Improved Collision Attack on Hash Function MD5

Jie Liang and Xuejia Lai

Department of Computer Science and Engineering
Shanghai Jiao Tong University
Shanghai 200240, China
Email: luckyaa@sjtu.edu.cn

Abstract. In this paper, we present a fast attack algorithm to find two-block collision of hash function MD5. The algorithm is based on the two-block collision differential path of MD5 that was presented by Wang et al. in EUROCRYPT 2005[6]. We found that the derived conditions for the desired differential path in [6] were not sufficient to guarantee the differential path to hold and that some conditions could be relaxed to enlarge the collision set. By using technique of small range searching and omitting the computing steps to check the characteristics in algorithm, we can speed up the attack of MD5 efficiently. Compared with the Advanced Message Modification technique [5,6], the small range searching technique can correct 4 more conditions for the first iteration differential and 3 more conditions for the second iteration differential, thus improving the probability and the complexity to find collisions. The whole attack on the MD5 can be accomplished within 5 hours using a PC with Pentium4 1.70GHZ CPU.

Key words: MD5, collision, differential attacks, fast attack algorithm.

1 Introduction

The hash function MD5 [2] was designed by Ronald Rivest in 1992 as a strengthened version of MD4 [1]. Though some weakness has been found by B.denBoer, A.Bosselaers [3] and H.Dobbertin [4] since its publication, MD5 is widely implemented in cryptography such as digital signature, data integrity, user authentication, key agreement, e-cash and many other cryptographic schemes and protocols. Consequently, MD5 also have been used in almost all commercial security systems and products.

In the past few years, there has been significant advances in the analysis of hash function MD5. At the rump session of Crypto'04, Wang et al. [10] presented the first collision of MD5. In EUROCRYPT 2005, Wang et al. presented a two-block collision differential path of MD5 [6] that allowed us to search collisions efficiently. The complexity of finding one collision is about 2^{41} MD5 operations using the attack algorithm presented in [6] (we consider the extra conditions that will be discussed in section 4). In March 2005, Klima present Multi-message modifications method to find collision of MD5 on a standard notebook PC roughly in 8 hours [7,8], the complexity is

about 2^{36} MD5 operations (we consider the extra conditions that will be discussed in section 4).

In this paper, we show that the conditions in Table 4 and 6 of [6] are not sufficient to ensure the occurrence of the collision path (which was also found in [12] with a different approach) and specify a set of truly sufficient conditions by adding some extra conditions. Additionally, We relax some conditions in Table 4 and 6 of [6] to enlarge the collision set. Finally we propose small range searching technique to correct more conditions in round 2 but keep all the conditions in round 1 hold. By using the small range searching technique and omitting the steps of checking characteristics in algorithm [6,7,8], we can reduce the searching complexity to about 2^{34} MD5 operations for the first block and about 2^{28} MD5 operations for the second block.

2 MD5 Algorithm

The MD5 Message-Digest Algorithm [2] is composed of integer modular addition, four auxiliary Boolean functions and left shift rotation. The processing of MD5 involves 64 steps, and can be described as following:

The chaining variables are initialized as:

a_0 = 0x67452301; d_0 = 0x10325476; c_0 = 0x98badcfe; b_0 = 0xefcdab89;

Step1: $\sum_1 = a_0 + F(b_0, c_0, d_0) + m_0 + 0xd76aa478$ $a_1 = b_0 + \sum_1 <<<7;$

Step2: $\sum_2 = d_0 + F(a_1, b_0, c_0) + m_1 + 0xe8c7b756$ $d_1 = a_1 + \sum_2 <<<12;$

Step3: $\sum_3 = c_0 + F(d_1, a_1, b_0) + m_2 + 0x242070db$ $c_1 = d_1 + \sum_3 <<<17;$

Step4: $\sum_4 = b_0 + F(c_1, d_1, a_1) + m_3 + 0xc1bdceee$ $b_1 = c_1 + \sum_4 <<<22;$

Step5: $\sum_5 = a_1 + F(b_1, c_1, d_1) + m_4 + 0xf57c0faf$ $a_2 = b_1 + \sum_5 <<<7;$

Step6: $\sum_6 = d_1 + F(a_2, b_1, c_1) + m_5 + 0x4787c62a$ $d_2 = a_2 + \sum_6 <<<12;$

Step7: $\sum_7 = c_1 + F(d_2, a_2, b_1) + m_6 + 0xa8304613$ $c_2 = d_2 + \sum_7 <<<17;$

Step8: $\sum_8 = b_1 + F(c_2, d_2, a_2) + m_7 + 0xfd469501$ $b_2 = c_2 + \sum_8 <<<22;$

Step9: $\sum_9 = a_2 + F(b_2, c_2, d_2) + m_8 + 0x698098d8$ $a_3 = b_2 + \sum_9 <<<7;$

Step10: $\sum_{10} = d_2 + F(a_3, b_2, c_2) + m_9 + 0x8b44f7af$ $d_3 = a_3 + \sum_{10} <<<12;$

Step11: $\sum_{11} = c_2 + F(d_3, a_3, b_2) + m_{10} + 0xffff5bb1$ $c_3 = d_3 + \sum_{11} <<<17;$

Step12: $\sum_{12} = b_2 + F(c_3, d_3, a_3) + m_{11} + 0x895cd7be$ $b_3 = c_3 + \sum_{12} <<<22;$

Step13: $\sum_{13} = a_3 + F(b_3, c_3, d_3) + m_{12} + 0x6b901122$ $a_4 = b_3 + \sum_{13} <<<7;$

Step14: $\sum_{14} = d_3 + F(a_4, b_3, c_3) + m_{13} + 0xfd987193$ $d_4 = a_4 + \sum_{14} <<<12;$

Step15: $\sum_{15} = c_3 + F(d_4, a_4, b_3) + m_{14} + 0xa679438e$ $c_4 = d_4 + \sum_{15} <<<17;$

Step16: $\sum_{16} = b_3 + F(c_4, d_4, a_4) + m_{15} + 0x49b40821$ $b_4 = c_4 + \sum_{16} <<<22;$

Step17: $\sum_{17} = a_4 + G(b_4, c_4, d_4) + m_1 + 0xf61e2562$ $a_5 = b_4 + \sum_{17} <<<5;$

Step18: $\sum_{18} = d_4 + G(a_5, b_4, c_4) + m_6 + 0xc040b340$ $d_5 = a_5 + \sum_{18} <<<9;$

Step19: $\sum_{19} = c_4 + G(d_5, a_5, b_4) + m_{11} + 0x265e5a51$ $c_5 = d_5 + \sum_{19} <<<14;$

Step20: $\sum_{20} = b_4 + G(c_5, d_5, a_5) + m_0 + 0xe9b6c7aa \qquad b_5 = c_5 + \sum_{20} <<< 20;$

…… …… …… …… …… …… ……

Step61: $\sum_{61} = a_{15} + I(b_{15}, c_{15}, d_{15}) + m_4 + 0xf7537e82,$

$a_{16} = b_{15} + \sum_{61} <<< 6, \qquad aa_0 = a_{16} + a_0;$

Step62: $\sum_{62} = d_{15} + I(a_{16}, b_{15}, c_{15}) + m_{11} + 0xbd3af235,$

$d_{16} = a_{16} + \sum_{62} <<< 10, \qquad dd_0 = d_{16} + d_0;$

Step63: $\sum_{63} = c_{15} + I(d_{16}, a_{16}, b_{15}) + m_2 + 0x2ad7d2bb,$

$c_{16} = d_{16} + \sum_{63} <<< 15, \qquad cc_0 = c_{16} + c_0;$

Step64: $\sum_{64} = b_{15} + I(c_{16}, d_{16}, a_{16}) + m_9 + 0xeb86d391,$

$b_{16} = c_{16} + \sum_{64} <<< 21, \qquad bb_0 = b_{16} + b_0;$

Where <<< K is cyclically left-shift by K bit positions.

Let aa_0, bb_0, cc_0 and dd_0 be the output of compressing one 512-bit message block. If there are more than one message block for compression, repeat the above 64 steps with the next 512-bit message block and (aa_0, bb_0, cc_0, dd_0) as inputs.

The four auxiliary Boolean functions used for MD5 are the following:

$F(x, y, z) = (x \wedge y) \vee (\neg x \wedge z)$

$G(x, y, z) = (x \wedge z) \vee (y \wedge \neg z)$

$H(x, y, z) = x \oplus y \oplus z$

$I(x, y, z) = y \oplus (x \wedge \neg z)$

where x, y, z are 32-bit words.

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

3 Collision Differentials for MD5

The collision differentials for MD5 with two iterations that Wang et al. presented in EUROCRYPT 2005 is as follows:

$$\Delta H_0 = 0 \xrightarrow{(M_0, M_0')} \Delta H_1 \xrightarrow{(M_1, M_1')} \Delta H_2 = 0$$

Such that

$\Delta M_0 = M_0' - M_0 = (0, 0, 0, 0, \ *2^{31}, 0, 0, 0, 0, 0, 0, \ 2^{15}, 0, 0, \ *2^{31}, 0)$

$\Delta M_1 = M_1' - M_1 = (0, 0, 0, 0, \ *2^{31}, 0, 0, 0, 0, 0, 0, -2^{15}, 0, 0, \ *2^{31}, 0)$

$\Delta H_1 = (*2^{31}, *2^{31} + 2^{25}, *2^{31} + 2^{25}, \ *2^{31} + 2^{25}).$

Where $*2^{31}$ means it can be -2^{31} or $+2^{31}$, which is limited to $+2^{31}$ in [6]. We found that it can be relaxed to -2^{31} or $+2^{31}$ but the collision differentials still hold. M_0, M_0', M_1 and M_1' each is one 512-bit message block. 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.

4 Sufficient Conditions for the Collision Path to Hold

In order to construct a fast attack algorithm without the need to test whether the characteristics really hold in every step, we first derive a set of truly sufficient conditions that guarantee the collision differential path described in Table 3 and 5 of [6] to hold. In this section, we show that it needs to add some extra conditions into Table 4 and 6 of [6] to keep the path attained. More detailed explanation has been discussed in [12], they also discover the lack of conditions independently.

For example, in the 5^{th} step computation of the first iteration:

Step5: $a_2 = b_1 + [\ a_1 + F(b_1, c_1, d_1) + m_4 + 0xf57c0faf\ ] <<<7$,

Let $\sum_5 = a_1 + F(b_1, c_1, d_1) + m_4 + 0xf57c0faf$.

Then, the output difference in the 5^{th} step caused by $\Delta m_4 = *2^{31}$ should depend on the value of bit 32 in $\sum_5$**:** if the bit 32 in $\sum_5$ is zero, then the output difference in the 5^{th} step is 2^6**;** if the bit 32 in $\sum_5$ is one, then the output difference in the 5^{th} step is -2^6. Thus, we need to add the necessary condition $\sum_{5,32}=1$ to keep the output difference -2^6 in Table 3 of [6]. In order to use basic modification technique, we add conditions $b_{1,5}=1$, $b_{1,6}=1$ and $a_{2,5}=0$ (these conditions are not necessary) instead of condition $\sum_{5,32}=1$ to the set of sufficient conditions. We show the 5^{th} step computation of the first iteration in Table 1. According to binary addition properties, for Z=X+Y, we have:

(1) If bit n in X is unknown, bit n in Y is 1 and bit n in Z is 0, then X+Y will have carry to bit n+1.

(2) If bit n in X is unknown, bit n in Y is 0 and bit n in Z is 1, then X+Y will have no carry to bit n+1.

So we can ensure condition $\sum_{5,32}=1$ occur if conditions $b_{1,5}=1$, $b_{1,6}=1$ and $a_{2,5}=0$ hold (see in Table 1).

Table 1. Compute the 5^{th} Step of the First Iteration

bit NO.		1	5	9	11	17	21	25	29
+	$\sum_5 <<<7$	????	?11?	????	????	????	????	????	????
	b_1	????	110?	???1	????	???1	???0	????	???1
=	a_2	1?1?	0100	0000	0000	0000	0010	?0?1	???1

From the example, we see that it is necessary to consider the left shift rotation operation when we derive sufficient conditions for keeping the collision path. Extra conditions for other steps are derived using the same method.

During our research, we found that some conditions in Table 4 and 6 of [6] could be relaxed to enlarge the collision set. We show them as follows:

(1) Conditions in bits $c_{4,32}$, $b_{4,32}$, $a_{5,32}$, $d_{5,32}$, $c_{5,32}$, $b_{5,32}$, $a_{6,32}$ and $d_{6,32}$ are not necessary to be zeros, they just need to be equal to each other for the first iteration differential and the second iteration differential.

(2) Conditions $a_{1,32} = d_{1,32} = c_{1,32} = b_{1,32} = 1$, $a_{2,32} = d_{2,32} = 0$, $c_{2,32} = b_{2,32} = a_{3,32} = d_{3,32} = c_{3,32} = b_{3,32} = 1$, $a_{4,32} = d_{4,32} = 0$ can be relaxed to $bb_{0,32} = a_{1,32} + 1 = d_{1,32} + 1 = c_{1,32} + 1 = b_{1,32} + 1 = a_{2,32} = d_{2,32} = c_{2,32} + 1 = b_{2,32} + 1 = a_{3,32} + 1 = d_{3,32} + 1 = c_{3,32} + 1 = b_{3,32} + 1 = a_{4,32} = d_{4,32}$ for the second iteration differential.

The conditions' relaxations do not have any influence on the differential characteristic in Table 3 and 5 of [6], which is easily verified by using the deriving method described in [6]. After we relax these conditions, the collision set should become at least 8 times larger than before.

Finally, we give out a set of sufficient conditions in Table 4 and 5 (in appendices) that guarantee the desired differential path to occur. The extra conditions in Table 4 and 5 are just one of the possible extra conditions sets derived from $\sum_i$, we choose them out as they are optimal. Compare with the Table 4 of [6], besides the relaxed conditions and the extra conditions derived from $\sum_i$, we also add conditions $d_{16,26} = 0$, $c_{16,26} = 0$, correct condition $c_{16,32} = a_{16,32}$ and delete condition $a_{16,27} = 0$ in Table 4. The same to extra condition $b_{15,26} = 0$ in Table 5. By using the deriving method described in [6], we confirm that these 5 conditions are lacked conditions or incorrect conditions in [6] (see counterexamples in appendices). In fact, we can derive other set of truly sufficient conditions, but the set that we show in Table 4 and 5 is the best in the sense that it contains the least number of conditions for keeping the collision path.

We note that the extra conditions $a_{2,27} = 0$, $a_{2,29} = 0$, $a_{2,30} = 0$ and $a_{2,31} = 0$, which are derived from $\sum_7$ of the first iteration, are not only sufficient but also necessary conditions for holding the collision path. We present a counterexample M_0 with standard IV that satisfies all the conditions in Table 4 except for conditions $a_{2,27} = 1$, $a_{2,29} = 1$, $a_{2,30} = 1$and $a_{2,31} = 1$:

m_0=0xe3b50fa3; m_1=0x4be14e05; m_2=0x16a10a8e; m_3=0xf70e2ebe;
m_4=0x5fa664b8; m_5=0xe60b9ef3; m_6=0xe4594b46; m_7=0x49813c40;
m_8=0x1e332d55; m_9=0x2ff43c05; m_{10}=0x8482ea2f; m_{11}=0x13823723;
m_{12}=0x31c034e3; m_{13}=0x234c14e1; m_{14}=0x14237194; m_{15}=0x34234335.

According to Section 3, M_1 can easy to get from M_0 but they are not collision pairs for the first block at all.

We find that Jun Yajima and Takeshi Shimoyama also discover the lack of conditions in [6]. They try to present a set of sufficient conditions in [13]. By comparing, we found that the sufficient condition set they showed still lacked of conditions. Here we show out a counterexample with standard IV that satisfies all the conditions for First Message Block listed in [13] as follow:

m_0=0x72bcc7d2; m_1=0x87fe0ffc; m_2=0xc7ee72f1; m_3=0x5c92b535;
m_4=0xb3fbb6d4; m_5=0xc03f68c8; m_6=0x95879481; m_7=0xf1f48bd6;
m_8=0x633ed41; m_9=0x3d3f800; m_{10}=0x7f80f83b; m_{11}=0x93410102;
m_{12}=0x90e148c1; m_{13}=0x129cff6; m_{14}=0xfffc2018; m_{15}=0x809c01c1.

Through verification, we found that $b_{1,5} = 1$, $b_{1,6} = 0$ and $a_{2,5} = 0$. These three conditions ensure $\sum_{5,32} = 0$ occur to break the collision path.

5 Construct a Fast Attack Algorithm

In this section, we propose small range searching technique to obtain a faster attack algorithm. The basic idea of this technique is that for N = U+ [L + F(X, Y, Z) +M +Constant] <<< k, we can change the value of bit n in N by searching bit n in U and bit n-k in L, X, Y, Z, M; We can also search the bits lower than n in U and bits lower than n-k in L, X, Y, Z, M to change the value of bit n in N by carry.

Through experiment, we found that the basic message modification technique described in [6] was not always success because of the Carry or Borrow in bit 32. So our basic message modification technique shown below different from the one presented in [6], and we can make the modification always succeed.

5.1 Fast Attack Algorithm for First Block

(a) Select random 32-bit value for m_0, m_1, $m_2 \ldots m_{15}$.

(b) Compute step 1 and step 2 of MD5 algorithm, modify $m_2 \ldots m_{13}$, m_{14} and m_{15} by basic message modification technique. For example, m_5 should be modified to ensure the conditions of d_2 in Table 4 hold (see in Table 2):

Step6: $d_2 = a_2 + [d_1 + F(a_2, b_1, c_1) + m_5 + 0x4787c62a] <<<12;$

Table 2. Conditions of d_2 for the First Iteration Differential

$d_{2,1} = 1$, $d_{2,2} = a_{2,2}$, $d_{2,3} = 0$, $d_{2,4} = a_{2,4}$, $d_{2,5} = a_{2,5}$, $d_{2,6} = 0$, $d_{2,7} = 1$, $d_{2,8} = 0$, $d_{2,9} = 0$,
$d_{2,10} = 0$, $d_{2,11} = 1$, $d_{2,12} = 1$, $d_{2,13} = 1$, $d_{2,14} = 1$, $d_{2,15} = 0$, $d_{2,16} = 1$, $d_{2,17} = 1$, $d_{2,18} = 1$
$d_{2,19} = 1$, $d_{2,20} = 1$, $d_{2,21} = 1$, $d_{2,22} = 1$, $d_{2,23} = 1$, $d_{2,24} = 0$, $d_{2,25} = a_{2,25}$, $d_{2,26} = 1$,
$d_{2,27} = a_{2,27}$, $d_{2,28} = 0$, $d_{2,29} = a_{2,29}$, $d_{2,30} = a_{2,30}$, $d_{2,31} = a_{2,31}$, $d_{2,32} = 0$

Basic Modification:

$d_2 = \{(d_2)\text{^}[(d_2)\mathbf{\&}(0xfd8043be)] \text{ ^ } [(a_2) \mathbf{\&} (0x7500001a)]\} \mid (0x027fbc41)$

$m_5 = [(d_2 - a_2) >>>12] - d_1 - F (a_2, b_1, c_1) - 0x4787c62a.$

In order to use small range searching technique in the next step, we add three extra conditions $c_{4,9} = 0$, $c_{4,21} = 0$ and $c_{4,23} = 0$ in c_4 when Modify m_{14}.

(c) Randomly select 32-bit value for a_5 but make the conditions $a_{5,4} = b_{4,4}$, $a_{5,16} = b_{4,16}$, $a_{5,18} = 0$, $a_{5,31} = 1$ and $a_{5,32} = b_{4,32}$ hold, then compute the 18th step:

$\sum_{18} = d_4 + G(a_5, b_4, c_4) + m_6 + 0xc040b340 \qquad d_5 = a_5 + \sum_{18} <<<9.$

If conditions $d_{5,18} = 1$, $d_{5,30} = a_{5,30}$ and $d_{5,32} = a_{5,32}$ are not all hold, we use small range searching technique to correct them. According to the 18th step computation and extra conditions $c_{4,9} = 0$, $c_{4,21} = 0$ and $c_{4,23} = 0$ in c_4, we notice that by searching bits $b_{4,9}$, $b_{4,21}$, $b_{4,23}$ in b_4, we can change the value of bits $d_{5,18}$, $d_{5,30}$, $d_{5,32}$ in d_5 to make the conditions $d_{5,18} = 1$, $d_{5,30} = a_{5,30}$ and $d_{5,32} = a_{5,32}$ hold, then we need to update the value of m_{15}:

$m_{15} = [\ (b_4 - c_4)\ >>>22\ ] - b_3 - F(c_4, d_4, a_4) - 0x49b40821.$

(d) Compute the 19th step:

$\sum_{19} = c_4 + G(d_5, a_5, b_4) + m_{11} + 0x265e5a51 \qquad c_5 = d_5 + \sum_{19} <<<14.$

If conditions $\sum_{19,4} \sim \sum_{19,8}$ not all ones, $c_{5,18} = 0$ and $c_{5,32} = d_{5,32}$ are not all fulfilled, we use small range searching technique to correct them. By searching bits $b_{3,4}$, $b_{3,5}$, $b_{3,6}$, $b_{3,7}$, $b_{3,21}$, $b_{3,22}$, $b_{3,23}$, $b_{3,24}$ in b_3 and update the value of m_{11} ($m_{11} = [(b_3 - c_3)\ >>>22\ ] - b_2 - F(c_3, d_3, a_3) - 0x895cd7be$), we can affect the value of c_5 to make the conditions $\sum_{19,4} \sim \sum_{19,8}$ not all ones, $c_{5,18} = 0$ and $c_{5,32} = d_{5,32}$ hold. Finally, we update the value of m_{12}, m_{13}, m_{14}, m_{15}:

$m_{12} = [\ (a_4 - b_3)\ >>>7\ ] - a_3 - F(b_3, c_3, d_3) - 0x6b901122,$

$m_{13} = [\ (d_4 - a_4)\ >>>12\ ] - d_3 - F(a_4, b_3, c_3) - 0xfd987193,$

$m_{14} = [\ (c_4 - d_4)\ >>>17\ ] - c_3 - F(d_4, a_4, b_3) - 0xa679438e,$

$m_{15} = [\ (b_4 - c_4)\ >>>22\ ] - b_3 - F(c_4, d_4, a_4) - 0x49b40821.$

(e) Randomly select 32-bit value for b_5 but make the conditions $\sum_{20,30} \sim \sum_{20,32}$ not all zeros and $b_{5,32} = c_{5,32}$ hold, then update the value of m_1,m_0, a_1, d_1,m_2, m_3, m_4 and m_5:

$m_1 = [\ (a_5 - b_4)\ >>>5\ ] - a_4 - G(b_4, c_4, d_4) - 0xf61e2562,$

$m_0 = [\ (b_5 - c_5)\ >>>20\ ] - b_4 - G(c_5, d_5, a_5) - 0xe9b6c7aa,$

$a_1 = b_0 + [\ a_0 + F(b_0, c_0, d_0) + m_0 + 0xd76aa478\ ] <<<7,$

$d_1 = a_1 + [\ d_0 + F(a_1, b_0, c_0) + m_1 + 0xe8c7b756\ ] <<<12,$

$m_2 = [\ (c_1 - d_1)\ >>>17\ ] - c_0 - F(d_1, a_1, b_0) - 0x242070db,$

$m_3 = [\ (b_1 - c_1)\ >>>22\ ] - b_0 - F(c_1, d_1, a_1) - 0xc1bdceee,$

$m_4 = [\ (a_2 - b_1)\ >>>7\ ] - a_1 - F(b_1, c_1, d_1) - 0xf57c0faf,$

$m_5 = [\ (d_2 - a_2)\ >>>12\ ] - d_1 - F(a_2, b_1, c_1) - 0x4787c62a.$

(f) Continuing compute with the remaining steps, if any condition in Table 4 is not satisfied, jump to step (c). If all the conditions are satisfied, go to the second block attack algorithm. We pass the output value aa_0, bb_0, cc_0 and dd_0 to the second block attack algorithm.

5.2 Fast Attack Algorithm for Second Block

(a) Select random 32-bit value for m_0, m_1, m_2 … m_{13}.

(b) Modify m_0, m_1, m_2 … m_{13} by basic message modification technique. We add three extra conditions $d_{4,21} = 1$, $d_{4,22} = 1$ and $d_{4,23} = 1$ in d_4 for small range searching technique when modifying m_{13}.

(c) Randomly select 32-bit value for c_4 but make the conditions $c_{4,4}=0$, $c_{4,16}=0$, $c_{4,17} = 0$, $c_{4,25}=1$, $c_{4,26}=0$, $c_{4,27}=1$, $c_{4,28}=1$, $c_{4,29}=1$, $c_{4,30}=1$ and $c_{4,31} = 1$ hold, then compute the value of m_{14}:

$m_{14} = [(c_4 – d_4) >>>17] – c_3 – F (d_4, a_4, b_3) – 0xa679438e.$

Randomly select 32-bit value for b_4 but make the conditions $b_{4,4} = 1$, $b_{4,16} = 1$, $b_{4,17} = 1$, $b_{4,29} = 0$ and $b_{4,32} = c_{4,32}$ hold, then compute the value of m_{15}:

$m_{15} = [(b_4 – c_4) >>>22] – b_3 – F (c_4, d_4, a_4) – 0x49b40821.$

(d) Compute the 17th step:

$\sum_{17} = a_4 + G(b_4, c_4, d_4) + m_1 + 0xf61e2562e \qquad a_5 = b_4 + \sum_{17} <<<5 .$

If conditions $\sum_{17,25} \sim \sum_{17,27}$ not all ones and $a_{5,32} = b_{4,32}$ are not all fulfilled, we search bits $d_{1,4}$, $d_{1,5}$ in d_1 and bits $b_{4,21}$, $b_{4,22}$, $b_{4,23}$, $b_{4,24}$ in b_4, then update the value of m_1, m_{15} ($m_1 = [(d_1 – a_1) >>>12] – dd_0 – F(a_1, bb_0, cc_0) – 0xe8c7b756$, $m_{15} = [(b_4 – c_4) >>>22] – b_3 – F (c_4, d_4, a_4) – 0x49b40821$) to correct them. Compute the 17th step again, if conditions $a_{5,4} = b_{4,4}$, $a_{5,16} = b_{4,16}$ and $a_{5,18} = 0$ are not all attained, we search bits $d_{1,11}$, $d_{1,23}$ and $d_{1,25}$ in d_1 and update the value of m_1 ($m_1 = [(d_1 – a_1) >>>12] – dd_0 – F(a_1, bb_0, cc_0) – 0xe8c7b756$) to correct them.

(e) Compute the 18th step:

$\sum_{18} = d_4 + G(a_5, b_4, c_4) + m_6 + 0xc040b340 \qquad d_5 = a_5 + \sum_{18} <<<9.$

If conditions $d_{5,18} = 1$ and $d_{5,32} = a_{5,32}$ are not all attained, jump to step (c). If $d_{5,30} \neq a_{5,30}$, we change bit $c_{2,6}$ in c_2 and update the value of m_6 ($m_6 = [(c_2 – d_2) >>>17] – c_1 – F(d_2, a_2, b_1) – 0xa8304613$) to correct it. Then we also need to update the value of m_7, m_8, m_9 and m_{10}:

$m_7 = [(b_2 – c_2) >>>22] – b_1 – F (c_2, d_2, a_2) – 0xfd469501,$

$m_8 = [(a_3 – b_2) >>>7] – a_2 – F (b_2, c_2, d_2) – 0x698098d8,$

$m_9 = [(d_3 – a_3) >>>12] – d_2 – F (a_3, b_2, c_2) – 0x8b44f7af,$

$m_{10} = [(c_3 – d_3) >>>17] – c_2 – F (d_3, a_3, b_2) – 0xffff5bb1.$

(f) Compute the 19th step:

$\sum_{19} = c_4 + G(d_5, a_5, b_4) + m_{11} + 0x265e5a51 \qquad c_5 = d_5 + \sum_{19} <<<14.$

If conditions $\sum_{19,4} \sim \sum_{19,18}$ not all ones, $c_{5,18} = 0$ and $c_{5,32} = d_{5,32}$ are not all attained, we use small range searching technique to correct them. By searching bits $b_{3,4}$, $b_{3,5}$,

$b_{3,6}$, $b_{3,7}$, $b_{3,21}$, $b_{3,22}$, $b_{3,23}$, $b_{3,24}$ in b_3 and update the value of m_{11} (m_{11} = [(b_3–c_3) >>>22] –b_2 –F(c_3, d_3, a_3) –0x895cd7be), we can affect the value of c_5 to make the conditions $\sum_{19,4} \sim \sum_{19,18}$ not all ones, $c_{5,18}$ = 0, $c_{5,32}$ = $d_{5,32}$ hold. Finally, we update the value of m_{12}, m_{13}, m_{14} and m_{15}:

m_{12}= [(a_4–b_3) >>>7] –a_3 – F (b_3, c_3, d_3) –0x6b901122,

m_{13}= [(d_4–a_4) >>>12] –d_3 – F (a_4, b_3, c_3) –0xfd987193,

m_{14}= [(c_4–d_4) >>>17] –c_3 – F (d_4, a_4, b_3) –0xa679438e,

m_{15}= [(b_4 – c_4) >>>22] –b_3 – F (c_4, d_4, a_4) –0x49b40821.

(g) Compute the 20th step:

$\sum_{20}$ = b_4 + G(c_5, d_5, a_5) + m_0 + 0xe9b6c7aa b_5 = c_5 + $\sum_{20}$ <<<20 .

If conditions $\sum_{20,30} \sim \sum_{20,32}$ not all zeros and $b_{5,32}$ = $c_{5,32}$ are not all attained, jump to step (c).

(h) In order to speed up the attack, we want to change the value of m_0 without updating the value of m_1, as updating the value of m_1 will cause the conditions in the 17th step not hold. By randomly select the value of $d_{1,7}$ = $a_{1,7}$, $d_{1,8}$ = $a_{1,8}$, $d_{1,13}$ = $a_{1,13}$, $d_{1,18}$ = $a_{1,18}$, $d_{1,19}$ = $a_{1,19}$, $d_{1,20}$ = $a_{1,20}$, $d_{1,21}$ = $a_{1,21}$, $d_{1,29}$ = $a_{1,29}$, $d_{1,30}$ = $a_{1,30}$ and $d_{1,31}$ = $a_{1,31}$ (see in Table 5), we may change the value of m_0 without updating the value of m_1 if the value of F(a_1, bb_0, cc_0) is unchanged. For example, according to the properties of Boolean Function $F(x, y, z) = (x \wedge y) \vee (\neg x \wedge z)$[5], if $bb_{0,7}$= $cc_{0,7}$, randomly choose the value of $d_{1,7}$ = $a_{1,7}$ will not change the value of F(a_1, bb_0, cc_0), then the value of m_1 (m_1= [(d_1–a_1) >>>12] –dd_0 –F(a_1, bb_0, cc_0) –0xe8c7b756) will not change at all. After randomly select the values of $d_{1,7}$ = $a_{1,7}$, $d_{1,8}$ =$a_{1,8}$, $d_{1,13}$ =$a_{1,13}$, $d_{1,18}$ = $a_{1,18}$, $d_{1,19}$ = $a_{1,19}$, $d_{1,20}$ = $a_{1,20}$, $d_{1,21}$ = $a_{1,21}$, $d_{1,29}$ = $a_{1,29}$, $d_{1,30}$ = $a_{1,30}$ and $d_{1,31}$ = $a_{1,31}$ without changing the value of m_1, we update the values of m_0, m_2, m_3, m_4 and m_5:

m_0= [(a_1–bb_0) >>>7] –aa_0 –F(bb_0, cc_0, dd_0) –0xd76aa478,

m_2= [(c_1–d_1) >>>17] –cc_0 –F(d_1, a_1, bb_0) –0x242070db,

m_3= [(b_1–c_1) >>>22] –bb_0 –F(c_1, d_1, a_1) –0xc1bdceee,

m_4= [(a_2 – b_1) >>>7] – a_1 –F(b_1, c_1, d_1) –0xf57c0faf,

m_5= [(d_2–a_2) >>>12] – d_1 –F(a_2, b_1, c_1) –0x4787c62a.

(i) Continuing with the remaining steps of MD5 from step20, if any condition in Table 5 is not satisfied, jump to step (h). If all the possible selections in step (h) fail, then jump to step (c).

5.3 The Speed of Our Algorithm

By using basic message modification technique and small range searching technique, about 35 conditions in round 2-4 are undetermined in the Table 4, and about 31 conditions in round 2-4 are undetermined in the table 5. Consider the extra conditions, the attack algorithm described in [6] should have about 39 conditions in round 2-4 undetermined in the table 4 and about 34 conditions in round 2-4 undetermined in the table 5. So, using out attack algorithm can speed up the attack of MD5 about 16 times. For each random selection of a_5 and b_5 in Fast Attack Algorithm for First Block or c_4 and b_4 in Fast Attack Algorithm for Second Block, we can reasonably assume that each random selection for searching takes about 32 steps of MD5 on average, then the first iteration differential holds within 2^{34} MD5 operations, and the second iteration differential holds within 2^{28} MD5 operations if ignore conditions $d_{5,18}$= 1 and $d_{5,32} = a_{5,32}$ because of their high success probability. The complexity of finding a 1024-bit collision message of MD5 does not exceed the time of running 2^{35} MD5 operations.

Compare with the Multi-message Modification technique [7], our small range searching technique should be more efficient for only searching some bits to correct the same conditions in round 2. It seems that the small range searching technique integrates the other two techniques to speed up the attack of MD5. Additionally, our attack algorithm is based on the truly sufficient conditions, we don't need to test whether the characteristics really hold in every step like [6,8], so our attack algorithm could be considered at least 2 times faster than the algorithm present in [7,8].

In experiment, the running time for determining the first block is within 4 hours using a PC with 1.70GHZ Pentium4 CPU, and within 20 minutes for the second block. Thus, we can find an example of MD5 collision within 5 hours. A collision example is given in Table 3 with $c_{4,32} = b_{4,32} = a_{5,32} = d_{5,32} = c_{5,32} = b_{5,32} = a_{6,32} = d_{6,32}$=1 for the first iteration.

Table 3. A Collision Example for MD5

IV	67452301	efcdab89	98badcfe	10325476				
M_0	055a604a 0632ed41	a3461df0 03f3f7fc	12221694 805eb737	6c449744 1300af02	25c44d2c befc06c7	a1b99a33 0099e023	92681957 ff80803f	3c554e32 0000bd93
M_1	c0e83a00 fe32fffd	37f3af4e a7e84fe0	95243bff 30803ffe	f2e16edf dc833c85	b4cc3b03 5f1330ed	fcbaa5a3 088bde83	852088e8 6f89b53d	c00d7bd1 819a57f0
$M_0^{'}$	055a604a 0632ed41	a3461df0 03f3f7fc	12221694 805eb737	6c449744 13012f02	a5c44d2c befc06c7	a1b99a33 0099e023	92681957 7f80803f	3c554e32 0000bd93
$M_1^{'}$	c0e83a00 fe32fffd	37f3af4e a7e84fe0	95243bff 30803ffe	f2e16edf dc82bc85	34cc3b03 5f1330ed	fcbaa5a3 088bde83	852088e8 ef89b53d	c00d7bd1 819a57f0
H	9cd5a4f9	3b375002	8ca3c972	901209ef				

6 Summary

In this paper, we present an algorithm to speed up the attack of hash function MD5. In order to construct the algorithm, we also discuss the sufficient conditions for keeping

the two-block collision differential path. By using small range searching technique and omitting the computing steps to check the characteristics, the probability and the complexity to find a collision of MD5 are greatly improved. The small range searching technique can also be used to speed up the attack of other hash functions such as MD4 and RIPEMD.

When we are summarizing our research results, we find that Yu Sasaki etc present their new message modification techniques in [14], they claim that their modification techniques can correct 14 more conditions in round 2 with probability about 1/2. Thus, the complexity is about 2^{33} MD5 operations considering the extra conditions in Table 4 and 5. In fact, by random selection of a_5 and b_5 in Fast Attack Algorithm for First Block, the conditions $a_{6,18} = b_{5,18}$, $d_{6,32} = a_{6,32} = b_{5,32}$, $c_{6,32} = 0$, $b_{6,32} = c_{6,32} + 1$ in round 2 can easily fulfilled with high probability.

References

1. Ronald Rivest, The MD4 Message Digest Algorithm, RFC1320, April 1992.
2. Ronald Rivest, The MD5 Message Digest Algorithm, RFC1321, April 1992.
3. B. den. Boer, A. Bosselaers, Collisions for the Compression Function of MD5, Advances in Cryptology, EUROCRYPT'93 Proceedings, Springer-Verlag, 1994.
4. H.Dobbertin, Cryptanalysis of MD5 compress, presented at the rump session of Eurocrypt'96.
5. Xiaoyun Wang, Xuejia Lai, Dengguo Feng, et al, Cryptanalysis of the Hash Functions MD4 and RIPEMD, EUROCRYPT 2005, LNCS 3494, pp.1-18, Springer-Verlag, 2005.
6. Xiaoyun Wang, Hongbo Yu, How to Break MD5 and Other Hash Functions, EUROCRYPT 2005, LNCS 3494, pp.19-35, Springer-Verlag, 2005.
7. Vlastimil Klima, Finding MD5 Collisions on a Notebook PC Using Multi-message Modifications, http://eprint.iacr.org/2005/102.pdf, 2005.
8. Patrick Stach, MD5 Collision Generator by Patrick Stach <pstach@stachliu.com>, http://www.stachliu.com.nyud.net:8090/md5coll.c
9. H. Dobbertin. "Cryptanalysis of MD4." FSE96,pp.53-69,1996.
10. Xiaoyun Wang, Dengguo Feng, Xuejia Lai, Hongbo Yu, Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD.rump session of Crypto'04, E-print, 2004/199.
11. Vlastimil Klima: Finding MD5 Collisions – a Toy For a Notebook, Cryptology ePrint Archive, Report 2005/075, http://eprint.iacr.org/2005/075.pdf, March 5, 2005.
12. Zhang-yi Wang, Huan-guo Zhang, Zhong-ping Qin, Qing-shu Meng, A Fast Attack on the MD5 Hash Function, paper submitted to The 2nd Information Security Practice and Experience Conference (ISPEC 2006).
13. Jun Yajima, Takeshi Shimoyama, Wang's sufficient conditions of MD5 are not sufficient, http://eprint.iacr.org/2005/263.pdf, 2005.
14. Yu Sasaki* Yusuke Naito* Noboru Kunihiro* Kazuo Ohta*, Improved Collision Attack on MD5, http://eprint.iacr.org/2005/400.pdf, 2005.

Appendices

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

$c_{1,7} = 0, c_{1,12} = 0, c_{1,20} = 0$	Extra conditions derived from $\sum_i$
$b_{1,7} = 0, b_{1,8} = c_{1,8}, b_{1,9} = c_{1,9}, b_{1,10} = c_{1,10}, b_{1,11} = c_{1,11}, b_{1,12} = 1, b_{1,13} = c_{1,13}, b_{1,14} = c_{1,14}, b_{1,15} = c_{1,15}, b_{1,16} = c_{1,16}, b_{1,17} = c_{1,17}, b_{1,18} = c_{1,18}, b_{1,19} = c_{1,19}, b_{1,20} = 1, b_{1,21} = c_{1,21}, b_{1,22} = c_{1,22}, b_{1,23} = c_{1,23}, b_{1,24} = 0, b_{1,32} = 1$	$\sum_5 \Rightarrow b_{1,5} = 1, b_{1,6} = 1, a_{2,5} = 0$
$a_{2,1} = 1, a_{2,3} = 1, a_{2,6} = 1, a_{2,7} = 0, a_{2,8} = 0, a_{2,9} = 0, a_{2,10} = 0, a_{2,11} = 0, a_{2,12} = 0, a_{2,13} = 0, a_{2,14} = 0, a_{2,15} = 0, a_{2,16} = 0, a_{2,17} = 0, a_{2,18} = 0, a_{2,19} = 0, a_{2,20} = 0, a_{2,21} = 0, a_{2,22} = 0, a_{2,23} = 0, a_{2,24} = 1, a_{2,26} = 0, a_{2,28} = 1, a_{2,32} = 1$	$\sum_7 \Rightarrow a_{2,27} = 0, a_{2,29} = 0, a_{2,30} = 0, a_{2,31} = 0$
$d_{2,1} = 1, d_{2,2} = a_{2,2}, d_{2,3} = 0, d_{2,4} = a_{2,4}, d_{2,5} = a_{2,5}, d_{2,6} = 0, d_{2,7} = 1, d_{2,8} = 0, d_{2,9} = 0, d_{2,10} = 0, d_{2,11} = 1, d_{2,12} = 1, d_{2,13} = 1, d_{2,14} = 1, d_{2,15} = 0, d_{2,16} = 1, d_{2,17} = 1, d_{2,18} = 1, d_{2,19} = 1, d_{2,20} = 1, d_{2,21} = 1, d_{2,22} = 1, d_{2,23} = 1, d_{2,24} = 0, d_{2,25} = a_{2,25}, d_{2,26} = 1, d_{2,27} = a_{2,27}, d_{2,28} = 0, d_{2,29} = a_{2,29}, d_{2,30} = a_{2,30}, d_{2,31} = a_{2,31}, d_{2,32} = 0$	
$c_{2,1} = 0, c_{2,2} = 0, c_{2,3} = 0, c_{2,4} = 0, c_{2,5} = 0, c_{2,6} = 1, c_{2,7} = 0, c_{2,8} = 0, c_{2,9} = 0, c_{2,10} = 0, c_{2,11} = 0, c_{2,12} = 1, c_{2,13} = 1, c_{2,14} = 1, c_{2,15} = 1, c_{2,16} = 1, c_{2,17} = 0, c_{2,18} = 1, c_{2,19} = 1, c_{2,20} = 1, c_{2,21} = 1, c_{2,22} = 1, c_{2,23} = 1, c_{2,24} = 1, c_{2,25} = 1, c_{2,26} = 1, c_{2,27} = 0, c_{2,28} = 0, c_{2,29} = 0, c_{2,30} = 0, c_{2,31} = 0, c_{2,32} = 0$	
$b_{2,1} = 0, b_{2,2} = 0, b_{2,3} = 0, b_{2,4} = 0, b_{2,5} = 0, b_{2,6} = 0, b_{2,7} = 1, b_{2,8} = 0, b_{2,9} = 1, b_{2,10} = 0, b_{2,11} = 1, b_{2,12} = 0, b_{2,14} = 0, b_{2,16} = 0, b_{2,17} = 1, b_{2,18} = 0, b_{2,19} = 0, b_{2,20} = 0, b_{2,21} = 1, b_{2,24} = 1, b_{2,25} = 1, b_{2,26} = 0, b_{2,27} = 0, b_{2,28} = 0, b_{2,29} = 0, b_{2,30} = 0, b_{2,31} = 0, b_{2,32} = 0$	
$a_{3,1} = 1, a_{3,2} = 0, a_{3,3} = 1, a_{3,4} = 1, a_{3,5} = 1, a_{3,6} = 1, a_{3,7} = 0, a_{3,8} = 0, a_{3,9} = 1, a_{3,10} = 1, a_{3,11} = 1, a_{3,12} = 1, a_{3,13} = b_{3,13}, a_{3,14} = 1, a_{3,16} = 0, a_{3,17} = 0, a_{3,18} = 0, a_{3,19} = 0, a_{3,20} = 0, a_{3,21} = 1, a_{3,25} = 1, a_{3,26} = 1, a_{3,27} = 0, a_{3,28} = 1, a_{3,29} = 1, a_{3,30} = 1, a_{3,31} = 1, a_{3,32} = 1$	
$d_{3,1} = 0, d_{3,2} = 0, d_{3,3} = 1, d_{3,7} = 0, d_{3,8} = 0, d_{3,9} = 0, d_{3,13} = 1, d_{3,14} = 0, d_{3,16} = 1, d_{3,17} = 1, d_{3,18} = 1, d_{3,19} = 1, d_{3,20} = 1, d_{3,21} = 1, d_{3,24} = 0, d_{3,31} = 1, d_{3,32} = 0$	$\sum_{11} \Rightarrow d_{3,29} = 1, d_{3,30} = 1, c_{3,29} = 0, c_{3,30} = 1$
$c_{3,1} = 0, c_{3,2} = 1, c_{3,7} = 1, c_{3,8} = 1, c_{3,9} = 0, c_{3,13} = 0, c_{3,14} = 0, c_{3,15} = d_{3,15}, c_{3,16} = 1, c_{3,17} = 1, c_{3,18} = 0, c_{3,19} = 0, c_{3,20} = 0, c_{3,31} = 0, c_{3,32} = 0$	
$b_{3,8} = 0, b_{3,9} = 1, b_{3,13} = 1, b_{3,14} = 0, b_{3,15} = 0, b_{3,16} = 0, b_{3,17} = 0, b_{3,18} = 0, b_{3,20} = 1, b_{3,25} = c_{3,25}, b_{3,26} = c_{3,26}, b_{3,19} = 0, b_{3,31} = 0, b_{3,32} = 0$	$\sum_{12} \Rightarrow b_{3,30} = 0$
$a_{4,4} = 1, a_{4,8} = 0, a_{4,9} = 0, a_{4,14} = 1, a_{4,15} = 1, a_{4,16} = 1, a_{4,17} = 1, a_{4,18} = 1, a_{4,19} = 1, a_{4,20} = 1, a_{4,25} = 1, a_{4,26} = 0, a_{4,31} = 1, a_{4,32} = 0$	
$d_{4,4} = 1, d_{4,8} = 1, d_{4,9} = 1, d_{4,14} = 1, d_{4,15} = 1, d_{4,16} = 1, d_{4,17} = 1, d_{4,18} = 1, d_{4,19} = 0, d_{4,20} = 1, d_{4,25} = 0, d_{4,26} = 0, d_{4,30} = 0, d_{4,32} = 0$	$\sum_{14} \Rightarrow d_{4,31} = 0$
$c_{4,4} = 0, c_{4,16} = 1, c_{4,25} = 1, c_{4,26} = 0, c_{4,30} = 1$	$\sum_{15} \Rightarrow c_{4,15} = 0$
$b_{4,30} = 1, b_{4,32} = c_{4,32}$	$\sum_{16} \Rightarrow c_{4,31} = 1, b_{4,22} = c_{4,22} + 1, b_{4,31} = 0$
$a_{5,4} = b_{4,4}, a_{5,16} = b_{4,16}, a_{5,18} = 0, a_{5,32} = b_{4,32}$	$\sum_{17} \Rightarrow a_{5,31} = b_{4,31} + 1 = 1$
$d_{5,18} = 1, d_{5,30} = a_{5,30}, d_{5,32} = a_{5,32}, c_{5,18} = 0, c_{5,32} = d_{5,32}$	$\sum_{19,4} \sim \sum_{19,18}$ not all ones
$b_{5,32} = c_{5,32}$	$\sum_{20,30} \sim \sum_{20,32}$ not all zeros
$a_{6,18} = b_{5,18}, d_{6,32} = a_{6,32} = b_{5,32}, c_{6,32} = 0, b_{6,32} = c_{6,32} + 1, b_{12,32} = d_{12,32}$	$\sum_{23,18} = 0, \sum_{35,16} = 0$
$a_{13,32} = c_{12,32}, d_{13,32} = b_{12,32} + 1, c_{13,32} = a_{13,32}, b_{13,32} = d_{13,32}, a_{14,32} = c_{13,32}, d_{14,32} = b_{13,32}, c_{14,32} = a_{14,32}, b_{14,32} = d_{14,32}, a_{15,32} = c_{14,32}, d_{15,32} = b_{14,32}, c_{15,32} = a_{15,32}, b_{15,26} = 0, b_{15,32} = d_{15,32} + 1, a_{16,26} = 1, a_{16,32} = c_{15,32}$	
$dd_{0,26} = 0, d_{16,26} = 0, d_{16,32} = b_{15,32}$	$\sum_{62,16} \sim \sum_{62,22}$ not all ones
$cc_{0,26} = 1, cc_{0,27} = 0, c_{16,26} = 0, c_{16,32} = a_{16,32}, bb_{0,26} = 0, bb_{0,27} = 0, bb_{0,6} = 0, bb_{0,32} = cc_{0,32} = dd_{0,32}$	

Table 5.A set of Sufficient Conditions for the Second Iteration Differential

$a_{1,6}=0, a_{1,12}=0, a_{1,22}=1, a_{1,26}=0, a_{1,27}=1, a_{1,28}=0, a_{1,32}=bb_{0,32}+1$	Extra conditions derived from $\sum_i$
$d_{1,2}=0, d_{1,3}=0, d_{1,6}=0, d_{1,7}=a_{1,7}, d_{1,8}=a_{1,8}, d_{1,12}=1, d_{1,13}=a_{1,13}, d_{1,17}=1,$ $d_{1,18}=a_{1,18}, d_{1,19}=a_{1,19}, d_{1,20}=a_{1,20}, d_{1,21}=a_{1,21}, d_{1,22}=0, d_{1,26}=0, d_{1,27}=1,$ $d_{1,28}=1, d_{1,29}=a_{1,29}, d_{1,30}=a_{1,30}, d_{1,31}=a_{1,31}, d_{1,32}=a_{1,32}$	$\sum_3 \Rightarrow a_{1,17}=1, d_{1,16}=0, c_{1,16}=1$
$c_{1,2}=1, c_{1,3}=1, c_{1,4}=d_{1,4}, c_{1,5}=d_{1,5}, c_{1,6}=1, c_{1,7}=1, c_{1,8}=0, c_{1,9}=1,$ $c_{1,12}=1, c_{1,13}=0, c_{1,17}=1, c_{1,18}=1, c_{1,19}=1, c_{1,20}=1, c_{1,21}=1, c_{1,22}=0,$ $c_{1,26}=1, c_{1,27}=1, c_{1,28}=1, c_{1,29}=1, c_{1,30}=1, c_{1,31}=0, c_{1,32}=d_{1,32}$	$\sum_5 \Rightarrow c_{1,1}=1$
$b_{1,1}=1, b_{1,2}=0, b_{1,3}=0, b_{1,4}=0, b_{1,5}=1, b_{1,6}=0, b_{1,7}=0, b_{1,8}=0, b_{1,9}=0, b_{1,10}=c_{1,10}, b_{1,11}=c_{1,11}, b_{1,12}=0, b_{1,13}=0,$ $b_{1,17}=0, b_{1,18}=0, b_{1,19}=1, b_{1,20}=0, b_{1,21}=0, b_{1,22}=0, b_{1,26}=1, b_{1,27}=0, b_{1,28}=1, b_{1,29}=1, b_{1,30}=1, b_{1,31}=0, b_{1,32}=c_{1,32}$	
$a_{2,1}=0, a_{2,2}=0, a_{2,3}=0, a_{2,4}=0, a_{2,5}=1, a_{2,6}=0, a_{2,7}=1, a_{2,8}=0, a_{2,9}=0,$ $a_{2,10}=1, a_{2,11}=1, a_{2,12}=1, a_{2,13}=0, a_{2,17}=1, a_{2,18}=1, a_{2,19}=1, a_{2,20}=1,$ $a_{2,21}=0, a_{2,22}=1, a_{2,27}=0, a_{2,28}=1, a_{2,29}=0, a_{2,30}=0, a_{2,31}=1, a_{2,32}=b_{1,32}+1$	$\sum_7 \Rightarrow d_{2,15}=1, c_{2,15}=0$
$d_{2,1}=0, d_{2,2}=1, d_{2,3}=1, d_{2,4}=0, d_{2,5}=1, d_{2,6}=0, d_{2,7}=1, d_{2,8}=0, d_{2,9}=0, d_{2,10}=0, d_{2,11}=1, d_{2,12}=1,$ $d_{2,13}=0, d_{2,17}=0, d_{2,18}=1, d_{2,21}=0, d_{2,22}=1, d_{2,26}=0, d_{2,27}=1, d_{2,28}=0, d_{2,29}=0, d_{2,32}=a_{2,32}$	
$c_{2,1}=1, c_{2,7}=0, c_{2,8}=0, c_{2,9}=0, c_{2,10}=1, c_{2,11}=1, c_{2,12}=1, c_{2,13}=1, c_{2,16}=d_{2,16}, c_{2,17}=1, c_{2,18}=0, c_{2,21}=0,$ $c_{2,22}=0, c_{2,24}=d_{2,24}, c_{2,25}=d_{2,25}, c_{2,26}=1, c_{2,27}=1, c_{2,28}=0, c_{2,29}=1, c_{2,32}=d_{2,32}+1$	
$b_{2,1}=0, b_{2,2}=c_{2,2}, b_{2,7}=1, b_{2,8}=1, b_{2,9}=1, b_{2,10}=1, b_{2,16}=1, b_{2,17}=0, b_{2,18}=1,$ $b_{2,21}=1, b_{2,22}=1, b_{2,24}=0, b_{2,25}=0, b_{2,26}=0, b_{2,27}=1, b_{2,28}=0, b_{2,29}=0, b_{2,32}=c_{2,32}$	$\sum_9 \Rightarrow b_{2,6}=1, a_{3,6}=0$
$a_{3,1}=1, a_{3,2}=0, a_{3,7}=1, a_{3,8}=1, a_{3,9}=1, a_{3,10}=0, a_{3,13}=b_{2,13}, a_{3,16}=0, a_{3,17}=1, a_{3,18}=0, a_{3,24}=0, a_{3,25}=0,$ $a_{3,26}=0, a_{3,27}=1, a_{3,28}=1, a_{3,29}=1, a_{3,32}=b_{2,32}$	
$d_{3,1}=0, d_{3,2}=0, d_{3,7}=1, d_{3,8}=1, d_{3,9}=1, d_{3,10}=1, d_{3,13}=0, d_{3,16}=1,$ $d_{3,17}=1, d_{3,18}=1, d_{3,19}=0, d_{3,24}=1, d_{3,25}=1, d_{3,26}=1, d_{3,27}=1, d_{3,32}=a_{3,32}$	$\sum_{11} \Rightarrow d_{3,12}=1, c_{3,12}=0$
$c_{3,1}=1, c_{3,2}=1, c_{3,7}=1, c_{3,8}=1, c_{3,9}=1, c_{3,10}=1, c_{3,13}=0, c_{3,14}=d_{3,14}, c_{3,15}=d_{3,15}, c_{3,16}=1, c_{3,17}=1,$ $c_{3,18}=0, c_{3,19}=1, c_{3,20}=d_{3,20}, c_{3,32}=d_{3,32}$	
$b_{3,8}=1, b_{3,13}=1, b_{3,14}=0, b_{3,15}=0, b_{3,16}=0, b_{3,17}=0, b_{3,18}=0, b_{3,19}=0, b_{3,20}=1, b_{3,25}=c_{3,25}, b_{3,26}=c_{3,26},$ $b_{3,27}=c_{3,27}, b_{3,28}=c_{3,28}, b_{3,29}=c_{3,29}, b_{3,30}=c_{3,30}, b_{3,31}=c_{3,31}, b_{3,32}=c_{3,32}$	
$a_{4,4}=1, a_{4,8}=0, a_{4,14}=1, a_{4,15}=1, a_{4,16}=1, a_{4,17}=1, a_{4,18}=1, a_{4,19}=1, a_{4,20}=1, a_{4,25}=1,$ $a_{4,25}=1, a_{4,26}=1, a_{4,27}=1, a_{4,28}=1, a_{4,29}=1, a_{4,30}=1, a_{4,31}=0, a_{4,32}=b_{3,32}+1$	$\sum_{14} \Rightarrow a_{4,24}=0, d_{4,24}=1$
$d_{4,4}=1, d_{4,8}=1, d_{4,14}=1, d_{4,15}=1, d_{4,16}=1, d_{4,17}=1, d_{4,18}=1, d_{4,19}=0, d_{4,20}=1 d_{4,25}=0, d_{4,26}=0, d_{4,27}=0,$ $d_{4,28}=0, d_{4,29}=0, d_{4,30}=0, d_{4,31}=1, d_{4,32}=a_{4,32}$	
$c_{4,4}=0, c_{4,16}=0, c_{4,25}=1, c_{4,26}=0, c_{4,27}=1, c_{4,28}=1, c_{4,29}=1, c_{4,30}=1, c_{4,31}=1, c_{4,32}=1$	$\sum_{15} \Rightarrow c_{4,17}=0$
$b_{4,4}=1, b_{4,32}=c_{4,32}$	$\sum_{16} \Rightarrow b_{4,16}=1, b_{4,17}=1, b_{4,29}=0$
$a_{5,4}=b_{4,4}, a_{5,16}=b_{4,16}, a_{5,18}=0, a_{5,32}=b_{4,32}$	$\sum_{17,25} \sim \sum_{17,27}$ not all ones
$d_{5,18}=1, d_{5,30}=a_{5,30}, d_{5,32}=a_{5,32}, c_{5,18}=0, c_{5,32}=d_{5,32}$	$\sum_{19,4} \sim \sum_{19,18}$ not all ones
$b_{5,32}=c_{5,32}$	$\sum_{20,30} \sim \sum_{20,32}$ not all zeros
$a_{6,18}=b_{5,18}, d_{6,32}=a_{6,32}=b_{5,32}, c_{6,32}=0, b_{6,32}=c_{6,32}+1, b_{12,32}=d_{12,32}$	$\sum_{23,18}=0, \sum_{35,16}=1$
$a_{13,32}=c_{12,32}, d_{13,32}=b_{12,32}+1, c_{13,32}=a_{13,32}, b_{13,32}=d_{14,32}, a_{13,32}=c_{13,32}, d_{13,32}=b_{14,32}, c_{14,32}=a_{14,32},$ $b_{14,32}=d_{14,32}, a_{15,32}=c_{14,32}, d_{15,32}=b_{14,32}, c_{15,32}=a_{15,32}, b_{15,26}=0, b_{15,32}=d_{15,32}+1, a_{16,26}=1, a_{16,32}=c_{16,32}$	
$d_{16,26}=1, d_{16,32}=b_{15,32}, c_{16,26}=1, c_{16,32}=a_{16,32}, b_{16,26}=1$	$\sum_{62,16} \sim \sum_{62,22}$ not all zeros

Table 6. A Counterexample Satisfied Cnditions in Table 5 Ecept for $b_{15,26} = 1$

H_1	aa14309f	80820546	c273be7a	cdc25be6				
M_1	e4473624	f5c3d164	9d285199	42a37cde	4513f7f9	4cadc2fe	a6208936	c2027bcd
	bdfbfefd	7f00fd6	301ffffe	4420063	fef41cbe	7f3b44f	580a4fb2	a21a69f2
H_2	d0fa79d	fa896b4e	8d8d224	81a9a7d9				
$H_1^{'}$	2a14309f	2820546	4473be7a	4fc25be6				
$M_1^{'}$	e4473624	f5c3d164	9d285199	42a37cde	c513f7f9	4cadc2fe	a6208936	c2027bcd
	bdfbfefd	7f00fd6	301ffffe	4418063	fef41cbe	7f3b44f	d80a4fb2	a21a69f2
$H_2^{'}$	d0fa79d	ba896c4e	8d8d324	81a9a7d9				

Table 7. A Collision Example for The First Block with $a_{16,27} = 1$

IV	67452301	efcdab89	98badcfe	10325476				
M_0	10e53c6b	57e8f46b	e861c5ea	5736e652	51c3a485	e111fe10	939aa559	e9039a39
	6332d51	2341c04	82b0fb4b	57b2e311	429126e3	4a8c20db	62ce4193	52823034
H	cdb59525	90550918	b204dbce	99a80f8f				
$M_0^{'}$	10e53c6b	57e8f46b	e861c5ea	5736e652	d1c3a485	e111fe10	939aa559	e9039a39
	6332d51	2341c04	82b0fb4b	57b36311	429126e3	4a8c20db	e2ce4193	52823034
$H^{'}$	4db59525	12550918	3404dbce	1ba80f8f				

Table 8. A Counterexample Satisfied Cnditions in Table 4 Ecept for $c_{16,32} = d_{16,32} \neq a_{16,32}$

IV	67452301	efcdab89	98badcfe	10325476				
M_0	b3ff8745	c72eb352	5f2ff952	a9b33ceb	fc7993f8	b7687688	c1548362	8180b92e
	6342d45	23f43bfd	7c42a22f	14a14334	332550f3	34ab42c4	7423620c	341f42b4
H	a9a392c8	6118e81d	7a12d3b4	d116e78				
$M_0^{'}$	b3ff8745	c72eb352	5f2ff952	a9b33ceb	7c7993f8	b7687688	c1548362	8180b92e
	6342d45	23f43bfd	7c42a22f	14a1c334	332550f3	34ab42c4	f423620c	341f42b4
$H^{'}$	29a392c8	e308e81d	fc12d3b4	8f116e78				

Table 9. A Counterexample Satisfied Cnditions in Table 4 Ecept for $d_{16,26} = 1$

IV	67452301	efcdab89	98badcfe	10325476				
M_0	b02f512a	c7be044b	13214fe0	c64ba7e7	80c78f18	fd09b7fa	9b9da541	fa069c37
	633ed51	3f40000	8060b83f	d2026306	df7ea316	59cfe3	fffdf	804c0267
H	69bf3aa8	612a7041	125353f8	24072fa0				
$M_0^{'}$	b02f512a	c7be044b	13214fe0	c64ba7e7	00c78f18	fd09b7fa	9b9da541	fa069c37
	633ed51	3f40000	8060b83f	d202e306	df7ea316	59cfe3	800fffdf	804c0267
$H^{'}$	e9bf3aa8	a329f241	945355f8	a6072fa0				

Table 10. A Counterexample Satisfied Cnditions in Table 4 Ecept for $c_{16,26} = 1$

IV	67452301	efcdab89	98badcfe	10325476				
M_0	fc552b76	58000108	ea1a196e	131f05e1	65ec44b7	74f3a5a7	9b9da541	fa069c37
	633ed51	3f40000	8060b84f	12010307	efe02fe	c0fdb	ff7fffef	7ffc018e
H	2e75bf44	7064f413	230c5438	5cd4be64				
$M_0^{'}$	fc552b76	58000108	ea1a196e	131f05e1	e5ec44b7	74f3a5a7	9b9da541	fa069c37
	633ed51	3f40000	8060b84f	12018307	efe02fe	c0fdb	7f7fffef	7ffc018e
$H^{'}$	ae75bf44	f2647413	a50c5438	ded4be64				