Improved Collision Attack on MD4

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

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

Abstract

In this paper, we propose an attack method to find collisions of MD4 hash function. This attack is the improved version of the attack which was invented by Xiaoyun Wang et al [1]. We were able to find collisions with probability almost 1, and the average complexity to find a collision is upper bounded by three times of MD4 hash operations. This result is improved compared to the original result of [1] where the probability were from 2^{-6} to 2^{-2}, and the average complexity to find a collision was upper bounded by 2^8 MD4 hash operations. We also point out the lack of sufficient conditions and imprecise modifications for the original attack in [1].

keywords: Collision Attack, MD4, Hash Function, Message Modification

1 Introduction

In August 2004, at the rump session of CRYPTO 2004, Wang et al. announced that their group discovered collisions of various hash functions such as MD4, MD5, HAVAL-128, RIPEMD [2]. In March 2005, details of their attack methods were published [1,3], and these papers brought big impact to cryptographic society.

The probability to find a collision on MD4 was 2^{-6} to 2^{-2} [1]. Although this probability was very high, we aimed at constructing a more efficient attack which finds a collision with probability almost 1. In the middle of the research, we found that there was an ambiguity on the estimation of the success probability in [1] so that we needed to remove the ambiguity. In this paper, we show how we removed the ambiguity and the result of an improved method.

Our improving process consists of roughly two stages. The first stage is removing the ambiguity of the probability mentioned in [1], and consequently, we were able to reconstruct the attack method to find a collision with probability exactly 2^{-2}. The second stage is improving the probability further in order to archive success probability almost 1. At this stage, we introduced new message modification techniques. Using these techniques, the complexity to find a collision can also be reduced. In our method, the complexity is upper bounded by three times of MD4 hash operations.

2 Improving Process

2.1 Discovering lacked sufficient conditions

We realized that all sufficient conditions were not listed on the paper [1]. Therefore, the first task of improvement was discovering the lacked sufficient conditions, and adding them into the

table of original sufficient conditions. Two sufficient conditions, $a_{6,30} = 0, b_{4,32} = c_{4,32}$, were newly discovered in this process. We show how they were discovered in Table 1.

Table 1: Discovery of "$a_{6,30} = 0$"

Step	Shift	Δm_i	The i-th output for M'
21	3	2^{31}	$a_6[-29, 30, -32]$

Table 2: Discovery of "$b_{4,32} = c_{4,32}$"

Step	Shift	Δm_i	The i-th output for M'
14	7		$d_4[-27, -29, 30]$
15	11		c_4
16	19		$b_4[19]$
17	3		$a_5[-26, 27, -29, -32]$
18	5		d_5

Table 1 and 2 are part of tables written in [1].

Table 1 indicates the situation where the value of $a_{6,30}$ has to be 0. In the right edge column of table 1, there is an output difference "$a_6[30]$". "$a_6[30]$" means that the value of $a_{6,30}$ has to be changed from 0 to 1 by modification. Therefore, we need to guarantee that the value of $a_{6,30}$ before the modification has been set to 0, but this condition is not listed in [1]. Therefore, we add the new sufficient condition "$a_{6,30} = 0$".

Table 2 shows the situation where the value of $b_{4,32}$ has to be equal to $c_{4,32}$. The value of d_5 is calculated by following expression,

$$d_5 \longleftarrow ((d_4 + G(a_5, b_4, c_4) + m_4 + 0x5a827999) \mod 32) \lll 5.$$

Here, the function G is as follows;

$$G(X, Y, Z) = (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z).$$

Now, we focus on the fact that the value of $a_{5,32}$ was changed from 1 to 0. The change in $a_{5,32}$ may cause the change in 32^{nd} bit of $G(a_5, b_4, c_4)$, and after bits are shifted by 5 bits, it may result in the change in $d_{5,5}$. However, table 2 also shows there must not be any difference in $d_{5,5}$. Therefore, the influence of the change in $a_{5,32}$ has to be canceled when $G(a_5, b_4, c_4)$ is calculated. From the feature of G function, if the value of $b_{4,32}$ and $c_{4,32}$ are the same, the value of G keeps unchanged in spite of the change in $a_{5,32}$. However, [1] doesn't guarantee this condition, and thus, we add the new sufficient condition "$b_{4,32} = c_{4,32}$"

2.2 Apply More Precise Modifications than [1]

Wang et al. introduced various message modification methods [1]. However, we found some of their modifications cannot modify message appropriately in some cases. Therefore, we propose more precise modification in order to surely modify message appropriately.

Modification of $d_{5,19}$ and $c_{5,32}$

There is a sufficient condition on $d_{5,19}$ and $c_{5,26}$. When both of them are not satisfied, it is necessary to modify both of them. In the modification method [1], the extra condition "$a_{2,17} = b_{2,17}$" has to be set in order to enable the value of $c_{5,26}$ to be modified. Whereas, in the case $d_{5,19}$ is modified, $a_{2,17}$ which is the corresponding value in round 1, is also modified. This results in breaking the existing extra condition "$a_{2,17} = b_{2,17}$". This situation happens with probability $\frac{1}{4}$. To avoid this conflict of modification, we change the way to modify $d_{5,19}$. When $d_{5,19}$ is modified, we use internal collision to cancel the modification to $a_{2,17}$.

Estimation of the Success Probability of [1] about $c_{5,26}$

As we explained above, the extra condition is broken with probability $\frac{1}{4}$. However, even if the extra condition is broken, there still remained a chance that modification of $c_{5,26}$ is done correctly. We explain the mechanism and evaluate the precise success probability of this modification. Table 3 shows the modification of $c_{5,26}$. We assume that $d_{5,19}$ has already been modified,

Table 3: Modification of "$c_{5,26}$"

step	shift	Modify m_i	Chaining value after message modification	Extra Conditions in 1^{st} round
6	7	$m_5 \rightarrow m_5 + 2^9$	$d_2[17], a_2, b_1, c_1$	$d_{2,17} = 0$
7	11		$c_2, d_2[17], a_2, b_1$	$a_{2,17} = b_{1,17}$
8	19		$b_2, c_2, d_2[17], a_2$	$c_{2,17} = 0$
9	3	$m_8 \rightarrow m_8 - 2^{16}$	$a_3, b_2, c_2, d_2[17]$	$b_{2,17} = 0$
10	7	$m_9 \rightarrow m_9 - 2^{16}$	d_3, a_3, b_2, c_2	

and consequently, the extra condition "$a_{2,17} = b_{2,17}$" has been broken. If we do nothing except for the modification shown in Table 3, the value of c_2 will be changed because of the broken extra condition. However, there is another way to prevent the influence of the change of $d_2[17]$ without using the extra condition. This method is modifying m_6 in order to guarantee that desirable c_2 will be produced with c_1, changed d_2, a_2, b_1 and changed m_6. In details, the following expression guarantees the desirable c_2.

$$m_6 \longleftarrow (c_2 \ggg 11) - c_1 - f(d_2', a_2, b_1) \tag{1}$$

The expression (1) can be seen only in the modification of $d_{6,29}$. Table 4 shows the modification

Table 4: Modification of "$d_{6,29}$"

$m_5 \longleftarrow m_5 \pm 2^{23}$
$m_6 \longleftarrow (c_2 \ggg 11) - c_1 - f(d_2', a_2, b_1)$
$m_7 \longleftarrow (b_2 \ggg 19) - b_1 - f(c_2, d_2', a_2)$
Add an extra condition "$b_{2,31} = 1$"
$m_9 \longleftarrow (d_3 \ggg 7) - d_2' - f(a_3, b_2, c_2)$

of $d_{6,29}$. The expression (1) is included in Table 4. Therefore, even though the extra condition "$a_{2,17} = b_{2,17}$" is broken, if $d_{6,29}$ is modified, $c_{5,26}$ is able to be modified appropriately.

As a result , the probability that $c_{5,26}$ satisfies the sufficient condition after multi-step modification is:

1. $c_{5,26}$ satisfies the sufficient condition without being modified: Prob.$\frac{1}{2}$
2. $c_{5,26}$ is modified and $d_{5,19}$ is not modified, so that the extra condition is hold: Prob.$\frac{1}{4}$
3. Both of $c_{5,26}$ and $d_{5,19}$ are modified and the extra condition is broken, but c_2 is modified when $d_{6,29}$ is modified: Prob.$\frac{1}{8}$

Therefore, the total success probability for $c_{5,26}$ is $\frac{1}{2} + \frac{1}{4} + \frac{1}{8} = \frac{7}{8}$.

Similar situation occurs when both of $c_{5,32}$ and $c_{6,32}$ are modified. According to [1], we need to set the extra condition $c_{2,23} = 0$ to modify $c_{5,32}$. [1] doesn't mention how to modify $c_{6,32}$. However, if we apply the same modification introduced in that paper, the value of $c_{2,23}$ has to be changed to modify $c_{6,32}$. This results in breaking the existing extra condition $c_{2,23} = 0$, and this time, there is no opportunity to modify $c_{6,32}$ without using extra conditions. This case happens with probability $\frac{1}{4}$. To avoid this conflict of modification, we change the way to modify $c_{5,32}$. We used another technique to modify $c_{5,32}$.

Modification of $c_{5,29}$

The modification of $c_{5,29}$ explained in [1] also has a problem. In this method, the value of $d_{2,20}$ is modified to make an internal collision. However, there is a sufficient condition on $d_{2,20}$, which is $d_{2,20} = a_{2,20}$. Therefore, if $c_{5,29}$ is modified, it breaks the sufficient condition $d_{2,20} = a_{2,20}$, and modification of $c_{5,29}$ doesn't make any sense. This happens with probability $\frac{1}{2}$. To avoid breaking a sufficient condition by modifying $d_{5,29}$, we change the way to modify $d_{5,29}$. We used another technique to modify $d_{5,29}$.

Now, all the ambiguous modifications are replaced with precise modifications, and we are able to satisfy all sufficient conditions in round 1 and 2.

2.3 Modifications of sufficient conditions in round 3

Although we are able to modify all sufficient conditions in round 1 and 2, two sufficient conditions in round 3 are still remained. However, we realized the method to modify these conditions. As a result of using our technique, the probability to find a collision becomes almost 1, and the complexity is upper bounded by three times of MD4 hash operations.

3 Conclusion

Without applying our techniques to [1], the probability to find a collision is from 2^{-6} to 2^{-2}, and the complexity to find a collision is upper bounded by 2^8 MD4 hash operations. We estimated that the probability of the result of [1] is $2^{-5.61\cdots}$ because of two sufficient conditions in round 3 ($\frac{1}{4}$), two lacked sufficient conditions ($\frac{1}{4}$), modification of $c_{5,29}$ ($\frac{1}{2}$), modification of $d_{5,19}$ ($\frac{7}{8}$), and modification of $c_{5,32}$ ($\frac{3}{4}$).

However, after our techniques are applied, the probability and the complexity to find a collision are greatly improved. The probability to find a collision becomes almost 1, and the complexity to find a collision is upper bounded by three times of MD4 hash operations.

In addition, this technique is expected to be able to be applied to MD5 hash function as well. We will announce the details of our technique later.

In the end of this paper, we put an example of a random message where [1] cannot generate a collision message pair and a collision message pair which was generated with the same random message by using our technique. Table 5 shows this result.

Table 5: Example of the generated collision pair

M_{random}	24ce9d37de4dfca0a3b88fc39c9f9e5c92ee86ada2c9e8b088f3a020c5368a69 0e503cc80c2368f978ff57bf21a1762ad018afb8daa431e9308bf382806a18a1
M_1	368b9d377e2dfc60b5b88fcb0c8fbe5601a6662d9ecc3929aa35aabf887f929f 2740a2c8c8c12039bbb401bdc1983331e45e1f61c150d565ee27d04af1dfec4c
M_1'	368b9d37fe2dfc6025b88fcb0c8fbe5601a6662d9ecc3929aa35aabf887f929f 2740a2c8c8c12039bbb401bdc1983331e45d1f61c150d565ee27d04af1dfec4c

References

[1] Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, Xiuyuan Yu: Cryptanalysis of the Hash Functions MD4 and RIPEMD, Advances in EUROCRYPT2005, LNCS 3494, pp. 1–18, 2005.

[2] Xiaoyun Wang, Dengguo Feng, Xuejia Lai, Hongbo Yu: Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD, rump session, CRYPTO 2004, e-Print, 2003.

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

[4] Ronald L. Rivest: The MD4 Message Digest Algorithm, CRYPTO'90 Proceedings, 1991, http://theory.lcs.mit.edu/ rivest/Rivest-MD4.txt