Fast Password Recovery Attack: Application to APOP

Fanbao Liu[1,2], Yi Liu[2], Tao Xie[1] and Yumeng Feng[2]

[1] School of Computer, National University of Defense Technology, Changsha, 410073, Hunan, P. R. China
[2] School of Computer, Beijing University of Technology, 100124, Beijing, P. R. China
liufanbao@gmail.com

Abstract. In this paper, we propose a fast password recovery attack to APOP application in local which can recover a password with 11 characters in less than one minute, recover a password with 31 characters extremely fast, about 4 minutes, and for 43 characters in practical time. These attacks truly simulate the practical password recovery attacks launched by *malware* in real life, and further confirm that the security of APOP is totally broken. To achieve these dramatical improvements, we propose a group satisfaction scheme, apply the divide-and-conquer strategy and a new suitable MD5 collision attack to greatly reduce the computational complexity in collision searching with high number of chosen bits. The average time of generating an "*IV Bridge*" is optimized to 0.17 second on ordinary PC, the average time of generating collision pairs for recovering passwords up to 11 characters is about 0.08 second, for 31 characters is about 0.15 second, for 39 characters is about 4.13 seconds, for 43 characters is about 20 seconds, and collisions for recovering passwords as long as 67 characters can be theoretically generated. These techniques can be further applied to reduce the complexity of producing a 1-bit-free collisions for recovering the first 11 characters, whose main target is that to reduce the number of challenges generated in APOP attack, to about 2^{36} MD5 compressions.

Keywords: MD5, APOP, Challenge and Response, Password Recovery, Group Satisfaction Scheme, Divide-and-Conquer, Collision Attack.

1 Introduction

MD5 [10] is one of the most widely used cryptographic hash functions, which was designed by Ron Rivest as the strengthen version of MD4 [9] in 1991. It has been widely applied in the areas of checksum, password shadow, challenge and response authentication protocol and digital signature, et al. Along with the widespread application, some weaknesses of MD5 have been found, for example, den Boer and Bosselaers found that a kind of pseudo-collision, which is also known as dBB collision, could be generated when a pair of initial chaining variables (*IVs*) exists a specific difference [2], in 1993. However, not until Wang

and Yu introduced the first real MD5 collision attack in 2004 [19], the security of MD5 and its applications was thought to be OK.

APOP (Authentication Post Office Protocol) [6], which has been widely employed in real POP3 (Post Office Protocol-Version 3) mail systems to provide origin identification and replay protection, is a challenge and response password authentication protocol based on MD5. The authentication involves two parties, an authenticator and a peer. The authenticator, who holds mails, sends a random and unique challenge string to the peer who wants to access his mail. After that, the peer generates a response by hashing the received challenge concatenated with the shared password and sends it to the authenticator. Finally, the authenticator redoes the hashing using the peer's password stored in his side, checks if it equates the received response to accomplish the peer verification.

In 1996, Preneel and van Oorschot showed how to apply divide-and-conquer strategy, which is based on hash collisions, to key recovery attack against Envelop MAC [8]. They generated collisions using birthday paradox, which was computation infeasible for the widely used hash functions like MD5 [10] and SHA-1 [3]. However, not long after the collision resistance of some prevalent hash functions was broken by Wang et al. [19, 17, 20, 18] and the collision searching efficiency was greatly improved [4], a serial of password recovery attacks on APOP [5, 12, 11, 15], all of which are based on chosen challenge attack, have been proposed in succession.

The best known result of the APOP attack [11] applied dBB collision proposed by den Boer and Bosselaers [2] to achieve more message freedom for recovering passwords. It claimed that collisions for recovering up to the first 11 characters of APOP passwords could be produced in less than one second, collisions for the first 31 characters could be produced in practical time. However, in order to apply dBB collision strategy, one additional "*IV Bridge*", which produces the intended differences of the initial chaining variables needed by dBB collision, must be first generated with computational complexity of 2^{42} MD5 compressions. In [15], the concept of bit-free collision, a pair of partially-fixed messages will collide regardless of the value of unfixed bits, was proposed to reduce the number of challenges generated in APOP attacks. The 1-bit-free collision for recovering the first 11 characters can be generated in 2^{46} MD5 compressions to reduce half of challenges.

Our Contributions. Based on tunnel and advanced message modification, we propose a group satisfaction scheme, which aims to determinately satisfy all conditions of the first three successive steps in the last tunnel, to further improve the collision searching efficiency.

We apply the combination of the divide-and-conquer strategy and the group satisfaction scheme to greatly reduce the computational complexity in MD5 collision searching with high number of chosen bits. As a result, the average time of generating collisions for recovering the first 11 characters of APOP password is about 0.08 second, for the first 31 characters about 0.15 second, for 39 characters about 4.13 seconds, for 43 characters about 20 seconds, and 67 characters can also be theoretically solved. Further, these techniques can be applied to reduce

the complexity of producing a 1-bit-free collision for recovering the first 11 characters, whose target is to reduce the number of challenges generated in APOP attacks, to about 2^{36} MD5 compressions. We also apply a new "*IV Bridge*" with complexity of 2^{19} MD5 compressions, which means it can be generated about 0.17 second in average.

We propose a really fast password recovery attack to APOP application in local PC that can recover a password up to 11 characters in less than one minute, recover password of 31 characters about 4 minutes, and for 43 characters in practical time. These attacks truly simulate the practical password recovery attacks launched by *malware* in real life, and further confirm that the security of APOP is totally broken, and the security of MD5 based applications should be seriously re-evaluated.

Organization of the Paper. The rest of this paper is organized as follows. In section two, we introduce some backgrounds of MD5 algorithm and APOP application. In section three, we present some related works about MD5 collision attack and password recovery attack to APOP. In section four, we apply both of the divide-and-conquer strategy and the group satisfaction scheme to achieve fast collisions searching with high number of chosen bits. In section five, we launch a really fast password recovery attack to APOP in local. We summarize the paper in the last section.

2 Backgrounds

2.1 MD5 Algorithm

MD5 [10] is a typical Merkle-Damgård structure hash function, it takes a variable-length message M as input and outputs a 128-bit hash value $MD5(M)$. First, M is padded with padding bits and the length of M, to the exact multiples of 512 bits. Then, the padded M' is divided into chunks of 512-bit blocks $(M_0, M_1, \ldots, M_{n-1})$. Finally, each block M_i is further divided into sixteen 32-bit words $(m_0, m_1, \ldots, m_{15})$.

MD5 Compression Function. Each block is processed by MD5 compression function (CF). CF takes M_i and a 128-bit chaining variable H_i as input, and outputs H_{i+1}. The initiate chaining variable H_0 is set to certain constants, $a_0 =$ `0x67452301`, $b_0 =$ `0xefcdab89`, $c_0 =$ `0x98badcfe`, $d_0 =$ `0x10325476`. The iterated procedure of MD5 algorithm is shown as follows, where H_n is the exact $MD5(M)$.

$$H_1 = CF(M_0, H_0), H_2 = CF(M_1, H_1), \ldots, H_n = CF(M_{n-1}, H_{n-1}).$$

CF consists of 64 steps. Steps 1-16, steps 17-32, steps 33-48 and step 49-64 are called the first, second, third and fourth round, respectively. Let Q_i represent the 32-bit state of step i, and $Q_{i,j}$ stand for the value of the j-th bit of Q_i. With initiated chaining variables $Q_{-3} = a_0$, $Q_0 = b_0$, $Q_{-1} = c_0$, $Q_{-2} = d_0$, $Q_i (1 \leq i \leq 64)$ is updated as follow.

$$Q_i = Q_{i-1} + (Q_{i-4} + f_i(Q_{i-1}, Q_{i-2}, Q_{i-3}) + w_i + t_i) \lll s_i$$

Each Q_i uses modular addition $+$, left rotation $\lll$ and round dependant Boolean function f_i. w_i is one of $(m_0, m_1, \ldots, m_{15})$. The details of f_i and w_i are shown as follows.

$$f_i = \begin{cases} (X \wedge Y) \vee (\neg X \wedge Z), & 1 \leq i \leq 16, \\ (X \wedge Z) \vee (Y \wedge \neg Z), & 17 \leq i \leq 32, \\ X \oplus Y \oplus Z, & 33 \leq i \leq 48, \\ Y \oplus (X \vee \neg Z), & 49 \leq i \leq 64. \end{cases}$$

$$w_i = \begin{cases} m_{i-1}, & 1 \leq i \leq 16, \\ m_{(5i-4)\ mod\ 16}, & 17 \leq i \leq 32, \\ m_{(3i+2)\ mod\ 16}, & 33 \leq i \leq 48, \\ m_{7(i-1)\ mod\ 16}, & 49 \leq i \leq 64. \end{cases}$$

$\oplus$, $\wedge$, $\vee$ and $\neg$ denote the logic operations *XOR*, *AND*, *OR* and *NOT*, respectively. X, Y and Z are 32-bit words.

The constant t_i is defined as $t_i = \lfloor 2^{32} \cdot |sin(i)| \rfloor$. $\lll s_i$ denotes left rotation by s_i bits, and $\ggg$ denotes right rotation, the details of s_i is shown as follow.

$$(s_i, s_{i+1}, s_{i+2}, s_{i+3}) = \begin{cases} (7, 12, 17, 22), & i = 1, 5, 9, 13, \\ (5, 9, 14, 20), & i = 17, 21, 25, 29, \\ (4, 11, 16, 23), & i = 33, 37, 41, 45, \\ (6, 10, 15, 21), & i = 49, 53, 57, 61. \end{cases}$$

If all steps are computed, the chaining variables are updated by adding the last four state words, which is shown as follows.

$$H_{i+1} = CF(M_i,\ H_i) = (Q_{-3} + Q_{61},\ Q_0 + Q_{64},\ Q_{-1} + Q_{63},\ Q_{-2} + Q_{62})$$

2.2 Properties of MD5 Collisions

If two arbitrary distinct messages $m, m' \in \{0,1\}^*$ satisfy $MD5(m) = MD5(m')$, then m and m' are called one collision pair. Let $|m|$, the length of m, be multiples of 512 bits, and $|m|=|m'|$ further, then for an arbitrary message x, $MD5(m\|x) = MD5(m'\|x)$ always holds, this equation is the base of the password recovery attack to APOP, which will be discussed in detail later.

2.3 APOP(Authenticated Post Office Protocol) Algorithm

APOP is a popular hash-based challenge and response authentication protocol, it is used by a mail server to authenticate the user who tries to access his e-mails stored on the server. The APOP is an optional command of the Post Office Protocol Version 3 [6] to provide extra security that to avoid sending the password in clear text over untrusted network. Let the pre-shared password be *passwd*, for mail accessing is always triggered by the user, the work routine of APOP is shown as follows.

1. The user sends a connection signal to the server.
2. The server generates a challenge nonce C(formatted as a message identifier), which is a random string with some restrictions shown later, and sends C to the user.
3. The user concatenates his password to C, and computes a response $R = MD5(C\|passwd)$, then sends R to the server.
4. Once a respond R is received, the server computes $R_S = MD5(C\|passwd)$ in the same way using the user's password stored in his side. It authenticates the user by comparing whether R_S equates R.

Advantage. With challenge and response, an eavesdropper will learn nothing about the password, provided that MD5 is a partial one-way function. Since the challenge C is randomly generated, this tactic can also provide replay protection for POP3 session.

Restrictions placed on APOP challenge strings.

1. Challenge must start from character '<'(`0x3c` in Hex) and include one character '@'(`0x40`).
2. Challenge must end with character '>'(`0x3e`).
3. Challenge must not include 'NULL'(`0x0`), '\n'(`0x0a`) and '<', '>' in the middle of the text.

3 Related Works

3.1 MD5 Collision Attack

Recently, the attacks on MD5 mostly focus on the collision attack, which is basically a differential attack first presented by Biham and Shamir for cryptanalysis of block ciphers [1].

dBB Collision Found by den Boer and Bosselaers. In 1993, a kind of pseudo-collision, known as dBB collision, was found by den Boer and Bosselaers [2]. If the initial chaining variables have fixed differences, then a message M may collide under those two variables. There are totally 47 conditions to be satisfied for dBB collision, $Q_{i,31} = Q_{0,31}$, $\{i \neq 64,\ i \in (round\,1, round\,2, round\,4)\}$. The authors reduced the complexity and found a dBB collision about 4 minutes with a 33MHz 80386 based PC. The fixed difference of H_0 is shown as follow.

$$\Delta H_0 = (H_0' - H_0) = \pm(2^{31},\ 2^{31},\ 2^{31},\ 2^{31})$$

$-$ denotes 2^{32} modular subtraction. Four $\pm$ mean that the signs of the four word differences of the chaining variables are just the same.

Collision Attack Presented by Wang et al. In 2004, Wang et al. showed the first real collision attack on MD5 [16, 19], and claimed that collisions could be generated with 2^{39} MD5 compressions. The collision consists of two blocks, which have input differences $\Delta M_0 = (M_0' - M_0) = (\Delta m_4 = 2^{31},\ \Delta m_{11} = 2^{15},\ \Delta m_{14} = 2^{31})$, and $\Delta M_1 = (\Delta m_4 = 2^{31},\ \Delta m_{11} = -2^{15},\ \Delta m_{14} = 2^{31})$, respectively. The main strategy of collision attack is shown as follows.

1. Choose an appropriate input difference (ΔM) that can efficiently generate a collision, according to some known knowledge[1].
2. Construct a differential path and deduce sufficient conditions, according to differential propagation and the properties of the Boolean functions.
3. Construct an optimized collision searching algorithm, applying some message freedom which can be used for efficient message modifications to satisfy sufficient conditions.

Collision Attacks Presented by Xie et al. In both 2008 and 2009, Xie et al. presented a series of collision attacks on MD5 [23, 24, 22], with details of how to choose input difference, and listed a lot of input difference patterns that may cause collision. Further, they greatly improved the efficiency of collision attack to about 2^{10} MD5 compressions [22], applying the divide-and-conquer strategy. In 2010, Xie et al. [21] proposed a single block MD5 collision, without no details.

Some Improved Collision Attacks. In 2006, Klima proposed the concept of tunnel [4], which is a efficient message modification technology, and greatly improved Wang's attack [19] to get a collision in one minute. The next year, Stevens further improved it to $2^{24.8}$ MD5 compressions [13]. In 2009, Stevens et al. claimed a collision attack with 2^{16} MD5 compressions [14].

3.2 Message Modification Technology

In order to search collisions efficiently, we need to employ some message modification technologies to satisfy the huge number of sufficient conditions, which hold the differential path. For example, in Wang et al.'s [19], there are more than 200 sufficient conditions in the first block, and after optimization, the least computational time is about $2^{24.8}$ MD5 compressions [13].

Basic and Advanced Message Modification. The basic and the advanced message modification in MD5 were both proposed by Wang et al. [19]. For the full control over $(m_0, ..., m_{15})$, the basic message modification can be applied like that we directly set the sufficient conditions of the first round to be satisfied, then compute the value of related w_i through Q_i shown as follow.

$$m_{i-1} = w_i = (Q_i - Q_{i-1}) \ggg s_i - Q_{i-4} - f_i(Q_{i-1}, Q_{i-2}, Q_{i-3}) - t_i$$

The advanced message modification is used to satisfy the sufficient conditions of the second round, by modifying related message words in the first round. For example, if the condition $Q_{25,31} = Q_{0,31}$ is not satisfied, we need to modify $m_{9,26}$, or introduce a difference of 2^{26}, according to the Q_{25} updating shown as follow.

$$Q_{25} = Q_{24} + (Q_{21} + f_{22}(Q_{24}, Q_{23}, Q_{22}) + (\underline{w_{25} = m_9}) + t_{25}) \lll 5$$

If we just modify m_9, the value of Q_{10} to Q_{14} in the first round will be changed, too. This will eventually destroy the differential path before Q_{25}. To

[1] However, the detail of this knowledge was not disclosed in the origin paper [19].

overcome this drawback, we keep the value of Q_{11} to Q_{14} unchanged, and recompute the related value of m_{10} to m_{13}, which are shown as belows. But there a more serious obstacle comes, we notice that the value of m_{10} and m_{11} have already been used in Q_{22} and Q_{19} in the second round. If modify m_{10} or m_{11} to hold the differential path of the first round, the path of the second round will be destroyed, which means that our former efforts are nonsense.

$$
\begin{aligned}
Q_9 &= Q_8 + (Q_5 + f_9(Q_8, Q_7, Q_6) + (w_9 = m_8) + t_9) \lll 7 \\
\underline{Q_{10}} &= Q_9 + (Q_6 + f_{10}(Q_9, Q_8, Q_7) + (w_{10} = \underline{m_9}) + t_{10}) \lll 12 \\
Q_{11} &= \underline{Q_{10}} + (Q_7 + f_{11}(\underline{Q_{10}}, Q_9, Q_8) + (w_{11} = \underline{m_{10}}) + t_{11}) \lll 17 \\
Q_{12} &= Q_{11} + (Q_8 + f_{12}(Q_{11}, \underline{Q_{10}}, Q_9) + (w_{12} = \underline{m_{11}}) + t_{12}) \lll 22 \\
Q_{13} &= Q_{12} + (Q_9 + f_{13}(Q_{12}, Q_{11}, \underline{Q_{10}}) + (w_{13} = \underline{m_{12}}) + t_{13}) \lll 7 \\
Q_{14} &= Q_{13} + (\underline{Q_{10}} + f_{14}(Q_{13}, Q_{12}, Q_{11}) + (w_{14} = \underline{m_{13}}) + t_{14}) \lll 12
\end{aligned}
$$

To find a way out, we notice that m_8 will be used after Q_{25}, in Q_{28}, which means that we can modify Q_9 to change m_9 indirectly. If flip the value of $m_{9,26}$ in Q_{10} equation, a difference 2^6 will come out in Q_{10} after rotation. So, we flip the value of $Q_{9,6}$, and set an extra condition $Q_{8,6} = Q_{7,6}$ to keep $f_{10}(Q_{9,6}, Q_{8,6}, Q_{7,6})$ unchanged, and recompute m_9 by the unchanged Q_{10}, then get the exact difference 2^{26}.

$$\underline{m_9} = (Q_{10} - \underline{Q_9}) \ggg 12 - Q_6 - f_{10}(Q_9, Q_8, Q_7) - t_{10}$$

Further, we set another extra condition $Q_{10,6} = 0$ to keep both Q_{11} and m_{10} unchanged while updating Q_{11}. Similarly, we set $Q_{11,6} = 1$ to keep both Q_{12} and m_{11} unchanged. Finally, we recompute m_{12} to keep Q_{13} unchanged. After these steps, we successfully keep the differential path before Q_{25} unchanged, and hold the condition $Q_{25,31} = Q_{0,31}$ with probability 1.

Tunnel. The concept of tunnel was first introduced by Klima [4]. The essence of tunnel is that it employs the free bits to do not only the brute force searching, but also move the start point of searching forward. For example, in the case of dBB collision searching, there are conditions $Q_{10,31} = Q_{0,31}$ in Q_{10}, and $Q_{9,31} = Q_{0,31}$ in Q_9, which means that the value of $Q_{9,31}$ can't be changed. Excluding $Q_{9,6}$ reserved for the advanced message modification, there are 30 free bits left in Q_9. If we set extra conditions $Q_{10} = 2^{31} \times (1 \wedge Q_{0,31}) + \texttt{0x0}$ and $Q_{11} = 2^{31} \times (1 \wedge Q_{0,31}) + \texttt{0x7fffffff}$. In step 25, we can flip any free bits of Q_9 to start a new searching without destroying the differential path before step 25. After step 25, there are only 22 conditions to be randomly satisfied, which means that we don't need to go back to step 1 for a successful collision. The tactic of tunnel greatly improve the collision searching efficiency.

Besides the Q_9 tunnel used in step 25, the Q_4 tunnel can be used in step 24. We also propose new Q_2 and Q_1 tunnels used in step 20 and step 17, respectively, for efficient meaningful collision searching. For the lack of space, we omit the details.

Submarine Modification. Naito et al. proposed a submarine modification [7] to improve the collision searching efficiency of SHA-0. They united both

the basic and advanced message modifications based on SHA-0 to satisfy the conditions after step 21.

3.3 APOP Password Recovery Attack

Client uses his mail account and password to login on the mail server, and access his mail. The account is always transferred in clear text, and the only way to keep his mail account safe is the password. The APOP is designed to do not leak the password in the untrusted network. Still, there are some ways to attack the password, such as social engineering, Trojan horse, brute force password recovery, dictionary attack. Here, we discuss the pure technological methods to recover APOP password.

Brute Force password Recovery. The brute force password recovery attack is direct and simple but not easy to be accomplished. In a extreme case, if the password consists of one bit, it may be '0' or '1', we need only one guess, whether it is right or wrong, to find the right answer. Actually, there are 52 letters and 10 numbers in total (we simply assume that the punctuations are excluded). If the password is only one character long, applying the brute force attack, the attacker needs to try 61 times in the worst case before he gets the right answer. When the password is 8 characters long, it may try $62^8 - 1 \approx 2^{47}$ times, which is too hard for the ordinary PC. Generally, passwords less than 8 characters are thought to be weak.

Divide-and-Conquer Strategy. The divide-and-conquer strategy is a well-known method to tackle problems with some inner independence. It was also introduced to improve the MD5 collision attack by Xie et al. in [24], by designing optimized differential path with huge message freedom supporting tunnels. The strategy is shown as follows.

If a problem P with computational complexity C_0 can be divided into k sub-problems $P_i (1 \leq i \leq k)$, and each sub-problem P_i can be solved independently with a sub-computational complexity C_i but without destroying other sub-problems' solutions, then the computational complexity to solve the original problem is reduced to $C_d = \sum_{i=1}^{k} C_i$ instead of $C_0 = \prod_{i=1}^{k} C_i$ [24].

Divide-and-Conquer Strategy to APOP Attack. If (m, m'), which is multiples of blocks long, is a collision pair, the equation of $MD5(m\|x) = MD5(m'\|x)$ always holds for arbitrary message x. This property was first applied by Preneel and van Oorschot while employing divide-and-conquer strategy to attack the Envelop MAC bit by bit [8]. The principle is that, first, we guess that the first *unknown* bit of the password is b, then generate a collision pair (m,m') which includes b at the end; second, we remove b from m and m', and send both as challenges; finally, if the both received responses equate, then the guess of b is correct. If apply the divide-and-conquer strategy bit by bit to a password of 64 bits, we need to try 64 times, each includes collision pair generation, to recover it, not the former brute force attack of $2^{64} - 1$ times.

In the case of APOP, the challenge must be multiples of characters. So we apply the divide-and-conquer strategy to APOP password recovery character by character, which is shown as follows.

1. Generate a collision pair (m, m') that the last part of it is the password candidate *cand* which includes the *known* part and the exact *guessing* character l at the tail. Let $C = m - cand$ and $C' = m' - cand$, send the C as challenge to the user.
2. Receive a response R, and send C' as another challenge to the user.
3. Receive a response R', and compare whether $R = R'$, if the equation holds, the *guessing* l is the very answer, go to the next *guessing* l'; else go to step 1 for another try.

Previous Password recovery Attacks to APOP

In 2007, Leurent [5] and Sasaki et al. [12] independently presented APOP password recovery attack which could recover the first 3 characters of the password, using Wang et al.'s MD5 collision attack. The cause of limitation is that Wang et al.'s attack needs input difference on the *MSB* bit of m_{14}, and MD5 process uses the little-endian, which leads to that only m_{15} can be used[2] for the recovery attack.

In 2008, Sasaki et al. [11] improved the attack to recover more characters by using an "*IV Bridge*", which aims to apply dBB collision and achieve great message controlling, and reduced the time of producing one collision pair for recovering the first 11 characters to less than one second. But the limitation is that it has to cost about 2^{42} MD5 compressions to construct an "*IV Bridge*" first.

In 2009, Wang et al. [15] improved the attack further by employing "*bit-free collision*" to reduce the number of chosen challenges. The *1-bit-free collision* means that if a origin MD5 collision pair can be generated in 2^t MD5 compressions, then *1-bit-free collision* can be generated in 2^{2t} MD5 compressions. They successfully reduced half of the challenges by generating *1-bit-free collision* for recovering the first 11 characters.

4 Collisions Searching with High Number of Chosen Bits

To apply the divide-and-conquer strategy against APOP password recovery attack, the introduced problem of generating MD5 collisions with high number of chosen bits must be solved efficiently. In this section, we propose a group satisfaction scheme to improve collision searching efficiency. With the employment of both of the group satisfaction scheme and the divide-and-conquer strategy, we implement efficient collision searching with more message controlling over fixed bits. The work routine of the optimized MD5 collision attack with high number of chosen bits is shown as follows.

1. Use the first block with input difference $\Delta M_0 = (\Delta m_8 = 2^{31})$, which was proposed by Xie et al. [24], as "*IV Bridge*". It goes through the initial chaining variable $\Delta H_0 = 0$ and gets the output of intended chaining variables

[2] The challenge C must end with a '>', this means that only 3 characters can be used.

with difference $\Delta H_1 = \pm(2^{31}, 2^{31}, 2^{31}, 2^{31})$, which is needed by den Boer et al.'s dBB collision.
2. Use dBB collision which needs input difference $\Delta M_1 = 0$, and apply the group satisfaction scheme and the divide-and-conquer strategy, to achieve great message controlling.

Group Satisfaction Scheme *Based on both of the tunnel and the advanced message modification, we determinately satisfy all of the conditions of the first three steps, in the last tunnel or the last phase of the divide-and-conquer strategy, to achieve full speed of collision searching.*

Advantage. Most of the former collision searching algorithms [4, 13], intended to satisfy conditions of the beginning step of the last tunnel, but we satisfy all conditions of the first three steps, which can greatly increase the collision probability and improve the searching efficiency.

Apply Q_9 Tunnel to Generate Collisions for Recovering the First 11 Characters. For example, we apply the divide-and-conquer strategy and the group satisfaction scheme to dBB collision searching with full control over 96 bits (*pre-chosen* m_{13}, m_{14}, m_{15}) in the way shown as follows.

1. There are 48 conditions[3] to be satisfied in total, and we can use full Q_9 tunnel at Q_{25}. We divide the collision searching into two phases. The first phase from Q_1 to Q_{24} (with 25 conditions). The second phase from Q_{25}, where Q_9 tunnel can be applied, to the final step of Q_{64} (with 23 conditions).
2. In the first phase we first apply the basic message modification from Q_1 to Q_{13}, after then there are 12 conditions left to be randomly satisfied, we handle it as follows.
   (a) Directly set the condition bits to be satisfied from Q_1 to Q_{13}, and randomly set non-condition bits. (The basic message modification is applied.)
   (b) Compute Q_{14} to Q_{16} orderly using the pre-chosen m_{13} to m_{15}, if any condition is not satisfied, go to step 2a for a new start.
   (c) Compute m_1, m_6, m_{11}, m_0, m_5, m_{10}, m_4, then orderly compute and check Q_{17} to Q_{24}, if any condition is not satisfied, go to step 2a.
   (d) Compute m_2, m_3, m_7, m_8, m_9, m_{12}.
3. After the group satisfaction scheme is applied (there are only 3 conditions in total from Q_{25} to Q_{27}), the second phase has 20 conditions left to be randomly satisfied, but a huge tunnel of 28 free bits.
   (a) We utilize the group satisfaction scheme to satisfy all three conditions of the first three steps of the last phase of the divide-and-conquer strategy. We notice that w_{25}=w_{10}=m_9.
      i. Randomly set the free bits of Q_9, and then re-compute m_9 by the unchanged Q_{10}.

[3] There is one more condition $Q_{23,22} = 1$ added to support the group satisfaction scheme.

ii. Compute Q_{25}, if the condition $Q_{25,31} = Q_{0,31}$ is not satisfied, then flip the value of $Q_{9,6}$, re-compute m_9 by unchanged Q_{10}, then re-compute Q_{25}.
iii. Compute Q_{26}, if the condition $Q_{26,31} = Q_{0,31}$ is not satisfied, then flip the valuer of $Q_{9,29}$, re-compute m_9, then re-compute Q_{25} and Q_{26}.
iv. Compute Q_{27}, if the condition $Q_{27,31} = Q_{0,31}$ is not satisfied, then flip the value of $Q_{9,24}$, re-compute m_9, then re-compute Q_{25}, Q_{26}, Q_{27}.
v. Check again if the three conditions are satisfied, if not, go to step 3(a)i.

(b) Re-compute m_8 from Q_9 and m_{12} from Q_{13}, compute and check from Q_{28} to the final step Q_{64} with early stop, if any condition is not satisfied, go to step 3a.

Complexity Analysis. Applying the divide-and-conquer strategy, the first and second phases have 12 and 20 conditions to be randomly satisfied, respectively. In this dividing, no other problem is introduced, and the start-point[4] of collision searching is moved from Q_1 to step 3a. The complexity of these two phases can be calculated independently, according to the divide-and-conquer strategy. The complexity of phase 1 is about 2^9 MD5 compressions, phase 2 is about 2^{18} MD5 compressions, and the full complexity is about $2^9 + 2^{18} \approx 2^{18}$ MD5 compressions. The searching can be done in average time of 0.08 second on ordinary PC (with intel 2G CPU), which is about 2^4 times faster than Sasaki et al.'s work [11].

Table 1. Complexity Comparison between Previous Works and Ours

Work	*IV Bridge*(block 0)	block 1	1-bit-free collision
Leurent [5]	2^{30}	5 s(3 c)	-
Sasaki et al. [12]	2^{30}	5 s(3 c)	-
Sasaki et al. [11]	2^{42}	within 1 s(11 c)	-
Wang et al. [15]	2^{42}	within 1 s(11 c)	2^{46}
Ours	2^{19}	0.08 s(11 c)	2^{36}

Utilizing both of the divide-and-conquer strategy and the group satisfaction scheme, the complexity of generating an "*IV Bridge*" is reduced to 2^{19} MD5 compressions, about 0.17 second in average time. Since the complexity of searching a collision pair for recovering the first 11 characters is 2^{18} MD5 compressions, and then the complexity of searching a 1-bit-free collision for recovering the first 11 characters is 2^{36} MD5 compressions. The previous works of collision searching

[4] In this situation, we start each collision searching in step 3a with huge message freedom when phase 2 is reached, no need to start from step 2a again.

and our results are compared in Table 1. In the table, s stands for second, c for character.

We also greatly improve the efficiency of collision searching for recovering more characters. We generate a collision pair for recovering the first 31 characters in 2^{19} MD5 compressions (about 0.15 second in average), using Q_4 tunnel in step 24. We generate a collision pair for recovering the first 39 characters in 2^{24} MD5 compressions (about 4.13 seconds in average), using Q_2 tunnel in step 20. We generate a collision pair for recovering the first 43 characters in 2^{26} MD5 compressions (about 20 seconds in average), using Q_1 tunnel in step 17. Applying the message controlling on m_{15} of the "*IV Bridge*"[5], we can also theoretically recover up to the first 67 characters of the password. In Table 2, we compare our results to the previous works of collisions with high number of chosen bits.

Table 2. Comparison of Collision with High Number of Chosen Bits

Work	11 c	31 c	35 c	39 c	43 c	47 c	51 c	theoretic recovery
Sasaki et al. [11]	within 1 s	5.86 s	2^{38}	2^{39}	2^{41}	2^{42}	2^{43}	61 c
Ours	0.08 s	0.15 s	2^{24}	2^{24}	2^{26}	2^{39}	2^{40}	67 c

5 Fast Password Recovery Attack to APOP

In order to implement a successful password recovery attack to APOP, we impersonate the mail server, construct the collided challenge strings and send them to the user when he tries to establish a connect. To avoid the restrictions placed on the challenge, we split the challenge to three parts shown as bellow, and each part is 512 bits long.

1. A block *pre* starts from a '<' and includes an exact '@'.
2. A block *mid* is the exact "*IV Bridge*".
3. A block *end* includes the password candidate *cand* at the end, and an exact '>' before the *cand*.

We construct a pair of challenges $C = pre\|mid\|(end - cand)$ and $C' = pre\|mid'\|(end - cand)$. The only difference between mid and mid' is the value of $m_{8,31}$ which satisfies $m_{8,31}{=}\neg m'_{8,31}$. If the mail user receives both C and C', he would think that these two challenges are totally different. So we can use the C and C' to launch password recovery attack to APOP.

An example of the collision pair, consisting of three blocks, for recovering the first 43 characters of the password is shown in Table 3. The candidate characters are at the end of the third block *end*, and them read as "*In the beginning God created the heaven and*", which are chosen from the beginning of the "*Bible*". The

[5] This message controlling over m_{15} will not influence the tunnel, even to say the divide-and-conquer strategy, so the complexity does not increase.

Table 3. An example collision pair for recovering the first 43 characters

H_0	0x67452301, 0xefcdab89, 0x98badcfe, 0x10325476
pre	0x6568543c, 0x6c6f4820, 0x69422079, 0x2e656c62 0x6e694b20, 0x614a2067, 0x2073656d, 0x73726556 0x2e6e6e69, 0x65685420, 0x646c4f20, 0x73655420 0x656d6174, 0x4720746e, 0x656e6e65, 0x40736973
$H_1 = H'_1$	0x6a8502b, 0xa9db20dd, 0x822f4299, 0x867a7298
mid	0x47409de4, 0xa1b6eced, 0x7e70ed35, 0xd8215bf0 0xd7b040e1, 0x7342a3a5, 0x79d9e8b3, 0xe707954f 0xac618677, 0x388dc5b7, 0xf7b86a0c, 0xa040309a 0x3a617c44, 0x3392ff6c, 0xf803241d, 0x4f3af7a7
mid'	0x47409de4, 0xa1b6eced, 0x7e70ed35, 0xd8215bf0 0xd7b040e1, 0x7342a3a5, 0x79d9e8b3, 0xe707954f 0x2c618677, 0x388dc5b7, 0xf7b86a0c, 0xa040309a 0x3a617c44, 0x3392ff6c, 0xf803241d, 0x4f3af7a7
H_2	0xe6fee770, 0x21018304, 0x15c61b85, 0x7151e9e2
H'_2	0x66fee770, 0xa1018304, 0x95c61b85, 0xf151e9e2
end	0x8d6f618, 0xb5db76d5, 0xa45975e8, 0x2841c42f 0x633f5fa5, 0x206e493e, 0x20656874, 0x69676562 0x6e696e6e, 0x6f472067, 0x72632064, 0x65746165 0x68742064, 0x65682065, 0x6e657661, 0x646e6120
$H_3 = H'_3$	0x2665f3c8, 0x1aa6bb7d, 0x78a0fc27, 0x2d485e7c

pre block reads as "<*The Holy Bible. King James Version. The Old Testament Gennesis@*". The *mid* block is an exact "*IV Bridge*", which is processed to output the intended difference $\Delta H_2 = -(2^{31}, 2^{31}, 2^{31}, 2^{31})$. The time of constructing such a pair is only 20 seconds.

5.1 Fast Local Password Recovery Attack to APOP

We launch a local password recovery attack to APOP, the experiment is based on one old version of open source software "Evolution" which supports APOP. We change the way of mail accessing, which is originally triggered by the user or time intervals, to be triggered by receiving a signal from a specific communication port at the source code level.

We impersonate the mail server(S) in local PC (with intel 2G CPU). U means the mail client. The work routine of the password recovery attack to APOP, character by character, is shown as follows.

1. S generates a challenge pair C and C' including a password candidate *cand* at the end of them, masquerades a signal of mail accessing and sends it to U.
2. U tries to connect S, upon a signal of mail accessing received from specific communication port.
3. S sends the challenge string C to U.
4. U computes $R = MD5(C||passwd)$, and sends it to S.

5. S sends another mail accessing signal to U once R is received.
6. U tries to connect S, upon a received signal.
7. S sends the challenge string C' to U.
8. U computes $R' = MD5(C'\|passwd)$, and sends it to S.
9. S compares whether R equates R', if so, the *cand* is confirmed, and if the password is fully recovered, stops; else, goes to step 1 for another guess of the *cand*.

How to Judge the Password is Fully Recovered. We notice that in step 9 of the work routine, it should be checked that whether the password is fully recovered. We handle this problem in the way shown as follows.

1. We always assume that the password is fully recovered, then the message following C and *passwd* is the exact padding bits.
2. We compute $MD5(C\|cand)$ and compare it with R, if the equation holds, the password is fully recovered. Otherwise, the password still has some *unknown* characters to be recovered.

Practical Analysis. In the experiment, we assume that the local *malware* can change the behavior of the mail client, and in real life, this is totally true; further, most of people prefer his password *being remembered* when he uses the mail client, for that he doesn't need to enter his password every time to check mail, so the experiment actually simulates a potential local password recovery attack to APOP launched by *malware*. We also assume that passwords just consist of letters with lower and upper cases and digits from '0' to '9', this assumption meets the ordinary use of actual life, so we just need to guess 61 but not 255 times to recover one character.

Usually, a password with more than 7 characters, is thought to be safe. We successfully recover passwords whose length ranges from 6 to 31 characters, and we can recover password of 43 characters in practical time. The average time of local attacks to recover passwords with different lengths is shown in Table 4.

Table 4. Average time to recover passwords with different lengths

Length	6 *c*	7 *c*	8 *c*	9 *c*	10 *c*	11 *c*	31 *c*
Time	30.66 *s*	35.74 *s*	40.88 *s*	45.80 *s*	50.95 *s*	55.98 *s*	251.43 *s*

The fast local password recovery attack to APOP shows that APOP is vulnerable to potential attack from some *malwares*, and the security of APOP and other MD5 based applications should be seriously re-evaluated.

6 Conclusion

In this paper, we propose a fast local password recovery attack to APOP application that can recover a password of 11 characters within one minute, the

recovery of 31 characters is also extremely fast, about 4 minutes, and for 43 characters in practical time. Therefore, these results further confirm that the security of APOP is totally broken, and the security of MD5 based applications should be seriously re-evaluated.

We propose a group satisfaction scheme and apply the divide-and-conquer strategy to greatly reduce the complexity of collision searching. The average time of generating collisions for recovering the first 11, and 31 characters is about 2^4 to 2^5 times faster than previous work [11]. The average time of generating collisions for recovering the first 39, and 43 characters is about 2^{15} times faster than [11]. These techniques can be further applied to reduce the complexity of producing 1-bit-free collision for recovering the first 11 characters, to about 2^{36} MD5 compressions.

Acknowledgements. This work was partially supported by the National High Technology Research and Development Program of China (2009AA01Z437).

References

1. Biham, E., Shamir, A.: Differential Cryptanalysis of the Data Encryption Standard. Springer-Verlag (1993)
2. den Boer, B., Bosselaers, A.: Collisions for the Compression Function of MD5. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 293-304. Springer, Heidelberg (1994)
3. FIPS 180-1, Secure Hash Standard. Federal Information Processing Standard (FIPS), Publication 180-1, National Institute of Standards and Technology, US Department of Commerce, Washington D.C., April 17 (1995)
4. Klima, V.: Tunnels in Hash Functions: MD5 Collisions Within a Minute. Cryptology ePrint Archive, Report 2006 /105 (2006)
5. Leurent, G.: Message freedom in MD4 and MD5 collisions: Application to APOP. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 309-328. Springer, Heidelberg (2007)
6. Myers, J., Rose, M.: Post Office Protocol - Version 3. RFC 1939 (Standard), Updated by RFCs 1957, 2449 (1996)
7. Naito, Y., Sasaki, Y., Shimoyama, T., Yajima, J., Kunihiro, N., Ohta, K.: Improved Collision Search for SHA-0. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 21-36. Springer, Heidelberg (2006)
8. Preneel, B., van Oorschot, P.C.: On the Security of Two MAC Algorithms. In: EUROCRYPT, pp. 19-32 (1996)
9. Rivest, R.L., The MD4 message-digest algorithm. Request for comments (RFC 1320), Internet Activities Board, Internet Privacy Task Force (1992)
10. Rivest, R.L., The MD5 message-digest algorithm. Request for comments (RFC 1321), Internet Activities Board, Internet Privacy Task Force (1992)
11. Sasaki, Y., Wang, L., Ohta, K., Kunihiro, N.: Security of MD5 challenge and response: Extension of APOP password recovery attack. In: Malkin, T.G. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 1-18. Springer, Heidelberg (2008)
12. Sasaki, Y., Yamamoto, G., Aoki, K.: Practical Password Recovery on an MD5 Challenge and Response. Cryptology ePrint Archive, Report 2007/101 (2007)

13. Stevens, M.: On Collisions for MD5. Masters Thesis, 2007. TU Eindhoven, Faculty of Mathematics and Computer Science, available at http://www.win.tue.nl/ (2007)
14. Stevens, M., Sotirov, A., Appelbaum, J., Lenstra, A., Molnar, D., Osvik, D., Weger, B.: Short chosen-prefix collisions for MD5 and the creation of a rogue CA certificate. In: Halevi, S.(ed.) CRYPTO 2009, LNCS, vol. 5677, pp. 55-69, Springer, Heidelberg (2009)
15. Wang, L., Sasaki, Y. Sakiyama, K., Ohta, K.: Bit-Free Collision: Application to APOP Attack. In: Takagi, T., Mambo, M.(Ed.) IWSEC 2009, LNCS, vol. 5824, pp. 3-21, Springer, Heidelberg (2009)
16. Wang, X., Feng. D., Lai., X., Yu, H.: Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMD. In: rump session of Crypto04, E-print (2004)
17. Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the Hash Functions MD4 and RIPEMD. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 1-18. Springer, Heidelberg (2005)
18. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17-36. Springer, Heidelberg (2005)
19. 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)
20. Wang, X., Yu, H., Yin, Y.L.: Efficient Collision Search Attacks on SHA-0. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 1-16. Springer, Heidelberg (2005)
21. Xie, T., Feng, D.: Construct MD5 Collisions Using Just A Single Block Of Message. Cryptology ePrint Archive, Report 2010/643 (2010)
22. Xie, T., Feng, D.: How To Find Weak Input Differences For MD5 Collision Attacks. Cryptology ePrint Archive, Report 2009/223 (2009)
23. Xie, T., Feng, D., Liu, F.: A New Collision Differential For MD5 With Its Full Differential Path. Cryptology ePrint Archive, Report 2008/230 (2008)
24. Xie, T., Liu, F., Feng, D.: Could The 1-MSB Input Difference Be The Fastest Collision Attack For MD5? LNCS 5479, the poster session of EUROCRYPT 2009. Cryptology ePrint Archive, Report 2008/391 (2008)