{11}[\mathbf{even}]$ are correct after computing the key schedule. Note that X_0 and X_1 can be chosen freely, because we can modify C_0 and C_1 such that $C_{-1} \oplus X_0$ and $C_0 \oplus X_1$ stay constant. Since a difference is introduced by X_1, we have after round 1 that $\Delta^{\oplus}(C_1) \neq 0$. Hence, X_2 can not be chosen freely.
   However, since we can choose the value of $C_0 \oplus X_1$ in the beginning of the attack, we can guarantee that the Hamming weight of $\Delta^{\oplus}(C_1)$ is small. Computer experiments show that the smallest weight we can get is 22. Consequential there are $2^{64-22} = 2^{42}$ possible choices for C_1 and X_2 such that $\Delta^{\oplus}(C_1 \oplus X_2)$ and $C_1 \oplus X_2$ stay constant. Hence, we have $2^{64+64+42} = 2^{170}$ degrees of freedom in the key schedule of Tiger. Therefore, we have to repeat the attack at most 2^{22} times to guarantee that X_8, X_9 and $X_{10}[\mathbf{even}], X_{11}[\mathbf{even}]$ are correct after applying the key schedule.

Hence, we can find a pseudo-collision in Tiger-21 with a complexity of about $2^{44+22} = 2^{66}$ applications of the compression function. Note that we assume in the analysis that it is computational easy to find suitable values for X_0, X_1, X_2.