How to Break MD5 and Other Hash Functions

Xiaoyun Wang and Hongbo Yu

Shandong University, Jinan 250100, China,
xywang@sdu.edu.cn, yhb@mail.sdu.edu.cn

Abstract. MD5 is one of the most widely used cryptographic hash functions nowadays. It was designed in 1992 as an improvement of MD4, and its security was widely studied since then by several authors. The best known result so far was a semi free-start collision, in which the initial value of the hash function is replaced by a non-standard value, which is the result of the attack. In this paper we present a new powerful attack on MD5 which allows us to find collisions efficiently. We used this attack to find collisions of MD5 in about 15 minutes up to an hour computation time. The attack is a differential attack, which unlike most differential attacks, does not use the exclusive-or as a measure of difference, but instead uses modular integer subtraction as the measure. We call this kind of differential a *modular differential*. An application of this attack to MD4 can find a collision in less than a fraction of a second. This attack is also applicable to other hash functions, such as RIPEMD and HAVAL.

1 Introduction

People know that digital signatures are very important in information security. The security of digital signatures depends on the cryptographic strength of the underlying hash functions. Hash functions also have many other applications in cryptography such as data integrity, group signature, e-cash and many other cryptographic protocols. The use of hash functions in these applications not only ensure the security, but also greatly improve the efficiency. Nowadays, there are two widely used hash functions – MD5 [18] and SHA-1 [12].

MD5 is a hash function designed by Ron Rivest as a strengthened version of MD4 [17]. Since its publication, some weaknesses has been found. In 1993, B. den Boer and A. Bosselaers [3] found a kind of pseudo-collision for MD5 which consists of the same message with two different sets of initial values. This attack discloses the weak avalanche in the most significant bit for all the chaining variables in MD5. In the rump session of Eurocrypt'96, H. Dobbertin [8] presented a semi free-start collision which consists of two different 512-bit messages with a chosen initial value IV_0'.

$$a_0 = \texttt{0x12ac2375},\ b_0 = \texttt{0x3b341042},\ c_0 = \texttt{0x5f62b97c},\ d_0 = \texttt{0x4ba763ed}$$

A general description of this attack was published in [9].

Although H. Dobbertin cannot provide a real collision of MD5, his attack reveals the weak avalanche for the full MD5. This provides a possibility to find a special differential with one iteration.

In this paper we present a new powerful attack that can efficiently find a collision of MD5. From H. Dobbertin's attack, we were motivated to study whether it is possible to find a pair of messages, each consists of two blocks, that produce collisions after the second block. More specifically, we want to find a pair (M_0, M_1) and (M_0', M_1') such that

$$\begin{aligned}(a, b, c, d) &= \mathrm{MD5}(a_0, b_0, c_0, d_0, M_0),\\ (a', b', c', d') &= \mathrm{MD5}(a_0, b_0, c_0, d_0, M_0'),\\ \mathrm{MD5}(a, b, c, d, M_1) &= \mathrm{MD5}(a', b', c', d', M_1'),\end{aligned}$$

where a_0, b_0, c_0, d_0 are the initial values for MD5. We show that such collisions of MD5 can be found efficiently, where finding the first blocks (M_0, M_0') takes about 2^{39} MD5 operations, and finding the second blocks (M_1, M_1') takes about 2^{32} MD5 operations. The application of this attack on IBM P690 takes about an hour to find M_0 and $M_0{}'$, where in the fastest cases it takes only 15 minutes. Then, it takes only between 15 seconds to 5 minutes to find the second blocks M_1 and $M_1{}'$. Two such collisions of MD5 were made public in the Crypto'04 rump session [19].

This attack is applicable to many other hash functions as well, including MD4, HAVAL-128 and RIPEMD ([17], [20], [15]). In the case of MD4, the attack can find a collision within less than a second, and can also find second pre-images for many messages.

In Crypto'04 Eli Biham and Rafi Chen presented a near-collision attack on SHA-0 [2], which follows the lines of the technique of [4]. In the rump session they described their new (and improved) results on SHA-0 and SHA-1 (including a multi-block technique and collisions of reduced SHA-1). Then, A.J̃oux presented a 4-block full collision of SHA-0 [14], which is a further improvement of these results. Both these works were made independently of this paper.

This paper is organized as follows: In Section 2 we briefly describe MD5. Then in Section 3 we give the main ideas of our attack, and in Section 4 we give a detailed description of the attack. Finally, in Section 5 we summarize the paper, and discuss the applicability of this attack to other hash functions.

2 Description of MD5

In order to conveniently describe the general structure of MD5, we first recall the iteration process for hash functions.

Generally a hash function is iterated by a compression function $X = f(Z)$ which compresses l-bit message block Z to s-bit hash value X where $l > s$. For MD5, $l = 512$, and $s = 128$. The iterating method is usually called the Merkle-Damgard meta-method (see [6], [16]). For a padded message M with multiples of l-bit length, the iterating process is as follows:

$$H_{i+1} = f(H_i, M_i), \quad 0 \le i \le t-1.$$

Here $M = (M_0, M_2, \cdots, M_{t-1})$, and $H_0 = IV_0$ is the initial value for the hash function.

In the above iterating process, we omit the padding method because it has no influence on our attack.

The following is to describe the compression function for MD5. For each 512-bit block M_i of the padded message M, divide M_i into 32-bit words, $M_i = (m_0, m_1, ..., m_{15})$. The compression algorithm for M_i has four rounds, and each round has 16 operations. Four successive step operations are as follows:

$$\begin{aligned}
a &= b + ((a + \phi_i(b, c, d) + w_i + t_i) \lll s_i),\\
d &= a + ((d + \phi_{i+1}(a, b, c) + w_{i+1} + t_{i+1}) \lll s_{i+1}),\\
c &= d + ((c + \phi_{i+2}(d, a, b) + w_{i+2} + t_{i+2}) \lll s_{i+2}),\\
b &= c + ((b + \phi_{i+3}(c, d, a) + w_{i+3} + t_{i+3}) \lll s_{i+3}),
\end{aligned}$$

where the operation + means ADD modulo 2^{32}. t_{i+j} and s_{i+j} $(j = 0,\ 1,\ 2,\ 3)$ are step-dependent constants. w_{i+j} is a message word. $\lll s_{i+j}$ is circularly left-shift by s_{i+j} bit positions. The details of the message order and shift positions can be seen in Table 3.

Each round employs one nonlinear round function, which is given below.

$$\begin{aligned}
&\Phi_i(X, Y, Z) = (X \wedge Y) \vee (\neg X \wedge Z), && 0 \leq i \leq 15,\\
&\Phi_i(X, Y, Z) = (X \wedge Z) \vee (Y \wedge \neg Z), && 16 \leq i \leq 31,\\
&\Phi_i(X, Y, Z) = X \oplus Y \oplus Z, && 32 \leq i \leq 47,\\
&\Phi_i(X, Y, Z) = Y \oplus (X \vee \neg Z), && 48 \leq i \leq 63,
\end{aligned}$$

where X, Y, Z are 32-bit words.

The chaining variables are initialized as:

$$a = \texttt{0x67452301},\ b = \texttt{0xefcdab89},\ c = \texttt{0x98badcfe},\ d = \texttt{0x10325476}.$$

We select a collision differential with two iterations as follows: Let $H_{i-1} = (aa, bb, cc, dd)$ be the chaining values for the previous message block. After four rounds, the compression value H_i is obtained by wordwise addition of the chaining variables to H_{i-1}.

3 Differential Attack for Hash Functions

3.1 The Modular Differential and the XOR Differential

The most important analysis method for hash functions is differential attack which is also one of most important methods for analyzing block ciphers. In general, the differential attack especially in block ciphers is a kind of XOR differential attack which uses exclusive-or as the difference. The differential attack was introduced by E. Biham and A. Shamir to analyze the security of DES-like cryptosystems. E. Biham and A. Shamir [1], described that differential cryptanalysis is a method which analyzes the effect of particular differences in plain text pairs on the differences of the resultant cipher text pairs.

The differential definition in this paper is a kind of precise differential which uses the difference in term of integer modular subtraction. A similar definition about the differential with the integer subtraction as the measure of difference were described in [5] for differential analysis of RC6.

We also use modular characteristics, which describe for each round with both the differences in term of integer modular subtraction and the differences in term of XOR. The combination of both kinds of differences give us more information than each of them keep by itself. For example, when the modular integer subtraction difference is $X' - X = 2^6$ for some value X, the XOR difference $X' \oplus X$ can have many possibilities, which are

1. One-bit difference in bit 7, i.e., `0x00000040`. In this case $X' - X = 2^6$ which means that bit 7 in X' is 1 and bit 7 in X is 0.
2. Two-bit difference, in which a different carry is transferred from bit 7 to bit 8, i.e., `0x000000C0`. In this case $X' - X = 2^6$, but the carry to bit 8 is different in X and X', so X'_7 is now 0, and $X_7 = 1$, while $X'_8 = 1$, and $X_8 = 0$. (i.e., bits 7 and 8 together in X' are 10 in binary, and in X there are 01 in binary).
3. Three-bit difference, in which a different carry is transferred from bit 7 to bit 8 and then to bit 9, i.e., `0x000001C0`. In this case bits 7, 8, and 9 in X' are 0, 0, and 1, respectively, and in X they are the complement of these values.
4. Similarly, there can be more carries to further bits, and the binary form of X' is $1000\ldots$, and of X is $0111\ldots$.
5. In case the former difference is negative, the XOR differences still look the same, but the values of X and X' are exchanged (i.e., X is of the form $1000\ldots$, and X' of the form $0111\ldots$).

In order to explain our attack clearly, we refer to the modular differences in the differential path (see Table 3) with both kinds of differences together, i.e., the difference is marked as a positive or a negative integer (modulo 2^{32}) and also with the XOR difference. But then the XOR difference is marked by the list of active bits with their relative sign, i.e., in the list of bits, the bits whose value in X is zero are marked without a sign, and the values whose value in X is 1 are marked with a negative sign. For example, the difference $-2^6, [7, 8, 9, \ldots, 22, -23]$ marks the integer modular subtraction difference $X' - X = -2^6$ (with $X' < X$), with many carries which start from bit 7 up to bit 23. All bits of X from bit 7 to bit 22 are 0, and bit 23 is 1, while all bits of X' from bit 7 to bit 22 are 1, and bit 23 is 0. A more complicated example is $-1 - 2^6 + 2^{23} - 2^{27}, [1, 2, 3, 4, 5, -6, 7, 8, 9, 10, 11, -12, -24, -25, -26, 27, 28, 29, 30, 31, -32]$, where the integer modular subtraction difference is composed of several (positive and negative) exponents of 2, and the XOR difference has many difference due to carries. Note that when the carry arrives to bit 32, a further (dropped) carry may happen, and then there is no negative sign in bit 32.

It should be noted that the modular differential has been used earlier to analyze some hash functions ([4], [7], [10]). Compared with these attacks, our attack has the following advantages:

1. Our attack is to find collisions with two iterations, i. e., each message in the collision includes two message blocks (1024-bit).

2. Our attack is a precise differential attack in which the characteristics are more restrictive than used, and that they gives values of bits in addition to the differences.
3. Our attack gives a set of sufficient conditions which ensure the differential to occur.
4. Our attack use a message modification technique to greatly improve the collision probability.

3.2 Differential Attacks on Hash Functions

The *difference* for two parameters X and X' is defined as $\Delta X = X' - X$. For any two messages M and M' with l-bit multiples, $M = (M_0, M_1, \cdots, M_{k-1})$, $M = ({M_0}', {M_1}', \cdots, {M_{k-1}}')$, a full *differential* for a hash function is defined as follows:

$$\Delta H_0 \overset{(M_0, M_0')}{\longrightarrow} \Delta H_1 \overset{(M_1, M_1')}{\longrightarrow} \Delta H_2 \overset{(M_2, M_2')}{\longrightarrow} \cdots\cdots \Delta H_{k-1} \overset{(M_{k-1}, M_{k-1}')}{\longrightarrow} \Delta H,$$

where ΔH_0 is the initial value difference which equals to zero. ΔH is the output difference for the two messages. $\Delta H_i = \Delta IV_i$ is the output difference for the i-th iteration, and also is the initial difference for the next iteration.

It is clear that if $\Delta H = 0$, there is a collision for M and M'. We call the differential that produces a collision *a collision differential*.

Provided that the hash function has 4 rounds, and each round has 16 step operations. For more details, we can represent the i-th iteration differential $\Delta H_i \overset{(M_i, M_i')}{\longrightarrow} \Delta H_{i+1}$ as follows:

$$\Delta H_i \overset{P_1}{\longrightarrow} \Delta R_{i+1,1} \overset{P_2}{\longrightarrow} \Delta R_{i+1,2} \overset{P_3}{\longrightarrow} \Delta R_{i+1,3} \overset{P_4}{\longrightarrow} \Delta R_{i+1,4} = \Delta H_{i+1}.$$

The round differential $\Delta R_{j-1} \longrightarrow \Delta R_j (j = 1, 2, 3, 4)$ with the probability P_j is expanded to the following differential characteristics.

$$\Delta R_{j-1} \overset{P_{j1}}{\longrightarrow} \Delta X_1 \overset{P_{j2}}{\longrightarrow} \cdots\cdots \overset{P_{j16}}{\longrightarrow} \Delta X_{16} = \Delta R_j,$$

where $\Delta X_{t-1} \overset{P_{jt}}{\longrightarrow} \Delta X_t, t = 1, 2, \cdots\cdots, 16$ is the differential characteristic in the t-th step of j-th round.

The probability P of the differential $\Delta H_i \overset{(M_i, M_i')}{\longrightarrow} \Delta H_{i+1}$ satisfies

$$P \geq \prod_{i=1}^{4} P_j \text{ and } P_j \geq \prod_{t=1}^{16} P_{jt}.$$

3.3 Optimized Collision Differentials for Hash Functions

In Section 3.1, we mentioned that our attack uses a message modification technique to improve the collision probability. According to the modification technique, we can get a rough method to search for optimized differentials (including collision differentials) of a hash function.

There are two kinds of message modifications:

1. For any two message blocks (M_i, M_i') and a 1-st round non-zero differential

$$\Delta H_i \overset{(M_i, M_i')}{\longrightarrow} \Delta R_{i+1,1}.$$

Our attack can easily modify M_i to guarantee the 1-st round differential to hold with probability $P_1 = 1$.
2. Using multi-message modification techniques, we can not only guarantee the first-round differential to hold with the probability 1, but also improve the second-round differential probability greatly.

To find an optimized differential for a hash function, it is better to select a message block difference which results in a last two-round differential with a high probability.

4 Differential Attack on MD5

4.1 Notation

Before presenting our attack, we first introduce some notation to simplify the discussion.

1. $M = (m_0, m_1, ..., m_{15})$ and $M' = (m_0', m_1', ..., m_{15}')$ represent two 512-bit messages. $\Delta M = (\Delta m_0, \Delta m_1, ..., \Delta m_{15})$ denotes the difference of two message blocks. That is, $\Delta m_i = m_i' - m_i$ is the $i - th$ word difference.
2. a_i, d_i, c_i, b_i respectively denote the outputs of the $(4i-3)$-th, $(4i-2)$-th $(4i-1)$-th and $4i$-th steps for compressing M, where $1 \le i \le 16$. a_i', b_i', c_i', d_i' are defined similarly.
3. $a_{i,j}$, $b_{i,j}$, $c_{i,j}$, $d_{i,j}$ represent respectively the $j - th$ bit of a_i, b_i, c_i, d_i, where the least significant bit is the 1-st bit, and the most significant bit is 32-th bit.
4. $\phi_{i,j}$ is the j-th bit of the output for the nonlinear function ϕ_i in the i-th step operation.
5. $\Delta x_{i,j} = x_{i,j}' - x_{i,j} = \pm 1$ is the bit difference that is produced by changing the j-bit of x_i. $x_i[j]$, $x_i[-j]$ (x can be a, b, c, d, ϕ) is the resulting values by only changing the $j-th$ bit of the word x_i. $x_i[j]$ is obtained by changing the j-th bit of x_i from 0 to 1, and $x_i[-j]$ is obtained by changing the j-th bit of x_i from 1 to 0.
6. $\Delta x_i[j_1, j_2, ..., j_l] = x_i[j_1, j_2, ..., j_l] - x_i$ denotes the difference that is produced by the changes of $j_1 - th$, $j_2 - th$, ..., $j_l - th$ bits of x_i. $x_i[\pm j_1, \pm j_2, ..., \pm j_l]$ is the value by change $j_1 - th$, $j_2 - th$, ..., $j_l - th$ bits of x_i. The "+" sign (usually is omitted) means that the bit is changed from 0 to 1, and the "–" sign means that the bit is changed from 1 to 0.

4.2 Collision Differentials for MD5

Our attack can find many real collisions which are composed of two 1024-bit messages (M_0, M_1) and $(M_0{}', M_1{}')$) with the original initial value IV_0 of MD5:

$IV_0 :\ a_0 =$ `0x67452301`, $b_0 =$ `0xefcdab89`, $c_0 =$ `0x98badcfe`, $d_0 =$ `0x10325476`.

We select a collision differential with two iterations as follows:

$$\Delta H_0 \overset{(M_0,M_0')}{\longrightarrow} \Delta H_1 \overset{(M_1,M_1')}{\longrightarrow} \Delta H = 0$$

where

$$\Delta M_0 = M_0' - M_0 = (0,0,0,0,2^{31},0,0,0,0,0,0,2^{15},0,0,2^{31},0)$$
$$\Delta M_1 = M_1' - M_1 = (0,0,0,0,2^{31},0,0,0,0,0,0,-2^{15},0,0,2^{31},0)$$
$$\Delta H_1 = (2^{31}, 2^{31}+2^{25}, 2^{31}+2^{25}, 2^{31}+2^{25}).$$

Non-zero entries of ΔM_0 and ΔM_1 are located at positions 5, 12 and 15. $\Delta H_1 = (\Delta a, \Delta b, \Delta c, \Delta d)$ is the difference of the four chaining values (a, d, c, b) after the first iteration.

We select ΔM_0 to ensure that both 3-4 round differential happens with a high probability. ΔM_1 is selected not only to ensure both 3-4 round differential happens with a high probability, but also to produce an output difference that can be cancelled with the output difference ΔH_1.

The collision differential with all the characteristics can be referred to Table 3 and Table 5. The columns of both tables have the same meanings. We just give the explanation for Table 3. The first column denotes the step, the second column is the chaining variable in each step for M_0, the third is the message word for M_0 in each step, the fourth is shift rotation, the fifth and the sixth are respectively the message word difference and chaining variable difference for M_0 and M_0', and the seventh is the chaining variable for M_0'. Especially, the empty items both in sixth and fifth columns denote zero differences, and steps those aren't listed in the table have zero differences both for message words and chaining variables.

4.3 Sufficient Conditions for the Characteristics to Hold

In what follows, we describe how to derive a set of sufficient conditions that guarantee the differential characteristic in Step 8 of MD5 (Table 3) to hold. Other conditions can be derived similarly.

The differential characteristic in Step 8 of MD5 is:

$$(\Delta c_2, \Delta d_2, \Delta a_2, \Delta b_1) \longrightarrow \Delta b_2.$$

Each chaining variable satisfies one of the following equations.

$b_1' = b_1$
$a_2' = a_2[7, ..., 22, -23]$
$d_2' = d_2[-7, 24, 32]$
$c_2' = c_2[7, 8, 9, 10, 11, -12, -24, -25, -26, 27, 28, 29, 30, 31, 32, 1, 2, 3, 4, 5, -6]$
$b_2' = b_2[1, 16, -17, 18, 19, 20, -21, -24]$

According to the operations in the 8-th step, we have

$$b_2 = c_2 + ((b_1 + F(c_2, d_2, a_2) + m_7 + t_7) \lll 22$$

$$b_2' = c_2' + ((b_1 + F(c_2', d_2', a_2') + m_7' + t_7) \lll 22$$

$$\phi_7 = F(c_2, d_2, a_2) = (c_2 \wedge d_2) \vee (\neg c_2 \wedge a_2)$$

In the above operations, c_2 occurs twice in the right hand side of the equation. In order to distinguish the two, let c_2^F denote the c_2 inside F, and c_2^{NF} denote the c_2 outside F.

The derivation is based on the following two facts:

1. Since $\Delta b_1 = 0$ and $\Delta m_7 = 0$, we know that $\Delta b_2 = \Delta c_2^{NF} + (\Delta\phi_7 \lll 22)$.
2. Fix one or two of the variables in F so that F is reduced to a single variable.

We get a set of sufficient conditions that ensure the differential characteristic holds.

1. **The conditions for each of the non-zero bits in Δb_2.**
   (a) The conditions $d_{2,11} = 1$ and $b_{2,1} = 0$ ensure the change of 1-st bit of b_2.
      i. If $d_{2,11} = \overline{a_{2,11}} = 1$, we know that $\Delta\phi_{7,11} = 1$.
      ii. After $\lll 22$, $\Delta\phi_{7,11}$ is in the position 1.
      iii. Since $\Delta c_{2,1}^{NF} = 0$, so, $\Delta b_{2,1} = \Delta c_{2,1}^{NF} + \Delta\phi_{7,11} = 1$.
   (b) The conditions $d_{2,26} = \overline{a_{2,26}} = 1$, $b_{2,16} = 0$ and $b_{2,17} = 1$ ensure the changes of 16-th bit and 17-th bit of b_2.
   (c) The conditions $d_{2,28} = \overline{a_{2,28}} = 0$, $b_{2,i} = 0, i = 18, 19, 20$ and $b_{2,21} = 1$ ensure the changes of 18-th, 19-th, 20-th, 21-th bits of b_2.
   (d) The conditions $d_{2,3} = \overline{a_{2,3}} = 0$ and $b_{2,24} = 1$ ensure the change of 24-th bit of b_2. This can be proven by the equation:

$$\Delta c_2^{NF}[-24, -25, -26, 27] + (\Delta\phi_7[3] \lll 22) = 2^{23} - 2^{24} = -2^{23}.$$

2. **The conditions for each of the zero bits in Δb_2.**
   (a) The condition $c_{2,17} = 0$ ensures the changed bits from 7-th bit to 12-th bit in ${c_2'}^{NF}$ and 17-th bit of a_2' result in no bit change in b_2. It is easily proven by the following equation:

$$\Delta c_2^{NF}[7, \ldots 11, -12] + (\Delta\phi_7[17] \lll 22) = -2^6 + 2^6 = 0.$$

   (b) The conditions $d_{2,i} = a_{2,i}$ ensure that the changed i-th bit in c_2^F result in no change in b_2, where $i \in \{1, 2, 4, 5, 25, 27, 29, 30, 31\}$.
   (c) The conditions $c_{2,i} = 1$ ensure that the changed i-th bit in a_2 result in no change in b_2, where $i \in \{13, 14, 15, 16, 18, 19, 20, 21, 22, 23\}$.
   (d) The condition $d_{2,6} = \overline{a_{2,6}} = 0$ ensures that the 6-th bit in c_2^F result in no change in b_2.
   (e) The condition $a_{2,32} = 1$ ensures that the changed 32-th bit in c_2^F and the 32-th bit in d_2 result in no change in b_2.
   (f) The condition $d_{2,i} = 0$ ensures that the changed i-th bit in a_2 and the i-th bit in c_2^F result in no change in b_2, where $i \in \{8, 9, 10\}$.
   (g) The condition $d_{2,12} = 1$ ensures that the changed 12-th bit in a_2 and the 12-th bit in c_2^F result in no change in b_2.

(h) The condition $a_{2,24} = 0$ ensures that the changed 24-th bit in c_2^F and the 24-th bit in d_2 result in no change in b_2.
(i) The changed 7-th bits in c_2^F, d_2 and a_2 result in no change in b_2.

By the similar method, we can derive a set of sufficient conditions (see Table 4 and Table 6) which guarantee all the differential characteristics in the collision differential to hold.

4.4 Message Modification

Single-message Modification In order to make the attack efficient, it is very attractive to improve over the probabilistic method that we describe, by fixing some of the message words to a prior fulfilling some of the conditions. We observe that it is very easy to generate messages that fulfill all the conditions of the first 16 steps of MD5. We call it *single-message modification*.

For each message block M_0 (or similarly M_1) and intermediate values (H_0, or for the second block H_1 and H_1'), we apply the following procedures to modify M_0 (or M_1, respectively), so that all the conditions of round 1 (the first 16 steps) in Table 4 and Table 6 hold.

It is easy to modify M_0 such that the conditions of round 1 in Table 4 hold with probability 1.

For example, to ensure that 3 conditions for c_1 in Table 4 hold, we modify m_2 as follows:

$$c_1^{new} \leftarrow c_1^{old} - c_{1,7}^{old} \cdot 2^6 - c_{1,12}^{old} \cdot 2^{11} - c_{1,20}^{old} \cdot 2^{19}$$

$$m_2^{new} \leftarrow ((c_1^{new} - c_1^{old}) \ggg 17) + m_2^{old}.$$

By modifying each message word of message M_0, all the conditions in round 1 of Table 4 hold. The first iteration differential hold with probability 2^{-43}.

The same modification is applied to M_1. After modification, the second iteration differential hold with probability 2^{-37}.

Multi-message Modification We further observe that it is even possible to fulfill a part of the conditions of the first 32 steps by an *multi-message modification*.

For example, if $a_{5,32} = 1$, we correct it into $a_{5,32} = 0$ by modifying m_1, m_2,m_3, m_4,m_5 such that the modification generates a partial collision from 2-6 steps, and remains that all the conditions in round 1 hold. See Table 1. Some other conditions can be corrected by the similar modification technique or other more precise modification techniques. By our modification, 37 conditions in round 2-4 are undetermined in the table 4, and 30 conditions in round 2-4 are undetermined in the table 6. So, the 1-st iteration differential holds with probability 2^{-37}, and the second iteration differential holds with probability 2^{-30}.

Table 1. The Message Modification for Correcting $a_{5,32}$

			Modify m_i	a^{new}, b^{new}, c^{new}, d^{new}
2	m_1	12	$m_1 \longleftarrow m_1 + 2^{26}$	d_1^{new}, a_1, b_0, c_0
3	m_2	17	$m_2 \longleftarrow ((c_1 - d_1^{new}) \ggg 17) - c_0 - \phi_2(d_1^{new}, a_1, b_0) - t_2$	c_1, d_1^{new}, a_1, b_0
4	m_3	22	$m_3 \longleftarrow (b_1 - c_1) \ggg 22) - b_0 - \phi_3(c_1, d_1^{new}, a_1) - t_3$	b_1, c_1, d_1^{new}, a_1
5	m_4	7	$m_4 \longleftarrow ((a_2 - b_1) \ggg 7) - a_1 - \phi_4(b_1, c_1, d_1^{new} - t_4$	a_2, b_1, c_1, d_1^{new}
6	m_5	12	$m_5 \longleftarrow ((d_2 - a_2) \ggg 12) - d_1^{new} - \phi_5(a_2, b_1, c_1) - t_5$	d_2, a_2, b_1, c_1

4.5 The Differential Attack on MD5

From the above description, it is very easy to show our attack on MD5.

The following is to describe how to find a two-block collision, of the following form

$$H_0 \overset{(M_0,M_0'),2^{-37}}{\longrightarrow} \Delta H_1 \overset{(M_1,M_1'),2^{-30}}{\longrightarrow} \Delta H = 0.$$

1. Repeat the following steps until a first block is found
   (a) Select a random message M_0.
   (b) Modify M_0 by the message modification techniques described in the previous subsection.
   (c) Then, M_0 and $M_0' = M_0 + \Delta M_0$ produce the first iteration differential
   $$\Delta M_0 \longrightarrow (\Delta H_1, \Delta M_1)$$
   with the probability 2^{-37}.
   (d) Test if all the characteristics really hold by applying the compression function on M_0 and M_0'.
2. Repeat the following steps until a collision is found
   (a) Select a random message M_1.
   (b) Modify M_1 by the message modification techniques described in the previous subsection.
   (c) Then, M_1 and $M_1 + \Delta M_1$ generate the second iteration differential
   $$(\Delta H_1, \Delta M_1) \longrightarrow \Delta H = 0$$
   with the probability 2^{-30}.
   (d) Test if this pair of messages lead to a collision.

The complexity of finding (M_0, M_0') doesn't exceed the time of running 2^{39} MD5 operations. To select another message M_0 is only to change the last two words from the previous selected message M_0. So, finding (M_0, M_0') only needs about one-time single-message modification for the first 14 words. This time can be neglected. For each selected message M_0, it is only needs two-time single-message modifications for the last two words and 7-time multi-message modifications for correcting 7 conditions in the second round, and each multi-message modification only needs about a few step operations, so the total time for both kinds of modifications is not exceeds about two MD5 operations for each selected message. According to the probability of the first iteration differential, it is easy to know that the complexity of finding (M_0, M'_0) is not exceeds 2^{39} MD5 operations.

Similarly, we can show that the complexity of finding (M_1, M'_1) is not exceeds 2^{32} MD5 operations.

Two collisions of MD5 are given in Table 2. It is noted that the two collisions start with the same 1-st 512-bit block, and that given a first block that satisfies all the required criteria, it is easy to find many second blocks M_1, M'_1 which lead to collisions.

Table 2. Two pairs of collision for MD5. H is the hash value with little-endian and no message padding, and H^* is the hash value with big-endian and message padding.

M_0	2dd31d1	c4eee6c5	69a3d69	5cf9af98	87b5ca2f	ab7e4612	3e580440	897ffbb8
	634ad55	2b3f409	8388e483	5a417125	e8255108	9fc9cdf7	f2bd1dd9	5b3c3780
M_1	d11d0b96	9c7b41dc	f497d8e4	d555655a	c79a7335	cfdebf0	66f12930	8fb109d1
	797f2775	eb5cd530	baade822	5c15cc79	ddcb74ed	6dd3c55f	d80a9bb1	e3a7cc35
$M_0{}'$	2dd31d1	c4eee6c5	69a3d69	5cf9af98	7b5ca2f	ab7e4612	3e580440	897ffbb8
	634ad55	2b3f409	8388e483	5a41f125	e8255108	9fc9cdf7	72bd1dd9	5b3c3780
$M_1{}'$	d11d0b96	9c7b41dc	f497d8e4	d555655a	479a7335	cfdebf0	66f12930	8fb109d1
	797f2775	eb5cd530	baade822	5c154c79	ddcb74ed	6dd3c55f	580a9bb1	e3a7cc35
H	9603161f	a30f9dbf	9f65ffbc	f41fc7ef				
H^*	a4c0d35c	95a63a80	5915367d	cfe6b751				
M_0	2dd31d1	c4eee6c5	69a3d69	5cf9af98	87b5ca2f	ab7e4612	3e580440	897ffbb8
	634ad55	2b3f409	8388e483	5a417125	e8255108	9fc9cdf7	f2bd1dd9	5b3c3780
M_1	313e82d8	5b8f3456	d4ac6dae	c619c936	b4e253dd	fd03da87	6633902	a0cd48d2
	42339fe9	e87e570f	70b654ce	1e0da880	bc2198c6	9383a8b6	2b65f996	702af76f
$M_0{}'$	2dd31d1	c4eee6c5	69a3d69	5cf9af98	7b5ca2f	ab7e4612	3e580440	897ffbb8
	634ad55	2b3f409	8388e483	5a41f125	e8255108	9fc9cdf7	72bd1dd9	5b3c3780
$M_1{}'$	313e82d8	5b8f3456	d4ac6dae	c619c936	34e253dd	fd03da87	6633902	a0cd48d2
	42339fe9	e87e570f	70b654ce	1e0d2880	bc2198c6	9383a8b6	ab65f996	702af76f
H	8d5e7019	61804e08	715d6b58	6324c015				
H^*	79054025	255fb1a2	6e4bc422	aef54eb4				

5 Summary

In this paper we described a powerful attack against hash functions, and in particular showed that finding a collision of MD5 is easily feasible.

Our attack is also able to break efficiently other hash functions, such as HAVAL-128, MD4, RIPEMD, and SHA-0. The analysis results for these hash functions are as follows:

1. The time complexity for finding a collision for MD4 is about 2^{23} MD4 operations without the multi-message modification, and is about 2^8 MD4 operations with the multi-message modification.
2. The time complexity for finding a collision for HAVAL-128 is about 2^{13} MD4 operations without the multi-message modification, and is 2^7 HAVAL-128 operations with the multi-message modification.
3. The time complexity for finding a collision for RIPEMD is about 2^{30} RIPEMD operations without the multi-message modification, and is 2^{18} RIPEMD operations with the multi-message modification.
4. The time complexity for finding a collision for SHA-0 is about 2^{61} SHA-0 operations without the multi-message modification, and is 2^{45} SHA-0 operations with the multi-message modification.

Acknowledgements

It is a pleasure to acknowledge Dengguo Feng for the conversations that led to this research on MD5. We would like to thank Eli Biham, Andrew C. Yao, and Yiqun Lisa Yin for their important advice, corrections, and suggestions, and for spending their precious time on our research. We would also like to thank Xuejia Lai, Hans Dobbertin, Magnus Daum for various discussions on this paper. The research is supported by the National Natural Science Foundation of China (Grant No. 90304009).

References

1. E. Biham, A. Shamir. Differential Cryptanalysis of the Data Encryption Standard, Springer-Verlag, 1993.
2. E. Biham, R. Chen, *Near collision for SHA-0,* Advances in Cryptology, Crypto'04, 2004, LNCS 3152, pp. 290-305.
3. B. den. Boer, A. Bosselaers. Collisions for the compression function of MD5, Advances in Cryptology, Eurocrypt'93 Proceedings, Springer-Verlag, 1994.
4. F. Chabaud, A. Joux. Differential collisions in SHA-0, Advances in Cryptology, Crypto'98 Proceedings, Springer-Verlag, 1998.
5. S. Cotini, R.L. Rivest, M.J.B. Robshaw, Y. Lisa Yin. Security of the RC6^{TM} Block Cipher, `http://www.rsasecurity.com/rsalabs/rc6/`.
6. I. B. Damgard. A design principle for hash functions, Advances in Cryptology, Crypto'89 Proceedings, Springer-Verlag, 1990.
7. H. Dobbertin. Cryptanalysis of MD4, Fast Software Encryption, LNCS 1039, Springer-Verlag, 1996, 53-69.
8. H. Dobbertin. Cryptanalysis of MD5 compress, presented at the rump session of Eurocrypt'96.
9. H. Dobbertin. The status of MD5 after a recent attack, CryptoBytes 2 (2), 1996, `ftp://ftp.rsasecurity.com/pub/cryptobytes/crypto2n2.pdf`.
10. H. Dobbertin. RIPEMD with two round compress function is not collision-free, Journal of Cryptology, 10:51-69, 1997.
11. H. Dobbertin, A. Bosselaers, B. Preneel. RIPEMD-160: A strengthened version of RIPEMD, Fast Software Encryption, LNCS 1039, Springer-Verlag, 1996.

12. FIPS 180-1. Secure hash standard, NIST, US Department of Commerce, Washington D.C., Springer-Verlag, 1996.
13. FIPS 180-2. Secure Hash Standard, `http://csrc.nist.gov/publications/`, 2002.
14. A. Joux. Collisions for SHA-0, rump session of Crypto'04, 2004.
15. RIPE. Integrity Primitives for Secure Information Systems. Final Report of RACE Integrity Primitives Evaluation (RIPE-RACE 1040), LNCS 1007, Springer-Verlag, 1995.
16. R.C. Merkle. One way hash function and DES, Advances in Cryptology, Crypto'89 Proceedings, Springer-Verlag, 1990.
17. R.L. Rivest. The MD4 message digest algorithm, Advances in Cryptology, Crypto'90, Springer-Verlag, 1991, 303-311.
18. R.L. Rivest. The MD5 message-digest algorithm, Request for Comments (RFC 1320), Internet Activities Board, Internet Privacy Task Force, 1992.
19. X.Y. Wang, F.D. Guo, X.J. Lai, H.B. Yu, Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMD, rump session of Crypto'04, E-print, 2004.
20. Y.L. Zheng, J. Pieprzyk, J. Seberry. HAVAL–A one-way hashing algorithm with variable length of output, Advances in Cryptology, Auscrypt'92 Proceedings, Springer-Verlag.

Table 3. The Differential Characteristics in the First Iteration Differential

Step	The output in i-th step for M_0	w_i	s_i	Δw_i	The output difference in i-th step	The output in i-th step for M_0'
4	b_1	m_3	22			
5	a_2	m_4	7	2^{31}	-2^6	$a_2[7,\ldots,22,-23]$
6	d_2	m_5	12		$-2^6+2^{23}+2^{31}$	$d_2[-7,24,32]$
7	c_2	m_6	17		$-1-2^6+2^{23}-2^{27}$	$c_2[7,8,9,10,11,-12,-24,-25,-26,$ $27,28,29,30,31,32,1,2,3,4,5,-6]$
8	b_2	m_7	22		$1-2^{15}-2^{17}-2^{23}$	$b_2[1,16,-17,18,19,20,-21,-24],$
9	a_3	m_8	7		$1-2^6+2^{31}$	$a_3[-1,2,7,8,-9,-32]$
10	d_3	m_9	12		$2^{12}+2^{31}$	$d_3[-13,14,32]$
11	c_3	m_{10}	17		$2^{30}+2^{31}$	$c_3[31,32]$
12	b_3	m_{11}	22	2^{15}	$-2^7-2^{13}+2^{31}$	$b_3[8,-9,14,\ldots,19,-20,32]$
13	a_4	m_{12}	7		$2^{24}+2^{31}$	$a_4[-25,26,32]$
14	d_4	m_{13}	12		2^{31}	$d_4[32]$
15	c_4	m_{14}	17	2^{31}	$2^3-2^{15}+2^{31}$	$c_4[4,-16,32]$
16	b_4	m_{15}	22		$-2^{29}+2^{31}$	$b_4[-30,32]$
17	a_5	m_1	5		2^{31}	$a_5[32]$
18	d_5	m_6	9		2^{31}	$d_5[32]$
19	c_5	m_{11}	14	2^{15}	$2^{17}+2^{31}$	$c_5[18,32]$
20	b_5	m_0	20		2^{31}	$b_5[32]$
21	a_6	m_5	5		2^{31}	$a_6[32]$
22	d_6	m_{10}	9		2^{31}	$d_6[32]$
23	c_6	m_{15}	14			c_6
24	b_6	m_4	20	2^{31}		b_6
25	a_7	m_9	5			a_7
26	d_7	m_{14}	9	2^{31}		d_7
27	c_7	m_3	14			c_7
...	...	...	...	...	...	...
34	d_9	m_8	11			d_9
35	c_9	m_{11}	16	2^{15}	2^{31}	$c_9[*32]$
36	b_9	m_{14}	23	2^{31}	2^{31}	$b_9[*32]$
37	a_{10}	m_1	4		2^{31}	$a_{10}[*32]$
38	d_{10}	m_4	11	2^{31}	2^{31}	$d_{10}[*32]$
39	c_{10}	m_7	16		2^{31}	$c_{10}[*32]$
...	...	...	...	...	...	...
45	a_{12}	m_9	4		2^{31}	$a_{12}[*32]$
46	d_{12}	m_{12}	11		2^{31}	$d_{12}[32]$
47	c_{12}	m_{15}	16		2^{31}	$c_{12}[32]$
48	b_{12}	m_2	23		2^{31}	$b_{12}[32]$
49	a_{13}	m_0	6		2^{31}	$a_{13}[32]$
50	d_{13}	m_7	10		2^{31}	$d_{13}[-32]$
51	c_{13}	m_{14}	15	2^{31}	2^{31}	$c_{13}[32]$
52	b_{13}	m_5	21		2^{31}	$b_{13}[-32]$
...	...	...	...	...	...	...
58	d_{15}	m_{15}	10		2^{31}	$d_{15}[-32]$
59	c_{15}	m_6	15		2^{31}	$c_{15}[32]$
60	b_{15}	m_{13}	21		2^{31}	$b_{15}[32]$
61	$aa_0=a_{16}+a_0$	m_4	6	2^{31}	2^{31}	$aa_0'=aa_0[32]$
62	$dd_0=d_{16}+d_0$	m_{11}	10	2^{15}	2^{31}	$dd_0'=dd_0[26,32]$
63	$cc_0=c_{16}+c_0$	m_2	15		2^{31}	$cc_0'=cc_0[-26,27,32]$
64	$bb_0=b_{16}+b_0$	m_9	21		2^{31}	$bb_0'=bb_0[26,-32]$

Table 4. A Set of Sufficient Conditions for the First Iteration Differential

c_1	$c_{1,7}=0,\ c_{1,12}=0,\ c_{1,20}=0$
b_1	$b_{1,7}=0,\ b_{1,8}=c_{1,8},\ b_{1,9}=c_{1,9},\ b_{1,10}=c_{1,10},\ b_{1,11}=c_{1,11},\ b_{1,12}=1,\ b_{1,13}=c_{1,13},$ $b_{1,14}=c_{1,14},\ b_{1,15}=c_{1,15},\ b_{1,16}=c_{1,16},\ b_{1,17}=c_{1,17},\ b_{1,18}=c_{1,18},b_{1,19}=c_{1,19},$ $b_{1,20}=1,\ b_{1,21}=c_{1,21},\ b_{1,22}=c_{1,22},\ b_{1,23}=c_{1,23},\ b_{1,24}=0,\ b_{1,32}=1$
a_2	$a_{2,1}=1,\ a_{2,3}=1,\ a_{2,6}=1,\ a_{2,7}=0,\ a_{2,8}=0,\ a_{2,9}=0,\ a_{2,10}=0,\ a_{2,11}=0,$ $a_{2,12}=0,\ a_{2,13}=0,\ a_{2,14}=0,\ a_{2,15}=0,\ a_{2,16}=0,\ a_{2,17}=0,\ a_{2,18}=0,\ a_{2,19}=0,$ $a_{2,20}=0,\ a_{2,21}=0,\ a_{2,22}=0,\ a_{2,23}=1,\ a_{2,24}=0,\ a_{2,26}=0,\ a_{2,28}=1,\ a_{2,32}=1$
d_2	$d_{2,1}=1,\ d_{2,2}=a_{2,2},\ d_{2,3}=0,\ d_{2,4}=a_{2,4},\ d_{2,5}=a_{2,5},\ d_{2,6}=0,\ d_{2,7}=1,\ d_{2,8}=0,$ $d_{2,9}=0,\ d_{2,10}=0,\ d_{2,11}=1,\ d_{2,12}=1,\ d_{2,13}=1,\ d_{2,14}=1,\ d_{2,15}=0,\ d_{2,16}=1,$ $d_{2,17}=1,\ d_{2,18}=1,\ d_{2,19}=1,\ d_{2,20}=1,\ d_{2,21}=1,\ d_{2,22}=1,\ d_{2,23}=1,\ d_{2,24}=0,$ $d_{2,25}=a_{2,25},\ d_{2,26}=1,\ d_{2,27}=a_{2,27},\ d_{2,28}=0,\ d_{2,29}=a_{2,29},\ d_{2,30}=a_{2,30},$ $d_{2,31}=a_{2,31},\ d_{2,32}=0$
c_2	$c_{2,1}=0,\ c_{2,2}=0,\ c_{2,3}=0,\ c_{2,4}=0,\ c_{2,5}=0,\ c_{2,6}=1,\ c_{2,7}=0,\ c_{2,8}=0,\ c_{2,9}=0,$ $c_{2,10}=0,\ c_{2,11}=0,\ c_{2,12}=1,\ c_{2,13}=1,\ c_{2,14}=1,\ c_{2,15}=1,\ c_{2,16}=1,\ c_{2,17}=0,$ $c_{2,18}=1,\ c_{2,19}=1,\ c_{2,20}=1,\ c_{2,21}=1,\ c_{2,22}=1,\ c_{2,23}=1,\ c_{2,24}=1,\ c_{2,25}=1,$ $c_{2,26}=1,\ c_{2,27}=0,\ c_{2,28}=0,\ c_{2,29}=0,\ c_{2,30}=0,\ c_{2,31}=0,\ c_{2,32}=0$
b_2	$b_{2,1}=0,\ b_{2,2}=0,\ b_{2,3}=0,\ b_{2,4}=0,\ b_{2,5}=0,\ b_{2,6}=0,\ b_{2,7}=1,\ b_{2,8}=0,\ b_{2,9}=1,$ $b_{2,10}=0,\ b_{2,11}=1,\ b_{2,12}=0,\ b_{2,14}=0,\ b_{2,16}=0,\ b_{2,17}=1,\ b_{2,18}=0,\ b_{2,19}=0,$ $b_{2,20}=0,\ b_{2,21}=1,\ b_{2,24}=1,\ b_{2,25}=1,\ b_{2,26}=0,\ b_{2,27}=0,\ b_{2,28}=0,\ b_{2,29}=0,$ $b_{2,30}=0,\ b_{2,31}=0,\ b_{2,32}=0$
a_3	$a_{3,1}=1,\ a_{3,2}=0,\ a_{3,3}=1,\ a_{3,4}=1,\ a_{3,5}=1,\ a_{3,6}=1,\ a_{3,7}=0,\ a_{3,8}=0,\ a_{3,9}=1,$ $a_{3,10}=1,\ a_{3,11}=1,\ a_{3,12}=1,\ a_{3,13}=b_{2,13},\ a_{3,14}=1,\ a_{3,16}=0,\ a_{3,17}=0,\ a_{3,18}=0,$ $a_{3,19}=0,\ a_{3,20}=0,\ a_{3,21}=1,\ a_{3,25}=1,\ a_{3,26}=1,\ a_{3,27}=0,\ a_{3,28}=1,\ a_{3,29}=1,$ $a_{3,30}=1,\ a_{3,31}=1,\ a_{3,32}=1$
d_3	$d_{3,1}=0,\ d_{3,2}=0,\ d_{3,7}=1,\ d_{3,8}=0,\ d_{3,9}=0,\ d_{3,13}=1,\ d_{3,14}=0,\ d_{3,16}=1,$ $d_{3,17}=1,\ d_{3,18}=1,\ d_{3,19}=1,\ d_{3,20}=1,\ d_{3,21}=1,\ d_{3,24}=0,\ d_{3,31}=1,\ d_{3,32}=0$
c_3	$c_{3,1}=0,\ c_{3,2}=1,\ c_{3,7}=1,\ c_{3,8}=1,\ c_{3,9}=0,\ c_{3,13}=0,\ c_{3,14}=0,\ c_{3,15}=d_{3,15},$ $c_{3,17}=1,\ c_{3,18}=0,\ c_{3,19}=0,\ c_{3,20}=0,\ c_{3,16}=1,\ c_{3,31}=0,\ c_{3,32}=0$
b_3	$b_{3,8}=0,\ b_{3,9}=1,\ b_{3,13}=1,\ b_{3,14}=0,\ b_{3,15}=0,\ b_{3,16}=0,\ b_{3,17}=0,\ b_{3,18}=0,$ $b_{3,20}=1,\ b_{3,25}=c_{3,25},\ b_{3,26}=c_{3,26},\ b_{3,19}=0,\ b_{3,31}=0,\ b_{3,32}=0$
a_4	$a_{4,4}=1,\ a_{4,8}=0,\ a_{4,9}=0,\ a_{4,14}=1,\ a_{4,15}=1,\ a_{4,16}=1,\ a_{4,17}=1,\ a_{4,18}=1,$ $a_{4,20}=1,\ a_{4,25}=1,\ a_{4,26}=0,\ a_{4,31}=1,\ a_{4,19}=1,\ a_{4,32}=0$
d_4	$d_{4,4}=1,\ d_{4,8}=1,\ d_{4,9}=1,\ d_{4,14}=1,\ d_{4,15}=1,\ d_{4,16}=1,\ d_{4,17}=1,\ d_{4,18}=1,$ $d_{4,19}=0,\ d_{4,20}=1,\ d_{4,25}=0,\ d_{4,26}=0,\ d_{4,30}=0,\ d_{4,32}=0$
c_4	$c_{4,4}=0,\ c_{4,16}=1,\ c_{4,25}=1,\ c_{4,26}=0,\ c_{4,30}=1,\ c_{4,32}=0$
b_4	$b_{4,30}=1,\ b_{4,32}=0$
a_5	$a_{5,4}=b_{4,4},\ a_{5,16}=b_{4,16},\ a_{5,18}=0,\ a_{5,32}=0$
d_5	$d_{5,18}=1,\ d_{5,30}=a_{5,30},\ d_{5,32}=0$
c_5	$c_{5,18}=0,\ c_{5,32}=0$
b_5	$b_{5,32}=0$
a_6-b_6	$a_{6,18}=b_{5,18},\ a_{6,32}=0,\ d_{6,32}=0,\ c_{6,32}=0,\ b_{6,32}=c_{6,32}+1$
$c_9,\ b_{12}$	$\phi_{34,32}=0,\ b_{12,32}=d_{12,32}$
$a_{13}-b_{13}$	$a_{13,32}=c_{12,32},\ d_{13,32}=b_{12,32}+1,\ c_{13,32}=a_{13,32},\ b_{13,32}=d_{13,32}$
$a_{14}-b_{14}$	$a_{14,32}=c_{13,32},\ d_{14,32}=b_{13,32},\ c_{14,32}=a_{14,32},\ b_{14,32}=d_{14,32}$
a_{15}	$a_{15,32}=c_{14,32}$
d_{15}	$d_{15,32}=b_{14,32}$
c_{15}	$c_{15,32}=a_{15,32}$
b_{15}	$b_{15,26}=0,\ b_{15,32}=d_{15,32}+1$
$aa_0=a_{16}+a_0$	$a_{16,26}=1,\ a_{16,27}=0,\ a_{16,32}=c_{15,32}$
$dd_0=d_{16}+d_0$	$dd_{0,26}=0,\ d_{16,32}=b_{15,32}$
$cc_0=c_{16}+c_0$	$cc_{0,26}=1,\ cc_{0,27}=0,\ cc_{0,32}=dd_{0,32},\ c_{16,32}=d_{16,32}$
$bb_0=b_{16}+b_0$	$bb_{0,26}=0, bb_{0,27}=0, bb_{0,6}=0, bb_{0,32}=cc_{0,32}$

Table 5. All the Differential Characteristics in the Second Iteration Differential

Step	The output in i-th step for M_1	w_i	s_i	Δw_i	The output Difference in i-th step	The output in i-th step for M_1'
IV	aa_0, dd_0 cc_0, bb_0					$aa_0[32]$, $dd_0[26,32]$ $cc_0[-26,27,32]$, $bb_0[26,-32]$
1	a_1	m_0	7		$2^{25}+2^{31}$	$a_1[26,-32]$
2	d_1	m_1	12		$2^5+2^{25}+2^{31}$	$d_1[6,26,-32]$
3	c_1	m_2	17		$2^5+2^{11}+2^{16}$ $+2^{25}+2^{31}$	$c_1[-6,-7,8,-12,13,$ -17,...,-21,22,-26,...,-30,31,-32]
4	b_1	m_3	22		$-2+2^5+2^{25}+2^{31}$	$b_1[2,3,4,-5,6,-26,27,-32]$
5	a_2	m_4	7	2^{31}	$1+2^6+2^8+2^9+2^{31}$	$a_2[1,-7,8,9,-10,-11,-12,13,32]$
6	d_2	m_5	12		$-2^{16}-2^{20}+2^{31}$	$d_2[17,-18,21,-22,32]$
7	c_2	m_6	17		$-2^6-2^{27}+2^{31}$	$c_2[7,8,9,-10,28,-29,-32]$
8	b_2	m_7	22		$2^{15}-2^{17}-2^{23}+2^{31}$	$b_2[-16,17,-18,24,25,26,-27,-32]$
9	a_3	m_8	7		$1+2^6+2^{31}$	$a_3[-1,2,-7,-8,-9,10,-32]$
10	d_3	m_9	12		$2^{12}+2^{31}$	$d_3[13,-32]$
11	c_3	m_{10}	17		2^{31}	$c_3[-32]$
12	b_3	m_{11}	22	-2^{15}	$-2^7-2^{13}+2^{31}$	$b_3[-8,14,15,16,17,18,19,-20,-32]$
13	a_4	m_{12}	7		$2^{24}+2^{31}$	$a_4[-25,\ldots,-30,31,32]$
14	d_4	m_{13}	12		2^{31}	$d_4[32]$
15	c_4	m_{14}	17	2^{31}	$2^3+2^{15}+2^{31}$	$c_4[4,16,32]$
16	b_4	m_{15}	22		$-2^{29}+2^{31}$	$b_4[-30,32]$
17	a_5	m_1	5		2^{31}	$a_5[32]$
18	d_5	m_6	9		2^{31}	$d_5[32]$
19	c_5	m_{11}	14	-2^{15}	$2^{17}+2^{31}$	$c_5[18,32]$
20	b_5	m_0	20		2^{31}	$b_5[32]$
21	a_6	m_5	5		2^{31}	$a_6[32]$
22	d_6	m_{10}	9		2^{31}	$d_6[32]$
23	c_6	m_{15}	14			$c_6[32]$
24	b_6	m_4	20	2^{31}		$b_6[32]$
25	a_7	m_9	5			a_7
26	d_7	m_{14}	9	2^{31}		d_7
27	c_7	m_3	14			c_7
...	...	...	...	...	...	...
34	d_9	m_8	11			d_9
35	c_9	m_{11}	16	-2^{15}	2^{31}	$c_9[*32]$
36	b_9	m_{14}	23	2^{31}	2^{31}	$d_9[*32]$
37	a_{10}	m_1	4		2^{31}	$a_{10}[*32]$
38	d_{10}	m_4	11	2^{31}	2^{31}	$d_{10}[*32]$
39	c_{10}	m_7	16		2^{31}	$c_{10}[*32]$
...	...	...	...	...	...	...
49	a_{13}	m_0	6		2^{31}	$a_{13}[32]$
50	d_{13}	m_7	10		2^{31}	$d_{13}[-32]$
51	c_{13}	m_{14}	15	2^{31}	2^{31}	$c_{13}[32]$
52	b_{13}	m_5	21		2^{31}	$b_{13}[-32]$
...	...	...	...	...	...	...
59	c_{15}	m_6	15		2^{31}	$c_{15}[32]$
60	b_{15}	m_{13}	21		2^{31}	$b_{15}[32]$
61	$a_{16}+aa_0$	m_4	6	2^{31}		$a_{16}+aa_0=a_{16}'+aa_0'$
62	$d_{16}+dd_0$	m_{11}	10	-2^{15}		$d_{16}+dd_0=d_{16}'+dd_0'$
63	$c_{16}+cc_0$	m_2	15			$c_{16}+cc_0=c_{16}'+cc_0'$
64	$b_{16}+bb_0$	m_9	21			$b_{16}+bb_0=b_{16}'+bb_0'$

Table 6. A Set of Sufficient Conditions for the Second Iteration Differential

a_1	$a_{1,6}=0$, $a_{1,12}=0$, $a_{1,22}=1$, $a_{1,26}=0$, $a_{1,27}=1$, $a_{1,28}=0$, $a_{1,32}=1$
d_1	$d_{1,2}=0$, $d_{1,3}=0$, $d_{1,6}=0$, $d_{1,7}=a_{1,7}$, $d_{1,8}=a_{1,8}$, $d_{1,12}=1$, $d_{1,13}=a_{1,13}$, $d_{1,16}=0$, $d_{1,17}=a_{1,17}$, $d_{1,18}=a_{1,18}$, $d_{1,19}=a_{1,19}$, $d_{1,20}=a_{1,20}$, $d_{1,21}=a_{1,21}$, $d_{1,22}=0$, $d_{1,26}=0$, $d_{1,27}=1$, $d_{1,28}=1$, $d_{1,29}=a_{1,29}$, $d_{1,30}=a_{1,30}$, $d_{1,31}=a_{1,31}$, $d_{1,32}=1$
c_1	$c_{1,2}=1$, $c_{1,3}=1$, $c_{1,4}=d_{1,4}$, $c_{1,5}=d_{1,5}$, $c_{1,6}=1$, $c_{1,7}=1$, $c_{1,8}=0$, $c_{1,9}=1$, $c_{1,12}=1$, $c_{1,13}=0$, $c_{1,17}=1$, $c_{1,18}=1$, $c_{1,19}=1$, $c_{1,20}=1$, $c_{1,21}=1$, $c_{1,22}=0$, $c_{1,26}=1$, $c_{1,27}=1$, $c_{1,28}=1$, $c_{1,29}=1$, $c_{1,30}=1$, $c_{1,31}=0$, $c_{1,32}=1$
b_1	$b_{1,1}=c_{1,1}$, $b_{1,2}=0$, $b_{1,3}=0$, $b_{1,4}=0$, $b_{1,5}=1$, $b_{1,6}=0$, $b_{1,7}=0$, $b_{1,8}=0$, $b_{1,9}=0$, $b_{1,10}=c_{1,10}$, $b_{1,11}=c_{1,11}$, $b_{1,12}=0$, $b_{1,13}=0$, $b_{1,17}=0$, $b_{1,18}=0$, $b_{1,19}=1$, $b_{1,20}=0$, $b_{1,21}=0$, $b_{1,22}=0$, $b_{1,26}=1$, $b_{1,27}=0$, $b_{1,28}=1$, $b_{1,29}=1$, $b_{1,30}=1$, $b_{1,31}=0$, $b_{1,32}=1$
a_2	$a_{2,1}=0$, $a_{2,2}=0$, $a_{2,3}=0$, $a_{2,4}=0$, $a_{2,5}=1$, $a_{2,6}=0$, $a_{2,7}=1$, $a_{2,8}=0$, $a_{2,9}=0$, $a_{2,10}=1$, $a_{2,11}=1$, $a_{2,12}=1$, $a_{2,13}=0$, $a_{2,17}=1$, $a_{2,18}=1$, $a_{2,19}=1$, $a_{2,20}=1$, $a_{2,27}=0$, $a_{2,28}=1$, $a_{2,29}=0$, $a_{2,30}=0$, $a_{2,21}=0$, $a_{2,22}=1$, $a_{2,31}=1$, $a_{2,32}=0$
d_2	$d_{2,1}=0$, $d_{2,2}=1$, $d_{2,3}=1$, $d_{2,4}=0$, $d_{2,5}=1$, $d_{2,6}=0$, $d_{2,7}=1$, $d_{2,8}=0$, $d_{2,9}=0$, $d_{2,10}=0$, $d_{2,11}=1$, $d_{2,12}=1$, $d_{2,13}=0$, $d_{2,17}=0$, $d_{2,18}=1$, $d_{2,21}=0$, $d_{2,22}=1$, $d_{2,26}=0$, $d_{2,27}=1$, $d_{2,28}=0$, $d_{2,29}=0$, $d_{2,32}=0$
c_2	$c_{2,1}=1$, $c_{2,7}=0$, $c_{2,8}=0$, $c_{2,9}=0$, $c_{2,10}=1$, $c_{2,11}=1$, $c_{2,12}=1$, $c_{2,13}=1$, $c_{2,16}=d_{2,16}$, $c_{2,17}=1$, $c_{2,18}=0$, $c_{2,21}=0$, $c_{2,22}=0$, $c_{2,24}=d_{2,24}$, $c_{2,25}=d_{2,25}$, $c_{2,26}=1$, $c_{2,27}=1$, $c_{2,28}=0$, $c_{2,29}=1$, $c_{2,32}=1$
b_2	$b_{2,1}=0$, $b_{2,2}=c_{2,2}$, $b_{2,7}=1$, $b_{2,8}=1$, $b_{2,9}=1$, $b_{2,10}=1$, $b_{2,16}=1$, $b_{2,17}=0$, $b_{2,18}=1$, $b_{2,21}=1$, $b_{2,22}=1$, $b_{2,24}=0$, $b_{2,25}=0$, $b_{2,26}=0$, $b_{2,27}=1$, $b_{2,28}=0$, $b_{2,29}=0$, $b_{2,32}=1$
a_3	$a_{3,1}=1$, $a_{3,2}=0$, $a_{3,7}=1$, $a_{3,8}=1$, $a_{3,9}=1$, $a_{3,10}=0$, $a_{3,13}=b_{2,13}$, $a_{3,16}=0$, $a_{3,17}=1$, $a_{3,18}=0$, $a_{3,24}=0$, $a_{3,25}=0$, $a_{3,26}=0$, $a_{3,27}=1$, $a_{3,28}=1$, $a_{3,29}=1$, $a_{3,32}=1$
d_3	$d_{3,1}=0$, $d_{3,2}=0$, $d_{3,7}=1$, $d_{3,8}=1$, $d_{3,9}=1$, $d_{3,10}=1$, $d_{3,13}=0$, $d_{3,16}=1$, $d_{3,17}=1$, $d_{3,18}=1$, $d_{3,19}=0$, $d_{3,24}=1$, $d_{3,25}=1$, $d_{3,26}=1$, $d_{3,27}=1$, $d_{3,32}=1$
c_3	$c_{3,1}=1$, $c_{3,2}=1$, $c_{3,7}=1$, $c_{3,8}=1$, $c_{3,9}=1$, $c_{3,10}=1$, $c_{3,13}=0$, $c_{3,14}=d_{3,14}$, $c_{3,15}=d_{3,15}$, $c_{3,16}=1$, $c_{3,17}=1$, $c_{3,18}=0$, $c_{3,19}=1$, $c_{3,20}=d_{3,20}$, $c_{3,32}=1$
b_3	$b_{3,8}=1$, $b_{3,13}=1$, $b_{3,14}=0$, $b_{3,15}=0$, $b_{3,16}=0$, $b_{3,17}=0$, $b_{3,18}=0$, $b_{3,19}=0$, $b_{3,20}=1$, $b_{3,25}=c_{3,25}$, $b_{3,26}=c_{3,26}$, $b_{3,27}=c_{3,27}$, $b_{3,28}=c_{3,28}$, $b_{3,29}=c_{3,29}$, $b_{3,30}=c_{3,30}$, $b_{3,31}=c_{3,31}$, $b_{3,32}=1$
a_4	$a_{4,4}=1$, $a_{4,8}=0$, $a_{4,14}=1$, $a_{4,15}=1$, $a_{4,16}=1$, $a_{4,17}=1$, $a_{4,18}=1$, $a_{4,19}=1$, $a_{4,20}=1$ $a_{4,25}=1$, $a_{4,26}=1$, $a_{4,27}=1$, $a_{4,28}=1$, $a_{4,29}=1$, $a_{4,30}=1$, $a_{4,31}=0$, $a_{4,32}=0$
d_4	$d_{4,4}=1$, $d_{4,8}=1$, $d_{4,14}=1$, $d_{4,15}=1$, $d_{4,16}=1$, $d_{4,17}=1$, $d_{4,18}=1$, $d_{4,19}=0$, $d_{4,20}=1$ $d_{4,25}=0$, $d_{4,26}=0$, $d_{4,27}=0$, $d_{4,28}=0$, $d_{4,29}=0$, $d_{4,30}=0$, $d_{4,31}=1$, $d_{4,32}=0$
c_4	$c_{4,4}=0$, $c_{4,16}=0$, $c_{4,25}=1$, $c_{4,26}=0$, $c_{4,27}=1$, $c_{4,28}=1$, $c_{4,29}=1$, $c_{4,30}=1$ $c_{4,31}=1$, $c_{4,32}=0$
b_4	$b_{4,30}=1$, $b_{4,32}=0$
a_5	$a_{5,4}=b_{4,4}$, $a_{5,16}=b_{4,16}$, $a_{5,18}=0$, $a_{5,32}=0$
d_5	$d_{5,18}=1$, $d_{5,30}=a_{5,30}$, $d_{5,32}=0$
c_5	$c_{5,18}=0$, $c_{5,32}=0$
b_5	$b_{5,32}=0$,
a_6-b_6	$a_{6,18}=b_{5,18}, a_{6,32}=0$, $d_{6,32}=0$, $c_{6,32}=0$, $b_{6,32}=c_{6,32}+1$
c_9, b_{12}	$\phi_{34,32}=1$, $b_{12,32}=d_{12,32}$,
$a_{13}-b_{13}$	$a_{13,32}=c_{12,32}$, $d_{13,32}=b_{12,32}+1$, $c_{13,32}=a_{13,32}$, $b_{13,32}=d_{13,32}$
$a_{14}-b_{14}$	$a_{14,32}=c_{13,32}$, $d_{14,32}=b_{13,32}$, $c_{14,32}=a_{14,32}$, $b_{14,32}=d_{14,32}$
$a_{15}-b_{15}$	$a_{15,32}=c_{14,32}$, $d_{15,32}=b_{14,32}$, $c_{15,32}=a_{15,32}$, $b_{15,32}=d_{15,32}+1$
a_{16}	$a_{16,26}=1$, $a_{16,32}=c_{15,32}$
d_{16}	$d_{16,26}=1, d_{16,32}=b_{15,32}$
c_{16}	$c_{16,26}=1, c_{16,32}=a_{16,32}$
b_{16}	$b_{16,26}=1$