New Message Difference for MD4

Yu Sasaki, Lei Wang, Kazuo Ohta and Noboru Kunihiro

The University of Electro-Communications
Chofugaoka 1-5-1, Chofu-shi, Tokyo, 182-8585, Japan
{yu339,wanglei,ota,kunihiro}@ice.uec.ac.jp

Abstract. This paper proposes several approaches to improve the collision attack on MD4 proposed by Wang et al. First, we propose a new local collision that is the best for the MD4 collision attack. Selection of a good message difference is the most important step in achieving effective collision attacks. This is the first paper to introduce an improvement to the message difference approach of Wang et al., where we propose a new local collision. Second, we propose a new algorithm for constructing differential paths. While similar algorithms have been proposed, they do not support the new local collision technique. Finally, we complete a collision attack, and show that the complexity is smaller than the previous best work.

keywords: Hash Function, Collision Attack, MD4, Local Collision, Message Difference, Differential Path

1 Introduction

Hash functions play an important role in modern cryptology. Hash functions must hold the property of *collision resistance*. This means that it must be computationally hard to find a pair of messages M and M' such that $H(M) = H(M')$ and $M \neq M'$.

MD4 is a hash function that was proposed by Rivest in 1990 [5]. MD4 has the Merkle-Damgård structure, and is designed for fast calculation. MD4 has been used to design other hash functions such as MD5 or SHA-1, which are widely used today. Therefore, cryptanalysis on MD4 is important since it affects the cryptanalysis of other hash functions based on MD4.

Several papers have found and reported the weaknesses of MD4. In 1996, the first collision attack was proposed by Dobbertin [1]. This attack finds a collision with probability of 2^{-22}. In 1998, Dobbertin pointed out that the first two rounds of MD4 did not have an property of one way. In 2004, at the CRYPTO rump session, Wang presented that collisions of MD4 could be generated very rapidly [7]. In 2005, Wang et al. reported the details of this attack which finds a collision with probability of 2^{-2} to 2^{-6}, and its complexity is less than 2^8 MD4 operations. This attack is highly efficient and many papers have suggested further improvements to this attack. In 2005, Naito et al. presented an improved attack [4]. They pointed out the mistakes of the sufficient conditions detailed in [8], and made improvements to the message modification techniques. This attack finds a collision with complexity less than 3 MD4 operations, and it is the fastest approach up to this time. In 2006, Schläffer and Oswald [6] analyzed how message differences given by [8] worked, and proposed an automated differential path search algorithm. As a result, they found a differential path that needed fewer sufficient conditions than Wang et al.'s.

An important characteristic of previous attacks is that they all used the same local collision, i.e. the message differences and differential path as proposed by Wang et al. In [8], the authors claim that their message differences, which are derived from their local collision, are optimized for collision attack. However, our research shows that local collision of [8] is not optimized.

For collision attacks, selecting good message differences is very important. Message differences are derived from a local collision, so selecting a good local collision is an important aspect of this work. If better local collisions or message differences are found, attack complexity can be drastically reduced. Moreover, an effective way to find a good local collision or message difference for MD4 may be applicable to other hash functions. This fact motivated our research on MD4.

This paper makes two contributions.

1. We propose a new local collision and new message differences that appear to be the best for collision attack on MD4. We use less than one half of the non-negligible sufficient conditions for new differences needed by Wang et al.'s approach.
2. We show that the differential path construction algorithm proposed by [6] does not work for the new local collision technique. Therefore, we analyze the problems of [6], and develop a new algorithm.

The improved collision attack is realized by combining the above two contributions and the message modification technique described by [3, 4]. Since local collision and message differences are improved, we can find a collision with complexity less than 2 MD4 operations, which is the fastest of all existing attacks. We show the new local collision in the right side of Table 1 and Figure 2.2, new message differences and differential path in Table 7, new sufficient conditions in Table 8, and all message modification procedures in Table 9 to Table 20; an example of a collision generated by our attack is shown in Table 4.

The organization of this paper is as follows. In section 2, we explain the specification of MD4 and related works. In section 3, we explain why the local collision of [8] is not best, and propose a new local collision for MD4. In section 4, we explain why the algorithm of [6] does not support the new local collision, and propose a new algorithm that constructs differential paths for the first round of MD4. Section 5 completes the improved attack, and compares its efficiency to previous works. Finally, we conclude this paper in section 6.

2 Preliminaries

2.1 Specification of MD4 [5]

MD4 input is an arbitrary length message M, and MD4 output is 128-bit data $H(M)$. MD4 has a Merkle-Damgård structure. First, the input message is padded to be a multiple of 512 bits. Since padding does not affect collision attack, we omit its explanation. The padded message M^* is divided into 512-bit messages $M^* = (M_0, \ldots, M_{n-1})$. The initial value (IV) for the hash value is set to be $H_0 = (0x67452301, 0xefcdab89, 0x98badcfe, 0x10325476)$. The output of the compression function H_1 is calculated using M_0 and H_0. In this paper, we call the calculation performed in a single run of the compression function 1 block. Next, H_2 is calculated using M_1 and H_1. Similarly, the compression function is calculated until the last message M_{n-1} is used. Let H_n be the hash value of M.

Compression Function of MD4

All calculations in the compression function are 32-bit. We omit the notation of "mod 2^{32}". The input to the compression function is a 512-bit message M_j and a 128-bit value H_j. First, M_j is divided into $(m_0, \ldots m_{15})$, where each m_i is a 32-bit message. The compression function consists of 48 steps. Steps 1-16 are called the first round (1R). Steps 17-32 and 33-48 are the second and third rounds (2R and 3R), respectively. In step i, chaining variables $a_i, b_i, c_i, d_i (1 \leq i \leq 48)$ are updated by the following expression (a_0, b_0, c_0, d_0 are the IV).

$a_i = d_{i-1}$,
$b_i = (a_{i-1} + f(b_{i-1}, c_{i-1}, d_{i-1}) + m_k + t) \lll s_i$,
$c_i = b_{i-1}$,
$d_i = c_{i-1}$,

where f is a Boolean function defined in each round, m_k is one of $(m_0, \ldots m_{15})$, and index k is defined in each step. If $m_0, \ldots m_{15}$ are fixed, all other $m_i (16 \leq i \leq 47)$ are also fixed. In this paper we call this "Message Expansion". t is a constant number defined in each round, $\lll s_i$ denotes left rotation by s_i bits, and s_i is defined in each step. Details of f and t are as follows.

$$1R : t = 0x00000000, f(X, Y, Z) = (X \wedge Y) \vee (\neg X \wedge Z),$$
$$2R : t = 0x5a827999, f(X, Y, Z) = (X \wedge Y) \vee (Y \wedge Z) \vee (X \wedge Z),$$
$$3R : t = 0x6ed9eba1, f(X, Y, Z) = X \oplus Y \oplus Z.$$

After 48 steps are calculated, H_{j+1} is calculated as follows.

$aa_0 \leftarrow a_{48} + a_0$, $\quad bb_0 \leftarrow b_{48} + b_0$,
$cc_0 \leftarrow c_{48} + c_0$, $\quad dd_0 \leftarrow d_{48} + d_0$,
$H_{j+1} \leftarrow (aa_0 | bb_0 | cc_0 | dd_0)$.

2.2 Related Work 1: Collision Attack by Wang et al. and its Improvement by Previous Works

The first collision attack on MD4 was proposed by Wang et al. in 2005. Many researchers have improved this attack since its publication. Our paper also improves this attack. In this section, we explain the attack procedure and which part of the attack has been improved so far.

This attack is a differential attack, and generates a collision with complexity less than 2^8 MD4 operations. Let m and m' be a pair of messages that yield a collision. Difference Δ is defined to be the value yielded by subtracting value for

m from value for m'. For example, $\Delta m = m' - m$. The attack procedure is as follows.

1. Find the "Message Difference (ΔM)" that yields a collision with high probability.
2. Determine how the impact of ΔM propagates. The propagation of this difference is called the "Differential Path (DP)."
3. Generate "Sufficient Conditions (SC)" for realizing the differential path on the value of chaining variables for calculating the hash value of m.
4. Determine the procedures of "Message Modification (MM)" that satisfy sufficient conditions.
5. Locate a message that satisfies all sufficient conditions by randomly generating messages and applying Message Modification. Let such a message be M_*.
6. Calculate $M'_* = M_* + \Delta M$. Finally, M_* and M'_* become a collision pair.

ΔM of Wang et al.'s attack is as follows.

$$\Delta M = M' - M = (\Delta m_0, \Delta m_1, \ldots, \Delta m_{15}),$$
$$\Delta m_1 = 2^{31}, \Delta m_2 = 2^{31} - 2^{28}, \Delta m_{12} = -2^{16}, \Delta m_i = 0, 0 \leq i \leq 15, i \neq 1, 2, 12.$$

[8] introduces two kinds of message modification: Single-Step Modification and Multi-Step Modification. Later, these names were changed to Basic Message Modification (BMM) and Advanced Message Modification (AMM). They are related as follows.

- Basic message modification is the technique that can satisfy all sufficient conditions in the first round with probability of 1.
- Advanced message modification can satisfy several sufficient conditions in the second round. (If the sufficient conditions exist in an early step of the second round, they may be satisfied by advanced message modification. On the other hand, if they exist in a later step of the second round, it's almost impossible to satisfy them.)
- No technique is known to satisfy any sufficient conditions in the third round.

For details about the differential path, sufficient condition and message modification, please refer to [8].

Strategy of Selecting ΔM

The strategy of selecting ΔM is explained by Schläffer and Oswald [6]. They showed that ΔM was determined by the Local Collision (LC) in the third round, although the reason was not explained. They then showed how Wang et al.'s local collision worked. We show the local collision of Wang et al. in the left side of Table 1, and its diagram in Figure 2.2.

Improvement of Collision Attack

So far, several papers [4, 6] have improved the collision attack proposed by Wang et al. Naito et al. [4] pointed out the sufficient condition mistakes in [8], and made

Table 1. Comparison of Wang et al.'s Local Collision and Ours

Wang et al.'s Local Collision					
Step	Δa_k	Δb_k	Δc_k	Δd_k	Δm_{k-1}
i		2^{31}			2^{31-s_i}
$i+1$		2^{31}	2^{31}		$2^{31-s_{i+1}}, 2^{31}$
$i+2$			2^{31}	2^{31}	
$i+3$	2^{31}			2^{31}	
$i+4$	2^{31}				
$i+5$					2^{31}

New Local Collision					
Step	Δa_k	Δb_k	Δc_k	Δd_k	Δm_{k-1}
i		2^{31}			2^{31-s_i}
$i+1$			2^{31}		2^{31}
$i+2$				2^{31}	2^{31}
$i+3$	2^{31}				2^{31}
$i+4$					2^{31}

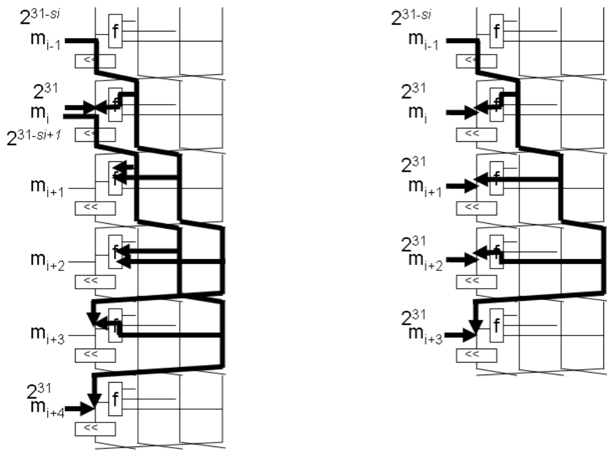

Fig. 1. Diagram of Wang et al.'s Local Collision and Ours

improvements to message modification. They used the same ΔM, differential path and sufficient condition as given by [8], but proposed advanced message modification that could satisfy all sufficient conditions in the second round and quickly satisfy sufficient conditions in the third round[1]. Their attack finds a collision of MD4 with complexity less than 3 MD4 operations. Schläffer and Oswald [6] proposed an algorithm that could construct differential paths when ΔM was given. They used the same ΔM as given by [8], and showed a differential path that needed fewer sufficient conditions than Wang et al.'s.

For Wang et al.'s attack, selection of ΔM gives the most significant impact to the attack complexity. However, no paper has attempted to improve the ΔM of Wang et al. In this paper, we propose new ΔM in section 3.

[1] This technique is restarting collision search from an intermediate step, which is different from message modification for the third round.

2.3 Related Work 2: Differential Path Search by Schläffer and Oswald [6]

At FSE06, Schläffer and Oswald [6] proposed an algorithm on how to construct differential paths for MD4 collision attack, when message differences are given. They used Wang et al.'s message differences. We show this algorithm can not be applied to the new ΔM (see section 4.2 and 4.3). In this section, we explain their algorithm as follows.

1. Calculate "Target Differences." In step i, target differences mean differences that are used to cancel Δm in steps $i + 4k(k = 0, \pm 1, \pm 2 \cdots)$.
2. Determine the actual output differences of f function. f function can not produce differences that are not elements of target differences. Resulted differential path usually has some contradictions in sufficient conditions.
3. Resolve contradictions in the sufficient conditions to construct a differential path.

For more details, please refer to [6].

3 New Local Collision

As mentioned in section 2.2, no paper has proposed a better ΔM than that of Wang et al. In this section, we propose a superior local collision and ΔM that are the best for collision attack on MD4.

3.1 Importance of Selecting ΔM

[6] determined ΔM was set by local collision in the third round, however, they didn't mention a reason for this. This motivated us to start with an explanation.

Thanks to the various message modification techniques which are available to us, following the differential path during the first two rounds (32 steps) of MD4 has very low cost. The computational cost of the attack comes from the probability of following the path during the third round. As a consequence, in order to get a more efficient collision attack on MD4, minimizing the number of sufficient conditions in the third round is the most important. Therefore, we start by looking for a very efficient local collision in the third round.

3.2 Problem of Wang et al.'s Local Collision and Constructing New Local Collision

In order to minimize the number of sufficient conditions in the third round, the method of Wang et al. applies local collision there. Their local collision is shown in left side of Table 1. They insert single message difference to step i in the third round, and insert other message differences to minimize the impact of the propagation of the inserted difference. As a result, the inserted difference is cancelled in following 6 steps. In Wang et al.'s local collision, two differences are made in the MSB of b_i and b_{i+1} for ensuring that differences don't propagate in step $i+2$, $i+3$ and $i+4$ (obviates the need to insert other message differences). Here, due to the addition and left rotation, the following fact is achieved. (We show the proof of this in Appendix A.)

Fact: To make $\Delta b_i = \pm 2^{31}$ from $\Delta m_k = \pm 2^{31-s_i}$, we need the following sufficient condition:

$$\begin{cases} b_{i,31} = 0 & (\text{if } \Delta m_k = +2^{31-s_i}) \\ b_{i,31} = 1 & (\text{if } \Delta m_k = -2^{31-s_i}) \end{cases}$$

Wang et al.'s local collision makes two differences. Therefore, it needs two sufficient conditions to realize the local collision.

In a collision attack on MD4, at least one message difference in the third round is necessary since collision messages must be different. Considering the Fact, it is obvious that if a local collision yielding one difference in the MSB of b_i can be constructed, that local collision is the best.

3.3 Construction of New Local Collision

We construct a new local collision by making only one difference in the MSB of b_i. The new local collision is summarized in right side of Table 1 and Figure 2.2. In step i, we make $\Delta b_i = \pm 2^{31}$ by inserting $\pm 2^{31-s_i}$ on m_{i-1}. From the Fact, we need one sufficient condition to make $\Delta b_i = \pm 2^{31}$. In step $i+1$, $\pm 2^{31}$ of b_i propagates through function $f = (b_i \oplus c_i \oplus d_i)$. Therefore, we insert $\pm 2^{31}$ on m_i in order to cancel $\pm 2^{31}$ from f. A similar situation occurs in step $i+2$ and $i+3$, so we insert $\pm 2^{31}$ on m, to cancel the difference that propagates from f. In step 4, we cancel $\Delta a_{i+3} = \pm 2^{31}$ by $\Delta m_{i+3} = \pm 2^{31}$, and finally all differences are cancelled. (Since all Δm are in MSB, any sign for Δm is acceptable.)

3.4 New Message Difference

As described in section 3.3, the new local collision makes only one difference, therefore, the new local collision needs only one sufficient condition in the third round, and this is the best local collision for MD4. In order to complete the entire collision attack, the impact of the message expansion on the previous rounds needs to be considered. The local collision constructed in section 3.3 does not fix step i that is the initial step of the local collision. Therefore, we analyze the impact of message expansion, and select the best i that minimizes the impact on the previous rounds.

We show details of this analysis in Appendix B. As a result, we found that $i = 33$ is the best, and obtained following message differences.

$$\Delta m_0 = 2^{28}, \Delta m_2 = 2^{31}, \Delta m_4 = 2^{31}, \Delta m_8 = 2^{31}, \Delta m_{12} = 2^{31},$$
$$\Delta m_i = 0 \text{ for } i = 1, 3, 5, 6, 7, 9, 10, 11, 13, 14, 15.$$

The differential path for the third round is also determined by local collision. It is shown in Table 7 in Appendix C.

Remaining work is to construct a differential path for the first and second round. In the first round, all sufficient conditions are satisfied by basic message modification, whereas, some of sufficient conditions in the second round may not be satisfied by advanced message modification. Therefore, we should minimize the number of sufficient conditions in the second round even if the differential path for the first round becomes more complicated. Since constructing the differential path for the second round is simple, we show only the result in Table 7 in Appendix C.

4 Differential Path Search Algorithm

In this section, we describe the DP construction algorithm, denoted as DPC.

4.1 Definition of Good DPC

Input of DPC is $\varDelta M$. A good DPC should not just work for one specified $\varDelta M$. It should be able to work for all $\varDelta M$. Moreover, it should be finished in reasonable time. If a DPC is able to work for all $\varDelta M$ in reasonable time, we will call it "Good DPC."

Crucial technique to Realize Good DPC
For DPC, how to control difference propagation through f function is important. Therefore, in design of good DPC, it is crucial to control difference propagation through f function to search DP in a large space efficiently.

4.2 Overview of Problems of Previous Work [6] and Our Improvement

Schläffer and Oswald proposed an algorithm to construct differential paths, based on Wang et al.'s message differences. Unfortunately the algorithm can not work for our new message differences. Therefore, we will propose a new DPC, which is able to work for our new $\varDelta M$. We will give an overview of comparison of these two algorithms.

Problem of Schläffer and Oswald's Algorithm
By Schläffer and Oswald's algorithm, the differences produced by f function in step i can only be used in step $i+4k$, and no others. Although their rule reduced the search space considerably, they found, fortunately, a better differential path than Wang et al's algorithm. However, reduced search space is too small, and consequently, their algorithm can not work for our new message differences.

Advantage of Our Algorithm
In our algorithm, we found a way to control difference propagation through f function efficiently in larger search space.

Important improvement: for step i, Schläffer and Oswald's algorithm only produces difference of f to cancel ΔM in step $i+4k$. By our algorithm, besides this kind of difference, f can also produce difference to guarantee that ΔM in other steps can be cancelled.

The search space of our algorithm is larger, which makes our algorithm work for the new ΔM proposed in section 3.

4.3 Details of Our New Proposed Algorithm for the First Round

The goal of our algorithm is to construct a differential path for the first round with given chaining variable differences after step 16 and given message differences. Our algorithm consists of three major parts: Forward search, Backward search and Joint algorithm.

i. Forward search

Forward search is done from step 1 to step 4. As mentioned in section 4.1, the crucial technique for making good DPC is controlling difference propagation through f function in a large search space. In forward search, we exhaustively search the possibility of difference through f. After forward search is finished, there are possible differences for chaining variables after step 4. Possible differences for a_4, b_4, c_4, d_4 are called "Potential Differences."

ii. Backward search

First of all, we will redefine the concept "Target Difference" for backward search. Backward Search is done from step 16 to step 8.

Target Differences: for step i, target differences Δt_i are calculated by differences of chaining variable b_i and message differences Δm_{i-1} as follows:

$$\Delta t_i = -\Delta m_{i-1} + (\Delta b_i \ggg s_i)(8 \leq i \leq 16)$$

Chaining variable differences after step 16 are already fixed, in other words, differences of $b_i(12 \leq i \leq 16)$ are fixed, it is easy to calculate target differences from step 16 to step 12. In step i, a target difference can be achieved by differences of a_{i-1} or f function. We need to determine each target difference which should be produced by a_{i-1} or f function. During backward search, we assume that differences of a_{i-1} and input differences of f function can always be produced in previous steps, so any target difference can be produced by a_{i-1}. This is main idea of Schläffer and Oswald's algorithm. We enlarge the search space by considering how to guarantee target differences produced by f function.

If we determine that a target difference will be achieved by f, input chaining variables b_{i-1}, c_{i-1}, or d_{i-1} should have a difference at the same position. If b_{i-1}, c_{i-1}, or d_{i-1} has a difference, it should be cancelled in the following step. If it can not be cancelled, it can not be used.

Considering that the chaining variable differences after step 16 are already fixed, from step 16 to step 12, it is very easy to check whether a difference of b_{i-1}, c_{i-1} or d_{i-1} can be cancelled in the following steps.

After backward search from step 16 to step 12 is done, there are some candidates for chaining variable differences in step 12. For every candidate, backward search from step 12 to step 8 is done by applying the same method.

After backward search from step 16 to 8 is done, there are some candidates for chaining variable differences in step 8, which are called "Aimed Differences."

iii. Joint algorithm

From step 4 to step 8, aimed differences will select potential differences. In the joint algorithm, only potential differences that match aimed differences can be produced by f function. Selected potential differences need to be cancelled in the following steps, To cancel these differences, carry can be used to expand input differences of f function. More differences can also be introduced.

After all the selected differences are canceled, that is, after the joint algorithm is finished, a differential path is constructed.

5 Attack Implementation

5.1 Message Modification

The results of Section 3 and 4 yield the new ΔM, differential path and sufficient conditions. To complete the collision attack, what remains is to propose advanced message modification procedures for satisfying sufficient conditions in the second and third round. Therefore, we propose advanced message modification, and evaluate the total attack complexity of proposed attack. Papers [3, 4] give some good ideas of advanced message modification. Since proposing advanced message modification is outside the scope of this paper, we show only the procedures of advanced message modification in Table 9 to Table 20 in Appendix D. As a result of applying advanced message modification, all sufficient conditions in the second round are satisfied with very high probability. Furthermore, by applying the technique called "Shortcut Modification" by [4] or "Tunnel" by [3], the sufficient condition in the third round is also quickly satisfied.

5.2 Attack Procedure and Complexity Estimation

The attack procedure is shown in Table 2. (Note extra conditions are set for advanced message modification. These are introduced in Appendix D.)

For line 1 to 5: Computation for m_i needs almost 1 MD4 step. Therefore, for line 4, we need 16 MD4 steps. Line 2 and 3 can be done very quickly compared to line 4.

For line 6 to 13: For line 7, computation for b_i is 1 MD4 step. Therefore, for line 7, we need 13 MD4 steps. For line 10, there are 11 message modification procedures. According to our evaluation, each procedure requires less than 3 MD4 steps (See Table 9 to 19 for details). Since each condition is satisfied with probability 1/2, we expect half of the message modification procedures are executed. Therefore, we need $3 \times 11 \div 2 = 16.5$ MD4 steps.

Table 2. Attack Procedure

1. **for** $(1 \le i \le 16)$ {
2. Randomly take the value of b_i.
3. Change b_i to satisfy all sufficient conds and extra conds for b_i.
4. Compute $m_{i-1} = (b_i \ggg s_i) - a_{i-1} - f(b_{i-1}, c_{i-1}, d_{i-1}) - t$.
5. }
6. **for** $(17 \le i \le 29)$ {
7. Compute b_i in standard way.
8. **for** $(0 \le j \le 31)$ {
9. **if** (sufficient conds or extra conds on $b_{i,j}$ is not satisfied) {
10. Do message modification described in Appendix D.
11. }
12. }
13. }
14. Compute b_{30} to b_{33} in standard way.
15. **if** ("$b_{33,31} = 0$" is not satisfied) {
16. Do message modification for "$b_{33,31} = 0$", and **goto** line 14.
17. }
18. Let the present message be M. Finally, M and $M + \Delta M$ become a collision pair.

For line 14 to 18: Line 14 needs 4 steps of MD4 operations. Then, if "$b_{33,31} = 0$" is not satisfied, we apply message modification. According to our evaluation, modifying message for "$b_{33,31} = 0$" takes less than 1 MD4 step, and the probability that this condition is satisfied after repeating line 14 to 16 is 1/2. Therefore, this condition is expected to be satisfied at most 2 trial. Complexity of satisfying in the first trial is 4 MD4 steps and by the second trial is $4 + 4 + 1 = 9$ MD4 steps. Therefore, average complexity is $(4 + 9) \div 2 = 6.5$ MD4 steps. Complexity for line 18 is negligible.

Total complexity: Total complexity is addition of above three parts, that is, 16+13+16.5+7.5=53 MD4 steps. Considering other trivial complexity, we evaluate that the complexity of our attack is less than 2 MD4 operations(=96 steps).

5.3 Comparison with Previous Attacks

The efficiencies of the proposed attack and previous attacks are compared in Table 3.

From Table 3, it can be said that improvement of local collision is very important. With the new local collision, number of sufficient conditions in the second and third round becomes less than half those of previous attacks, and this enables us to further reduce the complexity of generating a collision.

Table 3. Comparison of Previous Attacks and Our Result

Method	ΔM	#SCs in 2R	#SCs in 3R	Total complexity (Unit MD4 operations)
Wang et al.	Wang's ΔM	25	2	Less than 2^8
Naito et al.	Wang's ΔM	25	2	Less than 3
Schläffer and Oswald	Wang's ΔM	22	2	-
Proposed attack	ΔM for new LC	9	1	Less than 2

6 Conclusion

This paper proposed a new local collision that is the best for collision attack on MD4. This was achieved by the first improvement to the message difference approach of Wang et al.

For successful collision attacks, ΔM must be carefully selected to reduce the number of sufficient conditions that cannot be satisfied by message modification. In this paper, we showed how efficient the proposed improvement to ΔM is. With the new ΔM, number of sufficient conditions in the second and third round is less than half those of previous studies; attack complexity is also improved.

Improving the attack on MD4 is not the final goal. As future works, we will apply our analysis to other hash functions, and aim to find more efficient ΔM.

We realized that differential path construction algorithm proposed by [6] did not work with the new ΔM. Therefore, we analyzed the problems of [6], and proposed a new algorithm for constructing differential path for the first round.

Finally, we show a collision generated by using the new local collision in Table 4. We note that ΔM of this example is different from all previous works.

Table 4. An Example of MD4 Collision Generated with New Local Collision

M	m_0=0x$bcdd$2674 m_1=0x53fce1ed m_2=0x25d202ce m_3=0xe87d102e m_4=0xf45be728 m_5=0xacc992cc m_6=0x6$acfb$3ea m_7=0x7dbb29d4 m_8=0xed03bf75 m_9=0xc6$aedc$45 m_{10}=0xd442b710 m_{11}=0xfca27d99 m_{12}=0xa5f5eff1 m_{13}=0xfb2ee79b m_{14}=0x0f590d68 m_{15}=0x4989f380
M'	m'_0=0x$ccdd$2674 m'_1=0x53fce1ed m'_2=0xa5d202ce m'_3=0xe87d102e m'_4=0x745be728 m'_5=0xacc992cc m'_6=0x6$acfb$3ea m'_7=0x7dbb29d4 m'_8=0x6d03bf75 m'_9=0xc6$aedc$45 m'_{10}=0xd442b710 m'_{11}=0xfca27d99 m'_{12}=0x25f5eff1 m'_{13}=0xfb2ee79b m'_{14}=0x0f590d68 m'_{15}=0x4989f380
Hash	0x c257b7be 324f26ef 69d3d290 b01be001

Acknowledgements

We would like to thank Antoine Joux for improving our paper and anonymous reviewers for helpful comments.

References

1. H. Dobbertin. *Cryptanalysis of MD4*, Fast Software Encryption (FSE) 1996, LNCS 1039, pp53-69, Springer-Verlag, 1996
2. H. Dobbertin. *The First Two Rounds of MD4 are Not One-Way*, Fast Software Encryption (FSE) 1998, LNCS 1372, pp284-292, Springer-Verlag, 1998
3. V. Klima, *Tunnels in Hash Functions: MD5 Collisions Within a Minute*. Cryptology ePrint Archive, Report 2006/105.
4. Y. Naito, Y. Sasaki, N. Kunihiro and K. Ohta, *Improved Collision Attack on MD4 with Probability Almost 1*. Information Security and Cryptology (ICISC) 2005, LNCS 3935, pp129-145, Springer-Verlag, 2006.
5. R. Rivest. *The MD4 Message Digest Algorithm*. CRYPTO 1990, LNCS 537, pp303-311, Springer-Verlag, 1991, http://www.ietf.org/rfc/rfc1320.txt
6. M. Schläffer and E. Oswald. *Searching for Differential Paths in MD4*. Fast Software Encryption (FSE) 2006, LNCS 4047, pp242-261, Springer-Verlag, 2006.
7. X. Wang, D. Feng, H. Chen, X. Lai and X. Yu. *Collision for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD*. In Rump Session of CRYPTO 2004 and Cryptology ePrint Archive, Report 2004/199.
8. X. Wang, X. Lai, D. Feng, H. Chen and X. Yu. *Cryptanalysis of the Hash Functions MD4 and RIPEMD*. Eurocrypt 2005, LNCS 3494, pp1-18, Springer-Verlag, 2005.

A Proof of the Fact in Section 3.2

Proof. Remember that the calculation of b_i in step i is as follows.

$b_i = (a_{i-1} + f(b_{i-1}, c_{i-1}, d_{i-1}) + m_k + t) \lll s_i$

Here, we make $\Delta b_i = 2^{31}$ or -2^{31} by adding the difference $\Delta m_k = +2^{31-s_i}$ or -2^{31-s_i}. Let the value of $a_{i-1} + f(b_{i-1}, c_{i-1}, d_{i-1}) + m_k + t$ be Σ. We focus on the value of Σ in bit position $31 - s_i$ (In short, Σ_{31-s_i}). Σ_{31-s_i} is 0 or 1. First, we consider the case that Σ_{31-s_i} is 0. In this case, if Δm_k is positive, Σ_{31-s_i} changes from 0 to 1, and other bits are kept unchanged. Therefore, it becomes $\Delta b_i = +2^{31}$ after the rotation. However, if Δm_k is negative, Σ_{31-s_i} changes from 0 to 1, and bit position $31 - s_i + 1$ also changes because of carry. Here, assume that carry is stopped in bit position $31 - s_i + 1$, therefore, the value in bit position $31 - s_i + 1$ changes from 1 to 0, and other bits are unchanged. In this case, after the rotation, it becomes $\Delta b_i = +2^{31} - 2^0$, and thus, we cannot obtain the desired difference. The same analysis is applied in the case of Σ_{31-s_i} is 1. The result of the analysis is summarized below.

Table 5. Summary of Proof

	$\Delta m_k = +2^{31-s_i}$	$\Delta m_k = -2^{31-s_i}$
$\Sigma_{31-s_i} = 0$	Success	Failure
$\Sigma_{31-s_i} = 1$	Failure	Success

Consider the fact that $\Sigma_{31-s_i} = b_{i,31}$, we get following conclusion; To make success, $b_{i,31} = 0$ if $\Delta m_k = +2^{31-s_i}$, or $b_{i,31} = 1$ if $\Delta m_k = -2^{31-s_i}$. (Q.E.D)

B Selecting ΔM by Considering Message Expansion

As described in section 3.3, the new local collision makes one difference in the third round. This does not fix step i that is the initial step of the local collision. Therefore, we analyze the impact of message expansion, and select the best i that minimizes the impact on the previous rounds.

However, thanks to the basic message modification, even if the differential path becomes complicated and many sufficient conditions are required, all sufficient conditions in the first round can be satisfied with probability 1. Therefore, we only have to care about the impact on the second round.

The third round consists of step 33-48. Since the new local collision takes 5 steps to realize cancellation, we have 12 choices for the initial step of local collision ($IniLC$), ($33 \leq IniLC \leq 44$). To determine a good $IniLC$, we consider following characteristics of the second round.

- We need to cancel all differences by the final step of the second round since local collision in the third round must start from no-difference state.
- Let the last step with some message difference in the second round be *cp*, which stands for Cancellation Point. If all differences are cancelled at step *cp*, remaining steps of the second round are guaranteed to be no-difference without sufficient conditions.

Therefore, in order to reduce number of sufficient conditions in the second round, *cp* should be as early a step as possible. Finally, we select $IniLC$ to minimize *cp*. We show the analysis of message expansion in Table 6. In Table 6, gray highlight shows an example of the analysis for $IniLC = 33$. Messages that have differences on the second round and third round are colored. From the table, we see that *cp* for $IniLC = 33$ is 25. The same analysis is applied for all $IniLC$. *cp* for all $IniLC$ are listed in the bottom of Table 6.

Table 6. Analysis of Impact of Message Expansion

Message index for each step in 2R and 3R

2R	step number	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
	message index	0	4	8	12	1	5	9	13	2	6	10	14	3	7	11	15

3R	step number	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
	message index	0	8	4	12	2	10	6	14	1	9	5	13	3	11	7	15

The value of *cp* for all $IniLC$

$IniLC$	33	34	35	36	37	38	39	40	41	42	43	44
cp	25	27	27	28	28	28	28	28	29	31	31	32

From Table 6, finally, we find that $IniLC = 33$ is the best since its *cp* is the smallest. The obtained ΔM and differential path for the third round is shown in Table 7 in Appendix C[2].

[2] We also calculate the good ΔM for Wang et al.'s local collision by using this analysis. Then, we found that the resulting ΔM is the same as proposed by Wang et al.'s.

C Differential Path and Sufficient Conditions

Table 7. Differential Path of MD4 for New Local Collision

Step i	Shift s_i	Δm_{i-1}	Δb_i Numerical difference	Δb_i Difference in each bit
1	3	2^{28}	-2^{31}	$\Delta[-31]$
			-2^{0}	$\Delta[0,-1]$
2	7		-2^{7}	$\Delta[7]$
			-2^{8}	$\Delta[-9]$
3	11	2^{31}	-2^{11}	$\Delta[-11]$
4	19		-2^{19}	$\Delta[19,20,-21]$
5	3	2^{31}	2^{3}	$\Delta[3]$
			2^{22}	$\Delta[-22,-23,24]$
6	7		-2^{18}	$\Delta[18,19,-20]$
			2^{31}	$\Delta[31]$
7	11		2^{1}	$\Delta[-1,\cdots,9]$
			-2^{22}	$\Delta[22,23,-24]$
			2^{31}	$\Delta[31]$
8	19			
9	3	2^{31}	2^{22}	$\Delta[-22,23]$
			2^{25}	$\Delta[25]$
10	7		-2^{12}	$\Delta[-12]$
			-2^{25}	$\Delta[25,26,-27]$
			2^{29}	$\Delta[-29,-30,31]$
11	11		2^{12}	$\Delta[12]$
12	19		-2^{12}	$\Delta[-12]$
13	3	2^{31}	2^{25}	$\Delta[25]$
			-2^{28}	$\Delta[28,-29]$
14	7			
15	11			
16	19		-2^{31}	$\Delta[-31]$
17	3	2^{28}	2^{28}	$\Delta[28]$
18	5	2^{31}		
19	9	2^{31}		
20	13	2^{31}		
21	3		2^{31}	$\Delta[31]$
22	5			
23	9			
24	13			
25	3	2^{31}		
$\cdots$				
33	3	2^{28}	2^{31}	$\Delta[31]$
34	9	2^{31}		
35	11	2^{31}		
36	15	2^{31}		
37	3	2^{31}		

The symbol '$\Delta[i]$' means the value of chaining variable in bit position i changes from 0 to 1 by difference. The symbol '$\Delta[-i]$' means it changes from 1 to 0 instead.

Table 8. Sufficient Conditions and Extra Conditions for New Local Collision

Chaining variables	Conditions on bits			
	31 - 24	23 - 16	15 - 8	7 - 0
b_1	1 0̄ - - - - - -	ā - - - - - - -	- - - - - - a -	a - - - ā - 0 1
b_2	1 0̄ - 0̄ - ā - -	- - - - 0 - - -	- - - - a - 1 -	0 - - - - - 0 1
b_3	1 1̄ - 1̄ ā - ā 0	0̄ - a a 1 - ā ā	ā ā ā ā 1 ā 0 -	0 - - - 1̄ - 1 0
b_4	1 - ā - 1̄ 0̄ 1̄ 1	a a 1 0 0 - 0̄ 1̄	1̄ 1̄ 1̄ 1̄ 0 1̄ 1 a	1 a a a a - - -
b_5	a - - - 0̄ 0̄ 0̄ 0	1 1 0 0 0 a 1̄ 0̄	0̄ 0̄ 0̄ 0̄ 0 0̄ 0 1	1 1 1 1 1 - - -
b_6	0 - 0̄ - 1̄ - 1̄ 1	1 1 1 1 0 0 1̄ 1̄	1̄ 1̄ 1̄ 1̄ - 1̄ 0 0	0 0 0 0 0 a a -
b_7	0 - 1̄ - - - 1 1	0 0 - 0 1 0 - -	- - - - - - 0 1	1 1 1 1 1 1 1 -
b_8	1 - - - - - a 0	0 0 - 1 0 1 - -	- - - 0 - - 0 0	0 0 0 0 1 0 0 -
b_9	0 a a - a a 0 1	0 1 - - - - - -	- - - a - - 1 1	1 1 0 1 1 1 1 -
b_{10}	0 1 1 - 1 0 0 -	1 1 - - - ā - -	- - - 1 - 0̄ 0̄ -	- - - - - - - -
b_{11}	0 0 0 - 1 1 0 -	1 1 - - - - - -	- - - 0 - 1̄ 1̄ -	- - - - - - - -
b_{12}	0 1 1 a 0 0 1 -	- - - - - 1̄ - -	- - - 1 - 1̄ 0̄ -	- - - - - - - -
b_{13}	- - 1 0 - - 0 -	- - - - - 1̄ - -	- - - 0 - 0̄ 0̄ -	- - - - - - - -
b_{14}	- - 0 0 - - 0 -	- - - - ā - - -	- - - 0 - 1̄ 1̄ -	- - - - - - - -
b_{15}	a - 1 1 b̄ - 1 -	- - - - - - - b̄	- - - - - - - -	- - - - - - - -
b_{16}	1 - - a 0̄ - - -	- - - - c̄ - - -	- - - - - - - -	- - - - - - - -
b_{17}	b - - 0 - - - -	- - - - ā - - -	- - - - - - - -	- - - - - - - -
b_{18}	b - - c - - - -	- - - - - - - -	- - - - - - - -	- - - - - - - -
b_{19}	- - - a - - - -	- - - - - - - -	- - - - - - - -	- - - - - - - -
b_{20}	a - - - - - - -	- - - - - - - -	- - - - - - - -	- - - - - - - -
b_{21}	0 - - - - - - -	- b̄ - - - - - -	- - - - - - - -	- - - - - - - -
b_{22}	c - - - - - - -	- - - - - - - -	- - - - - - - -	- - - - - - - -
b_{23}	a - - - - - - -	- - - - - - - -	- - - - - - - -	- - - - - - - -
b_{24}	- - - - - - - -	- - - - - - - -	- - - - - - - -	- - - - - - - -
...				
b_{33}	0 - - - - - - -	- - - - - - - -	- - - - - - - -	- - - - - - - -
...				

The notation '0' stands for the conditions $b_{i,j} = 0$, the notation '1' stands for the conditions $b_{i,j} = 1$, the notation 'a' stands for the conditions $b_{i,j} = b_{i-1,j}$, 'b' stands for the condition $b_{i,j} \neq b_{i-1,j}$ and 'c' stands for the condition $b_{i,j} = b_{i-2,j}$. Conditions without upper bar are sufficient conditions for the new differential path. Conditions with upper bar are extra conditions that are introduced in Appendix D.

D List of Message Modification Procedures

Table 9 to Table 20 show the procedure for advanced message modification for each condition. These procedures are executed only if the target condition is not satisfied. Several procedures need to satisfy "extra conditions (EC)" in advance. ECs guarantee that each modification procedure is executed without breaking other sufficient conditions. ECs in the first round are set together with sufficient conditions in the first round by basic message modification.

We explain how these procedures work, by explaining Table 10 as an example. Before we execute Table 10, we check whether the target sufficient condition "$b_{17,28} = 0$" is satisfied or not. If it's satisfied, do nothing. Otherwise, execute procedures shown in Table 10. In the table, the first and second column show the rounds and steps where we are looking on. The third column shows the procedure to be executed. The last column shows how differences are used in each calculation. In case of Table 10, we first modify m_0. Then, by considering the expression in step 1, we modify b_1. We make sure that a carry doesn't occur in b_1. Therefore, if $b_{1,28} = 0$ we choose "+," otherwise, we choose "-." After that, to make sure that the result of step 2 keeps unchanged, we modify m_1. In 3rd step, from the property of f, if $b_{2,28}$ is fixed to be 0, the result of step 3 keeps unchanged. Therefore, we set the extra condition "$b_{2,28} = 0$." Step 4 and 5 are the almost same. After step 5 is calculated, all differences in the first round are cancelled. Since m_0 is modified, step 17 in the second round is also modified. From the expression of b_{17}, b_{17} has difference 2^{28}, and this difference flips the value of $b_{17,28}$. Therefore, the sufficient condition is satisfied.

In this section, "$b_i \leftarrow$ standard computation" means computing $b_i \leftarrow (a_{i-1} + f(b_{i-1}, c_{i-1}, d_{i-1}) + m_k + t) \lll s_i$. "$m_i \leftarrow$ inverse computation" means computing $m_i \leftarrow (b_i \ggg s_i) - a_{i-1} - f(b_{i-1}, c_{i-1}, d_{i-1}) - t$.

Table 9. Modification Procedure for Extra Condition "$b_{17,19} = c_{17,19}$"

Round	Step	Actual procedure	Expression with difference in each step
1R	16	$m_{15} \leftarrow m_{15} \pm 2^{29}$ $b_{16} \leftarrow b_{16} \pm 2^{16}$ (Take sign to avoid b_{16} carry.)	$b_{16} \leftarrow (a_{15} + f(b_{15}, c_{15}, d_{15}) + m_{15} \pm 2^{29} + t) \lll 19$
2R	17	EC:"$c_{16,16} \neq d_{16,16}$" $b_{17} \leftarrow$ standard computation	$b_{17} \leftarrow (a_{16} + f(b_{16}[\pm 16], c_{16}, d_{16}) + m_0 + t) \lll 3$

Table 10. Modification Procedure for "$b_{17,28} = 0$"

Round	Step	Actual procedure	Expression with difference in each step
1R	1	$m_0 \leftarrow m_0 \pm 2^{25}$ $b_1 \leftarrow b_1 \pm 2^{28}$ (Take sign to avoid b_1 carry.)	$b_1 \leftarrow (a_0 + f(b_0, c_0, d_0) + m_0 \pm 2^{25} + t) \lll 3$
	2	$m_1 \leftarrow$ inverse computation	$b_2 \leftarrow (a_1 + f(b_1[\pm 28], c_1, d_1) + m_1 + t) \lll 7$
	3	EC: "$b_{2,28} = 0$"	$b_3 \leftarrow (a_2 + f(b_2, c_2[\pm 28], d_2) + m_2 + t) \lll 11$
	4	EC: "$b_{3,28} = 1$"	$b_4 \leftarrow (a_3 + f(b_3, c_3, d_3[\pm 28]) + m_3 + t) \lll 19$
	5	$m_4 \leftarrow m_4 \mp 2^{28}$	$b_5 \leftarrow (a_4 \pm 2^{28} + f(b_4, c_4, d_4) + m_4 \mp 2^{28} + t) \lll 3$
2R	17	$b_{17} \leftarrow$ standard computation	$b_{17} \leftarrow (a_{16} + f(b_{16}, c_{16}, d_{16}) + m_0 \pm 2^{25} + t) \lll 3$

Table 11. Modification Procedure for "$b_{17,31} \neq b_{16,31}$"

Round	Step	Actual procedure	Expression with difference in each step
1R	1	$m_0 \leftarrow m_0 + 2^{27}$ $b_1 \leftarrow b_1 + 2^{30}$ EC: "$b_{1,30} = 0$"	$b_1 \leftarrow (a_0 + f(b_0, c_0, d_0) + m_0 + 2^{27} + t) \lll 3$
	2	$m_1 \leftarrow$ inverse operation	$b_2 \leftarrow (a_1 + f(b_1[+30], c_1, d_1) + m_1 + t) \lll 7$
	3	EC: "$b_{2,30} = 0$"	$b_3 \leftarrow (a_2 + f(b_2, c_2[+30], d_2) + m_2 + t) \lll 11$
	4	EC: "$b_{3,30} = 1$"	$b_4 \leftarrow (a_3 + f(b_3, c_3, d_3[+30]) + m_3 + t) \lll 19$
	5	$m_4 \leftarrow m_4 - 2^{30}$	$b_5 \leftarrow (a_4 + 2^{30} + f(b_4, c_4, d_4) + m_4 - 2^{30} + t) \lll 3$
1-2R	16	$m_{15} \leftarrow m_{15} + 2^8$ EC: "$b_{16,27} = 0$"	$b_{16} \leftarrow (a_{15} + f(b_{15}, c_{15}, d_{15}) + m_{15} + 2^8 + t) \lll 19$
	17	$b_{17} \leftarrow$ standard operation EC: "$c_{16,27} \neq d_{16,27}$"	$b_{17} \leftarrow (a_{16} + f(b_{16}[+27], c_{16}, d_{16}) + m_0 + 2^{27} + t) \lll 3$

Table 12. Modification Procedure for "$b_{18,28} = 0$"

Round	Step	Actual procedure	Expression with difference in each step
1R	2	$m_1 \leftarrow m_1 \pm 2^{16}$ $b_2 \leftarrow b_2 \pm 2^{23}$ (Take sign to avoid b_2 carry.)	$b_2 \leftarrow (a_1 + f(b_1, c_1, d_1) + m_1 \pm 2^{16} + t) \lll 7$
	3	EC: "$c_{2,23} = d_{2,23}$"	$b_3 \leftarrow (a_2 + f(b_2[\pm 23], c_2, d_2) + m_2 + t) \lll 11$
	4	EC: "$b_{3,23} = 0$"	$b_4 \leftarrow (a_3 + f(b_3, c_3[\pm 23], d_3) + m_3 + t) \lll 19$
	5	$m_4 \leftarrow m_4 \mp 2^{23}$ EC: "$b_{4,23} = 0$"	$b_5 \leftarrow (a_4 + f(b_4, c_4, d_4[\pm 23]) + m_4 \mp 2^{23} + t) \lll 3$
	6	$m_5 \leftarrow m_5 \mp 2^{23}$	$b_6 \leftarrow (a_5 \pm 2^{23} + f(b_5, c_5, d_5) + m_5 \mp 2^{23} + t) \lll 7$
2R	17	$b_{18} \leftarrow$ standard computation	$b_{18} \leftarrow (a_{17} + f(b_{17}, c_{17}, d_{17}) + m_4 \mp 2^{23} + t) \lll 5$

Table 13. Modification Procedure for "$b_{18,31} \neq b_{17,31}$"

Round	Step	Actual procedure	Expression with difference in each step
1R	5	$m_4 \leftarrow m_4 \pm 2^{26}$ $b_5 \leftarrow b_5 \pm 2^{29}$ (Take sign to avoid b_5 carry.)	$b_5 \leftarrow (a_4 + f(b_4, c_4, d_4) + m_4 \pm 2^{26} + t) \lll 3$
	6	EC: "$c_{5,29} = d_{5,29}$"	$b_6 \leftarrow (a_5 + f(b_5[\pm 29], c_5, d_5) + m_5 + t) \lll 7$
	7	EC: "$b_{6,29} = 0$"	$b_7 \leftarrow (a_6 + f(b_6, c_6[\pm 29], d_6) + m_6 + t) \lll 11$
	8	EC: "$b_{7,29} = 1$"	$b_8 \leftarrow (a_7 + f(b_7, c_7, d_7[\pm 29]) + m_7 + t) \lll 19$
	9	$m_8 \leftarrow m_8 \mp 2^{29}$	$b_9 \leftarrow (a_8 \pm 2^{29} + (f(b_8, c_8, d_8) + m_8 \mp 2^{29} + t) \lll 3$
2R	17	$b_{18} \leftarrow$ standard computation	$b_{18} \leftarrow (a_{17} + f(b_{17}, c_{17}, d_{17}) + m_4 \pm 2^{26} + t) \lll 5$

Table 14. Modification Procedure for "$b_{19,28} = b_{18,28}$"

Round	Step	Actual procedure	Expression with difference in each step
1-2R	15	$m_{14} \leftarrow m_{14} \pm 2^8$ $b_{15} \leftarrow b_{15} \pm 2^{19}$ (Take sign to avoid b_{15} carry.)	$b_{15} \leftarrow (a_{14} + f(b_{14}, c_{14}, d_{14}) + m_{14} \pm 2^8 + t) \lll 11$
	16	EC:"$c_{15,19} = d_{15,19}$"	$b_{16} \leftarrow (a_{15} + f(b_{15}[\pm 19], c_{15}, d_{15}) + m_{15} + t) \lll 19$
	17	EC:"$b_{16,19} = d_{16,19}$"	$b_{17} \leftarrow (a_{16} + f(b_{16}, c_{16}[\pm 19], d_{16}) + m_0 + t) \lll 3$
	18	EC:"$b_{17,19} = c_{17,19}$"	$b_{18} \leftarrow (a_{17} + f(b_{17}, c_{17}, d_{17}[\pm 19]) + m_4 + t) \lll 5$
	19	$b_{19} \leftarrow$ standard computation	$b_{19} \leftarrow (a_{18} \pm 2^{19} + f(b_{18}, c_{18}, d_{18}) + m_8 + t) \lll 9$

Table 15. Modification Procedure for "$b_{20,31} = b_{19,31}$"

round	step	Actual procedure	Expression with difference in each step
1R	11	$m_{10} \leftarrow m_{10} \pm 2^7$ $b_{11} \leftarrow b_{11} \pm 2^{18}$ take sign to avoid b_{11} carry	$b_{11} \leftarrow (a_{10} + f(b_{10}, c_{10}, d_{10}) + m_{10} \pm 2^7 + t) \lll 11$
	12	EC: "$c_{11,18} = d_{11,18}$"	$b_{12} \leftarrow (a_{11} + f(b_{11}[\pm 18], c_{11}, d_{11}) + m_{11} + t) \lll 19$
	13	$m_{12} \leftarrow m_{12} \mp 2^{18}$	$b_{13} \leftarrow (a_{12} + f(b_{12}, c_{12}[\pm 18], d_{12}) + m_{12} \mp 2^{18} + t) \lll 3$
	14	EC: "$b_{12,18} = 1$" EC: "$b_{13,18} = 1$"	$b_{14} \leftarrow (a_{13} + f(b_{13}, c_{13}, d_{13}[\pm 18]) + m_{13} + t) \lll 7$
	15	$m_{14} \leftarrow m_{14} \mp 2^{18}$	$b_{15} \leftarrow (a_{14} \pm 2^{18} + f(b_{14}, c_{14}, d_{14}) + m_{14} \mp 2^{18} + t) \lll 11$
2R	20	$b_{20} \leftarrow$ standard operation	$b_{20} \leftarrow (a_{19} + f(b_{19}, c_{19}, d_{19}) + m_{12} \mp 2^{18} + t) \lll 13$

Table 16. Modification Procedure for Extra Condition "$b_{21,22} \neq b_{20,22}$"

Round	Step	Actual procedure	Expression with difference in each step
1R	12	$m_{11} \leftarrow m_{11} \pm 2^{22}$ $b_{12} \leftarrow b_{12} \pm 2^9$ EC: "$b_{12,10} = 1$", "$b_{12,9} = 0$" (if sign is $+$, $b_3[9]$, else $b_3[-10, 9]$.)	$b_{12} \leftarrow (a_{11} + f(b_{11}, c_{11}, d_{11})+$ $+m_{11} \pm 2^{22} + t) \lll 19$
	13	$m_{12} \leftarrow m_{12} \mp 2^9$ (Take sign to avoid b_{20} carry.) EC: "$c_{12,10} = 1$", "$d_{12,10} = 0$", "$c_{12,9} = 1$", "$d_{12,9} = 0$"	$b_{13} \leftarrow (a_{12} + f(b_{12}[9]/[-10, 9], c_{12}, d_{12})$ $+m_{12} \mp 2^9 + t) \lll 3$
	14	EC: "$b_{13,10} = 0$", "$b_{13,9} = 1$"	$b_{14} \leftarrow (a_{13} + f(b_{13}, c_{13}[9]/[-10, 9], d_{13})$ $+m_{13} + t) \lll 7$
	15	EC: "$b_{14,10} = 1$", "$b_{14,9} = 1$"	$b_{15} \leftarrow (a_{14} + f(b_{14}, c_{14}, d_{14}[9]/[-10, 9])$ $+m_{14} + t) \lll 11$
	16	$m_{15} \leftarrow m_{15} \mp 2^9$	$b_{16} \leftarrow (a_{15} \pm 2^9 + f(b_{15}, c_{54}, d_{15})$ $+m_{15} \mp 2^9 + t) \lll 19$
2R	20	$b_{20} \leftarrow$ standard computation	$b_{20} \leftarrow (a_{19} + f(b_{19}, c_{19}, d_{19})$ $+m_{12} \mp 2^9 + t) \lll 13$
	21	$b_{21} \leftarrow$ standard computation ($b_{21,22}$ is never changed by this calculation).	$b_{21} \leftarrow (a_{20} + f(b_{20}[\mp 22], c_{20}, d_{20})$ $+m_1 + t) \lll 3$

Table 17. Modification Procedure for "$b_{21,31} = 0$"

Round	Step	Actual procedure	Expression with difference in each step
1R	2	$m_1 \leftarrow m_1 \pm 2^{28}$ $b_2 \leftarrow b_2 \pm 2^3$ (Take sign to avoid b_2 carry.)	$b_2 \leftarrow (a_1 + f(b_1, c_1, d_1) + m_1 \pm 2^{28} + t) \lll 7$
	3	EC: "$c_{2,3} = d_{2,3}$"	$b_3 \leftarrow (a_2 + f(b_2[\pm 3], c_2, d_2) + m_2 + t) \lll 11$
	4	$m_3 \leftarrow$ inverse computation	$b_4 \leftarrow (a_3 + f(b_3, c_3[\pm 3], d_3) + m_3 + t) \lll 19$
	5	EC: "$b_{4,3} = 1$"	$b_5 \leftarrow (a_4 + f(b_4, c_4, d_4[\pm 3]) + m_4 + t) \lll 3$
	6	$m_5 \leftarrow m_5 \mp 2^3$	$b_6 \leftarrow (a_5 \pm 2^3 + f(b_5, c_5, d_5) + m_5 \mp 2^3 + t) \lll 7$
2R	21	$b_{21} \leftarrow$ standard computation	$b_{21} \leftarrow (a_{20} + f(b_{20}, c_{20}, d_{20}) + m_1 \pm 2^{28} + t) \lll 3$

Table 18. Modification Procedure for "$b_{22,31} = b_{21,31}$"

Round	Step	Actual procedure	Expression with difference in each step
1R	3	$m_2 \leftarrow m_2 \pm 2^{15}$ $b_3 \leftarrow b_3 \pm 2^{26}$ (Take sign to avoid b_3 carry.)	$b_3 \leftarrow (a_2 + f(b_2, c_2, d_2) + m_2 \pm 2^{15} + t) \lll 11$
	4	EC: "$c_{3,26} = d_{3,26}$"	$b_4 \leftarrow (a_3 + f(b_3[\pm 26], c_3, d_3) + m_3 + t) \lll 19$
	5	EC: "$b_{4,26} = 0$"	$b_5 \leftarrow (a_4 + f(b_4, c_4[\pm 26], d_4) + m_4 + t) \lll 3$
	6	$m_5 \leftarrow m_5 \mp 2^{26}$ EC: "$b_{5,26} = 0$"	$b_6 \leftarrow (a_5 + f(b_5, c_5, d_5[\pm 26]) + m_5 + t) \lll 7$
	7	$m_6 \leftarrow m_6 \mp 2^{26}$	$b_7 \leftarrow (a_6 \pm 2^{26} + f(b_6, c_6, d_6) + m_6 \mp 2^{26} + t) \lll 11$
2R	21	$b_{22} \leftarrow$ standard computation	$b_{22} \leftarrow (a_{21} + f(b_{21}, c_{21}, d_{21}) + m_5 \mp 2^{26} + t) \lll 5$

Table 19. Modification Procedure for "$b_{23,31} = b_{22,31}$"

Round	Step	Actual procedure	Expression with difference in each step
1R	4	$m_3 \leftarrow m_3 \pm 2^{30}$ $b_4 \leftarrow b_4 \pm 2^{17}$ (Take sign to avoid b_4 carry.)	$b_4 \leftarrow (a_3 + f(b_3, c_3, d_3) + m_3 \pm 2^{30} + t) \lll 19$
	5	EC: "$c_{4,17} = d_{4,17}$"	$b_5 \leftarrow (a_4 + f(b_4[\pm 17], c_4, d_4) + m_4 + t) \lll 3$
	6	$m_5 \leftarrow m_5 \mp 2^{17}$ EC: "$b_{5,17} = 1$"	$b_6 \leftarrow (a_5 + f(b_5, c_5[\pm 17], d_5) + m_5 \mp 2^{17} + t) \lll 7$
	7	EC: "$b_{6,17} = 1$"	$b_7 \leftarrow (a_6 + f(b_6, c_6, d_6[\pm 17]) + m_6 + t) \lll 11$
	8	$m_7 \leftarrow m_7 \mp 2^{17}$	$b_8 \leftarrow (a_7 \pm 2^{17} + f(b_7, c_7, d_7) + m_7 \mp 2^{17} + t) \lll 19$
2R	22	$b_{22} \leftarrow$ standard computation (This breaks SC on $b_{22,31}$ with probability 2^{-9}.)	$b_{22} \leftarrow (a_{21} + f(b_{21}, c_{21}, d_{21}) + m_5 \mp 2^{17} + t) \lll 5$
	23	$b_{23} \leftarrow$ standard computation EC: "$c_{22,22} \neq d_{22,22}$"	$b_{23} \leftarrow (a_{22} + f(b_{22}[\mp 22], c_{22}, d_{22}) + m_9 + t) \lll 9$

Table 20. Modification Procedure for Starting Collision Search from Middle of the Second Round

In order to satisfy a condition "$b_{33,31} = 0$" in the third round, we modify the value of chaining variable in step 29, which is a late step of the second round, This difference will propagate until step 33, and can flip the value of $b_{33,31}$. In this modification, we need to set several extra conditions. We can set all extra conditions when $i = 0, 9, 11, 26, 28, 29, 30, 31$. Therefore, we can make 2^8 values for b_{29} that satisfy all sufficient conditions in previous steps. Since, we don't control how the difference is propagated from step 29 to 33, the probability of satisfying "$b_{33,31} = 0$" is not 1. However, 256 times of trials enable us to quickly find a message that also satisfies "$b_{33,31} = 0$."

Round	Step	Actual procedure	Expression with difference in each step
1R	4	$m_3 \leftarrow m_3 - 2^{i-3}$ $b_4 \leftarrow b_4 - 2^{i+16}$ EC: "$b_{4,i+16} = 0$"	$b_4 \leftarrow (a_3 + f(b_3, c_3, d_3) +$ $m_3 - 2^{i-3} + t) \lll 19$
	5	EC: "$c_{4,i+16} = d_{4,i+16}$"	$b_5 \leftarrow (a_4 + f(b_4[-(i+16)], c_4, d_4) +$ $m_4 + t) \lll 3$
	6	EC: "$b_{5,i+16} = 0$"	$b_6 \leftarrow (a_5 + f(b_5, c_5[-(i+16)], d_5) +$ $m_5 + t) \lll 7$
	7	EC: "$b_{6,i+16} = 1$"	$b_7 \leftarrow (a_6 + f(b_6, c_6, d_6[-(i+16)]) +$ $m_6 + t) \lll 11$
	8	$m_7 \leftarrow m_7 + 2^{i+16}$	$b_8 \leftarrow (a_7 - 2^{i+16} + (f(b_7, c_7, d_7) +$ $m_7 + 2^{i+16} + t) \lll 19$
2R	29	$b_{29} \leftarrow$ standard computation	$b_{29} \leftarrow (a_{28} + f(b_{28}, c_{28}, d_{28}) +$ $m_3 - 2^{i-3} + t) \lll 3$
	30	$b_{30} \leftarrow$ standard computation	Difference propagation is out of control. Probabilistically flip the value of $b_{33,31}$.
	31	$b_{31} \leftarrow$ standard computation	
	32	$b_{32} \leftarrow$ standard computation	
3R	33	$b_{33} \leftarrow$ standard computation	