Jan. 2007, Vol.22, No.1, pp.79–87 *J. Comput. Sci. & Technol.*

Improved Collision Attack on Hash Function MD5

Jie Liang (梁 杰) and Xue-Jia Lai (来学嘉)

Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China

E-mail: luckyaa@sjtu.edu.cn; lai-xj@cs.sjtu.edu.cn

Received November 29, 2005; revised June 7, 2006.

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 the Conference EUROCRYPT 2005. We found that the derived conditions for the desired collision differential path were not sufficient to guarantee the path to hold and that some conditions could be modified to enlarge the collision set. By using technique of small range searching and omitting the computing steps to check the characteristics in the attack algorithm, we can speed up the attack of MD5 efficiently. Compared with the Advanced Message Modification technique presented by Wang *et al.*, 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.

Keywords MD5, collision, differential attack, hash function

1 Introduction

The hash function MD5[1] was designed by Ronald Rivest in 1992 as a strengthened version of MD4[2]. Though some weakness has been found by B. den Boer, 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 have been significant advances in the analysis of hash function MD5. At the rump session of Crypto'04, Wang *et al.*[5] presented the first collision of MD5. In EUROCRYPT 2005, Wang *et al.* presented a two-block collision differential of MD5[6] that allowed us to search collisions efficiently. They state that the complexity of finding one collision does not exceed the time of running 2^{39} MD5 operations using the attack algorithm presented in [6]. In March 2005, Klima presented Multi-message modifications method to find collision of MD5 on a standard notebook PC in roughly 8 hours[7–9], the estimated complexity is about 2^{33} MD5 operations. However, considering the extra conditions that will be discussed in Section 4, the complexity of the algorithm presented in [6] should be about 2^{41} MD5 operations and the algorithm presented in [7, 8] should be about 2^{36} MD5 operations.

In this paper, we show that the conditions in Tables 4 and 6 of [6] are not sufficient to ensure the occurrence of the desired differential characteristics (which was also found in [10, 11] with a different approach) and specify a set of truly sufficient conditions by adding some extra conditions. Additionally, we modify some conditions in Tables 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 constructing algorithm based on the truly sufficient conditions to omit the steps of checking characteristics like algorithms in [6–8], 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[1] 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 follows:

The chaining variables are initialized as:

$a_0 = 0x67452301$; $d_0 = 0x10325476$;

$c_0 = 0x98badcfe$; $b_0 = 0xefcdab89$;

$S1$: $\sum_1 = a_0 + F(b_0, c_0, d_0) + m_0 + 0xd76aa478$
$\quad a_1 = b_0 + \sum_1 \lll 7$

$S2$: $\sum_2 = d_0 + F(a_1, b_0, c_0) + m_1 + 0xe8c7b756$
$\quad d_1 = a_1 + \sum_2 \lll 12$

$S3$: $\sum_3 = c_0 + F(d_1, a_1, b_0) + m_2 + 0x242070db$
$\quad c_1 = d_1 + \sum_3 \lll 17$

$S4$: $\sum_4 = b_0 + F(c_1, d_1, a_1) + m_3 + 0xc1bdceee$
$\quad b_1 = c_1 + \sum_4 \lll 22$

$\vdots$

$S61$: $\sum_{61} = a_{15} + I(b_{15}, c_{15}, d_{15}) + m_4 + 0xf7537e82$
$\quad a_{16} = b_{15} + \sum_{61} \lll 6$, $aa_0 = a_{16} + a_0$

$S62$: $\sum_{62} = d_{15} + I(a_{16}, b_{15}, c_{15}) + m_{11} + 0xbd3af235$
$\quad d_{16} = a_{16} + \sum_{62} \lll 10$, $dd_0 = d_{16} + d_0$

$S63$: $\sum_{63} = c_{15} + I(d_{16}, a_{16}, b_{15}) + m_2 + 0x2ad7d2bb$
$\quad c_{16} = d_{16} + \sum_{63} \lll 15$, $cc_0 = c_{16} + c_0$

$S64$: $\sum_{64} = b_{15} + I(c_{16}, d_{16}, a_{16}) + m_9 + 0xeb86d391$
$\quad b_{16} = c_{16} + \sum_{64} \lll 21$, $bb_0 = b_{16} + b_0$

where $\lll K$ is cyclically left-shift by K bit positions.

Regular Paper
Supported by the National Natural Science Foundation of China under Grant No. 60573032.

Let aa_0, bb_0, cc_0 and dd_0 be the outputs 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:

$$\begin{aligned} F(x,y,z) &= (x \wedge y) \vee (\neg x \wedge z),\\ G(x,y,z) &= (x \wedge z) \vee (y \wedge \neg z),\\ H(x,y,z) &= x \oplus y \oplus z,\\ I(x,y,z) &= y \oplus (x \vee \neg z), \end{aligned}$$

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 Differential for MD5

The collision differential 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

$$\begin{aligned} \Delta M_0 &= M_0' - M_0\\ &= (0,0,0,0,*2^{31},0,0,0,0,0,0,2^{15},0,0,*2^{31},0),\\ \Delta M_1 &= M_1' - M_1\\ &= (0,0,0,0,*2^{31},0,0,0,0,0,0,-2^{15},0,0,*2^{31},0),\\ \Delta H_1 &= (*2^{31}, *2^{31}+2^{25}, *2^{31}+2^{25}, *2^{31}+2^{25}) \end{aligned}$$

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 differential 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 Differential 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 differential characteristics described in Tables 3 and 5 of [6] to hold. Base on experiments, Yajima and Shimoyama[11] have proposed partial correction about sufficient conditions in August 2005.

4.1 Modified Conditions

During our research, we found that some conditions in Tables 4 and 6 of [6] could be modified to enlarge the collision set. This claim has also been discussed in [11], but limiting in the second iteration differential. We show our new modification 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 one another 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 modified 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.

These conditions' modifications do not have any influence on the differential characteristic in Tables 3 and 5 of [6] except some positive distortions at the higher-order (most significant) bit. We can make these modifications, because of the properties[12] of Boolean functions $F(x,y,z) = (x \wedge y) \vee (\neg x \wedge z)$ and $G(x,y,z) = (x \wedge z) \vee (y \wedge \neg z)$ and the useless carry of bit 32. Regardless the carry of bit 32, the differential output of the Boolean functions F and G have no influence at all after these modifications.

4.2 Derive Extra Conditions

In this Subsection, we show with examples that it is necessary to add some extra conditions into Tables 4 and 6 of [6] in order to keep the desired differential characteristics attained. More detailed explanation is also discussed in [10], where they show the lack of conditions independently.

By computing the 5th step of MD5 in the first iteration:

$$a_2 = b_1 + [a_1 + F(b_1, c_1, d_1) + m_4 + 0xf57c0faf] \lll 7.$$

Let $\sum_5 = a_1 + F(b_1, c_1, d_1) + m_4 + 0xf57c0faf$, then the output difference in the 5th 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 5th step is 2^6; if the bit 32 in $\sum_5$ is one, then the output difference in the 5th 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[6], 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 5th 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 Table 1).

Table 1. Compute the 5th Step of the First Iteration

	bit NO.	1	5	9	⋯
+	$\sum_5 \lll 7$	????	?1**1**?	????	⋯
	b_1	????	**11**0?	???1	⋯
=	a_2	1?1?	**0**100	0000	⋯

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 differential. Extra conditions for other steps are derived using the same method.

We note that there is one important lemma of the extra conditions that we prove as follows.

Lemma. *Extra conditions* $a_{2,27} = 0$*,* $a_{2,29} = 0$*,* $a_{2,30} = 0$ *and* $a_{2,31} = 0$ *of the first iteration are not only sufficient but also necessary conditions for the collision differential described in Section* 3 *to hold.*

Proof. According to the collision differential of MD5 presented in [6], we know the differential characteristic in Step 7 of MD5 should be:

$$(\Delta d_2, \Delta a_2, \Delta b_1, \Delta c_1) \to \Delta c_2$$

where $\Delta c_1 = c_1' - c_1 = 0$, $\Delta b_1 = b_1' - b_1 = 0$,

$$\begin{aligned}\Delta a_2 = a_2' - a_2 = &[7,8,9,10,11,12,13,14,15,16,17,18,\\ &19,20,21,22,-23],\\ \Delta d_2 = d_2' - d_2 = &[-7,24,32],\\ \Delta c_2 = c_2' - c_2 = &[1,2,3,4,5,-6,7,8,9,10,11,-12,-24,\\ &-25,-26,27,28,29,30,31,32].\end{aligned}$$

The operation of Step 7 of MD5:

$$\sum_7 = c_1 + F(d_2, a_2, b_1) + m_6 + 0xa8304613, \quad (1)$$

$$c_2 = d_2 + \sum_7 \lll 17. \quad (2)$$

1) We need the differential characteristic of c_2 is $\Delta c_2 = [1,2,3,4,5,-6,7,8,9,10,11,-12,-24,-25,-26,27,28,29,30,31,32]$, thus c_2 first need to be satisfied conditions: $c_{2,1} = 0, c_{2,2} = 0$, $c_{2,3} = 0, c_{2,4} = 0$, $c_{2,5} = 0, c_{2,6} = 1, \ldots, c_{2,31} = 0$, $c_{2,32} = 0$.

2) As differential $\Delta d_2[-7,24,32]$, so the differential $\Delta c_2[7,8,9,10,11,-12,-24,-25,-26,27,32]$ is passed (or satisfied) from the differential of $\Delta d_2[-7,24,32]$, see (2).

3) According to the differential characteristics of MD5 presented in [6], the differential $\Delta c_2[1,2, 3,4,5,-6,28,29,30,31]$ is passed (or satisfied) from the differential of $\Delta a_2[11,12,13,14, 16,17,18,19,20,21,22,-23]$ after cyclically left-shift by 17 bit positions, see (1). So the differential $\Delta a_2[11,12,13,14,16,17,18,19,20,21,22,-23]$ in $F(d_2,a_2,b_1)$ should be kept. Through the properties of Boolean function $F(x,y,z) = (x \wedge y) \vee (\neg x \wedge z)$, conditions for keeping differential $\Delta a_2[11,12,13,14,16,17,18,19,20,21,22,-23]$ are: $d_{2,11} = 1$, $d_{2,12} = 1$, $d_{2,13} = 1$, $d_{2,14} = 1$, $d_{2,16} = 1$, $d_{2,17} = 1$, $d_{2,18} = 1$, $d_{2,19} = 1$, $d_{2,20} = 1$, $d_{2,21} = 1$, $d_{2,22} = 1$, $d_{2,23} = 1$.

4) The differential of $\Delta d_2[-7,24,32]$ and $\Delta a_2[7,8,9,10,15]$ in $F(d_2,a_2,b_1)$ should be removed or deleted. According to the properties of Boolean function F[12], conditions for canceling differential $\Delta d_2[-7,24,32]$ and $\Delta a_2[7,8,9,10,15]$ in the Boolean function F are: $b_{1,7} = 0, a_{2,24} = b_{1,24}$, $a_{2,32} = b_{1,32}, d_{2,8} = 0$, $d_{2,9} = 0, d_{2,10} = 0$, $d_{2,15} = 0$.

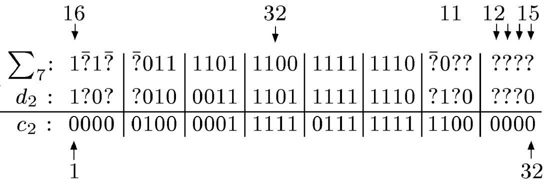
5) As the differential $\Delta a_2[11,12,13,14,16,17,18,19, 20,21,22,-23]$ has been kept (see Step 3), we still need to make sure that this differential can really be passed to satisfy differential $\Delta c_2[1,2,3,4,5,-6,28,29,30,31]$ after the left shift operation.

Notice that $\sum_7 = c_1 + F(d_2,a_2,b_1) + m_6 + 0xa8304613$, and the differential of $\sum_7$ passed from $\Delta a_2[11,12,13,14,16,17,18,19,20,21,22,-23]$ can be expressed in many kinds, such as:

$$\begin{aligned}&\Delta{\textstyle\sum_7}[11,12,13,14,16,17,18,19,20,21,22,-23],\\ &\Delta{\textstyle\sum_7}[-11,15,16,17,18,19,20,21,22,-23],\\ &\Delta{\textstyle\sum_7}[-11,15,-16],\\ &\Delta{\textstyle\sum_7}[-11,-15],\\ &\Delta{\textstyle\sum_7}[11,12,13,14,-16],\\ &\vdots\end{aligned}$$

What kinds of differential of $\sum_7$ may be depended on the value of c_1, d_2, a_2, b_1 and m_6. We can find out that some kinds of expressions should be forbidden in order to guarantee the desired differential characteristics. For this instance, expressions like pattern $\Delta \sum_7[\ldots,-15]$ cannot get the differential $\Delta c_2[1,2,3,4,5,-6,28,29,30,31]$ at all after cyclically left-shift by 17 bit positions.

6) Now we prove that only the expression: $\Delta \sum_7[11,12,13,14,-16]$ is satisfied for the whole differential. For $c_2 = d_2 + \sum_7 \lll 17$, some bit value of d_2 and c_2 have been used to keep the differential characteristics, thus part bit value of $\sum_7$ can be computed out as in Fig.1.



$\sum_7$:	$1\bar{?}1\bar{?}$	$\bar{?}011$	1101	1100	1111	1110	$\bar{?}0??$	????
+ d_2 :	1?0?	?010	0011	1101	1111	1110	?1?0	???0
c_2 :	0000	0100	0001	1111	0111	1111	1100	0000

Fig.1. Compute the 7th step of the first iteration.

From Fig.1, we know that $\sum_{7,16} = 1$, so only nine expressions of differential of $\sum_7$ left, they are: $\Delta \sum_7[-11,-15]$, $\Delta \sum_7[11,-12,-15]$, $\Delta \sum_7[11,12,-13,-15]$, $\Delta \sum_7[11,12,13,-14,-15]$, $\Delta \sum_7[11, 12,13,14,-16]$, $\Delta \sum_7[-11,15,-16]$, $\Delta \sum_7[11,-12,15,-16]$, $\Delta \sum_7[11, 12,-13,15,-16]$ and $\Delta \sum_7[11,12,13, -14,15,-16]$, but $\Delta \sum_7[-11,-15]$, $\Delta \sum_7[11,-12,-15]$, $\Delta \sum_7[11,12, -13,-15]$ and $\Delta \sum_7[11,12,13,-14,-15]$ can not get the differential

$\Delta c_2[1,2,3,4,5,-6,28,29,30,31]$ at all after cyclically left-shift by 17 bit positions. For $\Delta\sum_7[-11,15,-16]$, it needs extra condition: $\sum_{7,11}=1$; then according to the binary addition property (2), $c_2=d_2+\sum_7 \lll 17$ will have carry at bits 28, 29, 30, 31, 32; the carry of bit 31 and condition $d_{2,32}=0$ can make sure that bit 32 of $\sum_7 \lll 17$ should be one or $\sum_{7,15}=1$. For $\sum_{7,15}=1$, expression $\Delta\sum_7[-11,15,-16]$ will not hold at all. For $\Delta\sum_7[11,12,13,14,-16]$, $\Delta\sum_7[11,-12,15,-16]$, $\Delta\sum_7[11,12,-13,15,-16]$ and $\Delta\sum_7[11,12,13,-14,15,-16]$, they all need extra condition: $\sum_{7,11}=0$; see Fig.1, binary addition operation: $d_2+\sum_7 \lll 17$ will have no carry at bit 28, so we can ensure that $\sum_{7,12}=0$, $d_{2,29}=0$, $\sum_{7,13}=0, d_{2,30}=0$, $\sum_{7,14}=0, d_{2,31}=0$, $\sum_{7,15}=0$ hold. For $\sum_{7,12}=0$, $\sum_{7,13}=0$, $\sum_{7,14}=0$ and $\sum_{7,15}=0$, only the expression: $\Delta\sum_7[11,12,13,14,-16]$ is satisfied for the whole differential.

7) Now we prove that the extra conditions $a_{2,27}=0, a_{2,29}=0$, $a_{2,30}=0$ and $a_{2,31}=0$ of the first iteration are not only sufficient but also necessary conditions. In other words, we need to prove that the only expression: $\Delta\sum_7[11,12,13,14,-16]$ need the conditions $a_{2,27}=0, a_{2,29}=0$, $a_{2,30}=0$ and $a_{2,31}=0$ to make sure it occurs. It is easy to be proved according to the binary addition properties. For $\Delta\sum_7[11,12,13,14,-16]$, we need extra conditions: $\sum_{7,11}=0, \sum_{7,12}=0$, $\sum_{7,13}=0$ and $\sum_{7,14}=0$, see Fig.2.

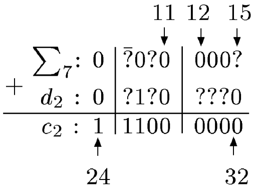


Fig.2. Part of the 7th step computing of the first iteration.

If $d_{2,27}=1$, then carry will happen at bit 27 and cause $\sum_{7,11}=1$; as there is no carry happening at bit 26, condition $d_{2,27}=0$ can ensure $\sum_{7,11}=0$ hold. If $\sum_{7,11}=0$, then $d_{2,29}=0$, $d_{2,30}=0$ and $d_{2,31}=0$ happen. So we need conditions $d_{2,27}=0, d_{2,29}=0$, $d_{2,30}=0$ and $d_{2,31}=0$ to make sure that $\Delta\sum_7[11,12,13,14,-16]$ hold. For the next step differential computing of MD5, we need conditions: $a_{2,27}=d_{2,27}$, $a_{2,29}=d_{2,29}$, $a_{2,30}=d_{2,30}$ and $a_{2,31}=d_{2,31}$ to keep the differential characteristic of b_2, so we have completely proved that conditions $a_{2,27}=0, a_{2,29}=0, a_{2,30}=0$ and $a_{2,31}=0$ of the first iteration are not only sufficient but also necessary conditions. □

For the other steps of MD5, maybe there is more than one expression of the differential about $\sum_*$ that can satisfy to keep the whole collision differential. In this case, we choose the extra conditions that can maximize the satisfied set of differential expressions of $\sum_*$. Generally, the higher order bits are prior to choose as extra conditions for maximizing the satisfied set.

In [11], Jun Yajima and Takeshi Shimoyama also discover the lack of conditions in [6] basing on experiments. They claim that if we omit one of the conditions $a_{2,27}=0, a_{2,29}=0, a_{2,30}=0$ and $a_{2,31}=0$ of the first iteration, the success rate of the modification become extremely low. In this paper, we have proved that these 4 extra conditions are not only sufficient but also necessary conditions for the collision differential described in Section 3. They also try to present a set of sufficient conditions. 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 [11] as follows:

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

Through verification, we know 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 differential.

4.3 Lacked Conditions and Incorrect Conditions

Now we give out a set of truly sufficient conditions in Tables A1 and A2 (in Appendix) that guarantee the desired differential characteristics to occur. The extra conditions in Tables A1 and A2 are just one of the possible extra condition sets derived from $\sum_i$; we choose them out as they are optimal. Comparing with the Table 4 of [6], besides the modified conditions and the extra conditions derived from $\sum_i$, we also add conditions $d_{16,26}=0, c_{16,26}=0$, correct condition $c_{16,32}=d_{16,32}$ to $c_{16,32}=a_{16,32}$ and delete condition $a_{16,27}=0$ in Table A1. The same to adding extra condition $b_{15,26}=0$ in Table A2. We confirm that these 5 conditions are lacked conditions or incorrect conditions in [6], see the deriving conditions details as follows:

In Table A1, we derive conditions $d_{16,26}=0, \sum_{62,16} \sim \sum_{62,22}$ not all ones and $a_{16,32}=c_{15,32}$ to make sure that the differential $\Delta d_{16}=[26,*32]$ occur; conditions $c_{16,26}=0$, $d_{16,32}=b_{15,32}$ and $b_{15,26}=0$ to make sure that the differential $\Delta c_{16}=[26,*32]$ occur; conditions $a_{16,26}=1$ and $c_{16,32}=a_{16,32}$ to make sure that only the differential $\Delta c_{16}=[26,*32]$ can be passed to differential Δb_{16}. Condition $a_{16,27}=0$ is no need at all for keeping the differential characteristics.

In Table A2, we derive conditions $d_{16,26}=1$, $\sum_{62,16} \sim \sum_{62,22}$ not all zeros and $a_{16,32}=c_{15,32}$ to make sure that the differential $\Delta d_{16}=[-26,*32]$ occur; conditions $c_{16,26}=1$, $d_{16,32}=b_{15,32}$ and $b_{15,26}=0$ to make sure that the differential $\Delta c_{16}=[-26,*32]$ occur; conditions $a_{16,26}=1$ and $c_{16,32}=a_{16,32}$ to make sure that only the differential $\Delta c_{16} = [-26, *32]$ can be passed to differential Δb_{16}.

In fact, the output differential characteristics for Δd_{16} and Δc_{16} in the first iteration differential can be randomly chosen to be $[26, *32], [-26, 27, *32]$, $[-26, -27, 28, *32]$, $[-26, -27, -28, 29, *32]$, $[-26, -27, -28, -29, 30, *32]$, $[-26, -27, -28, -29, -30, 31, *32]$ or $[-26, -27, -28, -29, -30, -31]$ (the same to the output differential characteristics for Δd_{16} and Δc_{16} in the second iteration differential). So conditions $d_{16,26} = 0$, $c_{16,26} = 0$, $d_{16,32} = b_{15,32}$, and $c_{16,32} = a_{16,32}$ are not necessary conditions in Table A1 (conditions $d_{16,26} = 1$, $c_{16,26} = 1$, $d_{16,32} = b_{15,32}$ and $c_{16,32} = a_{16,32}$ are also not necessary conditions in Table A2). But here the differential characteristics for Δd_{16} and Δc_{16} chosen to be $[26, *32]$ ($[-26, *32]$ for the second iteration differential) is the best in the sense that it needs the least number of conditions for keeping the differential characteristics.

4.4 How to Construct Sufficient Conditions

From the condition-deriving method described in [6] and the extra-condition deriving method described in the above section, we can summarize the method of constructing sufficient conditions for a collision differential of MD5. These sufficient conditions are derived to ensure that the collision differential and its corresponding differential characteristics will always hold. When deriving sufficient conditions there are three aspects need to be considered:

1) Necessary conditions of the differential characteristic itself. For example, $\Delta X_* = X'_* - X_* = [1, 2, 3, 4, -5, -6, 7]$, then the necessary conditions for this differential characteristic are $X_{*,1} = 0$, $X_{*,2} = 0$, $X_{*,3} = 0$, $X_{*,4} = 0$, $X_{*,5} = 1$, $X_{*,6} = 1$, and $X_{*,7} = 0$.

2) Necessary conditions to control the differential output of the Boolean function for the need of the next differential characteristic.

3) Extra conditions to control the differential output of left shift rotation operation for the need of the next differential characteristic. The necessary conditions (see Steps 1 and 2) should be derived first, then consider the extra conditions to control the differential output of left rotation operation.

A set of conditions like Tables A1 and A2 derived from the three aspects must be the sufficient conditions set, as it makes sure that the collision differential always holds.

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 $N = U + [L + F(X, Y, Z) + M + \mathit{Constant}] \lll k$, 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 the 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. Our basic message modification technique shown below is 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 A1 hold (see Table 2):

Step 6: $d_2 = a_2 + [d_1 + F(a_2, b_1, c_1) + m_5 + 0x4787c62a] \lll 12$;

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:

$$\begin{aligned} d_2 &= \{(d_2) \wedge [(d_2)\&(0xfd8043be)] \wedge [(a_2)\& \\ &\quad (0x7500001a)]\}|(0x027fbc41), \\ m_5 &= [(d_2 - a_2) \ggg 12] - d_1 - F(a_2, b_1, c_1) - 0x4787c62a. \end{aligned}$$

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 modifying 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:

$$\begin{aligned} \textstyle\sum_{18} &= d_4 + G(a_5, b_4, c_4) + m_6 + 0xc040b340, \\ d_5 &= a_5 + \textstyle\sum_{18} \lll 9. \end{aligned}$$

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}:

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

(d) Compute the 19th step:

$$\begin{aligned} \textstyle\sum_{19} &= c_4 + G(d_5, a_5, b_4) + m_{11} + 0x265e5a51 \\ c_5 &= d_5 + \textstyle\sum_{19} \lll 14. \end{aligned}$$

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} ($m_{11} = [(b_3 - c_3) \ggg 22] - b_2 - F(c_3, d_3, a_3) - 0x895cd7be$), 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 values of $m_{12}, m_{13}, m_{14}, m_{15}$:

$$
\begin{aligned}
m_{12} &= [(a_4 - b_3) \ggg 7] - a_3 - F(b_3, c_3, d_3) - 0x6b901122,\\
m_{13} &= [(d_4 - a_4) \ggg 12] - d_3 - F(a_4, b_3, c_3) - 0xfd987193,\\
m_{14} &= [(c_4 - d_4) \ggg 17] - c_3 - F(d_4, a_4, b_3) - 0xa679438e,\\
m_{15} &= [(b_4 - c_4) \ggg 22] - b_3 - F(c_4, d_4, a_4) - 0x49b40821.
\end{aligned}
$$

(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 values of $m_1, m_0, a_1, d_1, m_2, m_3, m_4$ and m_5:

$$
\begin{aligned}
m_1 &= [(a_5 - b_4) \ggg 5] - a_4 - G(b_4, c_4, d_4) - 0xf61e2562,\\
m_0 &= [(b_5 - c_5) \ggg 20] - b_4 - G(c_5, d_5, a_5) - 0xe9b6c7aa,\\
a_1 &= b_0 + [a_0 + F(b_0, c_0, d_0) + m_0 + 0xd76aa478] \lll 7,\\
d_1 &= a_1 + [d_0 + F(a_1, b_0, c_0) + m_1 + 0xe8c7b756] \lll 12,\\
m_2 &= [(c_1 - d_1) \ggg 17] - c_0 - F(d_1, a_1, b_0) - 0x242070db,\\
m_3 &= [(b_1 - c_1) \ggg 22] - b_0 - F(c_1, d_1, a_1) - 0xc1bdceee,\\
m_4 &= [(a_2 - b_1) \ggg 7] - a_1 - F(b_1, c_1, d_1) - 0xf57c0faf,\\
m_5 &= [(d_2 - a_2) \ggg 12] - d_1 - F(a_2, b_1, c_1) - 0x4787c62a.
\end{aligned}
$$

(f) Continuing compute with the remaining steps, if any condition in Table A1 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 $m_0, m_1, m_2 \ldots m_{13}$.

(b) Modify $m_0, m_1, m_2 \ldots m_{13}$ 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}:

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

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}:

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

(d) Compute the 17th step:

$$
\begin{aligned}
\textstyle\sum_{17} &= a_4 + G(b_4, c_4, d_4) + m_1 + 0xf61e2562e\\
a_5 &= b_4 + \textstyle\sum_{17} \lll 5.
\end{aligned}
$$

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 values of m_1, m_{15} ($m_1 = [(d_1 - a_1) \ggg 12] - dd_0 - F(a_1, bb_0, cc_0) - 0xe8c7b756$, $m_{15} = [(b_4 - c_4) \ggg 22] - b_3 - F(c_4, d_4, a_4) - 0x49b40821$) 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 ($m_1 = [(d_1 - a_1) \ggg 12] - dd_0 - F(a_1, bb_0, cc_0) - 0xe8c7b756$) to correct them.

(e) Compute the 18th step:

$$
\begin{aligned}
\textstyle\sum_{18} &= d_4 + G(a_5, b_4, c_4) + m_6 + 0xc040b340\\
d_5 &= a_5 + \textstyle\sum_{18} \lll 9.
\end{aligned}
$$

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 values of m_6 ($m_6 = [(c_2 - d_2) \ggg 17] - c_1 - F(d_2, a_2, b_1) - 0xa8304613$) to correct it. Then we also need to update the values of m_7, m_8, m_9 and m_{10}:

$$
\begin{aligned}
m_7 &= [(b_2 - c_2) \ggg 22] - b_1 - F(c_2, d_2, a_2) - 0xfd469501,\\
m_8 &= [(a_3 - b_2) \ggg 7] - a_2 - F(b_2, c_2, d_2) - 0x698098d8,\\
m_9 &= [(d_3 - a_3) \ggg 12] - d_2 - F(a_3, b_2, c_2) - 0x8b44f7af,\\
m_{10} &= [(c_3 - d_3) \ggg 17] - c_2 - F(d_3, a_3, b_2) - 0xffff5bb1.
\end{aligned}
$$

(f) Compute the 19th step:

$$
\begin{aligned}
\textstyle\sum_{19} &= c_4 + G(d_5, a_5, b_4) + m_{11} + 0x265e5a51\\
c_5 &= d_5 + \textstyle\sum_{19} \lll 14.
\end{aligned}
$$

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} ($m_{11} = [(b_3 - c_3) \ggg 22] - b_2 - F(c_3, d_3, a_3) - 0x895cd7be$), 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 values of m_{12}, m_{13}, m_{14} and m_{15}:

$$
\begin{aligned}
m_{12} &= [(a_4 - b_3) \ggg 7] - a_3 - F(b_3, c_3, d_3) - 0x6b901122,\\
m_{13} &= [(d_4 - a_4) \ggg 12] - d_3 - F(a_4, b_3, c_3) - 0xfd987193,\\
m_{14} &= [(c_4 - d_4) \ggg 17] - c_3 - F(d_4, a_4, b_3) - 0xa679438e,\\
m_{15} &= [(b_4 - c_4) \ggg 22] - b_3 - F(c_4, d_4, a_4) - 0x49b40821.
\end{aligned}
$$

(g) Compute the 20th step:

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

$$b_5 = c_5 + \sum\nolimits_{20} \lll 20.$$

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 Table A2), 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 $F(x, y, z) = (x \wedge y) \vee (\neg x \wedge z)$[12], 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 ($m_1 = [(d_1 - a_1) \ggg 12] - dd_0 - F(a_1, bb_0, cc_0) - 0xe8c7b756$) 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:

$$\begin{aligned}
m_0 &= [(a_1 - bb_0) \ggg 7] - aa_0 - F(bb_0, cc_0, dd_0) - 0xd76aa478,\\
m_2 &= [(c_1 - d_1) \ggg 17] - cc_0 - F(d_1, a_1, bb_0) - 0x242070db,\\
m_3 &= [(b_1 - c_1) \ggg 22] - bb_0 - F(c_1, d_1, a_1) - 0xc1bdceee,\\
m_4 &= [(a_2 - b_1) \ggg 7] - a_1 - F(b_1, c_1, d_1) - 0xf57c0faf,\\
m_5 &= [(d_2 - a_2) \ggg 12] - d_1 - F(a_2, b_1, c_1) - 0x4787c62a.
\end{aligned}$$

(i) Continuing with the remaining steps of MD5 from Step 20, if any condition in Table A2 is not satisfied, jump to Step (h). If all the possible selections in Step (h) fail, then jump to Step (c).

5.3 Speed of Our Algorithm

By using the basic message modification technique and small range searching technique, about 35 conditions in Rounds 2∼4 are undetermined in Table A1, and about 31 conditions in Rounds 2∼4 are undetermined in Table A2. Consider the extra conditions: the attack algorithm described in [6] should have about 39 conditions in Rounds 2∼4 undetermined in Table A1 and about 34 conditions in Rounds 2∼4 undetermined in Table A2. 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 algorithm 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.

Compared 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 do not need test whether the characteristics really hold in every step like [6, 8], thus our attack algorithm could be considered at least 2 times faster than the algorithm present in [7, 8].

In the 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. Collision Example for MD5

IV	67452301	efcdab89	98badcfe	10325476
M_0	055a604a	a3461df0	12221694	6c449744
	25c44d2c	a1b99a33	92681957	3c554e32
	0632ed41	03f3f7fc	805eb737	1300af02
	befc06c7	0099e023	ff80803f	0000bd93
M_1	c0e83a00	37f3af4e	95243bff	f2e16edf
	b4cc3b03	fcbaa5a3	852088e8	c00d7bd1
	fe32fffd	a7e84fe0	30803ffe	dc833c85
	5f1330ed	088bde83	6f89b53d	819a57f0
M_0'	055a604a	a3461df0	12221694	6c449744
	a5c44d2c	a1b99a33	92681957	3c554e32
	0632ed41	03f3f7fc	805eb737	13012f02
	befc06c7	0099e023	7f80803f	0000bd93
M_1'	c0e83a00	37f3af4e	95243bff	f2e16edf
	34cc3b03	fcbaa5a3	852088e8	c00d7bd1
	fe32fffd	a7e84fe0	30803ffe	dc82bc85
	5f1330ed	088bde83	ef89b53d	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. 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 *et al.* present their new message modification techniques in [13], they claim that their modification techniques can correct 14 conditions in Round 2 with success probability about 1/2. Thus, use the same complexity estimation method described in [6], we claim that the complexity is about 2^{33} MD5 operations considering the extra conditions in Tables A1 and A2. 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 be easily fulfilled with high probability.

References

[1] Ronald Rivest. The MD5 message digest algorithm. RFC1321, April 1992, http://rfc.net/rfc1321.html.

[2] Ronald Rivest. The MD4 message digest algorithm. RFC1320, April 1992, http://rfc.net/rfc1320.html.

[3] B den Boer, A. Bosselaers. Collisions for the compression function of MD5. In *Proc. Advances in Cryptology, EUROCRYPT'93, LNCS 765*, Helleseth T (ed.), Springer-Verlag, Berlin, Germany, 1994, pp.293–304.

[4] Dobbertin H. Cryptanalysis of MD5 compress. Rump session of Eurocrypt'96, http://www.cs.ucsd.edu/users/bsy/dobbertin.ps, 1996.

[5] Xiaoyun Wang, Dengguo Feng, Xuejia Lai, Hongbo Yu. Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMD. Rump session of Crypto'04, Cryptology ePrint Archive, Report2004/199, http://eprint.iacr.org/2004/199.pdf, 2004.

[6] Xiaoyun Wang, Hongbo Yu. How to break MD5 and other hash functions. In *Proc. Advances in Crytolog—EUROCRYPT 2005, LNCS3494*, Cramer R (ed.), Springer-Verlag, pp.19–35.

[7] Vlastimil Klima. Finding MD5 collisions on a notebook PC using multi-message modifications. In *Proc. International Scientific Conference Security and Protection of Information 2005*, Brno, Czech Republic, May 3–5, 2005, http://eprint.iacr.org/2005/102.pdf.

[8] Patrick Stach. MD5 Collision Generator. ⟨pstach@stachliu.com⟩, http://www.stachliu.com. nyud.net:8090/md5coll.c.

[9] Vlastimil Klima. Finding MD5 collisions—A toy for a notebook. Cryptology ePrint Archive, Report2005/075, March, 2005, http://eprint.iacr.org/2005/075.pdf.

[10] Zhang-Yi Wang, Huan-Guo Zhang, Zhong-Ping Qin, Qing-Shu Meng. A fast attack on the MD5 hash function. *Journal of Shanghai Jiaotong University*, 2006, 11(2): 140–145, 151.

[11] Jun Yajima, Takeshi Shimoyama. Wang's sufficient conditions of MD5 are not sufficient. Cryptology ePrint Archive, Report2005/263, 2005, http://eprint.iacr.org/2005/263.pdf.

[12] Xiaoyun Wang, Xuejia Lai, Dengguo Feng. Cryptanalysis of the hash functions MD4 and RIPEMD. In *Proc. Advances in Cryptology—EUROCRYPT 2005, LNCS 3494*, Cramer R (ed.), Springer-Verlag, 2005, pp.1–18.

[13] Yu Sasaki, Yusuke Naito, Noboru Kunihiro, Kazuo Ohta. Improved collision attack on MD5. Cryptology ePrint Archive, Report2005/400, Nov, 2005, http://eprint.iacr.org/2005/400.pdf.

Jie Liang was born in 1980. He received his B.Sc. degree in July 2004 from Dept. Electronic Science and Engineering, Jilin Univ., China. He is currently a master candidate of Shanghai Jiao Tong University in the Dept. Computer Science and Engineering, working in computer information security and cryptography under the superb supervision of Prof. Dr. Xuejia Lai. His research interests are hash function, trusted computing and DRM scheme.

Xue-Jia Lai received his B.Sc. degree in El. Ing. in 1982 and M.Sc. degree in math. from the Xidian Univ., China, 1984. He received his Ph.D. degree of Sc. Techn. in 1992 from the Swiss Federal Institute of Technology, Zurich. He is a professor of Shanghai Jiao Tong University since 2004. His work has been concentrated in cryptography and PKI during the past 20 years, especially in the design and analysis of practical cryptosystems (including block ciphers and stream ciphers), differential cryptanalysis of block ciphers, and analysis hash functions. He is a co-inventor (together with Prof. J. L. Massey) of the IDEA cipher. In 1994, he joined r^3 security engineering ag which became Entrust Technologies Switzerland since June 1998. He was senior consultant S.W.I.S. GROUP, Switzerland since 2001. He was editor of 3 ISO IT-security standards. He has published a book "On the Design and Security of Block Ciphers" (Hartung-Gorre Verlag, 1992) and more than 40 papers.

Appendix

Table A1. 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 $\sum_i$
$b_{1,7}=0, b_{1,8}=c_{1,8}, b_{1,9}=c_{1,9}, b_{1,10}=c_{1,10}, b_{1,11}=c_{1,11}, b_{1,12}=1, b_{1,13}=c_{1,13},$ $b_{1,14}=c_{1,14}, b_{1,15}=c_{1,15}, b_{1,16}=c_{1,16}, b_{1,17}=c_{1,17}, b_{1,18}=c_{1,18}, b_{1,19}=c_{1,19},$ $b_{1,20}=1, b_{1,21}=c_{1,21}, b_{1,22}=c_{1,22}, b_{1,23}=c_{1,23}, b_{1,24}=0, b_{1,32}=1$	$\sum_5 \Rightarrow b_{1,5}=1, b_{1,6}=1, a_{2,5}=0$
$a_{2,1}=1, a_{2,3}=1, a_{2,6}=1, a_{2,7}=0, a_{2,8}=0, a_{2,9}=0, a_{2,10}=0, a_{2,11}=0,$ $a_{2,12}=0, a_{2,13}=0, a_{2,14}=0, a_{2,15}=0, a_{2,16}=0, a_{2,17}=0, a_{2,18}=0, a_{2,19}=0, a_{2,20}=0,$ $a_{2,21}=0, a_{2,22}=0, a_{2,23}=1, a_{2,24}=0, a_{2,26}=0, a_{2,28}=1, a_{2,32}=1$	$\sum_7 \Rightarrow a_{2,27}=0, a_{2,29}=0$, $a_{2,30}=0, a_{2,31}=0$
$d_{2,1}=1, d_{2,2}=a_{2,2}, d_{2,3}=0, d_{2,4}=a_{2,4}, d_{2,5}=a_{2,5}, d_{2,6}=0, d_{2,7}=1, d_{2,8}=0, d_{2,9}=0, d_{2,10}=0, d_{2,11}=1, d_{2,12}=1,$ $d_{2,13}=1, d_{2,14}=1, d_{2,15}=0, d_{2,16}=1, d_{2,17}=1, d_{2,18}=1, d_{2,19}=1, d_{2,20}=1, d_{2,21}=1, d_{2,22}=1, d_{2,23}=1, d_{2,24}=0,$ $d_{2,25}=a_{2,25}, d_{2,26}=1, d_{2,27}=a_{2,27}, d_{2,28}=0, d_{2,29}=a_{2,29}, d_{2,30}=a_{2,30}, d_{2,31}=a_{2,31}, d_{2,32}=0$	
$c_{2,1}=0, c_{2,2}=0, c_{2,3}=0, c_{2,4}=0, c_{2,5}=0, c_{2,6}=1, c_{2,7}=0, c_{2,8}=0, c_{2,9}=0, c_{2,10}=0, c_{2,11}=0, c_{2,12}=1, c_{2,13}=1,$ $c_{2,14}=1, c_{2,15}=1, c_{2,16}=1, c_{2,17}=0, c_{2,18}=1, c_{2,19}=1, c_{2,20}=1, c_{2,21}=1, c_{2,22}=1, c_{2,23}=1, c_{2,24}=1, c_{2,25}=1,$ $c_{2,26}=1, c_{2,27}=0, c_{2,28}=0, c_{2,29}=0, c_{2,30}=0, c_{2,31}=0, c_{2,32}=0$	
$b_{2,1}=0, b_{2,2}=0, b_{2,3}=0, b_{2,4}=0, b_{2,5}=0, b_{2,6}=0, b_{2,7}=1, b_{2,8}=0, b_{2,9}=1, b_{2,10}=0, b_{2,11}=1, b_{2,12}=0,$ $b_{2,14}=0, b_{2,16}=0, b_{2,17}=1, b_{2,18}=0, b_{2,19}=0, b_{2,20}=0, b_{2,21}=1, b_{2,24}=1, b_{2,25}=1, b_{2,26}=0, b_{2,27}=0,$ $b_{2,28}=0, b_{2,29}=0, b_{2,30}=0, b_{2,31}=0, b_{2,32}=0$	

To be continued on the next page

Continuing from the previous page

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

Table A2. 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 $\sum_i$
$d_{1,2} = 0, d_{1,3} = 0, d_{1,6} = 0, d_{1,7} = a_{1,7}, d_{1,8} = a_{1,8}, d_{1,12} = 1, d_{1,13} = a_{1,13},$ $d_{1,17} = 1, d_{1,18} = a_{1,18}, d_{1,19} = a_{1,19}, d_{1,20} = a_{1,20}, d_{1,21} = a_{1,21}, d_{1,22} = 0, d_{1,26} = 0,$ $d_{1,27} = 1, d_{1,28} = 1, d_{1,29} = a_{1,29}, d_{1,30} = a_{1,30}, d_{1,31} = a_{1,31}, d_{1,32} = a_{1,32}$	$\sum_3 \Rightarrow a_{1,17} = 1, d_{1,16} = 0, c_{1,16} = 1$
$c_{1,2} = 1, c_{1,3} = 1, c_{1,4} = d_{1,4}, c_{1,5} = d_{1,5}, c_{1,6} = 1, c_{1,7} = 1, c_{1,8} = 0, c_{1,9} = 1,$ $c_{1,12} = 1, c_{1,13} = 0, c_{1,17} = 1, c_{1,18} = 1, c_{1,19} = 1, c_{1,20} = 1, c_{1,21} = 1, c_{1,22} = 0, c_{1,26} = 1,$ $c_{1,27} = 1, c_{1,28} = 1, c_{1,29} = 1, c_{1,30} = 1, c_{1,31} = 0, c_{1,32} = d_{1,32}$	$\sum_5 \Rightarrow c_{1,1} = 1$
$b_{1,1} = 1, b_{1,2} = 0, b_{1,3} = 0, b_{1,4} = 0, b_{1,5} = 1, b_{1,6} = 0, b_{1,7} = 0, b_{1,8} = 0, b_{1,9} = 0, b_{1,10} = c_{1,10}, b_{1,11} = c_{1,11}, b_{1,12} = 0, b_{1,13} = 0,$ $b_{1,17} = 0, b_{1,18} = 0, b_{1,19} = 1, b_{1,20} = 0, b_{1,21} = 0, b_{1,22} = 0, b_{1,26} = 1, b_{1,27} = 0, b_{1,28} = 1, b_{1,29} = 1, b_{1,30} = 1, b_{1,31} = 0, b_{1,32} = c_{1,32}$	
$a_{2,1} = 0, a_{2,2} = 0, a_{2,3} = 0, a_{2,4} = 0, a_{2,5} = 1, a_{2,6} = 0, a_{2,7} = 1, a_{2,8} = 0, a_{2,9} = 0,$ $a_{2,10} = 1, a_{2,11} = 1, a_{2,12} = 1, a_{2,13} = 0, a_{2,17} = 1, a_{2,18} = 1, a_{2,19} = 1, a_{2,20} = 1, a_{2,21} = 0,$ $a_{2,22} = 1, a_{2,27} = 0, a_{2,28} = 1, a_{2,29} = 0, a_{2,30} = 0, a_{2,31} = 1, a_{2,32} = b_{1,32} + 1$	$\sum_7 \Rightarrow d_{2,15} = 1, c_{2,15} = 0$
$d_{2,1} = 0, d_{2,2} = 1, d_{2,3} = 1, d_{2,4} = 0, d_{2,5} = 1, d_{2,6} = 0, d_{2,7} = 1, d_{2,8} = 0, d_{2,9} = 0, d_{2,10} = 0, d_{2,11} = 1, d_{2,12} = 1,$ $d_{2,13} = 0, d_{2,17} = 0, d_{2,18} = 1, d_{2,21} = 0, d_{2,22} = 1, d_{2,26} = 0, d_{2,27} = 1, d_{2,28} = 0, d_{2,29} = 0, d_{2,32} = a_{2,32}$	
$c_{2,1} = 1, c_{2,7} = 0, c_{2,8} = 0, c_{2,9} = 0, c_{2,10} = 1, c_{2,11} = 1, c_{2,12} = 1, c_{2,13} = 1, c_{2,16} = d_{2,16}, c_{2,17} = 1, c_{2,18} = 0, c_{2,21} = 0,$ $c_{2,22} = 0, c_{2,24} = d_{2,24}, c_{2,25} = d_{2,25}, c_{2,26} = 1, c_{2,27} = 1, c_{2,28} = 0, c_{2,29} = 1, c_{2,32} = d_{2,32} + 1$	
$b_{2,1} = 0, b_{2,2} = c_{2,2}, b_{2,7} = 1, b_{2,8} = 1, b_{2,9} = 1, b_{2,10} = 1, b_{2,16} = 1, b_{2,17} = 0, b_{2,18} = 1,$ $b_{2,21} = 1, b_{2,22} = 1, b_{2,24} = 0, b_{2,25} = 0, b_{2,26} = 0, b_{2,27} = 1, b_{2,28} = 0, b_{2,29} = 0, b_{2,32} = c_{2,32}$	$\sum_9 \Rightarrow b_{2,6} = 1, a_{3,6} = 0$
$a_{3,1} = 1, a_{3,2} = 0, a_{3,7} = 1, a_{3,8} = 1, a_{3,9} = 1, a_{3,10} = 0, a_{3,13} = b_{2,13}, a_{3,16} = 0, a_{3,17} = 1, a_{3,18} = 0, a_{3,24} = 0, a_{3,25} = 0,$ $a_{3,26} = 0, a_{3,27} = 1, a_{3,28} = 1, a_{3,29} = 1, a_{3,32} = b_{2,32}$	
$d_{3,1} = 0, d_{3,2} = 0, d_{3,7} = 1, d_{3,8} = 1, d_{3,9} = 1, d_{3,10} = 1, d_{3,13} = 0, d_{3,16} = 1,$ $d_{3,17} = 1, d_{3,18} = 1, d_{3,19} = 0, d_{3,24} = 1, d_{3,25} = 1, d_{3,26} = 1, d_{3,27} = 1, d_{3,32} = a_{3,32}$	$\sum_{11} \Rightarrow d_{3,12} = 1, c_{3,12} = 0$
$c_{3,1} = 1, c_{3,2} = 1, c_{3,7} = 1, c_{3,8} = 1, c_{3,9} = 1, c_{3,10} = 1, c_{3,13} = 0, c_{3,14} = d_{3,14}, c_{3,15} = d_{3,15}, c_{3,16} = 1, c_{3,17} = 1,$ $c_{3,18} = 0, c_{3,19} = 1, c_{3,20} = d_{3,20}, c_{3,32} = d_{3,32}$	
$b_{3,8} = 1, b_{3,13} = 1, b_{3,14} = 0, b_{3,15} = 0, b_{3,16} = 0, b_{3,17} = 0, b_{3,18} = 0, b_{3,19} = 0, b_{3,20} = 1, b_{3,25} = c_{3,25}, b_{3,26} = c_{3,26},$ $b_{3,27} = c_{3,27}, b_{3,28} = c_{3,28}, b_{3,29} = c_{3,29}, b_{3,30} = c_{3,30}, b_{3,31} = c_{3,31}, b_{3,32} = c_{3,32}$	
$a_{4,4} = 1, a_{4,8} = 0, a_{4,14} = 1, a_{4,15} = 1, a_{4,16} = 1, a_{4,17} = 1, a_{4,18} = 1, a_{4,19} = 1, a_{4,20} = 1,$ $a_{4,25} = 1, a_{4,26} = 1, a_{4,27} = 1, a_{4,28} = 1, a_{4,29} = 1, a_{4,30} = 1, a_{4,31} = 0, a_{4,32} = b_{3,32} + 1$	$\sum_{14} \Rightarrow a_{4,24} = 0, d_{4,24} = 1$
$d_{4,4} = 1, d_{4,8} = 1, d_{4,14} = 1, d_{4,15} = 1, d_{4,16} = 1, d_{4,17} = 1, d_{4,18} = 1, d_{4,19} = 0, d_{4,20} = 1, d_{4,25} = 0, d_{4,26} = 0, d_{4,27} = 0,$ $d_{4,28} = 0, d_{4,29} = 0, d_{4,30} = 0, d_{4,31} = 1, d_{4,32} = a_{4,32}$	
$c_{4,4} = 0, c_{4,16} = 0, c_{4,25} = 1, c_{4,26} = 0, c_{4,27} = 1, c_{4,28} = 1, c_{4,29} = 1, c_{4,30} = 1, c_{4,31} = 1$	$\sum_{15} \Rightarrow c_{4,17} = 0$
$b_{4,30} = 1, b_{4,32} = c_{4,32}$	$\sum_{16} \Rightarrow b_{4,16} = 1, b_{4,17} = 1, b_{4,29} = 0$
$a_{5,4} = b_{4,4}, a_{5,16} = b_{4,16}, a_{5,18} = 0, a_{5,32} = b_{4,32}$	$\sum_{17,25} \sim \sum_{17,27}$ 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
$b_{5,32} = c_{5,32}$	$\sum_{20,30} \sim_{20,32}$ not all zeros
$a_{6,18} = b_{5,18}, d_{6,32} = a_{6,32}, c_{6,32} = 0, b_{6,32} = c_{6,32} + 1, b_{12,32} = d_{12,32}$	$\sum_{23,18} = 0, \sum_{35,16} = 1$
$a_{13,32} = c_{12,32}, d_{13,32} = b_{12,32} + 1, c_{13,32} = a_{13,32}, b_{13,32} = d_{13,32}, a_{14,32} = c_{13,32}, d_{14,32} = b_{13,32}, c_{14,32} = a_{14,32},$ $b_{14,32} = d_{14,32}, a_{15,32} = c_{14,32}, d_{15,32} = b_{14,32}, c_{15,32} = a_{15,32}, \boldsymbol{b_{15,26} = 0}, b_{15,32} = d_{15,32} + 1, a_{16,26} = 1, a_{16,32} = c_{15,32}$	
$d_{16,26} = 1, d_{16,32} = b_{15,32}, c_{16,26} = 1, c_{16,32} = a_{16,32}, b_{16,26} = 1$	$\sum_{62,16} \sim \sum_{62,22}$ not all zeros