Preimage Attacks on One-Block MD4, 63-Step MD5 and More

Kazumaro Aoki and Yu Sasaki

NTT, 3-9-11 Midoricho, Musashino-shi, Tokyo, 180-8585 Japan

Abstract. This paper shows preimage attacks on one-block MD4 and MD5 reduced to 63 (out of 64) steps. Our attacks are based on the meet-in-the-middle attack, and many additional improvements make the preimage computable faster than that of the brute-force attack, 2^{128} hash computation. A preimage of one-block MD4 can be computed in the complexity of the 2^{107} MD4 compression function computation, and a preimage of MD5 reduced to 63 steps can be computed in the complexity of the 2^{121} MD5 compression function computation. Moreover, we optimize the computational order of the brute-force attack against MD5, and a preimage of full-round MD5 can be computed in the complexity of the 2^{127} MD5 compression function computation.

Keywords: MD5, MD4, meet-in-the-middle, local collision, one-way, preimage.

1 Introduction

A cryptographic hash function is an important primitive of cryptographic techniques. There are many applications to make a scheme secure using a hash function: message compression in digital signatures and message authentication, for example. However, surprisingly, unlike block ciphers, there are not many concrete instantiations of hash functions. MD5 [12] and SHA-1 [14] are the de facto standards of a hash function and their security is not analyzed well.

A hash function should have several security properties such as collision resistance and one-wayness. After the breakthrough of Wang's work [15], a lot of study has been applied to collision resistance of hash functions. However, the one-wayness of hash functions is not analyzed much.

At FSE 2008, Leurent showed that a preimage attack of MD4, which is a predecessor of MD5 and consists of 48 steps, can be computed in the complexity of the $2^{100.5}$ MD4 compression function [10]. (Hereafter, we omit the unit of complexity, which is the computational complexity of the compression function of the corresponding hash function.) The attack is based on the pioneering work by Dobbertin [4] and its extension [8]. The techniques used in that paper made extensive use of the property of MD4 such as simple step function, not

R. Avanzi, L. Keliher, and F. Sica (Eds.): SAC 2008, LNCS 5381, pp. 103–119, 2009.

well-mixed message expansion, and so on. Therefore, applying those techniques to MD5 directly seems difficult. Recently, [13] have tried to compute a preimage of MD5, which consists of 64 steps, utilizing the techniques in [10]. However, [13] can compute a preimage of reduced variants of MD5 up to only 44 steps faster than the brute-force attack. While De et al. proposed preimage attacks on reduced variants of MD4 and MD5 based on SAT-solver [3].

This paper applies the meet-in-the-middle attack to MD5. With newly developed techniques, a preimage of MD5 reduced to 63 steps can be computed in 2^{121}, and a pseudo-preimage of full-round MD5 can be computed in $2^{125.7}$, which is faster than the brute-force attack. On the concrete preimage of full-round MD5, we develop a clever brute-force algorithm, and it finds a preimage of full-round MD5 in 2^{127}. Moreover, utilizing our technique with absorption properties of Boolean functions used in MD4, we can compute a one-block preimage of MD4 in 2^{107}, while [10] computes a preimage of more than 1 block.

A summary of our results and previously published results is shown in Table 1[1]. Note that we do not think that our attack can be used to practically compute a preimage by using currently available resources, since all of our attacks need very high complexity. Since the storage requirements for our attacks are 2^{32} blocks at most, we do not mention the precise memory requirement in this paper.

Table 1. Comparison of preimage attacks against MD4 and MD5

Target	Attack	Attacked steps	Complexity	
			Pseudo-preimage	Preimage
MD4 (Total 48 steps)	[4]	32	2^{32} †	
	[8]	32	2^{32} †	
	[3]	39	Not given (8 hours) †	
	[10]	48 (Full)	2^{96}	$2^{100.5}$
	Our result (Sect. 5.2)	48 (Full)	2^{107} †	
MD5 (Total 64 steps)	[3]	26	Not given	
	[13]	44	2^{96} †	
	[1]	47	2^{96}	2^{102}
	Our result (Sect. 3.2)	55	2^{96}	2^{113}
	Our result (Sect. 3.3)	59	2^{96}	*
	Our result (Sect. 3.4)	63	2^{112}	2^{121}
	Our result (Sect. 4)	64 (Full)	$2^{125.7}$ ‡	2^{127} †‡

† One-block attack.
‡ The attack is just the brute-force attack, but the computation is optimized.
* This attack only computes a pseudo-preimage. If a very long preimage is accepted, the attack can be converted to a preimage attack whose preimage length is $\approx 2^{64}$ blocks and computed in 2^{113}.

[1] Aumasson et al. independently shows a preimage attack in [1]. We refer their result in the table for convenience.

2 Description of MD5 and MD4

2.1 MD5 Specification and Its Properties

This section briefly describes the specification of MD5. Refer to details in [12].

MD5 is one of the Merkle-Damgård hash functions, that is, the hash value is computed as follows:

$$\begin{cases} H_0 \leftarrow IV, \\ H_{i+1} \leftarrow \mathrm{md5}(H_i, M_i) \qquad \text{for } i = 0, 1, \ldots, n-1, \end{cases} \tag{1}$$

where IV is the initial value defined in the specification, md5: $\{0,1\}^{128} \times \{0,1\}^{512} \rightarrow \{0,1\}^{128}$ is the compression function of MD5, and the output of the hash function is H_n. Before applying (1), the messages string M is processed as follows:

- The messages are padded in 512-bit multiples.
- The padded string includes the length of the message, which is represented by 64-bits, and the length string is placed at the end of the padding.

After the process, the message string is divided into 512-bit blocks, M_i ($i = 0, 1, \ldots, n-1$).

The compression function $H_{i+1} \leftarrow \mathrm{md5}(H_i, M_i)$ is computed as follows.

1. M_i is divided into 32-bit message words m_j ($j = 0, 1, \ldots, 15$).
2. Do the following recurrence:

$$\begin{cases} p_0 \leftarrow H_i, \\ p_{j+1} \leftarrow R_j^{\mathrm{MD5}}(p_j, m_{\pi^{\mathrm{MD5}}(j)}) \qquad \text{for } j = 0, 1, \ldots, 63. \end{cases}$$

3. Output H_{i+1} ($= p_{64} + H_i$), where "+" denotes 32-bit word-wise addition. In this paper, we similarly use "$-$" to denote 32-bit word-wise subtraction.

R_j^{MD5} is the step function for Step j. Let Q_j be a 32-bit value that satisfies $p_j = (Q_{j-3} \| Q_j \| Q_{j-1} \| Q_{j-2})$. R_j^{MD5} is defined as follows:

$$\begin{aligned} R_j^{\mathrm{MD5}}(p_j, m_{\pi^{\mathrm{MD5}}(j)}) &= (Q_{j-2} \| Q_{j+1} \| Q_j \| Q_{j-1}), \quad \text{where } Q_{j+1} \\ &= Q_j + (Q_{j-3} + \Phi_j(Q_j, Q_{j-1}, Q_{j-2}) + m_{\pi^{\mathrm{MD5}}(j)} + k_j) \lll s_j, \end{aligned} \tag{2}$$

where Φ_j, k_j, and s_j are bitwise Boolean function, constant value, and left rotation defined in the specification. $\pi^{\mathrm{MD5}}(j)$ is a function for MD5 message expansion shown in Table 2. Note that $(R_j^{\mathrm{MD5}})^{-1}(\cdot, m_{\pi^{\mathrm{MD5}}(j)})$ can be computed in almost the same complexity as that of R_j^{MD5}.

2.2 MD4 Specification and Its Properties

The structure of MD4 is similar to that of MD5. The compression function of MD4 consists of 48 steps. The step function R_j^{MD4} for Step j is defined as follows:

$$Q_{j+1} = (Q_{j-3} + \Phi_j(Q_j, Q_{j-1}, Q_{j-2}) + m_{\pi^{\mathrm{MD4}}(j)} + k_j) \lll s_j, \tag{3}$$

Table 2. MD5 message expansion

$\pi^{\mathrm{MD5}}(0), \pi^{\mathrm{MD5}}(1), \ldots, \pi^{\mathrm{MD5}}(15)$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\pi^{\mathrm{MD5}}(16), \pi^{\mathrm{MD5}}(17), \ldots, \pi^{\mathrm{MD5}}(31)$	1	6	11	0	5	10	15	4	9	14	3	8	13	2	7	12
$\pi^{\mathrm{MD5}}(32), \pi^{\mathrm{MD5}}(33), \ldots, \pi^{\mathrm{MD5}}(47)$	5	8	11	14	1	4	7	10	13	0	3	6	9	12	15	2
$\pi^{\mathrm{MD5}}(48), \pi^{\mathrm{MD5}}(49), \ldots, \pi^{\mathrm{MD5}}(63)$	0	7	14	5	12	3	10	1	8	15	6	13	4	11	2	9

Table 3. MD4 Boolean functions and message expansion

$\Phi_j(X,Y,Z),\ 0 \le j \le 15$	$(X \wedge Y) \vee (\neg X \wedge Z)$															
$\Phi_j(X,Y,Z), 16 \le j \le 31$	$(X \wedge Y) \vee (Y \wedge Z) \vee (X \wedge Z)$															
$\Phi_j(X,Y,Z), 32 \le j \le 47$	$X \oplus Y \oplus Z$															
$\pi^{\mathrm{MD4}}(0), \pi^{\mathrm{MD4}}(1), \ldots, \pi^{\mathrm{MD4}}(15)$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\pi^{\mathrm{MD4}}(16), \pi^{\mathrm{MD4}}(17), \ldots, \pi^{\mathrm{MD4}}(31)$	0	4	8	12	1	5	9	13	2	6	10	14	3	7	11	15
$\pi^{\mathrm{MD4}}(32), \pi^{\mathrm{MD4}}(33), \ldots, \pi^{\mathrm{MD4}}(47)$	0	8	4	12	2	10	6	14	1	9	5	13	3	11	7	15

where Φ_j, k_j, s_j, and $\pi^{\mathrm{MD4}}(j)$ are defined *differently* than in MD5. Φ_j and $\pi^{\mathrm{MD4}}(j)$ are shown in Table 3. Note that $(R_j^{\mathrm{MD4}})^{-1}(\cdot, m_{\pi^{\mathrm{MD4}}(j)})$ can be computed in almost the same complexity as that of R_j^{MD4}.

Hereafter, we omit superscripts of $R_j^{\mathrm{MD5}}, R_j^{\mathrm{MD4}}, \pi^{\mathrm{MD5}}$, and π^{MD4} if the hash function discussed is obvious from the context.

3 Preimage Attacks against Reduced MD5

3.1 Converting Pseudo-preimages to a Preimage

First, we describe the generic algorithm that converts pseudo-preimages to a preimage [11, Fact 9.99]. Assume that there is an algorithm that finds $(H_1, (M_1, M_2, \ldots, M_{n-1}))$ such that $H_{i+1} = \mathrm{md5}(H_i, M_i)$ $(i = 1, 2, \ldots, n-1)$ in the complexity of 2^x and H_1 looks random. Prepare a table that includes $2^{64-x/2}$ entries of $(H_1, (M_1, M_2, \ldots, M_{n-1}))$. Compute $2^{64+x/2}$ $\mathrm{md5}(H_0, M_0)$ for random M_0, then one of them agrees with one of the entries in the table with high probability. The required complexity of the attack is about $2^{65+x/2}$. Therefore, showing how to compute (H_1, M_1) from a given hash value within 2^x where $x < 126$ is enough for theoretical preimage attack.

3.2 A Preimage Attack against MD5 Reduced to 55 Steps

The proposed attack finds a preimage of MD5 reduced to 55 steps from a given hash value H_n. Our attack target is MD5 reduced to 55 steps, and the steps lie from Step 5 to Step 59[2]. We propose a new technique called the splice-and-cut technique.

[2] We confirmed that Step 5 to Step 59, which are 55 steps in total, are the longest section that can be attacked with only the splice-and-cut technique.

Technique 1: Splice-and-Cut
We consider the first and last steps of the attack target as consecutive steps. Then, we divide the attack target into two chunks of steps so that each chunk includes at least one message word that is independent from the other chunk. We call such message words "neutral words." Then, we find pseudo-preimages by the meet-in-the-middle approach.

Observe the message expansion described in Table 2 and notice that Steps 23-37 do not contain m_0, m_6, m_{10}, m_{15}, and Steps 5-22 and 38-59 do not contain m_4 as shown in Fig. 1.

Our attack finds a 2-block preimage, so first, the appropriate padding strings for 2-block messages are set in m_{13}, m_{14}, and m_{15}. For a given H_2, an attack procedure is given below.

Attack Procedure

1. Choose m_i ($i \notin \{4, 6, 13, 14, 15\}$) and p_{38} randomly.
2. For all m_6, do the following:

$$\begin{cases} p_{j+1} \leftarrow R_j(p_j, m_{\pi(j)}) & \text{for } j = 38, 39, \ldots, 59, \\ p_5 \leftarrow H_2 - p_{60}, & \\ p_{j+1} \leftarrow R_j(p_j, m_{\pi(j)}) & \text{for } j = 5, 6, \ldots, 22. \end{cases}$$

3. Make a table of (m_6, p_{23})s which are computed in the last step.
4. For all m_4, do the following:

$$p_j \leftarrow R_j^{-1}(p_{j+1}, m_{\pi(j)}) \qquad \text{for } j = 37, 36, \ldots, 23,$$

   and examine that the computed p_{23} is in the table made by the previous step. If p_{23} is in the table, the corresponding message and H_1 is just a pseudo-preimage of H_2.

Note that p_5 in the attack is just H_1.

The computational complexity of the above attack procedure is about 2^{32} ($= 2^{32}\frac{40}{55} + 2^{32}\frac{15}{55}$), and the success probability is about 2^{-64} ($= 2^{32} \cdot 2^{32}/2^{128}$). Thus, by iterating the above procedure 2^{64} times, we expect to find one pseudo-preimage (H_1, M_1), and its complexity is about 2^{96} ($= 2^{64} \cdot 2^{32}$). By applying the technique in Section 3.1, we expect that a preimage of MD5 reduced to 55 steps can be computed in 2^{113} ($= 2^{65+96/2}$).

3.3 A Preimage Attack against MD5 Reduced to 59 Steps

We propose an attack that finds a preimage of MD5 reduced to 59 steps starting from Step 3 and ending with Step 61. We notice that this attack cannot deal with the message padding, therefore, the attack can only find a pseudo-preimage.

In the attack against MD5 reduced to 55 steps, two chunks reach the same p_i, and we examine 128-bit matching. Here, we do not have to check all 128 bits, but we check part of them e.g. only 32 bits.

Assume that one chunk produces 2^{32} p_is and the other chunk produces 2^{32} p_{i-3}s. Since $p_i = (Q_{i-3} \| Q_i \| Q_{i-1} \| Q_{i-2})$ and $p_{i-3} = (Q_{i-6} \| Q_{i-3} \| Q_{i-4} \| Q_{i-5})$,

Step	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
index	⓪	1	2	3	④	5	⑥	7	8	9	⑩	11	12	13	14	⑮	
	excluded					first chunk											
Step	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
index	1	⑥	11	⓪	5	⑩	⑮	④	9	14	3	8	13	2	7	12	
	first chunk							second chunk									
Step	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	
index	5	8	11	14	1	④	7	⑩	13	⓪	3	⑥	9	12	⑮	2	
	second chunk						first chunk										
Step	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	
index	⓪	7	14	5	12	3	⑩	1	8	⑮	⑥	13	④	11	2	9	
	first chunk												excluded				

Fig. 1. Message word distribution in MD5 observed in 55-step attack

Step	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
index	0	1	②	3	4	5	6	7	8	9	10	11	12	13	14	⑮	
	excluded							first chunk									
Step	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
index	1	6	11	0	5	10	⑮	4	9	14	3	8	13	②	7	12	
	first chunk							second chunk									
Step	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	
index	5	8	11	14	1	4	7	10	13	0	3	6	9	12	⑮	②	
	second chunk													skip			
Step	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	
index	0	7	14	5	12	3	10	1	8	⑮	6	13	4	11	②	9	
	first chunk														excluded		

Fig. 2. Message word distribution in MD5 observed in 59-step attack

we can examine 32-bit matching without computing three steps. This enables us to find longer sections that are vulnerable against our attack.

Technique 2: Partial Matching
By executing only one-word matching instead of all-word matching, up to three consecutive steps can be skipped from the attack target.

Observe the message expansion described in Table 2 and notice that Steps 23-44 do not contain m_{15}, and Steps 3-22 and 48-61 do not contain m_2 as shown in Fig. 2. For a given H_2, the rough sketch of the attack procedure is as follows[3].

Attack Procedure
1. Choose m_i ($i \notin \{2, 15\}$) and p_{23} randomly.
2. For all m_{15}, do the following:

[3] In this attack, skipping two steps is enough. However, we explain the attack procedure for skipping three steps to show the generality of our attack.

$$\begin{cases} p_j \leftarrow R_j^{-1}(p_{j+1}, m_{\pi(j)}) & \text{for } j = 22, 21, \ldots, 3, \\ p_{62} \leftarrow H_2 - p_3, & \\ p_j \leftarrow R_j^{-1}(p_{j+1}, m_{\pi(j)}) & \text{for } j = 61, 60, \ldots, 48, \end{cases}$$

and store (m_{15}, p_{48})s in a table.

3. For all m_2, do the following:

$$p_{j+1} \leftarrow R_j(p_j, m_{\pi(j)}) \qquad \text{for } j = 23, 24, \ldots, 44.$$

Since $p_{48} = (Q_{45} \| Q_{48} \| Q_{47} \| Q_{46})$ and $p_{45} = (Q_{42} \| Q_{45} \| Q_{44} \| Q_{43})$ we can examine Q_{45} is in the table. If Q_{45} is in the table, we compute Q_{46} to Q_{48} by the corresponding m_i, and check whether all of Q_{46} to Q_{48} are matched.

The computational complexity and the success probability are almost the same with the attack against MD5 reduced to 55 steps. Therefore, a pseudo-preimage of MD5 reduced to 59 steps can be found at the complexity of 2^{96}. Note that we can compute a very long preimage by using the technique in Section 3.1 and expandable message introduced in [6], where the length is determined by m_{15}.

Note that we can attack MD5 reduced up to 50 steps even if we restrict that the reduced MD5 should start with the first step (Step 0). The first chunk starts with Step 17 and is 19 steps long, and the second chunk starts with Step 36 and is 28 steps long. The neutral words are m_1 and m_{14}.

3.4 A Preimage Attack against MD5 Reduced to 63 Steps

We propose an attack that finds a preimage of the last 63 steps of MD5. In addition to the splice-and-cut and partial-matching techniques, we use partial-fixing technique.

In previous attack variants, neutral words are totally free when we execute the meet-in-the-middle attack, and thus, both chunks can produce 2^{32} outputs. In this attack, we fix the lower 16 bits of a neutral word. By this effort, computation of one chunk can be partially continued even if the message word for the other chunk appears.

Let us see the inversion of the step function R_j^{-1}. $R_j^{-1}(\cdot, m_{\pi(j)})$ is written by using Q_j as follows:

$$Q_{j-3} = ((Q_{j+1} - Q_j) \ggg s_j) - \Phi_j(Q_j, Q_{j-1}, Q_{j-2}) - m_{\pi(j)} - k_j. \qquad (4)$$

When the lower n bits of Q_{j-1}, Q_{j-2}, and $m_{\pi(j)}$ are fixed and other variables are fully fixed, we can compute the lower n bits of $R_j^{-1}(\cdot, m_{\pi(j)})$ independently from the higher $32 - n$ bits of Q_{j-1}, Q_{j-2}, and $m_{\pi(j)}$. As a consequence, we can partially compute 3 more steps if neutral words are partially fixed. This is graphically explained in Appendix A.

Technique 3: Partial Fixing
By partially fixing neutral words in chunks, up to three consecutive steps can be additionally skipped from the attack target.

Observe the message expansion described in Table 2 and notice that Steps 19-42 do not contain m_6, and Steps 1-18 and 49-63 do not contain m_0 as shown in Fig. 3.

For a given H_2, the rough sketch of the attack procedure is as follows.

Attack Procedure

1. Set m_{13}, m_{14}, and m_{15} to appropriate padding for 2-block messages.
2. Choose m_i ($i \notin \{0, 6\}$), p_{19}, and the lower 16 bits of m_0, randomly.
3. For all higher 16 bits of m_0, do the following:

$$p_{j+1} \leftarrow R_j(p_j, m_{\pi(j)}) \qquad \text{for } j = 19, 20, \ldots, 42,$$

   and store (m_0, p_{43})s in a table, where $p_{43} = (Q_{40} \| Q_{43} \| Q_{42} \| Q_{41})$.
4. (a) For all m_6, do the following:

$$\begin{cases} p_j \leftarrow R_j^{-1}(p_{j+1}, m_{\pi(j)}) & \text{for } j = 18, 17, \ldots, 1, \\ p_{64} \leftarrow H_2 - p_1, & \\ p_j \leftarrow R_j^{-1}(p_{j+1}, m_{\pi(j)}) & \text{for } j = 63, 62, \ldots, 49. \end{cases}$$

   (b) From obtained $p_{49} = (Q_{46} \| Q_{49} \| Q_{48} \| Q_{47})$, by the partial-fixing technique, we can compute the lower 16 bits of Q_{45}, Q_{44}, and Q_{43}.
   (c) From the partial-matching technique described in Section 3.3, we can examine 16-bit matching by Q_{43}.

Step 3 of the above procedure needs the complexity of 2^{16}, and steps 4(a) and 4(b) need the complexity of 2^{32}. Therefore, the total complexity is 2^{32}. At step 4(c), we examine 16-bit matching for 2^{48} pairs, and we obtain $2^{48} \times 2^{-16} = 2^{32}$ pairs whose 16 bits are matched. Finally, by repeating the above procedure 2^{80} times, we obtain a pair, where all 128 bits are matched. Therefore, the final complexity of the pseudo-preimage attack is $2^{32} \times 2^{80} = 2^{112}$, and this is converted to a preimage attack whose complexity is 2^{121}.

Step	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
index	⓪	1	2	3	4	5	⑥	7	8	9	10	11	12	13	14	15
excluded		first chunk														
Step	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
index	1	⑥	11	⓪	5	10	15	4	9	14	3	8	13	2	7	12
	first chunk			second chunk												
Step	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
index	5	8	11	14	1	4	7	10	13	⓪	3	⑥	9	12	15	2
	second chunk											skip				
Step	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
index	⓪	7	14	5	12	3	10	1	8	15	⑥	13	4	11	2	9
skip		first chunk														

Fig. 3. Message word distribution in MD5 observed in 63-step attack

4 Notes on Preimage Attack against Full-Round MD5

This section studies the preimage resistance against full-round MD5. We, unfortunately, cannot find any "cryptanalytic attacks" against full-round MD5. While, we find clever technique to perform a brute-force attack. Using this technique, we can find a pseudo-preimage of MD5 at the complexity of 2^{126}, and a preimage of MD5 at the complexity of about 2^{127}.

4.1 Finding Pseudo-preimage of MD5

As we learned by the partial-matching and partial-fixing techniques, a few steps can be skipped from the attack target. Based on this finding, we searched for the minimum number of steps that must be skipped to attack the full-round MD5. The best selection of two chunks where the number of skipped steps is 19 is shown in Fig. 4. (Only this pattern allows skipped steps to be less than 20.)

Since the number of skipped steps is large, we cannot find an efficient way to check whether results from both chunks are matched or not. In this attack, we exhaustively search for the pair that can be matched. Assume we obtain values of p_{14}, p_{33}, and all message words. Whether the computation for that message from p_{14} reaches p_{33} can be checked at the complexity of computing only 13 steps with negligible cost since the complexity of computing 6 steps can be saved by the partial-matching and partial-fixing techniques.

When we only consider pseudo-preimage of the compression function md5, the attack procedure becomes very simple. However, we later want to discuss the conversion from pseudo-preimage(s) to a preimage in Section 4.3. So, we stress that m_{14} is selected as a neutral word, Therefore, some effort is necessary to adjust the padding part. Since m_5 is selected as a neutral word, the last message block must be longer than or equal to 192 bits. As explained later, this attack needs at least a 2-block message. Therefore, we fix 9 bits of m_{14} to guarantee that the value of m_{14} is $192 + 512n, n \geq 1$ for any choice of other bits. Details of messages we select are as follows:

Step	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
index	0	1	2	3	4	⑤	6	7	8	9	10	11	12	13	⑭	15
	first chunk														skip	
Step	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
index	1	6	11	0	⑤	10	15	4	9	⑭	3	8	13	2	7	12
	skip															
Step	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
index	⑤	8	11	⑭	1	4	7	10	13	0	3	6	9	12	15	2
	skip	second chunk														
Step	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
index	0	7	⑭	⑤	12	3	10	1	8	15	6	13	4	11	2	9
	2nd chunk			first chunk												

Fig. 4. Message word distribution in MD5 observed in full-round MD5

- $m_0, \ldots, m_4 \leftarrow$ Randomly chosen fixed value,
- Lower 16 bits of $m_5 \leftarrow$ Randomly chosen fixed value,
- $m_6 \leftarrow$ `0x00000080`,
- $m_7, \ldots, m_{13} \leftarrow$ `0x00000000`,
- m_{14} is chosen to be $192 + 512n$, $n \geq 1$,
- $m_{15} \leftarrow$ `0x00000000`.

For a given hash value H_n, the attack procedure is as follows.

Attack Procedure

1. Set messages as explained above and choose p_{51} randomly.
2. For all the 23 free-bits of m_{14}, do the following:

$$\begin{cases} p_j \leftarrow R_j^{-1}(p_{j+1}, m_{\pi(j)}) \text{ for } j = 50, 49, \ldots, 33, \\ \text{Partially compute } Q_{29}, Q_{28}, \text{ and } Q_{27} \text{ by the partial-fixing technique,} \end{cases}$$

   and store $(m_{14}, p_{33}, \text{partial } Q_{29}, \text{partial } Q_{28}, \text{partial } Q_{27})$s in a table.
3. For all of higher 16-bits of m_5, do the following:

$$\begin{cases} p_{j+1} \leftarrow R_j(p_j, m_{\pi(j)}) & \text{for } j = 51, 52, \ldots, 63, \\ p_0 \leftarrow H_n - p_{64}, & \\ p_{j+1} \leftarrow R_j(p_j, m_{\pi(j)}) & \text{for } j = 0, 1, \ldots, 13, \end{cases}$$

   and keep the values of (m_5, p_{14}).

   (a) For all $(m_{14}, p_{33}, \text{partial } Q_{29}, \text{partial } Q_{28}, \text{partial } Q_{27})$ stored in a table, do the following:

$$p_{j+1} \leftarrow R_j(p_j, m_{\pi(j)}) \qquad \text{for } j = 14, 15, \ldots, 26,$$

   and examine the lower 16-bit match of Q_{27}.

   (b) If lower 16 bits of Q_{27} are matched, compute all bits of Q_{29}, Q_{28}, and Q_{27} by using p_{33} and m_5. Then, examine the higher 16-bit match of Q_{27}.

   (c) If higher 16 bits of Q_{27} are matched, compute $p_{28} = R_{27}(p_{27}, m_{\pi(27)})$, and examine the match of Q_{28}.

   (d) If Q_{28} is matched, compute $p_{29} = R_{28}(p_{28}, m_{\pi(28)})$, and examine the match of Q_{29}.

   (e) If Q_{29} is matched, compute $p_{30} = R_{29}(p_{29}, m_{\pi(29)})$, and examine the match of Q_{30}. If matched, corresponding (p_0, M) is a pseudo-preimage.

Step 2 of the above procedure needs the complexity of $2^{23}\frac{21}{64}$. For each m_5, the first 4 lines of step 3 need the complexity of $\frac{27}{64}$. Step 3(a) needs the complexity of $2^{23}\frac{13}{64}$. As a result of lower 16-bit match of Q_{27}, $2^{23} \times 2^{-16} = 2^7$ pairs are expected to be remained. Step 3(b) needs the complexity of $2^7\frac{3}{64}$. As a result of higher 16-bit match of Q_{27}, $2^7 \times 2^{-16} = 2^{-9}$ pair is expected to be remained. Step 3(c) needs the complexity of $2^{-9}\frac{1}{64}$. After the match of Q_{28}, $2^{-9} \times 2^{-32} = 2^{-41}$ pair is expected to be remained. The complexities of 3(d) and 3(e) are negligible. Hence, the complexity of step 3 is $2^{23}\frac{21}{64} + 2^{16}\frac{27}{64} + 2^{16}(2^{23}\frac{13}{64} + 2^7\frac{3}{64} + 2^{-9}\frac{1}{64}) < 2^{37}$.

Finally, by repeating this procedure 2^{89} times, we obtain a pair, where all 128 bits are matched. Therefore, the final complexity of the pseudo-preimage attack is $2^{89} \times (2^{16} \times 2^{23}\frac{13}{64}) < 2^{126}$.

4.2 Increase the Speed of the Naive Search

When we compute hash values of 2^{128} different messages, we do not have to compute 2^{128} times of md5. For example, two different messages whose m_0 to m_{14} are the same and whose m_{15} are different will have the same computation result after the first 15 steps. This saves us the cost of computing the first 15 steps of the second message. By extending this idea, the complexity of computing hash values of 2^{128} different messages becomes $2^{127} = 2^{128}\frac{32}{64}$.

We use a technique named *Q4 Tunnel* by Klima [7], which enables us to compute md5 from an intermediate step. In this technique, the value of m_3 in the first round is changed, however, any change in m_3 can be offset by modifying m_4 and m_7 so that all other chaining variables in the first round are kept unchanged. Since m_3, m_4, and m_7 appear in Steps 23, 26, and 30, respectively in the second round, any choice of m_3 does not impact on chaining variables up to Step 22. Therefore, this technique saves us the cost for computing the first 23 steps. Moreover, we can save the complexity of a few more steps:

1. Since the initial value and hash value are fixed and message words used in Steps 61, 62, and 63 are not m_3, m_4, and m_7, we can compute p_{63}, p_{62}, and p_{61} independently of m_3, m_4, and m_7. This saves us the complexity of three steps.
2. The partial-matching and partial-fixing techniques described in Section 3.3 save us the complexity of six steps.

Finally, the complexity to compute a hash value becomes $64 - 23 - 3 - 6 = 32$ steps, we can compute hash values of 2^{128} messages at the complexity of $2^{128}\frac{32}{64}$.

4.3 Discussion on Converting a Pseudo-preimage to a Preimage

Let $E(x) = 1 - \exp(-x)$, then the success probability of a brute-force attack for computing preimage is $b = E(2^{128}/2^{128}) \approx 0.63$. The attack described in Section 4.2 finds a pseudo-preimage at the complexity of $2^{125.70}$ $(= 2^{128}\frac{13}{64})$ with probability b. If we directly use the conversion described in Section 3.1, a preimage will be found at the complexity of $2^{127.85}$ with probability b^2. This complexity is higher than $2^{127.00}$ $(= 2^{128}\frac{32}{64})$ described in Section 4.2. Applying the technique in Section 4.2 to the computation of $\mathrm{md5}(H_0, M_0)$ in the conversion described in Section 3.1, resulting preimage attack still requires $2^{127.39}$ with probability b^2.

Using the idea of expandable message [6] as in [10], one preimage is enough to compute a preimage[4]. This attack requires $2^{126.86}$ $(= 2^{128}\frac{13}{64} + 2^{128}\frac{32}{64}/2)$. However, the success probability is b^2, which is lower than that of the brute-force attack, b. Spending $c_1 2^{128}\frac{13}{64}$ work for computing a pseudo-preimage and $c_2 2^{128}\frac{32}{64}/2$ for brute-force attack, the success probability of the attack is $E(c_1)$ $E(c_2)$ and the complexity is $2^{128}\frac{13c_1+16c_2}{64}$. To achieve the same success probability $b = E(c_1)E(c_2)$, the attack requires $2^{127.52}$, where $c_1 \approx 1.672$ and $c_2 \approx 1.507$.

[4] Appendix B shows extensions of the attack.

Even if computing a pseudo-preimage fails, we can continue to seek a preimage using brute-force attack. The complexity of the attack is also $2^{128\frac{13c_1+16c_2}{64}}$, and the success probability increases to $E(c_1)E(c_2)+(1-E(c_1))E(c_2/2)$. To achieve the same success probability b, the attack requires $2^{126.94}$, where $c_1 \approx 0.354$ and $c_2 \approx 1.636$.

5 Preimage Attacks against MD4

The first preimage attack against full-round MD4 was proposed by Leurent [10]. It finds a preimage of MD4 with the complexity of 2^{102} by using messages of 34 blocks[5]. Therefore, no one has succeeded in attacking MD4 by using one-block messages. A one-block attack is particularly interesting since an attacker cannot use the characteristics of the Merkle-Damgård structure. A one-block attack analyzes the security of the compression function md4.

In this section, we first show a preimage attack using messages of 2 blocks to show that the splice-and-cut approach can be also applied to MD4. This attack finds a preimage of MD4 at the complexity of 2^{121}. Second, we show a one-block attack that finds a preimage at the complexity of 2^{107}.

5.1 Two-Block Preimage Attack against MD4

MD4 can be analyzed in a manner similar to MD5. By using the splice-and-cut, partial-matching, and partial-fixing techniques, we can find a pseudo-preimage at the complexity of 2^{112}, and this attack is converted to a preimage attack at the complexity of 2^{121}. The selection of two chunks is shown in Fig. 5.

5.2 One-Block Preimage Attack against MD4

By checking the details of the step function of MD4, we can find a preimage that consists of a one-block message. The key idea is fixing the value of p_0 to the original MD4 IV when we compute a chunk. To achieve this, we use a local-collision approach. By this approach, the value of p_0 can be kept unchanged even if the value of a neutral word in a chunk is changed. (The similar idea is used by Sasaki et al. [13] to analyze MD5.) We thus search for a pair of chunks in which one chunk includes one neutral word and the other chunk includes two neutral words where changes of one neutral word can be offset by changing the other neutral word. The selected chunks are shown in Fig. 6.

When we compute the first chunk by changing the value of m_7, the corresponding chaining variable Q_4 is updated according to the selection of m_7. In this attack, by selecting m_3 adaptively and fixing p_7, m_0 to m_2, and m_4 to m_6 in advance, Q_0 to Q_{-3} can be fixed to the original IV of MD4 for any m_7. This attack heavily uses the absorption properties of Boolean functions of MD4, so readers who are not familiar with them are recommended to read [10, Section 2.1]. The method to

[5] This attack can be easily converted to an attack that finds a preimage with the complexity of 2^{113} by using messages of 2 blocks.

Step	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
index	0	1	2	3	4	5	6	⑦	⑧	9	10	11	12	13	14	15
	first chunk								second chunk							
Step	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
index	0	4	⑧	12	1	5	9	13	2	6	10	14	3	⑦	11	15
	second chunk													skip		
Step	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
index	0	⑧	4	12	2	10	6	14	1	9	5	13	3	11	⑦	15
	skip		first chunk													

Fig. 5. Message word distribution in MD4 observed in 2-block attack

Step	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
index	0	1	2	③	4	5	6	⑦	⑧	9	10	11	12	13	14	15
	first chunk								second chunk							
Step	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
index	0	4	⑧	12	1	5	9	13	2	6	10	14	③	⑦	11	15
	second chunk												skip			
Step	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
index	0	⑧	4	12	2	10	6	14	1	9	5	13	③	11	⑦	15
	skip		first chunk													

Fig. 6. Message word distribution in MD4 observed in 1-block attack

select p_7 and m_0 to m_7 is shown in Table 4. $\mathbf{0}$, $\mathbf{1}$, $\mathbf{C}_i$, and $*$ denote `0x00000000`, `0xffffffff`, a randomly fixed value, and a flexible value which depends on the value of m_7, respectively.

The attack procedure is as follows:

Precomputation

1. Set the values of chaining variables Q_j as shown in Table 4. Note the value of $*$ is left undetermined.
2. Compute $m_j, j \in \{0, 1, 2, 4, 5, 6\}$ by the following equation:

$$m_{\pi(j)} = (Q_{j+1} \ggg s_j) - Q_{j-3} - \Phi_j(Q_j, Q_{j-1}, Q_{j-2}) - k_j. \tag{5}$$

Computation of the first chunk including m_7 and m_3

3. For all 32-bits of m_7, compute the value of $*$.
4. For each m_7, compute m_3 by the following equation:

$$m_{\pi(3)} = (Q_4 \ggg s_3) - Q_0 - \Phi_3(Q_3, Q_2, Q_1) - k_3. \tag{6}$$

Table 4. Fixed values for MD4 one-block preimage attack

step j	$m_{\pi(j)}$	Q_{j-2}	Q_{j+1}	Q_j	Q_{j-1}
0	m_0	Q_{-3}	Q_0	Q_{-1}	Q_{-2}
1	m_1	Q_{-2}	$\mathbf{C}_4$	Q_0	Q_{-1}
2	m_2	Q_{-1}	$\mathbf{C}_3$	$\mathbf{C}_4$	Q_0
3	$\textcircled{m}_3$	Q_0	$\mathbf{C}_3$	$\mathbf{C}_3$	$\mathbf{C}_4$
4	m_4	$\mathbf{C}_4$	$*$	$\mathbf{C}_3$	$\mathbf{C}_3$
5	m_5	$\mathbf{C}_3$	$\mathbf{0}$	$*$	$\mathbf{C}_3$
6	m_6	$\mathbf{C}_3$	$\mathbf{1}$	$\mathbf{0}$	$*$
7	$\textcircled{m}_7$	$*$	$\mathbf{C}_2$	$\mathbf{1}$	$\mathbf{0}$
8		$\mathbf{0}$	$\mathbf{C}_1$	$\mathbf{C}_2$	$\mathbf{1}$

At Step 2 of the above procedure, the value of $*$ is involved in the computation for m_4, m_5, and m_6. However, due to the absorption properties, m_4, m_5, and m_6 can be computed independently on $*$.

In this attack, we fix lower 11-bits of m_8 to an arbitrary value. Then, we compute the second chunk for all the remaining 21-bits of m_8 and store the results. After that, we compute the first chunk for all m_7, then check whether they are matched with stored items by comparing the lower 11-bits of Q_{27} and Q_{28}[6]. Finally, we can find a one-block preimage at the complexity of 2^{107}.

6 Conclusion

This paper has shown the preimage attacks of one-block MD4 and MD5 reduced to 63 (out of 64) steps. A preimage of MD5 reduced to 63 steps can be computed in 2^{121} MD5 computations, which is faster than the brute-force attack, and a pseudo-preimage of full-round MD5 can be computed in $2^{125.7}$ MD5 computations. On a preimage of full-round MD5, we optimize the computational order of the brute-force attack against MD5, and a preimage of full-round MD5 can be computed in the complexity of the 2^{127} MD5 compression function computation. Moreover, a one-block preimage of MD4 can be computed in 2^{107} MD4 computations, while the previous work [10] computes a preimage of more than 1 block. The key idea of our attacks, which are based on the meet-in-the-middle technique, is quite simple, but very effective for preimage attacks. We left the application of our attack to other hash functions as a problem.

Acknowledgments

The authors wish to thank Christophe De Cannière and Christian Rechberger for providing [2] and anonymous referees for many useful comments on this paper.

[6] In MD4, up to four steps can be additionally skipped by the partial-fixing technique.

References

1. Aumasson, J.-P., Meier, W., Mendel, F.: Preimage attacks on 3-pass HAVAL and step-reduced MD5. In: Avanzi, R., Keliher, L., Sica, F. (eds.) Selected Areas in Cryptography — Workshop Records of 15th Annual International Workshop, SAC 2008, Sackville, New Brunswick, Canada, pp. 99–114 (2008); also appeared in IACR Cryptology ePrint Archive: Report 2008/183 http://eprint.iacr.org/2008/183
2. De Cannière, C., Rechberger, C.: Preimages for reduced SHA-0 and SHA-1. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 179–202. Springer, Heidelberg (2008); slides on preliminary results were appeared at ESC 2008 seminar, http://wiki.uni.lu/esc/
3. De, D., Kumarasubramanian, A., Venkatesan, R.: Inversion attacks on secure hash functions using SAT solvers. In: Marques-Silva, J., Sakallah, K.A. (eds.) SAT 2007. LNCS, vol. 4501, pp. 377–382. Springer, Heidelberg (2007)
4. Dobbertin, H.: The first two rounds of MD4 are not one-way. In: Vaudenay, S. (ed.) FSE 1998. LNCS, vol. 1372, pp. 284–292. Springer, Heidelberg (1998)
5. Flajolet, P., Odlyzko, A.M.: Random mapping statistics. In: Quisquater, J.-J., Vandewalle, J. (eds.) EUROCRYPT 1989. LNCS, vol. 434, pp. 329–354. Springer, Heidelberg (1990)
6. Kelsey, J., Schneier, B.: Second preimages on n-bit hash functions for much less than 2^n work. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 474–490. Springer, Heidelberg (2005)
7. Klima, V.: Tunnels in hash functions: MD5 collisions within a minute (IACR Cryptology ePrint Archive: Report 2006/105) (2006), http://eprint.iacr.org/2006/105
8. Kuwakado, H., Tanaka, H.: New algorithm for finding preimages in a reduced version of the MD4 compression function. IEICE Transactions Fundamentals of Electronics, Communications and Computer Sciences (Japan) E83-A(1), 97–100 (2000)
9. Lai, X., Massey, J.L.: Hash functions based on block ciphers. In: Rueppel, R.A. (ed.) EUROCRYPT 1992. LNCS, vol. 658, pp. 55–70. Springer, Heidelberg (1993)
10. Leurent, G.: MD4 is not one-way. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 412–428. Springer, Heidelberg (2008)
11. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of applied cryptography. CRC Press, Boca Raton (1997)
12. Rivest, R.L.: Request for Comments 1321: The MD5 Message Digest Algorithm. The Internet Engineering Task Force (1992), http://www.ietf.org/rfc/rfc1321.txt
13. Sasaki, Y., Aoki, K.: Preimage attacks on step-reduced MD5. In: Mu, Y., Susilo, W., Seberry, J. (eds.) ACISP 2008. LNCS, vol. 5107, pp. 282–296. Springer, Heidelberg (2008)
14. U.S. Department of Commerce, National Institute of Standards and Technology. Announcing the SECURE HASH STANDARD (Federal Information Processing Standards Publication 180-2) (2002), http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf
15. Wang, X., Yu, H.: How to break MD5 and other hash functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)

A Graphical Explanation of Partial-Matching and Partial-Fixing Techniques

The way partial-matching and partial-fixing techniques work when we skip 6 steps from the attack target are shown in Fig. 7. The numbers in Fig. 7 denote the number of bits that can be computed independently of the neutral word for the other chunk.

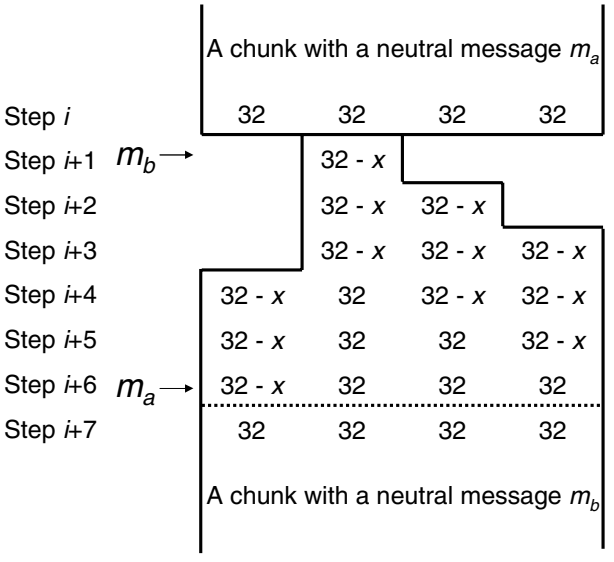

Fig. 7. Graphical explanation of the partial-matching and partial-fixing techniques

First, we store results of the computation of the chunk including m_a for all possible values of m_a. Note that to use the partial-fixing technique, we fix the lower x-bits of m_a to any value. Since m_b is used in Step $i+1$, this computation can be carried out until Step i. Then, we compute the chunk including m_b for all possible values of m_b. Until Step $i+7$, all 128-bit values can be computed independently of m_a.

Since the lower x-bits of m_a are fixed by the partial-fixing technique, we can partially execute inverse computation in Step $i+6$ independently of the higher $(32-x)$-bits of m_a. A similar situation occurs in Steps $i+5$ and $i+4$. Finally, by applying the partial-matching technique, we can compare $32-x$ bits of results of the two chunks.

B Notes on MD-Strengthening

Section 3.1 describes how to convert pseudo-images to a preimage on Merkle-Damgård structure. When the length of preimages is not fixed and MD-strengthening [9] is used in the target hash function, the method cannot be applied. The

problem can be solved using "expandable message" introduced in [6]. An (a, b)-expandable message inputs a fixed chaining value, and can take any message length between a and b blocks, and outputs the same chaining value. The expandable message to adjust the length of preimage is already applied to MD4 [10]. Actually, when the compression function is constructed by Davies-Meyer, an expandable message is easily found, and [6] showed how to construct $(n, n+2^n-1)$-expandable message for a generic compression function. However, when the computational cost of computing preimages is nearly the complexity of brute-force attack, [6] may not be efficient to compute a preimage. The following algorithm efficiently produce $(k+1, k+n)$-expandable message for given k. Note that the algorithm is not efficient when n is large compared with [6].

Assume we need to generate a multi-collision that consists of messages whose length are $(k+1)$-block, $(k+2)$-block, ..., $(k+n)$-block. Such a multi-collision is generated as follows:

1. Randomly generate a k-block message M_k and compute $H_k = h(H_0, M_k)$.
2. Randomly generate a 1-block message M_{k+1} and compute $H_{k+1} = h(H_k, M_{k+1})$.
3. For $i = 1, 2, \ldots, n-1$, search for a 1-block message M_{k+i+1} such that $h(H_k, M_{k+i+1}) = h(H_{k+i}, M_{k+i+1})$. Let the generated value be H_{k+i+1}.
4. Finally, $(M_k \| M_{k+n})$, $(M_k \| M_{k+n-1} \| M_{k+n})$, ..., $(M_k \| M_{k+1} \| \cdots \| M_{k+n-1} \| M_{k+n})$ are multi-collision messages of $(k+1)$, $(k+2)$, ..., $(k+n)$ blocks.

In the above procedure, generating M_{k+i+1} costs the complexity of the birthday paradox, which is sufficiently low in the preimage attacks.

Very recently, [2] introduced the use of P^3graph. When a random directed graph has n nodes which are a part of chaining values, $2n$ edges are sufficient to connect from IV to a given hash value, and the path from IV to the given hash value can take any length if the length is large enough. On the other hand, we know n edges are sufficient to connect to the given hash value with high probability, and we conjectured that there exists paths to the given hash value from about $\sqrt{n}$ nodes. The conjecture is true for the case of random map [5], and we examine the conjecture by computer simulations. Though we only examined that the number of nodes is less than 4096, about $1.1n$ edges make $\sqrt{n}$ nodes connect to the given hash value. Followed by the idea in Section 3.1 with expandable message and above conjecture, a preimage can be computed and its complexity is about half compared with that in [2]. More precisely, the number of nodes in P^3graph is small, a preimage can be computed more efficiently.