Improved Collision Attack on MD5

Yu Sasaki* Yusuke Naito* Noboru Kunihiro* Kazuo Ohta*

*The University of Electro-Communications, Japan
{ yu339, tolucky } @ice.uec.ac.jp

Abstract

In EUROCRYPT2005, a collision attack on MD5 was proposed by Wang et al. In this attack, conditions which are sufficient to generate collisions (called "sufficient condition") are introduced. This attack raises the success probability by modifing messages to satisfy these conditions. In this attack, 37 conditions cannot be satisfied even messages are modified. Therefore, the complexity is 2^{37}. After that, Klima improved this result. Since 33 conditions cannot be satisfied in his method, the complexity is 2^{33}.

In this paper, we propose new message modification techniques which are more efficient than attacks proposed so far. In this method, 29 conditions cannot be satisfied. However, this method is probabilistic, and the probability that this method work correctly is roughly 1/2. Therefore, the complexity of this attack is 2^{30}. Furthermore, we propose a more efficient collision search algorithm than that of Wang et al. By using this algorithm, the total complexity is reduced into roughly 5/8.

keywords: MD5, collision attack, message modification, sufficient condition

1 Introduction

MD5 is one of the hash fucntions which compress an arbitrary length message into a defined length random message. (In case of MD5, the output is 128-bit.) Since hash functions are composed of addition, XOR, bit-rotation and other light operations, they can be calculated quickly. One of the important properties of hash functions is called collision resistance. Let x be a message and $h(x)$ be a hash value of x. Collision resistance means that it is difficult to find a pair of message (x, y), where $h(x) = h(y)$.

In 1992, Rivest proposed a hash function named MD4 [3]. After that, in 1992, MD5 [4] which is an improved version of MD4 was proposed by Rivest. Then various hash functions based on MD4 such as RIPEMD, SHA-0 and SHA-1 were proposed.

In 1996, Dobbertin proposed a collision attack on MD5 [1]. However, since this attack used modified initial value, this attack was not real attack for MD5. In 2005, Wang et al. proposed an efficient collision attack on MD5 [6]. This attack is a kind of differential attack. Although almost all differential attacks use XOR operation to calculate differential value, Wang et al. uses modular subtraction instead of XOR. In the method of Wang et al, the input differenatial is decided in advance, and the goal of the attack is finding messages which cancel the input differential and make a differential of the hashed values of input messages 0. This attack introduces conditions on chaining variablesto make the output differential 0. These conditions are called "sufficient condition." If an input message satisfies all conditions, collision messages can deterministically be generated. According to [6], the number of conditions is 290. Therefore, the probability that a randomly message satisfies all conditions is 2^{-290}, and this is very small. However, this probability can be improved by modifying a message to satisfy conditions. In this attack, message modification techniques which satisfy except for 37 conditions are proposed. Therefore, the complexity of this attack is 2^{37}.

In 2005, Klima improved the attack of Wang et al. [2]. Message modification techniques discribed in [2] can satisfy except for 33 conditions. Therefore, the complexity of this attack is 2^{33}. In CRYPTO2005, Wang et al. represented an attack to SHA-1 [7]. In this paper, they claimed that they found a technique to generate collisions of MD5 with complexity 2^{32} though details are not mentioned.

In this paper, we propose new message modification techniques which can satisfy except for 29 conditions with probability 1/2. Therefore, the complexity of our attack is 2^{30}. We also propose an efficient collision search algorithm which can reduce the complexity into roughly 5/8 compared to an algorithm of Wang et al.

2 Description of MD5

MD5 is a compression function which calculates a 128-bit random value from an arbitrary length message. When a message M is inputted, the hash value of M is calculated by the following way:

1. A message M is devided into 512-bit messages.

$$M = (M_0, M_1, ..., M_n),\ |M_i| = 512$$

2. Let h_i be the output of i-th operation. h_i is calculated by the MD5 compression function with M_{i-1} and h_{i-1}. Repeat this process from M_0 to M_n.
   The initial value h_0 is defined as follows,
   $h_0 = (0x67452301, 0xefcdab89, 0x98badcfe, 0x10325476)$.
3. The output of the operation for the last message is the hash value of M.

Compression Function

A message M_j is compressed by the compression function. This operation is called the $(j+1)$-th block operation. Each block consists of 64 steps. Step 1 to step 16 are called the first round, step 17 to step 32 are called the second round, step 33 to step 48 are called the third round and step 49 to step 64 are called the forth round. In each step, one of the chaining variables a, b, c, d is calculated. The order of the calculated chaining variables is $a_1, d_1, c_1, b_1, a_2, \cdots$. The output of each block h_i is $h_i = (a_0 + a_{16}, b_0 + b_{16}, c_0 + c_{16}, d_0 + d_{16})$.

Chaining variables are calculated in the following way. First, M_j is devieded into 32-bit messages $m_0, \ldots, m_{15}$. Then, a function ψ is defined as follows,

$$\psi(x, y, z, w, m, s, t) = y + ((x + \phi(y, z, w) + m + t) \mod 2^{32}) \lll s,$$

where m denotes a message inputted in that step. Which m is inputted is defined in advance. s denotes a number for left cyclic shift defined in each step. t denotes a constant number defined in each step. ϕ denotes a non-linear function defined in each round. Details of the function ϕ are listed below.

$1R: \quad \phi(y, z, w) = (y \wedge z) \vee (\neg y \wedge w),$
$2R: \quad \phi(y, z, w) = (y \wedge w) \vee (z \wedge \neg w),$
$3R: \quad \phi(y, z, w) = y \oplus z \oplus w,$
$4R: \quad \phi(y, z, w) = z \oplus (y \vee \neg w).$

Chaining variables a_i, b_i, c_i, d_i are calculated in the following way.

$a_i = \psi(a_{i-1}, b_{i-1}, c_{i-1}, d_{i-1}, m, s, t),$
$d_i = \psi(d_{i-1}, a_i, b_{i-1}, c_{i-1}, m, s, t),$
$c_i = \psi(c_{i-1}, d_i, a_i, b_{i-1}, m, s, t),$
$b_i = \psi(b_{i-1}, c_i, d_i, a_i, m, s, t).$

3 Description of the Attack by Wang et al. [6]

3.1 Notation

We define notations which are used in this paper. $M = (m_0, ..., m_{15})$ and $M' = (m'_0, ..., m'_{15})$ are messeages where each m_i is 32-bit integer. $\Delta M = (\Delta m_0, ..., \Delta m_{15})$ is the differential of M and M', where $\Delta m_i = m'_i - m_i$. $a_i, b_i, c_i, d_i (1 \leq i \leq 16)$ represent chaining variables, and $a_{i,j}, b_{i,j}, c_{i,j}, d_{i,j} (1 \leq j \leq 32)$ represent the j-th bit of respective chaining variables. ϕ_i is the output of the nonlinear function ϕ in the i-th step, and $\phi_{i,j}$ denotes the j-th bit of ϕ_i. $x_i[j]$ and $x_i[-j]$ represent differentials of the j-th bit in chaining variable x_i. $x_i[j]$ represents that differential of the j-th bit in x_i is 1, that is, $x'_{i,j} - x_{i,j} = 1$. Whereas, $x_i[-j]$ represents that differential of the j-th bit in x_i is -1, that is, $x'_{i,j} - x_{i,j} = -1$.

3.2 Collision Differentials

The attack of Wang et al. is a differential attack. It generated collision messages by canceling the input differential, and by making output differential 0. It generated collision messages which are composed of two 1024-bit messages (M_0, M_1) and (M'_0, M'_1). In this attack, a collision differential is given in advance. The differential is as follows,

$$\Delta H_0 = 0 \overset{(M_0,M'_0)}{\longrightarrow} \Delta H_1 \overset{(M_1,M'_1)}{\longrightarrow} \Delta H = 0,$$

where,

$$\begin{aligned}
\Delta M_0 &= M'_0 - M_0 \\
&= (0, 0, 0, 0, 2^{31}, 0, 0, 0, 0, 0, 0, 2^{15}, 0, 0, 2^{31}, 0), \\
\Delta M_1 &= M'_1 - M_1 \\
&= (0, 0, 0, 0, 2^{31}, 0, 0, 0, 0, 0, 0, -2^{15}, 0, 0, 2^{31}, 0), \\
\Delta H_1 &= (2^{31}, 2^{31} + 2^{25}, 2^{31} + 2^{25}, 2^{31} + 2^{25}).
\end{aligned}$$

3.3 Sufficient Condition

Sufficient condition is a set of conditions of chaining variables for generating collisions. If all conditions for compressing M are satisfied, M and $M + \Delta M$ becomes a collision pair with probability 1. According to [6], there are 290 conditions for M_0, therefore, the probability that a randomly chosen message satisfies all conditions is very small. However, it is possible to drastically raise the probability by modifying a message (Described in 3.5). Conditions in the first round and in the early steps of the second round are able to be corrected by modifying a message. However, 37 conditions are not able to be corrected. Therefore, if a random message does not satisfy all of these conditions, it is necessary to choose another random message and modify it again.

3.4 Collision Search Algorithm

An algorithm for searching collisions is as follows.

1. Generate a 512-bit random message M_0 for the first block.
2. Modify M_0 in order to satisfy the sufficient condition for the first block as much as possible.

3. If all conditions are satisfied, calculate the output of the first block with the modified message and the MD5 initial value. If all conditions are not satisfied, randomly rechoose m_{14} and m_{15}, and go back to (2).
4. Generate a 512-bit random message M_1 for the second block.
5. By the similar way to the first block, calculate the output of the second block. However, in this time, use the result of the first block instead of using the MD5 initial value.
6. Calculate $M_0' = M_0 + \Delta M_0$, $M_1' = M_1 + \Delta M_1$
7. (M_0, M_1) and (M_0', M_1') are the collision messages.

3.5 Message Modification

The purpose of message modification is deterministically (or with very high probability) satisfying the sufficient condition. There are two kinds of message modification:"single-message modification" and "multi-message modification". Single-message modification is a message modification for the first round, and multi-message modification is for the second round. Single-message modification modifies only one message, whereas, multi-message modification affects some messages. Therefore, to do multi-message modification, we need additional work in order to cancel the influence to other messages.

3.5.1 Single-Message Modification

Single-message modification is a message modification for satisfying the sufficient condition in the first round.

For example, we consider the situation where we modify a message in order to satisfy conditions in the third step of the first round. According to [6], there are three conditions on c_1: "$c_{1,8} = 0$","$c_{1,12} = 0$" and "$c_{1,20} = 0$". On the other hand, the expression for calculating c_1 is as follows,

$$c_1 = (d_1 + (c_0 + \phi(d_1, a_1, b_0) + m_2 + t) \lll 17) \bmod 2^{31}.$$

In order to guarantee that these conditions on c_1 are satisfied, we modify m_2 by the following way.

1. Generate a 32-bit random message m_2^{old}, and calculate the value of c_1^{old}.
2. Change the value of c_1^{old} into desirable c_1^{new} by the expression below.
$$c_1^{new} \leftarrow c_1^{old} - c_{1,7}^{old} \cdot 2^6 - c_{1,12}^{old} \cdot 2^{11} - c_{1,20}^{old} \cdot 2^{19}$$
3. Rewrite the m_2 in order to guarantee that c_1^{new} is always obtained.
$$m_2^{new} \leftarrow ((c_1^{new} - d_1) \ggg 17) - c_0 - \phi(d_1, a_1, b_0) - t$$

The similar procedure is applied to all steps in the first round, and all conditions in the first round are satisfied with high probability.

3.5.2 Multi-Message Modification

Multi-message modification is a message modification for satisfying the sufficient condition in the second round. Some of the conditions in early steps of the second round can be corrected by this modification.

For example, we consider the situation where the condition on $a_{5,32}$ needs to be corrected. According to [6], the condition on $a_{5,32}$ is $a_{5,32} = 0$. Therefore, if the value of $a_{5,32}$ is happen to be 0, $a_{5,32}$ does not have to be corrected. However, if $a_{5,32} = 1$, $a_{5,32}$ has to be corrected by the following way:

First, a_5 is calculated as follows,

$$a_5 \leftarrow b_4 + (a_4 + \phi(b_4, c_4, d_4) + m_1 + t) \lll 5$$

1. Since m_1 is used to calculate a_5, and the bit rotation number in this step is 5, we modify the 27-th bit of m_1 in order to change $a_{5,32}$.
2. Since m_1 is used not only in the second round but in the first round, the change on m_1 affects other steps in the first round. Therefore, we need additional change in the first round.

Details of the correcting $a_{5,32}$ are listed in Table 1. Wang et al. claimed that except for 37 conditions can be corrected by single-message modifications and multi-message modifications.

Table 1: The message modification for correcting "$a_{5,32}$"

step	m	shift	Modify m_i	Chaining variables
2	m_1	12	$m_1 \leftarrow m_1 + 2^{26}$	d_1^{new}, a_1, b_0, c_0
3	m_2	17	$m_2 \leftarrow ((c_1 - d_1^{new}) \ggg 17) - c_0 - \phi(d_1^{new}, a_1, b_0) - t$	c_1, d_1^{new}, a_1, b_0
4	m_3	22	$m_3 \leftarrow ((b_1 - c_1) \ggg 22) - b_0 - \phi(c_1, d_1^{new}, a_1) - t$	b_1, c_1, d_1^{new}, a_1
5	m_4	7	$m_4 \leftarrow ((a_2 - b_1) \ggg 7) - a_1 - \phi(b_1, c_1, d_1^{new}) - t$	a_2, b_1, c_1, d_1^{new}
6	m_5	12	$m_5 \leftarrow ((d_2 - a_2) \ggg 12) - d_1^{new} - \phi(a_2, b_1, c_1) - t$	d_2, a_2, b_1, c_1

4 New Multi-Message Modification

Wang et al. claimed that except for 37 conditions can be corrected. This means that they succeeded in correcting 6 conditions in the second round. In our research, we found that 14 conditions in the second round could be corrected. In this section, we explain which conditions can be corrected and how to correct those conditions.

4.1 Extra Condition

To correct many conditions in the second round, it is necessary to make "extra condition". Therefore, we first explain the extra condition. The terminology of the "extra condition" was introduced by Wang et al. [5]. The purpose of setting extra condition is to guarantee that conditions in the second round can be corrected. Extra conditions are not conditions for generating collisions, therefore, messages do not have to satisfy extra conditions after they are modified in order to correct unsatisfied sufficient conditions.

Extra conditions in the first round are set together with the sufficient condition in the first round by the single-message modification. Extra conditions in the second round are set by the multi-message modification. A collision search algorithm for the first block including the extra condition is as follows,

1. Generate a random message.
2. Modify the message to satisfy the sufficient condition and the extra condition in the first round by single-message modification.
3. If the sufficient condition and the extra condition in the second round are not satisfied, correct them by multi-message modification.
4. If modified message does not satisfy all of the sufficient condition, choose m_{14} and m_{15} again, and go back to (2), otherwise, output the calculated value.

A collision search algorithm for the second block is almost same with the first block.

4.2 Details of the Multi-Message Modification

In our research, we have found that 14 conditions in the second round are able to be corrected. Correctable conditions are $a_{5,4}$, $a_{5,16}$, $a_{5,18}$, $a_{5,32}$, $d_{5,18}$, $d_{5,30}$, $d_{5,32}$, $c_{5,18}$, $c_{5,32}$, $b_{5,32}$, $a_{6,18}$, $a_{6,32}$, $d_{6,32}$ and $c_{6,32}$.

4.2.1 Corrections for $a_{5,i}(i = 4, 16, 18, 32)$

a_5 is calculated by the following expression.

$$a_5 = b_4 + (a_4 + \phi(b_4, c_4, d_4) + m_1 + t) \lll 5$$

The i-th bit of a_5 can be corrected by the message modification shown in Table 2.

Table 2: The message modification for correcting "$a_{5,i}$"

	shift	Modify m_i
2	12	$m_1 \leftarrow m_1 \pm 2^{i-6}$
3	17	$m_2 \leftarrow ((c_1 - d_1^{new}) \ggg 17) - c_0 - \phi(d_1^{new}, a_1, b_0) - t$
4	22	$m_3 \leftarrow ((b_1 - c_1) \ggg 22) - b_0 - \phi(c_1, d_1^{new}, a_1) - t$
5	7	$m_4 \leftarrow ((a_2 - b_1) \ggg 7) - a_1 - \phi(b_1, c_1, d_1^{new}) - t$
6	12	$m_5 \leftarrow ((d_2 - a_2) \ggg 12) - d_1^{new} - \phi(a_2, b_1, c_1) - t$

In Table 2, '±' is chosen depending on the value of a chaining variable in the first round. Since m_1 is used to calculate d_1 in step 2, the value of d_1 is changed. Considered the bit rotation number in step 2, the value of $d_{1,i+7}$ is changed. If carry occurs in $d_{1,i+7}$, upper bits of $d_{1,i+7}$ are changed, and this may result in breaking conditions in the upper bits of $d_{1,i+7}$. Therefore, if $d_{1,i+7} = 0$, we choose '+', and if $d_{1,i+7} = 1$, we choose '−'.

Basically, corrections are done from lower bit in order to avoid undesirable carry. However, there are some exceptions. From the expression for a_5, it can be said that $m_{1,31}$ is relevant to $a_{5,4}$, whereas, according to Table 2, $m_{1,27}$ is changed when $a_{5,32}$ is corrected. Therefore, if $a_{5,32}$ is corrected and the carry is transmitted from $m_{1,27}$ to $m_{1,31}$, the condition of $a_{5,4}$ is broken. To avoid this, we correct conditions in the following order: $i = 16, 18, 32, 4$. However, when $a_{5,4}$ is corrected and the carry is transmitted from $a_{5,4}$ to $a_{5,16}$, the condition on $a_{5,16}$ is broken. To avoid this, we set extra conditions $b_{4,4} = 1$ and $d_{1,11} = 0$. By setting $b_{4,4} = 1$, the value of $a_{5,4}$ becomes always 0 when $a_{5,4}$ is corrected because of a condition $a_{5,4} = b_{4,4}$. $d_{1,11} = 0$ guarantees that the carry never occurs in step 2 when $a_{5,4}$ is corrected.

4.2.2 Corrections for $d_{5,i}(i = 18, 30)$

d_5 is calculated by the following expression.

$$d_5 = a_5 + (d_4 + \phi(a_5, b_4, c_4) + m_6 + t) \lll 9$$

i-th bit of d_5 can be corrected by the message modification shown in Table 3.

Two extra conditions are set in Table 3. The purpose of the extra condition $d_{4,i-9} = 1$ is ignoring the change of c_4 in ϕ_{17}. The purpose of the extra condition $a_{5,i-9} \neq b_{4,i-9}$ is reflecting the change of c_4 in ϕ_{18} in order to make change in $d_{5,i}$.

4.2.3 A Correction for $d_{5,32}$

From the expression of calculating d_5, if $m_{6,23}$ is changed, $d_{5,32}$ is corrected. Table 4 shows how to cancel the effect of the change of $m_{6,23}$ in the first round.

Table 3: The message modification for correcting "$d_{5,i}$"

step	shift	Modify m_i	Chaining Variables	Extra Conditions
15	17	$m_{14} \leftarrow m_{14} \pm 2^{i-27}$	$c_4[\pm i-9], d_4, a_4, b_4$	
16	22	$m_{15} \leftarrow ((b_4 - c_4^{new}) \ggg 22) - b_4 - \phi(c_4^{new}, d_4, a_4) - t$	$b_4, c_4[\pm i-9], d_4, a_4$	
17	5		$a_5, b_4, c_4[\pm i-9], d_4$	$d_{4,i-9} = 1$
18	9		$d_5[\pm i], a_5, b_4, c_4[\pm i-9]$	$a_{5,i-9} \neq b_{4,i-9}$

Table 4: The message modification for correcting "$d_{5,32}$"

step	shift	Modify m_i	Chaining Variables	Extra Conditions
3	17	$m_2 \leftarrow m_2 + 2^6$	$c_1[23], d_1, a_1, b_0$	$c_{1,23} = 0$
4	22	$b_1 \leftarrow b_1 + 2^{22}$ $m_3 \leftarrow ((b_1^{new} - c_1^{new}) \ggg 22) - b_0 - \phi(c_1^{new}, d_1, a_1) - t$	$b_1[23], c_1[23], d_1, a_1$	
5	7	$m_4 \leftarrow ((a_2 - b_1^{new}) \ggg 7) - a_1 - \phi(b_1^{new}, c_1^{new}, d_1) - t$	$a_2, b_1[23], c_1[23], d_1$	
6	12	$m_5 \leftarrow ((d_2 - a_1) \ggg 12) - d_1 - \phi(a_2, b_1^{new}, c_1^{new}) - t$	$d_2, a_2, b_1[23], c_1[23]$	
7	17	$m_6 \leftarrow m_6 - 2^{22}$	$c_2, d_2, a_2, b_1[23]$	$d_{2,23} = 1$
8	22	$m_7 \leftarrow ((b_2 - c_2) \ggg 22) - b_1 - \phi(c_2, d_2, a_2) - t$	b_2, c_2, d_2, a_2	

In Table 4, the purpose of setting $c_{1,23} = 0$ is limiting the direction of change. Since a sufficient condition $b_{1,23} = c_{1,23}$ exists, we have to correct the value of $b_{1,23}$ after we modify m_2.

4.2.4 Corrections for Other Conditions

Corrections for other conditions are shown in Table 5 to Table 11.

Table 5: The message modification for correcting "$c_{5,18}$"

step	shift	Modify m_i	Chaining Variables	Extra Conditions
10	12	$m_9 \leftarrow m_9 + 2^{23}$	$d_3[4], a_3, b_2, c_2$	$d_{3,4} = 0$
11	17	$m_{10} \leftarrow ((c_3 - d_3^{new}) \ggg 17) - c_2 - \phi(d_3^{new}, a_3, b_2) - t$	$c_3, d_3[4], a_3, b_2$	
12	22	$m_{11} \leftarrow m_{11} - 2^3$	$b_3, c_3, d_3[4], a_3$	$c_{3,4} = 1$
13	7	$m_{12} \leftarrow ((a_4 - b_3) \ggg 7) - a_3 - \phi(b_3, c_3, d_3^{new}) - t$	$a_4, b_3, c_3, d_3[4]$	
14	12	$m_{13} \leftarrow ((d_4 - a_4) \ggg 12) - d_3^{new} - \phi(a_4, b_3, c_3) - t$	d_4, a_4, b_3, c_3	

Table 6: The message modification for correcting "$c_{5,32}$"

step	shift	Modify m_i	Chaining Variables	Extra Conditions
13	7	$m_{12} \leftarrow m_{12} + 2^5$	$a_4[13], b_3, c_3, d_3$	$a_{4,13} = 0$
14	12	$m_{13} \leftarrow ((d_4 - a_4^{new}) \ggg 12) - d_3 - \phi(a_4^{new}, b_3, c_3) - t$	$d_4, a_4[13], b_3, c_3$	
15	17	$m_{14} \leftarrow ((c_4 - d_4) \ggg 17) - c_3 - \phi(d_4, a_4^{new}, b_3) - t$	$c_4, d_4, a_4[13], b_3$	
16	22	$b_4 \leftarrow b_4 - 2^{17} + 2^{22}$ $m_{15} \leftarrow ((b_4^{new} - c_4) \ggg 22) - b_3 - \phi(c_4, d_4, a_4^{new}) - t$	$b_4[-18, 23], c_4, d_4, a_4[13]$	$b_{4,18} = 1, b_{4,23} = 0$
17	5		$a_5, b_4[-18, 23], c_4, d_4$	$d_{4,23} = 0, (d_{4,18} = 1)$
18	9		$d_5, a_5, b_4[-18, 23], c_4$	$c_{4,18} = 1, c_{4,23} = 1$
19	14		$c_5[-32, 5], d_5, a_5, b_4[-18, 23]$	

Table 7: The message modification for correcting "$b_{5,32}$"

step	shift	Modify m_i	Chaining Variables	Extra Conditions
13	7	$m_{12} \leftarrow m_{12} + 2^{31}$	$a_4[7], b_3, c_3, d_3$	$a_{4,7} = 0$
14	12	$m_{13} \leftarrow ((d_4 - a_4^{new}) \ggg 12) - d_3 - \phi(a_4^{new}, b_3, c_3) - t$	$d_4, a_4[7], b_3, c_3$	
15	17	$m_{14} \leftarrow ((c_4 - d_4) \ggg 17) - c_3 - \phi(d_4, a_4^{new}, b_3) - t$	$c_4, d_4, a_4[7], b_3$	
16	22	$b_4 \leftarrow b_4 - 2^{11}$ $m_{15} \leftarrow ((b_4^{new} - c_4) \ggg 22) - b_3 - \phi(c_4, d_4, a_4^{new}) - t$	$b_4[-12], c_4, d_4, a_4[7]$	$b_{4,12} = 1$
17	5		$a_5, b_4[-12], c_4, d_4$	$d_{4,12} = 0$
18	9		$d_5, a_5, b_4[-12], c_4$	$c_{4,12} = 1$
19	14		$c_5, d_5, a_5, b_4[-12]$	$d_{5,12} = a_{5,12}$
20	20		$b_5[-32], c_5, d_5, a_5$	

Table 8: The message modification for correcting "$a_{6,18}$"

step	shift	Modify m_i	Chaining Variables		Extra Conditions
13	7	$m_{12} \leftarrow m_{12} - 2^{25}$	$a_4[-1], b_3, c_3, d_3$		$a_{4,1} = 1$
14	12	$m_{13} \leftarrow ((d_4 - a_4^{new}) \ggg 12) - d_3 - \phi(a_4^{new}, b_3, c_3) - t$	$d_4, a_4[-1], b_3, c_3$		
15	17	$m_{14} \leftarrow ((c_4 - d_4) \ggg 17) - c_3 - \phi(d_4, a_4^{new}, b_3) - t$	$c_4, d_4, a_4[-1], b_3$		
16	22	$m_{15} \leftarrow ((b_4 - c_4) \ggg 22) - b_3 - \phi(c_4, d_4, a_4^{new}) - t$	$b_4, c_4, d_4, a_4[-1]$		
17,2	5,12	$m_1 \leftarrow m_1 + 1$	a_5, b_4, c_4, d_4	$d_1[13], a_1, b_0, c_0$	$d_{1,13} = 0$
3	17	$m_2 \leftarrow ((c_1 - d_1^{new}) \ggg 17) - c_0 - \phi(d_1^{new}, a_1, b_0) - t$		$c_1, d_1[13], a_1, b_0$	
4	22	$m_3 \leftarrow ((b_1 - c_1) \ggg 22) - b_0 - \phi(c_1, d_1^{new}, a_1) - t$		$b_1, c_1, d_1[13], a_1$	
5	7	$m_4 \leftarrow ((a_2 - b_1) \ggg 7) - a_1 - \phi(b_1, c_1, d_1^{new}) - t$		$a_2, b_1, c_1, d_1[13]$	
6	12	$m_5 \leftarrow m_5 - 2^{12}$		d_2, a_2, b_1, c_1	

Table 9: The message modification for correcting "$a_{6,32}$"

step	shift	Modify m_i	Chaining Variables	Extra Conditions
4	22	$m_3 \leftarrow m_3 + 2^4$	$b_1[27], c_1, d_1, a_1$	$b_{1,27} = 0$
5	7	$m_4 \leftarrow ((a_2 - b_1^{new}) \ggg 7) - a_1 - \phi(b_1^{new}, c_1, d_1) - t$	$a_2, b_1[27], c_1, d_1$	
6	12	$m_5 \leftarrow m_5 - 2^{26}$	$d_2, a_2, b_1[27], c_1$	$a_{2,27} = 1$
7	17		$c_2, d_2, a_2, b_1[27]$	$(d_{2,27} = a_{2,27})$
8	22	$m_7 \leftarrow m_7 - 2^{26}$	b_2, c_2, d_2, a_2	

Table 10: The message modification for correcting "$d_{6,32}$"

step	shift	Modify m_i	Chaining Variables	Extra Conditions
9	7	$m_8 \leftarrow m_8 + 2^{15}$	$a_3[23], b_2, c_2, d_2$	$a_{3,23} = 0$
10	12	$m_9 \leftarrow ((d_3 - a_3^{new}) \ggg 12) - d_2 - \phi(a_3^{new}, b_2, c_2) - t$	$d_3, a_3[23], b_2, c_2$	
11	17	$m_{10} \leftarrow m_{10} - 2^{22}$	$c_3, d_3, a_3[23], b_2$	$d_{3,23} = 1$
12	22		$b_3, c_3, d_3, a_3[23]$	$c_{3,23} = 1$
13	7	$m_{12} \leftarrow ((a_4 - b_3) \ggg 9) - a_3^{new} - \phi(b_3, c_3, d_3) - t$	a_4, b_3, c_3, d_3	

Table 11: The message modification for correcting "$c_{6,32}$"

step	shift	Modify m_i	Chaining Variables	Extra Conditions
11	17	$m_{10} \leftarrow m_{10} + 2^{12}$	$c_3[30], d_3, a_3, b_2$	$c_{3,30} = 0$
12	22	$m_{11} \leftarrow m_{11} - 2^7$	$b_3, c_3[30], d_3, a_3$	$d_{3,30} = a_{3,30}$
13	7	$m_{12} \leftarrow ((a_4 - b_3) \ggg 7) - a_3 - \phi(b_3, c_3^{new}, d_3) - t$	$a_4, b_3, c_3[30], d_3$	
14	12	$m_{13} \leftarrow ((d_4 - a_4) \ggg 12) - d_3 - \phi(a_4, b_3, c_3^{new}) - t$	$d_4, a_4, b_3, c_3[30]$	
15	17	$m_{14} \leftarrow m_{14} + 2^{22} - 2^{29}$	$c_4[8], d_4, a_4, b_3$	$c_{4,8} = 0$
16	22	$m_{15} \leftarrow m_{15} - 2^7 - 2^{17}$	$b_4, c_4[8], d_4, a_4$	$(a_{4,8} = 0), (d_{4,8} = 1)$
17	5		$a_5, b_4, c_4[8], d_4$	$(d_{4,8} = 1)$
18	9		$d_5, a_5, b_4, c_4[8]$	$a_{5,8} = b_{4,8}$
19	14	$(m_{11} \leftarrow m_{11} - 2^7$: modified in step 12)	c_5, d_5, a_5, b_4	

4.3 Estimation of the efficiency of corrections

Corrections mentioned in this section are probabilistic. We experimentally confirmed that all of these 14 conditions are satisfied after the message modifications with probability about 1/2.

Remark

We have found many other modification techniques for the second block. By applying these modications, 14 conditions in the second round for the second block can be corrected. Therefore, the complexity of the attack for the second block becomes 2^{23}.

5 Modification Techniques for Shorten Repetition

In the method of Wang et al, if all of the sufficient condition are not satisfied after messages are modified, the algorithm goes back to step 15, and restart from randomly choosing m_{14} and m_{15}. Therefore, whenever a message does not satisfy all of the sufficient condition, single-message modification for m_{14} and m_{15}, multi-message modification and MD5 calculation for step 15 to step 64 are calculated. This complexity is big. In our research, we have found that it would be possible to go back to step 25 rather than step 15. This saves us the complexity of single-message modification, multi-message modification and calculation for step 15 to 24. By our modification, we estimated that the complexity becomes roughly 5/8 compared to the algorithm of Wang et al. In this section, we explain how to restart the algorithm from step 25, and the estimation of the effect.

The calculation of step 25 is as follows,

$$a_7 = b_6 + (a_6 + \phi(b_6, c_6, d_6) + m_9 + t) \lll 5$$

Therefore, we change several bits of m_9 and this results in randomly changing the value of latter chaining variables. When an i-th bit of m_9 is changed, if we modify messages as shown in Table 12, the effect of the change is cancelled.

Table 12: The message modification for shorten repetition

step	shift	Modify m_i	Chaining Variables	Extra Conditions
9	7	$m_8 \leftarrow m_8 + 2^{i+4}$	$a_3[i+12], b_2, c_2, d_2$	$a_{3,i+12} = 0$
10	12	$m_9 \leftarrow m_9 - 2^{i-1}$	$d_3, a_3[i+12], b_2, c_2$	$b_{2,i+12} = c_{2,i+12}$
11	17		$c_3, d_3, a_3[i+12], b_2$	$d_{3,i+12} = 0$
12	22		$b_3, c_3, d_3, a_3[i+12]$	$c_{3,i+12} = 1$
13	7	$m_{12} \leftarrow ((a_4 - b_3) \ggg 7) - a_3^{new} - \phi(b_3, c_3, d_3) - t$	a_4, b_3, c_3, d_3	

The number of i, where we can set corresponding extra conditions, is 2. Therefore, we can restart the collision search algorithm from the 25-th step $2^2 - 1 (= 3)$ times for each message satisfying all conditions untill step 25. We have estimated that the total complexity from step 15 to step 64 including both message modification would be 81 steps, and the complexity from step 25 to step 64 including our modification technique would be 41 steps. Therefore, the complexity of the attack by Wang et al. is $81 \times 4 = 324$ steps, whereas, the complexity of our attack is $81 + 41 \times 3 = 204$ steps. This indicates that the complexity becomes roughly 5/8 by applying our modification technique.

Remark

This modifiation technique is more effective for the second block than the first block. The complexity for the second block will become less than half.

6 Conclusion

In this paper, we proposed the method to correct 14 conditions in the second round. These corrections are probabilistic. We experimentally confirmed that all of these conditions were satisfied after the message modifications with probability about 1/2. On the other hand, 29 conditions are remained uncorrectable. Therefore, the complexity of our attack is 2^{30}.

We also proposed the more efficient collision search algorithm than that of Wang et al. We showed that the total complexity of the collision search algorithm would be reduced to roughly 5/8. This result is the best of all existing attacks to MD5.

Finally, we show a collision message generated by our proposed method in Table 13. The message (M_0, M_1) and (M_0', M_1') have the same hash value.

Table 13: Generated collision messages in our proposed attack

M_0	0xcfdcc99611b8fc4b9dcf1099bab3b0f2c1d51f7112e9d1dc2110fa3e9f01eea5 06332cb1e0f307f88f6cf5ef9fb00f66a65fe4dbf50ed81f553b6443bc59b6e2
M_1	0xd958e84d2f5d1b9b53a46fce7ba577da94ac52f6ddc5506ae72e090ca18cf8cc f37eeff5085695806a77fb8a3e65b89590032d1d9513e57a7d283757ea659e11
M_0'	0xcfdcc99611b8fc4b9dcf1099bab3b0f241d51f7112e9d1dc2110fa3e9f01eea5 06332cb1e0f307f88f6cf5ef9fb08f66a65fe4dbf50ed81fd53b6443bc59b6e2
M_1'	0xd958e84d2f5d1b9b53a46fce7ba577da14ac52f6ddc5506ae72e090ca18cf8cc f37eeff5085695806a77fb8a3e65389590032d1d9513e57afd283757ea659e11
Hash value	0x1afe687725129c5fa5d52829e9bd5080

Acknowledgement

We would like to thank Dr. T. Shimoyama and Mr. J. Yajima at Fujitsu Laboratories, and Dr. Y. Lisa Yin at Independent Security Consultant for helpful advice.

References

[1] H. Dobbertin: The status of MD5 after a recent attack, CryptoBytes 2 (2), 1996

[2] V. Klima: Finding MD5 Collisions on a Notebook PC Using Multi-message Modifications, e-Print 102, 2005.

[3] R. Rivest: The MD4 Message Digest Algorithm, CRYPTO'90 Proceedings, 1992, http://theory.lcs.mit.edu/˜
rivest/Rivest-MD4.txt

[4] R. Rivest: The MD5 Message Digest Algorithm, CRYPTO'90 Proceedings, 1992, http://theory.lcs.mit.edu/˜
rivest/Rivest-MD5.txt

[5] X. Wang, X. Lai, D. Feng, H. Chen, X. Yu: Cryptanalysis of the Hash Functions MD4 and RIPEMD, Advances in EUROCRYPT2005, LNCS 3494, pp. 1–18, 2005.

[6] X. Wang, H. Yu: How to break MD5 and Other Hash Functions, Advances in EUROCRYPT2005, LNCS 3494, pp. 19–35, 2005.

[7] X. Wang, Y. Lisa Yin, H. Yu: Finding Collisions in the Full SHA-1, Crypto 2005, LNCS 3621, pp. 17-36, 2005