Collision Attack for the Hash Function Extended MD4*

Gaoli Wang[1,2]

[1] School of Computer Science and Technology, Donghua University, Shanghai 201620, China
wanggaoli@dhu.edu.cn
[2] State Key Laboratory of Information Security, Institute of Software, Chinese Academy of Sciences, Beijing 100049, China

Abstract. Extended MD4 is a hash function proposed by Rivest in 1990 with a 256-bit hash value. The compression function consists of two different and independent parallel lines called Left Line and Right Line, and each line has 48 steps. The initial values of Left Line and Right Line are denoted by IV_0 and IV_1 respectively. Dobbertin proposed a collision attack for the compression function of Extended MD4 with a complexity of about 2^{40} under the condition that the value for $IV_0 = IV_1$ is prescribed. In this paper, we gave a collision attack on the full Extended MD4 with a complexity of about 2^{37}. Firstly, we propose a collision differential path for both lines by choosing a proper message difference, and deduce a set of sufficient conditions that ensure the differential path hold. Then by using some precise message modification techniques to improve the success probability of the attack, we find two-block collisions of Extended MD4 with less than 2^{37} computations. This work provides a new reference to the collision analysis of other hash functions such as RIPEMD-160 etc. which consist of two lines.

Keywords: Cryptanalysis, Collision, Differential path, Hash functions, Extended MD4.

1 Introduction

Cryptographic hash functions remain one of the most used cryptographic primitives, and they can be used to guarantee the security of many cryptosystems and protocols such as digital signature, message authentication code and so on. In 1990, Rivest introduced the first dedicated hash function MD4 [20, 21]. After the publication of MD4, several dedicated hash functions were proposed, and these functions are called MD4-family. Depend on the method of the message expansion and the number of parallel of computations, the MD4-family are divided into three subfamilies. The first is MD-family, which consists of MD4 [20, 21], MD5 [23] and HAVAL [33]. The characteristics of MD-family is using roundwise permutations for the message expansion

* This work is supported by "Chen Guang" project (supported by Shanghai Municipal Education Commission and Shanghai Education Development Foundation), the National Natural Science Foundation of China (No. 61103238), the Fundamental Research Funds for the Central Universities, and the open research fund of State Key Laboratory of Information Security.

S. Qing et al. (Eds.): ICICS 2011, LNCS 7043, pp. 228–241, 2011.

and only one line of computation. The second is RIPEMD-family, which consists of RIPEMD [19], RIPEMD-{128,160,256,320} [10] and Extended MD4 [20, 21]. The crucial difference between MD-family and RIPEMD-family is that RIPEMD-family uses two parallel lines of computations. The third is SHA-family, which consists of SHA-{0,1,224,256,384,512} [13–15]. These functions use only one line of computation, but the message expansion is achieved by some recursively defined function. Several important breakthroughs have been made in the cryptanalysis against hash functions and they imply that most of the currently used standard hash functions are vulnerable against these attacks. In this circumstance, National Institute of Standards and Technology (NIST) launches the NIST Hash Competition, a public competition to develop a new hash standard, which will be called SHA-3 [18] and announced by 2012.

The first analysis of MD4 and MD5 were made by Vaudenay [24] and by den Boer and Bosselaers [7]. Along with the development of the hash functions, there are several analysis on them [4, 8, 9, 11, 12, 22]. The continuous works on analysis of hash functions reveal that most of them are not as secure as claimed [2, 3, 17, 25]. Wang et al. presented a series collision attacks on the most prevailing hash functions including MD4, RIPEMD, RIPEMD-128, MD5, SHA-0, SHA-1 and HAVAL [16, 26–30, 32] using an attack technique which is based on differential cryptanalysis [1]. Wang's method was also used in searching the second-preimage of MD4 [31], and was further developed and used in the analysis of SHA-1 [5, 6].

There are four steps in attacking a hash function by using Wang's method. The first step is to select an appropriate message difference, which determines the success probability of the attack. The second step is to select a feasible differential path according to the selected message difference. The third step is to derive a set of sufficient conditions on the chaining variables to ensure the differential path hold, and a correct differential path implies that all the chaining variable conditions don't contradict each other. The last step is to use message modification techniques to force the modified messages to satisfy most of the sufficient conditions, so to greatly improve the success probability of the attack.

Every step is important for the attack. Many studies have been conducted on the security of MD-family and SHA-family hash functions using Wang's method, such as MD4, MD5, SHA-0, SHA-1 and HAVAL [26–30, 32]. However, owing to the two parallel lines of RIPEMD-family, it is hard for the attacker to deduce the correct differential path and to use message modification technique to improve the success probability for RIPEMD-family. The security of RIPEMD-family hash functions against collision attack has been strengthened greatly. [26] reported that among 30 selected collision differential paths, only one can produce the real collision, and in other paths, the conditions of both lines in some step cannot hold simultaneously. Extended MD4 is such a representative hash function of RIPEMD-family. It is difficult to deduce a correct differential path for both lines and to modify the message to greatly improve the success probability of the attack, so to find a practical collision.

Extended MD4 was proposed in the original article [20] by Rivest in 1990 with 256-bit result. Its compression function consists of two parallel lines called Left Line and Right Line, and each line has 48 steps. It is difficult to deduce a correct collision differential path because the sufficient conditions for the path in both lines are more likely to

contradict each other. Dobbertin proposed a collision attack for the compression function of Extended MD4 with the same prescribed initial values in both lines with the complexity of about 2^{40}. However, no collision of the full Extended MD4 was found yet. In this paper, we propose a practical collision attack on the full Extended MD4 using Wang's method. By choosing a proper message difference, we can find a differential path of both Left Line and Right Line, and deduce the corresponding sufficient conditions that ensure the differential path hold. We use the message modification techniques to modify the message so that almost all sufficient conditions hold. Our attack requires less than 2^{37} computations to get a collision of the full Extended MD4. To the best of our knowledge, this is the first work that a practical collision attack on the full Extended MD4 has been proposed.

The rest of the paper is organized as follows. Section 2 introduces some notations and describes the Extended MD4 algorithm. Section 3 introduces some useful properties of the nonlinear functions in Extended MD4. Section 4 proposes the detailed description of the collision attack on Extended MD4. Finally, Section 5 concludes the paper.

2 Background and Definitions

2.1 Notation

In order to describe our analysis conveniently, we introduce some notation, where $0 \leq j \leq 31$.

1. $M = (m_0, m_1, \ldots, m_{15})$: 512-bit block M, where m_i $(0 \leq i \leq 15)$ is a 32-bit word
2. $\neg, \wedge, \oplus, \vee$: bitwise complement, AND, XOR and OR
3. $\lll s$: circular shift s-bit positions to the left
4. $x||y$: concatenation of the two bitstrings x and y
5. $+, -$: addition and subtraction modulo 2^{32}
6. $x_{i,j}$: the j-th bit of 32-bit word x_i, where the most significant bit is the 31-st bit
7. $\Delta x_i = x_i' - x_i$: the modular subtraction difference of two words x_i' and x_i
8. $x_i' = x_i[j]$: the value obtained by modifying the j-th bit of x_i from 0 to 1, i.e. $x_{i,j} = 0$, $x_{i,j}' = 1$, and the other bits of x_i and x_i' are all equal. Similarly, $x_i[-j]$ is the value obtained by modifying the j-th bit of x_i from 1 to 0
9. $x_i[\pm j_1, \pm j_2, ..., \pm j_k]$: the value obtained by modifying the bits in positions j_1,...,j_k of x_i according to the $\pm$ signs
10. (a_i, b_i, c_i, d_i), (aa_i, bb_i, cc_i, dd_i) $(0 \leq i \leq 12)$: the chaining variables corresponding to the message block M_1 of Left Line and Right Line respectively
11. (a_i', b_i', c_i', d_i'), $(aa_i', bb_i', cc_i', dd_i')$ $(0 \leq i \leq 12)$: the chaining variables corresponding to the message block M_1' of Left Line and Right Line respectively

Note that the differential definition in Wang's method is a kind of precise differential which uses the difference in terms of integer modular subtraction and the difference in terms of XOR. The combination of both kinds of differences gives attackers more information. For example, the output difference in Step 1 of Table 5 is $\Delta a_1 = a_1' - a_1 = 2^{19}$, for the specific differential path, we need to expand the one-bit difference in bit 19 into a three-bit differences in bits 19, 20, 21. That is, we expand $a_1[19]$ to $a_1[-19, -20, 21]$, which means the 19-th and 20-th bits of a_1 are 1, and the 21-st bit of a_1 is 0, while the 19-th and 20-th bits of a_1' are 0, and the 21-st bit of a_1' is 1.

2.2 Description of Extended MD4

The hash function Extended MD4 compresses a message of length less than 2^{64} bits into a 256-bit hash value. Firstly, the algorithm pads any given message into a message with the length of 512-bit multiple. We don't describe the padding process here because it has little relation with our attack, and the details of the message padding can refer to [20, 21]. Each 512-bit message block invokes a compression function of Extended MD4. The compression function takes a 256-bit chaining value and a 512-bit message block as inputs and outputs another 256-bit chaining value. The initial chaining value is a set of fixed constants. The compression function consists of two parallel operations called Left Line and Right Line, which have the same structure. Each line has three rounds, and the nonlinear functions in each round are as follows:

$$\begin{aligned} F(X,Y,Z) &= (X \wedge Y) \vee (\neg X \wedge Z), \\ G(X,Y,Z) &= (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z), \\ H(X,Y,Z) &= X \oplus Y \oplus Z. \end{aligned}$$

Here X, Y, Z are 32-bit words. The operations of the three functions are all bitwise. Each round of the compression function in each line consists of sixteen similarly steps, and in each step one of the four chaining variables a, b, c, d is updated.

$$\begin{aligned} \phi_0(a,b,c,d,m_k,s) &= (a + F(b,c,d) + m_k) \lll s \\ \phi_1(a,b,c,d,m_k,s) &= (a + G(b,c,d) + m_k + 0x5a827999) \lll s \\ \phi_2(a,b,c,d,m_k,s) &= (a + H(b,c,d) + m_k + 0x6ed9eba1) \lll s \\ \psi_0(a,b,c,d,m_k,s) &= (a + F(b,c,d) + m_k) \lll s \\ \psi_1(a,b,c,d,m_k,s) &= (a + G(b,c,d) + m_k + 0x50a28be6) \lll s \\ \psi_2(a,b,c,d,m_k,s) &= (a + H(b,c,d) + m_k + 0x5c4dd124) \lll s \end{aligned}$$

The initial value of Left Line is (a_0, b_0, c_0, d_0) = (67452301, $efcdab89$, 98$badcfe$, 10325476). The initial value of Right Line is (aa_0, bb_0, cc_0, dd_0) = (33221100, 77665544, $bbaa9988$, $ffeeddcc$).

Compression Function of Extended MD4. For a 512-bit message block $M = (m_0, m_1, \ldots, m_{15})$ of the padded message $\overline{M}$, the compression function consists of Left Line and Right Line.

Left Line. For the 512-bit block M, Left Line is as follows:

1. Let (a_0, b_0, c_0, d_0) be the input of Left Line for M. If M is the first message block to be hashed, then (a_0, b_0, c_0, d_0) are set to be the initial value. Otherwise they are the output from compressing the previous message block by Left Line.
2. Perform the following 48 steps (three rounds):
   For $j = 0, 1, 2$,
   For $i = 0, 1, 2, 3$,
   $a = \phi_j(a, b, c, d, m_{ord(j,16j+4i+1)}, s_{j,16j+4i+1})$,
   $d = \phi_j(d, a, b, c, m_{ord(j,16j+4i+2)}, s_{j,16j+4i+2})$,
   $c = \phi_j(c, d, a, b, m_{ord(j,16j+4i+3)}, s_{j,16j+4i+3})$,
   $b = \phi_j(b, c, d, a, m_{ord(j,16j+4i+4)}, s_{j,16j+4i+4})$.

The compressing result of Left Line is $(A, B, C, D) = (a_0 + a, b_0 + b, c_0 + c, d_0 + d)$.

Right Line. For the 512-bit block M, Right Line is as follows:

1. Let (aa_0, bb_0, cc_0, dd_0) be the input of Right Line process for M. If M is the first block to be hashed, (aa_0, bb_0, cc_0, dd_0) are the initial value. Otherwise they are the output from compressing the previous message block by Right Line.
2. Perform the following 48 steps (three rounds):
   For $j = 0, 1, 2$,
   For $i = 0, 1, 2, 3$,
   $aa = \psi_j(aa, bb, cc, dd, m_{ord(j,16j+4i+1)}, s_{j,16j+4i+1})$,
   $dd = \psi_j(dd, aa, bb, cc, m_{ord(j,16j+4i+2)}, s_{j,16j+4i+2})$,
   $cc = \psi_j(cc, dd, aa, bb, m_{ord(j,16j+4i+3)}, s_{j,16j+4i+3})$,
   $bb = \psi_j(bb, cc, dd, aa, m_{ord(j,16j+4i+4)}, s_{j,16j+4i+4})$.

The compressing result of Right Line is $(AA, BB, CC, DD) = (aa_0 + aa, bb_0 + bb, cc_0 + cc, dd_0 + dd)$.

Note that after every 16-word block is processed, the values of the a register in Left Line and the aa register in Right Line are exchanged. The ordering of message words and the details of the shift positions can be seen in Table 1.

If M is the last block of $\overline{M}$, $(A\|B\|C\|D\|AA\|BB\|CC\|DD)$ is the hash value of the message $\overline{M}$. Otherwise, repeat the above process with the next 512-bit message block and (A, B, C, D) as the input chaining variables of Left Line, (AA, BB, CC, DD) as the input chaining variables of Right Line.

Table 1. Order of the message words and Shift positions in Extended MD4

Step i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$ord(0, i)$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$s_{0,i}$	3	7	11	19	3	7	11	19	3	7	11	19	3	7	11	19
Step i	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
$ord(1, i)$	0	4	8	12	1	5	9	13	2	6	10	14	3	7	11	15
$s_{1,i}$	3	5	9	13	3	5	9	13	3	5	9	13	3	5	9	13
Step i	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
$ord(2, i)$	0	8	4	12	2	10	6	14	1	9	5	13	3	11	7	15
$s_{2,i}$	3	9	11	15	3	9	11	15	3	9	11	15	3	9	11	15

3 Some Basic Conclusions of the Three Nonlinear Functions

We will recall some well-known properties of the three nonlinear Boolean functions in the following because they are very helpful for determining the collision differential path and the corresponding sufficient conditions.

Proposition 1. For the nonlinear function $F(x, y, z) = (x \wedge y) \vee (\neg x \wedge z)$, here and in the follows, $x \in \{0, 1\}$, $y \in \{0, 1\}$ and $z \in \{0, 1\}$, there are the following properties:

1. (a) $F(x,y,z) = F(\neg x,y,z)$ if and only if $y = z$.

   (b) $F(x,y,z) = x$ and $F(\neg x,y,z) = \neg x$ if and only if $y = 1$ and $z = 0$.

   (c) $F(x,y,z) = \neg x$ and $F(\neg x,y,z) = x$ if and only if $y = 0$ and $z = 1$.
2. (a) $F(x,y,z) = F(x,\neg y,z)$ if and only if $x = 0$.

   (b) $F(x,y,z) = y$ and $F(x,\neg y,z) = \neg y$ if and only if $x = 1$.
3. (a) $F(x,y,z) = F(x,y,\neg z)$ if and only if $x = 1$.

   (b) $F(x,y,z) = z$ and $F(x,y,\neg z) = \neg z$ if and only if $x = 0$.

Proposition 2. For the nonlinear function $G(x,y,z) = (x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$, there are the following properties:

1. $G(x,y,z) = G(\neg x,y,z)$ if and only if $y = z$.
   $G(x,y,z) = x$ and $G(\neg x,y,z) = \neg x$ if and only if $y \neq z$.
2. $G(x,y,z) = G(x,\neg y,z)$ if and only if $x = z$.
   $G(x,y,z) = y$ and $G(x,\neg y,z) = \neg y$ if and only if $x \neq z$.
3. $G(x,y,z) = G(x,y,\neg z)$ if and only if $x = y$.
   $G(x,y,z) = z$ and $G(x,y,\neg z) = \neg z$ if and only if $x \neq y$.

Proposition 3. For the nonlinear function $H(x,y,z) = x \oplus y \oplus z$, there are the following properties:

1. $H(x,y,z) = \neg H(\neg x,y,z) = \neg H(x,\neg y,z) = \neg H(x,y,\neg z)$.
2. $H(x,y,z) = H(\neg x,\neg y,z) = H(x,\neg y,\neg z) = H(\neg x,y,\neg z)$.

4 The Practical Collision Attack against Extended MD4

In this section, we present a practical collision attack on Extended MD4. Each message in the collision includes two 512-bit message blocks. We search the collision pair $(M_0\|M_1, M_0\|M_1')$ in the following four parts:

1. Denote Extended MD4 by h and the hash value $h(M_0)$ by $(a\|b\|c\|d\|aa\|bb\|cc\|dd)$. (a,b,c,d) and (aa,bb,cc,dd) are also the input chaining variables of Left Line and Right Line respectively of the next compression function. Find a message block M_0 such that $h(M_0)$ satisfy some conditions which are part of the sufficient conditions that ensure the differential path hold, and the conditions of $h(M_0)$ are $b_i = c_i (i = 19,21)$, $b_i = 0 (i = 20,27,28)$, $c_{20} = 1$, $bb_i = cc_i (i = 19,21)$, $bb_i = 0 (i = 20,27,28)$ and $cc_{20} = 1$.
2. Choose an appropriate message difference $\Delta M_1 = M_1' - M_1$ and deduce the differential path according to the specified message difference.
3. Derive a set of sufficient conditions which ensures the differential path hold. This means that if $h(M_1)$ satisfies all the conditions in Table 6, then $(M_0\|M_1, M_0\|M_1')$ consist of a collision.
4. Modify the message M_1 to fulfill most of the sufficient conditions.

Obviously the first part is easy to be carried out. We will describe the last three parts in details.

4.1 Collision Differential Path for Extended MD4

Constructing the differential path and deriving the sufficient conditions go on simultaneously. On one hand, we derive the sufficient conditions according to the differential path. On the other hand, we can adjust the differential path to avoid the contradictory conditions. If the sufficient conditions contradict each other, the corresponding differential path is error and a collision cannot be found. The sufficient conditions in Left Line and Right Line of Extended MD4 in some step cannot hold simultaneously, so we must search other differential paths from scratch.

Almost all the conditions in the first round and some conditions in the second round can be modified to be hold by the message modification technique, the other conditions in the last rounds are difficult to be modified to hold. Therefore, we will ensure the sufficient conditions in the last rounds to be as less as possible. In order to find such a differential path, we select a difference between two messages as follows: $\Delta M_1 = M_1' - M_1 = (\Delta m_0, \Delta m_1, \ldots, \Delta m_{15})$, where $\Delta m_0 = 2^{16}, \Delta m_i = 0, 0 < i \leq 15$.

The whole differential path are shown in Table 5. The first column denotes the step, the second is the chaining variable in each step for M_1, the third is the message word of M_1 in each step, the fourth is the shift rotation, the fifth is the message difference between M_1 and M_1', the sixth is the chaining variable difference for M_1 and M_1', and the seventh is the chaining variable for M_1'. The empty items both in the fifth and the sixth columns denote zero differences, and steps those aren't listed in the table have zero differences for message words and chaining variables.

4.2 Deriving the Sufficient Conditions for Differential Path

In light of the propositions of the nonlinear Boolean functions given in Section 3, we can derive the conditions that guarantee the differential path in Table 5 hold. A set of sufficient conditions is shown in Table 6.

We give an example to describe how to derive a set of sufficient conditions that guarantees the differential path in Step 9 of Table 5 hold. Other conditions can be derived similarly. The differential path in Step 9 of Table 5 is

$$(a_2[-10, 22, -30], d_2[-2, -3, 4], c_2[17], b_2[21, -22])$$

$$\longrightarrow (d_2[-2, -3, 4], c_2[17], b_2[21, -22], a_3[1, -2, -13]).$$

1. According to $a_3 = (a_2 + F(b_2, c_2, d_2) + m_8) \lll 3$ and 1(b) of Proposition 1, the conditions $c_{2,22} = 1$ and $d_{2,22} = 0$ ensure that the change of $b_{2,22}$ results in $F(b_2[-22], c_2, d_2) - F(b_2, c_2, d_2) = -2^{22}$, combined with $\Delta a_2 = 2^{22}$, which will lead to no change in a_3.
2. According to 1(a) of Proposition 1, the condition $c_{2,21} = d_{2,21}$ ensures that the change of $b_{2,21}$ results in no change in a_3.
3. According to 2(a) of Proposition 1, the condition $b_{2,17} = 0$ ensures that the change of $c_{2,17}$ results in no change in a_3.
4. According to 3(a) of Proposition 1, the conditions $b_{2,2} = 1$, $b_{2,3} = 1$ and $b_{2,4} = 1$ ensure that the changes in the 2-nd, 3-rd and 4-th bits of d_2 result in no change in a_3.

5. Because the shift is 3 in Step 9, $\Delta a_2 = -2^{10}$ must lead to $\Delta a_3 = -2^{13}$, and the condition $a_{3,13} = 1$ results in $a'_{13} = a_{13}[-13]$.
6. Similarly, $\Delta a_2 = -2^{30}$ must lead to $\Delta a_3 = -2$, and the condition $a_{3,1} = 0$ and $a_{3,2} = 1$ result in $a'_3 = a_3[1, -2]$.

This means the above 10 conditions consist of a set of sufficient conditions for the differential path in Step 9.

4.3 Message Modification

In order to improve the collision probability, we modify M_1 so that most of the sufficient conditions in Table 6 hold. The modification includes the basic and advanced techniques. Because Extended MD4 has two lines, the modification is much more complicated than that of MD4, MD5, HAVAL etc. which only have one line operation.

1. We modify M_1 word by word so that both lines with the modified M_1 satisfy almost all the conditions in the first round.
   (a) By using the basic modification technique, we modify m_{i-1} such that the i-th step conditions in the first round of Left Line hold. For example, to ensure the 8 conditions of d_1 in Table 6 hold, we modify m_1 as follows:
   $d_1 \longleftarrow d_1 \oplus (d_{1,19} \ll 19) \oplus (d_{1,20} \ll 20) \oplus (d_{1,21} \ll 21) \oplus (d_{1,28} \ll 28) \oplus ((d_{1,27} \oplus 1) \ll 27) \oplus ((d_{1,6} \oplus a_{1,6}) \ll 6) \oplus ((d_{1,7} \oplus a_{1,7}) \ll 7) \oplus ((d_{1,8} \oplus a_{1,8}) \ll 8)$,

   $$m_1 \longleftarrow (d_1 \ggg 7) - d_0 - F(a_1, b_0, c_0).$$

   (b) By using the advanced modification technique, we modify the message word from low bit to high bit to correct the corresponding conditions in the first round of Right Line.
      i. Firstly, we can correct the conditions by bit carry. For example, because there is no constraint conditions in $a_{1,18}$ in Table 6, we can correct $aa_{1,19} = 0$ to $aa_{1,19} = 1$ as follows. If $a_{1,18} = 0$ and $aa_{1,18} = 1$, let $m_0 \longleftarrow m_0 + 2^{15}$, then there is a bit carry in Right Line and no bit carry in Left Line, so the condition in $aa_{1,19}$ can be corrected, and the corrected $a_{1,19}$ will not be changed. Similarly, if $a_{1,18} = 1$ and $aa_{1,18} = 0$, let $m_0 \longleftarrow m_0 - 2^{15}$, then $aa_{1,19}$ can be corrected and $a_{1,19}$ will not be changed. If $a_{1,18} = aa_{1,18}$, we can use the lower bit carry to change $a_{1,18}$ or $aa_{1,18}$ such that $a_{1,18} \neq aa_{1,18}$, and then use the bit carry. The details for correcting $aa_{1,19}$ are given in Table 2.
      ii. Secondly, we can correct the condition on $a_{i,j}$ by changing the corresponding variables in the previous steps. For example, we can correct $dd_{1,20} = 1$ to $dd_{1,20} = 0$ as follows. If $b_{0,13} \oplus c_{0,13} \neq bb_{0,13} \oplus cc_{0,13}$ (which means when $b_{0,13} = c_{0,13}$, then $bb_{0,13} \neq cc_{0,13}$; when $b_{0,13} \neq c_{0,13}$, then $bb_{0,13} = cc_{0,13}$), let $m_0 \longleftarrow m_0 \pm 2^{10}$, then $a_{1,13}$ and $aa_{1,13}$ will be changed, and the changed $a_{1,13}, aa_{1,13}$ only cause one of $d_{1,20}$ and $dd_{1,20}$ change according to 1 of Proposition 1. Then if $d_{1,20} = dd_{1,20} = 1$, let $m_1 \longleftarrow m_1 - 2^{13}$, if $d_{1,20} = dd_{1,20} = 0$, modify the next bit of dd_1. Note that there is no condition in $a_{1,13}$ and $aa_{1,13}$ in Table 6, so the changes in $a_{1,13}$ and $aa_{1,13}$ don't invalidate the differential path. The details for correcting $dd_{1,20}$ are given in Table 3.

2. There are $18 \times 2 = 36$ conditions in total in the second round in both lines. We can utilize some more precise modification techniques to correct some conditions in the second round. Sometimes, it needs to add some extra conditions in the first round in advance such that the change of any condition doesn't affect all the corrected conditions.

Table 2. The message modification for correcting $aa_{1,19} = 0$ to $aa_{1,19} = 1$

Known Conditions	The Modified m_0	New Chaining Variables
$a_{1,18} = 0$ $aa_{1,18} = 1$	$m_0 \longleftarrow m_0 + 2^{15}$	$aa_{1,19} = 1$ $a_{1,19}$ unchanged
$a_{1,18} = 1$ $aa_{1,18} = 0$	$m_0 \longleftarrow m_0 - 2^{15}$	$aa_{1,19} = 1$ $a_{1,19}$ unchanged
$a_{1,18} = aa_{1,18} = 1$ $a_{1,17} = 0$ $aa_{1,17} = 1$	$m_0 \longleftarrow m_0 + 2^{14}$	$aa_{1,19} = 1$ $a_{1,19}$ unchanged
$a_{1,18} = aa_{1,18} = 1$ $a_{1,17} = 1$ $aa_{1,17} = 0$	$m_0 \longleftarrow m_0 - 2^{14} - 2^{15}$	$aa_{1,19} = 1$ $a_{1,19}$ unchanged
$a_{1,18} = aa_{1,18} = 0$ $a_{1,17} = 0$ $aa_{1,17} = 1$	$m_0 \longleftarrow m_0 + 2^{14} + 2^{15}$	$aa_{1,19} = 1$ $a_{1,19}$ unchanged
$a_{1,18} = aa_{1,18} = 0$ $a_{1,17} = 1$ $aa_{1,17} = 0$	$m_0 \longleftarrow m_0 - 2^{14}$	$aa_{1,19} = 1$ $a_{1,19}$ unchanged

4.4 Overview of the Collision Attack Algorithm

From the above description, an overview of the collision attack algorithm on Extended MD4 can be expressed as follows.

1. Find a message block M_0 such that $h(M_0) = (a||b||c||d||aa||bb||cc||dd)$ satisfy $b_i = c_i (i = 19, 21)$, $b_i = 0 (i = 20, 27, 28)$, $c_{20} = 1$, $bb_i = cc_i (i = 19, 21)$, $bb_i = 0 (i = 20, 27, 28)$ and $cc_{20} = 1$.
2. Repeat the following steps until we can find a message block M_1 which satisfies all the sufficient conditions in the first round of Left Line and Right Line in Table 6.
   (a) Select a random message block M_1.
   (b) Modify M_1 by Step 1 of message modification described above.
   (c) Test if the hash value of M_1 satisfy all the sufficient conditions in the first round in both lines in Table 6.
3. Repeat the following steps until a collision $(M_0||M_1, M_0||M_1')$ is found.
   (a) Select random message words m_{14} and m_{15} of M_1.
   (b) Modify M_1 by Step 1 of message modification described above such that all the conditions in c_4, b_4, cc_4 and bb_4 satisfied.

Table 3. The message modification for correcting $dd_{1,20} = 1$ to $dd_{1,20} = 0$

Case	Step	m_i	Shift	Modified m_i	Chaining Variables	Conditions
	1	m_0	3	$m_0 \longleftarrow m_0 \pm 2^{10}$	$a_{1,13}$, $aa_{1,13}$ changed	
Case 1	2	m_1	7		$d_{1,20}$ changed i.e. $d_{1,20} = 1$	$b_{0,13} \neq c_{0,13}$
					$dd_{1,20}$ unchanged i.e. $dd_{1,20} = 1$	$bb_{0,13} = cc_{0,13}$
				$m_1 \longleftarrow m_1 - 2^{13}$	$d_{1,20} = 0$ $dd_{1,20} = 0$	
Case 2	2	m_2	7		$d_{1,20}$ unchanged i.e. $d_{1,20} = 0$	$b_{0,13} = c_{0,13}$
					$dd_{1,20}$ changed i.e. $dd_{1,20} = 0$	$bb_{0,13} \neq cc_{0,13}$

(c) Modify M_1 by Step 2 of message modification described above such that some conditions in the second round satisfied.
(d) Then M_1 and $M_1' = M_1 + \Delta M_1$ satisfy all the sufficient conditions in both lines in Table 6 with the probability higher than 2^{-36}.
(e) Test if the hash value of M_1 is equal to the hash value of M_1'.

It is easy to find proper M_0 in Step 1 and to find M_1 which satisfy all the sufficient conditions in the first round in both lines in Table 6, and the complexity can be neglected. There are 36 conditions in total in the second round in both lines, so M_1 and M_1' lead to a collision with probability higher than 2^{-36}, and the complexity to find a collision $(M_0\|M_1, M_0\|M_1')$ is less than 2^{37} Extended MD4 computations. A collision for Extended MD4 can be seen in Table 4.

Table 4. A collision of Extended MD4. H_0 is the hash value for the message block M_0 with little-endian and no message padding. H is the common hash value for the message $M_0 \parallel M_1$ and $M_0 \parallel M_1'$ with little-endian and no message padding.

M_0	a4eff7cd 87afe33e b96f8657 1054fe49 8397de8d 23bc04b8 b683a020 3b2a5d9f
	c69d71b3 f9e99198 d79f805e a63bb2e8 45dd8e31 97e31fe5 2794bf08 b9e8c3e9
H_0	b5aac7e7 c1664fe2 01705583 ac3cc062 65c931e6 452829ae 527e12c7 30fafffb
M_1	54b7b7e1 65336f98 7621fe73 ffa42822 13dda7e1 1c28a008 bf20c341 3e8e28f2
	578bdb87 afd42be4 b9ecab2e 0aaa9293 02e7070b eab6f4cf 2e96aaf7 5ab41efd
M_1'	54b8b7e1 65336f98 7621fe73 ffa42822 13dda7e1 1c28a008 bf20c341 3e8e28f2
	578bdb87 afd42be4 b9ecab2e 0aaa9293 02e7070b eab6f4cf 2e96aaf7 5ab41efd
H	3055a689 7fe0b4a6 88d59251 af8afd0f 3826bda2 942f0939 c2673493 a6c56bac

Table 5. Collision Differential Path of Extended MD4

Step	Chaining Value for M	m_i	Shift	Δm_i	The step difference	Chaining Value for M'
1	a_1	m_0	3	2^{16}	2^{19}	$a_1[-19,-20,21]$
2	d_1	m_1	7		2^{27}	$d_1[-27,28]$
3	c_1	m_2	11		2^{6}	$c_1[-6,-7,8]$
4	b_1	m_3	19			b_1
5	a_2	m_4	3		$-2^{10}+2^{22}-2^{30}$	$a_2[-10,22,-30]$
6	d_2	m_5	7		2^{2}	$d_2[-2,-3,4]$
7	c_2	m_6	11		2^{17}	$c_2[17]$
8	b_2	m_7	19		-2^{21}	$b_2[21,-22]$
9	a_3	m_8	3		$-2-2^{13}$	$a_3[1,-2,-13]$
10	d_3	m_9	7		2^{9}	$d_3[-9,10]$
11	c_3	m_{10}	11		2^{28}	$c_3[28]$
12	b_3	m_{11}	19		-2^{8}	$b_3[8,-9]$
13	a_4	m_{12}	3		$-2^{4}-2^{16}$	$a_4[-4,-16]$
14	d_4	m_{13}	7			d_4
15	c_4	m_{14}	11		2^{7}	$c_4[-7,8]$
16	b_4	m_{15}	19			b_4
17	a_5	m_0	3	2^{16}	-2^{7}	$a_5[7,-8]$
18	d_5	m_4	5			d_5
19	c_5	m_8	9			c_5
20	b_5	m_{12}	13			b_5
21	a_6	m_1	3		-2^{10}	$a_6[-10]$
22	d_6	m_5	5			d_6
23	c_6	m_9	9			c_6
24	b_6	m_{13}	13			b_6
25	a_7	m_2	3		-2^{13}	$a_7[-13]$
26	d_7	m_6	5			d_7
27	c_7	m_{10}	9			c_7
28	b_7	m_{14}	13			b_7
29	a_8	m_3	3		-2^{16}	$a_8[-16]$
30	d_8	m_7	5			d_8
31	c_8	m_{11}	9			c_8
32	b_8	m_{15}	13			b_8
33	a_9	m_0	3	2^{16}		a_9
...	...	...	...	...	...	...
48	b_{12}	m_{15}	15			b_{12}

Table 6. A Set of Sufficient Conditions for the Differential Path given in Table 5

c_0 $c_{0,20} = 1$
b_0 $b_{0,19} = c_{0,19}, b_{0,20} = 0, b_{0,21} = c_{0,21}, b_{0,27} = 0, b_{0,28} = 0$
a_1 $a_{1,19} = 1, a_{1,20} = 1, a_{1,21} = 0, a_{1,27} = 1, a_{1,28} = 1$
d_1 $d_{1,6} = a_{1,6}, d_{1,7} = a_{1,7}, d_{1,8} = a_{1,8}, d_{1,19} = 0, d_{1,20} = 0, d_{1,21} = 0, d_{1,27} = 1, d_{1,28} = 0$
c_1 $c_{1,6} = 1, c_{1,7} = 1, c_{1,8} = 0, c_{1,19} = 1, c_{1,20} = 1, c_{1,21} = 1, c_{1,27} = 0, c_{1,28} = 0$
b_1 $b_{1,6} = 0, b_{1,7} = 1, b_{1,8} = 0, b_{1,10} = c_{1,10}, b_{1,22} = c_{1,22}, b_{1,27} = 0, b_{1,28} = 1, b_{1,30} = c_{1,30}$
a_2 $a_{2,2} = b_{1,2}, a_{2,3} = b_{1,3}, a_{2,4} = b_{1,4}, a_{2,6} = 1, a_{2,7} = 1,$ $a_{2,8} = 1, a_{2,10} = 1, a_{2,22} = 0, a_{2,30} = 1$
d_2 $d_{2,2} = 1, d_{2,3} = 1, d_{2,4} = 0, d_{2,10} = 0, d_{2,17} = a_{2,17}, d_{2,22} = 0, d_{2,30} = 0$
c_2 $c_{2,2} = 1, c_{2,3} = 0, c_{2,4} = 0, c_{2,10} = 1, c_{2,17} = 0, c_{2,21} = d_{2,21}, c_{2,22} = 1, c_{2,30} = 1$
b_2 $b_{2,1} = c_{2,1}, b_{2,2} = 1, b_{2,3} = 1, b_{2,4} = 1, b_{2,13} = c_{2,13}, b_{2,17} = 0, b_{2,21} = 0, b_{2,22} = 1$
a_3 $a_{3,1} = 0, a_{3,2} = 1, a_{3,9} = b_{2,9}, a_{3,10} = b_{2,10}, a_{3,13} = 1, a_{3,17} = 1, a_{3,21} = 0, a_{3,22} = 0$
d_3 $d_{3,1} = 0, d_{3,2} = 0, d_{3,9} = 1, d_{3,10} = 0, d_{3,13} = 0, d_{3,21} = 1, d_{3,22} = 1, d_{3,28} = a_{3,28}$
c_3 $c_{3,1} = 1, c_{3,2} = 1, c_{3,8} = d_{3,8}, c_{3,9} = 0, c_{3,10} = 0, c_{3,13} = 1, c_{3,28} = 0$
b_3 $b_{3,4} = c_{3,4}, b_{3,8} = 0, b_{3,9} = 1, b_{3,10} = 1, b_{3,16} = c_{3,16}, b_{3,28} = 0$
a_4 $a_{4,4} = 1, a_{4,8} = 0, a_{4,9} = 1, a_{4,16} = 1, a_{4,28} = 1$
d_4 $d_{4,4} = 0, d_{4,7} = a_{4,7}, d_{4,8} = 1, d_{4,9} = 1, d_{4,16} = 0$
c_4 $c_{4,4} = 1, c_{4,7} = 1, c_{4,8} = 0, c_{4,16} = 1$
b_4 $b_{4,7} = d_{4,7}, b_{4,8} = d_{4,8}$
a_5 $a_{5,7} = 0, a_{5,8} = 1$
d_5 $d_{5,7} \neq b_{4,7}, d_{5,8} \neq b_{4,8}$
c_5 $c_{5,7} = d_{5,7}, c_{5,8} = d_{5,8}$
b_5 $b_{5,10} = c_{5,10}$
a_6 $a_{6,10} = 1$
d_6 $d_{6,10} = b_{5,10}$
c_6 $c_{6,10} = d_{6,10}$
b_6 $b_{6,13} = c_{6,13}$
a_7 $a_{7,13} = 1$
d_7 $d_{7,13} = b_{6,13}$
c_7 $c_{7,13} = d_{7,13}$
b_7 $b_{7,16} = c_{7,16}$
a_8 $a_{8,16} = 1$
d_8 $d_{8,16} = b_{7,16}$
c_8 $c_{8,16} = d_{8,16}$

5 Conclusions

In this study, the security of the hash function Extended MD4 which consists of two parallel lines against collision analysis is examined. A practical attack on Extended MD4 for finding 2-block collision is proposed. A true collision instance of Extended MD4 can be found with less than 2^{37} computations. Future analysis should be able to explore the security of other hash functions which consist of two parallel lines against the collision analysis.

References

1. Biham, B., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. Journal of Cryptology 4(1), 3–72 (1991)
2. Biham, E., Chen, R.: Near-Collisions of SHA-0. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 290–305. Springer, Heidelberg (2004)
3. Biham, E., Chen, R., Joux, A., Carribault, P., Lemuet, C., Jalby, W.: Collisions of SHA-0 and Reduced SHA-1. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 36–57. Springer, Heidelberg (2005)
4. Chabaud, F., Joux, A.: Differential Collisions in SHA-0. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 56–71. Springer, Heidelberg (1998)
5. De Cannière, C., Rechberger, C.: Finding SHA-1 Characteristics: General Results and Applications. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 1–20. Springer, Heidelberg (2006)
6. De Cannière, C., Mendel, F., Rechberger, C.: Collisions for 70-Step SHA-1: On the Full Cost of Collision Search. In: Adams, C., Miri, A., Wiener, M. (eds.) SAC 2007. LNCS, vol. 4876, pp. 56–73. Springer, Heidelberg (2007)
7. den Boer, B., Bosselaers, A.: Collisions for the Compression Function of MD-5. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 293–304. Springer, Heidelberg (1994)
8. Dobbertin, H.: Cryptanalysis of MD4. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 53–69. Springer, Heidelberg (1996)
9. Dobbertin, H.: Cryptanalysis of MD5 Compress. In: The Rump Session of EUROCRYPT 1996 (1996)
10. Dobbertin, H., Bosselaers, A., Preneel, B.: RIPEMD-160: A Strengthened Version of RIPEMD. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 71–82. Springer, Heidelberg (1996)
11. Dobbertin, H.: RIPEMD with Two Round Compress Function Is Not Collision-Free. Journal of Cryptology 10(1), 51–70 (1997)
12. Dobbertin, H.: The First Two Rounds of MD4 are Not One-Way. In: Vaudenay, S. (ed.) FSE 1998. LNCS, vol. 1372, pp. 284–292. Springer, Heidelberg (1998)
13. FIPS 180: Secure Hash Standard, Federal Information Processing Standards Publication£NIST£ (May 1993)
14. FIPS 180-1: Secure Hash Standard, Federal Information Processing Standards Publication, NIST, US Department of Commerce, Washington D.C. (1996)
15. FIPS 180-2: Secure Hash Standard, Federal Information Processing Standards Publication, NIST (2002), http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf
16. Wang, G., Wang, M.: Cryptanalysis of reduced RIPEMD-128. Journal of Software 19(9), 2442–2448 (2008)

17. Joux, A.: Collisions for SHA-0. In: The Rump Session of CRYPTO 2004 (2004)
18. National Institute of Standards and Technology, Cryptographic hash project, http://csrc.nist.gov/groups/ST/hash/index.html
19. RIPE: Integrity Primitives for Secure Information Systems, Final Report of RACE Integrity Primitives Evalution (RIPE-RACE 1040). LNCS, vol. 1007. Springer, Heidelberg
20. Rivest, R.: The MD4 Message Digest Algorithm. In: Menezes, A., Vanstone, S. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 303–311. Springer, Heidelberg (1991)
21. Rivest, R.: The MD4 message-digest algorithm. Request for Comments (RFC) 1320, Internet Activities Board, Internet Privacy Task Force (1992)
22. Van Rompay, B., Biryukov, A., Preneel, B., Vandewalle, J.: Cryptanalysis of 3-pass HAVAL. In: Laih, C.S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 228–245. Springer, Heidelberg (2003)
23. The MD5 Message-digest Algorithm. In: Request for Comments (RFC) 1321, Internet Activities Board, Internet Privacy Task Force (1992)
24. Vaudenay, S.: On the Need for Multipermutations: Cryptanalysis of MD4 and SAFER. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 286–297. Springer, Heidelberg (1995)
25. Wang, X., Feng, D., Lai, X., Yu, H.: Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD. In: The Rump Session of CRYPTO 2004 (2004)
26. Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the Hash Functions MD4 and RIPEMD. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 1–18. Springer, Heidelberg (2005)
27. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)
28. Wang, X., Yu, H., Yin, Y.L.: Efficient Collision Search Attacks on SHA-0. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 1–16. Springer, Heidelberg (2005)
29. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)
30. Wang, X., Feng, D., Yu, X.: An attack on HAVAL function HAVAL-128. Science in China Ser. F Information Sciences 48(5), 1–12 (2005)
31. Yu, H., Wang, G., Zhang, G., Wang, X.: The Second-preimage Attack on MD4. In: Desmedt, Y.G., Wang, H., Mu, Y., Li, Y. (eds.) CANS 2005. LNCS, vol. 3810, pp. 1–12. Springer, Heidelberg (2005)
32. Yu, H., Wang, X., Yun, A., Park, S.: Cryptanalysis of the Full HAVAL with 4 and 5 Passes. In: Robshaw, M. (ed.) FSE 2006. LNCS, vol. 4047, pp. 89–110. Springer, Heidelberg (2006)
33. Zheng, Y., Pieprzyk, J., Seberry, J.: HAVAL-A One-way Hashing Algorithm with Variable Length of Output. In: Zheng, Y., Seberry, J. (eds.) AUSCRYPT 1992. LNCS, vol. 718, pp. 81–104. Springer, Heidelberg (1993)