PAPER Special Section on Information and Communication System Security

Cryptanalysis of Two MD5-Based Authentication Protocols: APOP and NMAC*

Lei WANG[†a)], ***Student Member***, **Kazuo OHTA**[†], **Yu SASAKI**[†,††], **Kazuo SAKIYAMA**[†], ***and*** **Noboru KUNIHIRO**[†††], ***Members***

SUMMARY Many hash-based authentication protocols have been proposed, and proven secure assuming that underlying hash functions are secure. On the other hand, if a hash function compromises, the security of authentication protocols based on this hash function becomes unclear. Therefore, it is significantly important to verify the security of hash-based protocols when a hash function is broken.

In this paper, we will re-evaluate the security of two MD5-based authentication protocols based on a fact that MD5 cannot satisfy a required fundamental property named *collision resistance*. The target protocols are APOP (Authenticated Post Office Protocol) and NMAC (Nested Message Authentication Code), since they or their variants are widely used in real world. For security evaluation of APOP, we will propose a modified password recovery attack procedure, which is twice as fast as previous attacks. Moreover, our attack is more realistic, as the probability of being detected is lower than that of previous attacks. For security evaluation of MD5-based NMAC, we will propose a new key-recovery attack procedure, which has a complexity lower than that of previous attack. The complexity of our attack is 2^{76}, while that of previous attack is 2^{100}.[**]Moreover, our attack has another interesting point. NMAC has two keys: the inner key and the outer key. Our attack can recover the outer key partially without the knowledge of the inner key.

key words: *authentication protocol, APOP, NMAC, MD5*

1. Introduction

Authentication is the act of confirming someone or something, such as the identity of a person or the integrity of a message. The authentication action is indispensable, especially when the two interacting parties are communicating with each other through Network. One popular approach to design an authentication protocols is utilizing a *hash function*. Hash function compresses arbitrary-length messages into fixed-length short random values. The hash-based authentication protocols have been proven secure based on the assumption that the underlying hash functions are secure. On the other hand, if a dedicated hash function compromises, the security of the authentication protocols based on this hash function will become unclear. Therefore, it is significantly important to verify the security of the hash-based authentication protocols when their underlying hash functions are broken.

Recently cryptographers pay a lot of attention to cryptanalysis of hash functions. Weakness of many hash functions has been revealed [14], [15]. It leads to the necessity of re-analyzing the hash-based protocols implemented with a dedicated hash function, which compromised. This is the main motivation of our research.

One of the most popular hash functions is MD5, which was broken in 2005 [14]. We will deal with two authentication protocols based on MD5: APOP [8] and NMAC [1].

APOP is an authentication command for the server to authenticate the users in the Post Office Protocol Version 3 (POP3) [8], which is implemented for a local user to retrieve his emails from a remote server over the Internet channel. Interestingly, quoting to [8], APOP is defined with MD5 as its underlying hash function.

NMAC, proposed in [1], does not have any application. However, its variant HMAC (Keyed-Hashing for Message Authentication Code), proposed in [1], has been applied in many protocols such as TLS (Transport Layer Security) and IPSec (Internet Protocol Security). MD5-based HMAC (HMAC-MD5 for simplicity) has been standardized and implemented in many Internet protocols. We stress that the security of NMAC is the foundation of the security of HMAC: the security proof of HMAC is based on the assumption that NMAC is secure [1], [2]. If MD5-based NMAC (NMAC-MD5 for simplicity) is insecure, the security proof of HMAC-MD5 will collapse. Consequently the security of the protocols based on HMAC-MD5 will become unclear. Therefore cryptanalysis of NMAC-MD5 is significantly important, even though it does not have any practical application.

1.1 Our Results

We will attack APOP and NMAC-MD5 by utilizing the weakness of MD5.

For APOP, we will propose a modified password recovery attack. Previous password recovery attacks are *chosen-challenge* attacks, which impersonate the server to send chosen challenges to the user. Our attack procedure adopts the same strategy and scenario. Our main contribution is half-reducing the number of necessary impersonations. Namely our attack is twice as faster as

Manuscript received August 3, 2009.
Manuscript revised December 18, 2009.

[†]The authors are with the University of Electro-Communications, Chofu-shi, 182–8585 Japan.
[††]The author is with NTT Information Sharing Platform Laboratories, NTT Corporation, Musashino-shi, 180–8585 Japan.
[†††]The author is with the University of Tokyo, Kashiwa-shi, 277–8561 Japan.

[*]This paper has been partially presented in IWSEC2009 and in EUROCRYPT2008. Refer to [13] and [12].

a) E-mail: wanglei@ice.uec.ac.jp

DOI: 10.1587/transinf.E93.D.1087

[**]We follow the complexity evaluation method in [4].

previous attacks [7], [10], [11]. Moreover, during impersonating the server, the attack has a risk of being detected. Therefore reducing the number of impersonations will lower the probability of the attack being detected, which makes the attack more realistic.

For NMAC-MD5, there are two secret keys: the inner key and the outer key. Our main contribution is a new attack procedure to recover the outer key. By combining previous inner-key recovery attack [4], we can obtain a new full-key recovery attack on NMAC-MD5. Our attack can recover the full key within 2^{75} online queries and 2^{75} offline MD5 computations, while previous attack needs 2^{28} online queries and 2^{100} offline MD5 computations [6]. Following the evaluation method of the total complexity in [4], which is the sum of online queries and offline complexity, the complexity of our attack is 2^{76}, while that of previous attack is 2^{100}. Of independent interest, our outer-key recovery attack can recover the outer key without the knowledge of the inner key. Finally we emphasize that our attack reveals a new weakness of NMAC-MD5, which is based on the feed-forward operation in compression function of MD5. Many other hash functions such as SHA-1 have similar structure. Therefore our attack procedure might be applicable to SHA-1-based NMAC.

1.2 Related Previous Attacks

For APOP, Leurent published a password recovery attack [7] by utilizing Wang *et al.*'s collision attacks on MD5 [14], which can recover 3 password characters. This was independently found by Sasaki *et al.* [10]. Sasaki *et al.* published an improved password recovery attack [11] by utilizing Den Boer *et al.*'s collision attack on MD5 [3], which can recover 31 password characters. For NMAC, Contini *et al.* proposed an inner-key-recovery attack on NMAC based on several hash functions including MD5 [4]. Fouque *et al.* proposed an outer-key-recovery attack on NMAC with several hash functions including MD5, which can be extended to a full-key-recovery attack by combining [4].

2. Background and Related Works

2.1 MD5 and Related Attacks on MD5

2.1.1 MD5 Algorithm

MD5 [9] follows Merkle-Damgård structure. An input message M will be padded and divided into 512-bit blocks $\{M_1, \ldots, M_l\}$. The padding rule is first adding a single '1' and the minimum number of '0's until the bit length becomes 448 on modulo 512, and then adding the bit length of original M to the last 64 bits. The message blocks will be hashed sequentially by a primitive usually described as *compression function*. Hereafter we will denote by *md5* the compression function of MD5. md5 is a mapping function: $\{0,1\}^{128} \times \{0,1\}^{512} \rightarrow \{0,1\}^{128}$. The MD5 procedure is as follows:

$h_0 \leftarrow$ a public initial value IV;
$h_i \leftarrow md5(h_{i-1}, M_i)$.

md5 function.
M_i and h_{i-1} will be divided into 32-bit variables $(m_0, \ldots, m_{15})$ and (a_0, b_0, c_0, d_0) respectively. md5 consists of 64 steps, regrouped into four 16-step rounds. Each step is as follows:

$a_i = d_{i-1}$, $c_i = b_{i-1}$, $d_i = c_{i-1}$, and
$b_i = b_{i-1} + (a_{i-1} + f(b_{i-1}, c_{i-1}, d_{i-1}) + m_k + t) \lll s_i$,
where m_k is one of $(m_0, \ldots, m_{15})$, t is a constant defined at each step, $\lll s_i$ means a left-rotation by s_i bits, and f is a Boolean function depending on the round. The details of these parameters are shown in Table 1. The final output is as follows:

$$(a_0 + a_{64}, b_0 + b_{64}, c_0 + c_{64}, d_0 + d_{64}).$$

Table 1 Md5 algorithm.

1 ~ 16 steps	$f(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z)$ k: (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15)
17 ~ 32 steps	$f(X,Y,Z) = (X \wedge Z) \vee (Y \wedge \neg Z)$ k: (1, 6, 11, 0, 5, 10, 15, 4, 9, 14, 3, 8, 13, 2, 7, 12)
33 ~ 48 steps	$f(X,Y,Z) = X \oplus Y \oplus Z$ k: (5, 8, 11, 14, 1, 4, 7, 10, 13, 0, 3, 6, 9, 12, 15, 2)
49 ~ 64 steps	$f(X,Y,Z) = (X \vee \neg Z) \oplus Y$ k: (0, 7, 14, 5, 12, 3, 10, 1, 8, 15, 6, 13, 4, 11, 2, 9)

2.1.2 Collision Attacks on Md5

A collision attack on md5 has been proposed by Den Boer and Bosselaers in [3]: $md5(h, M)$=$md5(h', M)$, where the differences only exist between h and h'. Moreover, the difference between h and h' has a special form. From the specification of md5, h and h' will be divided into (a_0, b_0, c_0, d_0) and (a'_0, b'_0, c'_0, d'_0) respectively. The differences between h and h' have to satisfy
$(a_0 \oplus a'_0, b_0 \oplus b'_0, c_0 \oplus c'_0, d_0 \oplus d'_0) =$
$(0x80000000, 0x80000000, 0x80000000, 0x80000000)$,
which will be denoted as Δ^{MSB} hereafter for simplicity. Moreover, the MSBs of b_0, c_0 and d_0 should be equal.

For random h and M, the probability that $md5(h, M) = md5(h \oplus \Delta^{MSB}, M)$ is 2^{-46}, because 46 conditions have to be satisfied during the computation $md5(h, M)$. These conditions are as follows:

1∼14 steps: $b_{i,32} = b_{i-1,32}$;
16∼31 steps: $b_{i,32} = b_{i-1,32}$;
49∼63 steps: $b_{i,32} = b_{i-2,32}$,
where $b_{i,j}$ means j-th bit of b_i.

2.1.3 IV Bridge

Collision attacks on md5 in Sect. 2.1.2 can not be trivially extended to attack MD5. The main reason is that IV of MD5

Table 2 Tunnel Q9.

step index	message	conditions
9	m_9	$b_{10} = 0xffffffff$;
10	m_{10}	$b_{11} = 0x00000000$;

has been fixed. The attacker has to generate two intermediate hash values h_i and h'_i, which have XOR difference Δ^{MSB}. Towards this problem, Sasaki *et al.* proposed a technique named *IV bridge* [11]. Conceptually, IV bridge is a pair of messages M and M' which link IV to a pair intermediate hash values h and h' satisfying $h \oplus h' = \Delta^{MSB}$. Sasaki *et al.* provided one IV bridge [11], which has been detailed in Appendix A. We will utilize their IV bridge for improving the previous attacks on APOP.

2.1.4 Tunnel Technique

Klima proposed a technique named *Tunnel* [5], which can speed up searching collisions on MD5. Here we will only describe "Q9 Tunnel", which will be used in our improved attacks on APOP.

Q9 tunnel, detailed in Table 2, is a local collision inside md5 from the 8-th step until the 12-th step: for any m_8 at the 8-th step, there are corresponding m_9 and m_{12} that make the intermediate chaining variables at the 12-th step remain the same. Moreover, m_8, m_9 and m_{12} locate at steps 28, 25 and 32 in the second round respectively. Therefore all the intermediate chaining variables from the 12-th step until the 24-th step will also remain the same. During searching a collision on md5, by setting Q9 tunnel, after a message satisfying all the conditions from the first step until the 24-th step is found, we will change its m_8 freely, then adaptively compute m_9 and m_{12} without breaking the intermediate chaining variables until the 24-th step. For more details, refer to [5].

2.2 APOP and Related Attacks on APOP

2.2.1 APOP

APOP is a MD5-based challenge and response authentication protocol, which is used in POP3 [8] by mail servers to authenticate users. The procedure of APOP is detailed as follows.

1. The user sends an access request to the server.
2. The server sends a random challenge to the user.
3. The user sends a MD5 hash digest MD5(IV, challenge||password) to the server, where || means concatenation.
4. The server performs the same calculation on his side to obtain another MD5 hash digest, and then checks whether the two MD5 digests match.
5. If the two digests are the same, authentication succeeds. Otherwise, authentication fails.

2.2.2 Password Recovery Attacks on APOP

Password recovery attacks on APOP [7], [10], [11] are chosen-challenge attacks. The attacker impersonates the server and sends chosen challenges to the user. The main novelty of the previous password recovery attack is recovering the former part of the password following the *exhaustively guess-then-verify* approach without the knowledge of the latter part. Consequently, the attacker can recover the password one part by one part sequentially, and the complexity will be significantly lower than the expected complexity.

Denote the password by $P_1\|\cdots\|P_l$, where P_i is one part of password. Suppose the attacker has recovered P_1, $\ldots$, P_{i-2} and P_{i-1} ($i \le l$). The high-level description of recovering P_i is as follows.

1. Guess the value of P_i.
2. Generate a pair of challenges (C, C') satisfying the following three conditions.
   a. C and C' have the same bit length;
   b. the length of $C\|P_1\|P_2\|\cdots\|P_i$ is a multiple of block-length;
   c. MD5($IV, C\|P_1\cdots\|P_i$)= MD5($IV, C'\|P_1\|\cdots\|P_i$). Here MD5 computation is without padding the messages.

   From the specification of MD5, these three conditions guarantee that MD5($IV, C\|P$)=MD5($IV, C'\|P$) no matter what $P_{i+1}, \cdots, P_l$ are.
3. Send C to the user to obtain a response R.
4. Send C' to the user to obtain a response R'.
5. If $R = R'$, the current guess value is the true P_i.
6. If $R \neq R'$, change the guess value, and go to step 2.

Suppose P_i has n bits. There are 2^n possible candidates for P_i. As a result, steps 2~6 will be repeated 2^n times in the worst case. So the bit length of P_i should be as short as possible. From the specification of APOP [8], the length of challenges must be a multiple of 8. Therefore, the minimum bit length of P_i is 8, namely, one character. So previous APOP attacks [7], [10], [11] recover the password characters one by one. For one password character, $2^8 - 1$ pairs of challenges are necessary in the worst case (the attacker does not need to confirm the last candidate if all the other candidates have been proven wrong).

2.3 NMAC and Related Attacks on NMAC

2.3.1 NMAC

NMAC is designed based on the keyed Merkle-Damgård hash functions. A keyed Merkle-Damgård hash functions is derived from a public Merkle-Damgård hash function by replacing the public IV with a secret key. Denote by $H(IV, \cdot)$ a Merkle-Damgård hash function. The structure of NMAC is as follows:

$$\text{NMAC}(M)=H(k_1, H(k_2, M)),$$

where k_1 and k_2 are denoted as the outer key and the inner key respectively.

2.3.2 Previous Key-Recovery Attacks on NMAC-MD5

Previous key-recovery attacks on NMAC-MD5 [4], [6] are in the related-key setting, which means that the attacker is given other NMAC-MD5 oracles with different keys, and the relations between the keys in different oracles are known. Denote by MD5(k_1, MD5($k_2, \cdot$)) the target oracle. The attacker first recovers the inner key k_2 by utilizing another oracle MD5(k_1, MD5($k_2 \oplus \Delta^{MSB}, \cdot$)) [4], and then recovers the outer key k_1 by utilizing the inner key k_2 and another oracle MD5($k_1 \oplus \Delta^{MSB}$, MD5($k_2, \cdot$)) [6]. The procedure of recovering k_1 and of recovering k_2 are similar. How to recover k_b ($b \in \{1,2\}$) is as follows:

1. Search a message M satisfying md5(k_b, M)= md5($k_b \oplus \Delta^{MSB}, M$). Such a message M can be found after 2^{46} trials according to the collision attack on md5 in Sect. 2.1.2.
2. Recover an intermediate chaining value at *some* step (a_i, b_i, c_i, d_i) $(0 \leq i \leq 64)$ during the computation of *md*5(k_b, M). This technique is the main novelty of the previous attacks. However, our attack does not utilize this technique at all, so we will omit the description. For more details of this technique, refer to [4], [6].
3. Obtain k_b by the inverse calculation from step i until the initial value k_b. The step function of md5 is invertible: $b_{i-1} = c_i$, $c_{i-1} = d_i$, $d_{i-1} = a_i$, and $a_{i-1} = (b_i - c_i) \ggg s_i - f(c_i, d_i, a_i) - m_k - t$.

3. Improving the Previous Attacks on APOP

At first, we will show two problems of the previous password recovery attacks on APOP and point out one plausible approach. Then we will introduce a new variant of collision on hash function and apply it to MD5. Finally we illustrate our improved attack on APOP.

3.1 Two Problems of the Previous Attacks

3.1.1 Problem I: APOP is Triggered by the User

From the specification of APOP, APOP can not be executed unless the user sends access requests first. Therefore the attacker impersonating the server can not send chosen challenges unless the user sends access requests, which will make the time complexity of the previous attacks depend on the number of chosen challenges and the frequency of the user sending requests. Pick one password character as an example. The number of necessary chosen challenges is 510 in the worst case. Suppose the user checks new emails once per hour. The time complexity is at least 510 hours in the worst case.

Reducing the number of necessary chosen challenges will improve the time complexity.

3.1.2 Problem II: the Attack Might be Detected

Following the previous attack procedure, the attacker impersonates the mail server when the user sends access requests for new emails. Finally the attacker responds with "No new email" to the user. The attack might be detected by the user:

(a) the user does not get any new email for a long time when the attacker continuously impersonates the server;
(b) the user gets a new email delayed when the attacker impersonates the server from time to time. Suppose the user checks new emails once per day. Then when he gets a new email one day delayed, he may suspect being attacked.

Reducing the number of necessary chosen challenges will reduce the number of impersonations, which will lower the probability of the attack being detected. This improvement will make the attack on APOP more realistic.

3.2 A Variant of Collisions on Hash Functions: Applied to MD5

3.2.1 Bit-Free Collision

Definition 1. *If a pair of partially-fixed messages (M, M') satisfies the following conditions*[†]*, it is denoted as a bit-free collision on a hash function H:*

1. *M and M' have the same bit-length.*
2. *the unfixed bits of M and of M' have the same bit positions, and are equal.*
3. *any pair of messages, derived by setting the value of the unfixed bits of M and M', will be a collision on H.*

where the unfixed bits are denoted as free bits.

Denote by *t-bit-free collision* a bit-free collision with t-free bits for simplicity. 2^t pairs of colliding messages can be derived from a t-bit-free collision. So a t-bit-free collision is a set of 2^t colliding message pairs. Note that the collision pairs differ from each other at the bit values of the t-free bits. For any two messages from different pairs, namely with different bit values at the t-free bits, the probability that they contribute to a collision is negligible.

To make the definition clear, we will pick a 1-bit-free collision (M, M') as an example. M and M' have the same bit-length, and have one same bit position (the free bit), where the value is not fixed. By setting the free bit to 0, a pair of messages (M_0, M'_0) is derived from (M, M'). By setting the free bit to 1, a pair of messages (M_1, M'_1) is derived from (M, M'). Both (M_0, M'_0) and (M_1, M'_1) are collisions on H. The probability that (M_0, M'_1) or (M'_0, M_1) is a collision on H is negligible.

[†]The conditions are restrictive. In fact we can give more general definition for bit-free collision. For example, conditions 1 and 2 are not necessary. Since this paper deals with the application to attacking APOP, we define the bit-free collision according to this application for the consistency.

3.2.2 Bit-Free Collision Attack on MD5

Our bit-free collision attack on MD5 adopts the collision attack on md5 in Sect. 2.1.2, the IV bridge in Sect. 2.1.3 and the Tunnel technique in Sect. 2.1.4. Following the notations in Appendix A, denote by $(M_1\|M_2, M_1\|M_2')$ the pair of messages of IV bridge in [11], and by h and h' the values $\mathrm{MD5}(M_1\|M_2)$ and $\mathrm{MD5}(M_1\|M_2')$ respectively, where messages are not padded during MD5 computations. $h \oplus h' = \Delta^{MSB}$, and h satisfies that $b_{0,32} = c_{0,32} = d_{0,32}$. We will locate the free bits in the third message block M_3 to search a bit-free collision $(M_1\|M_2\|M_3, M_1\|M_2'\|M_3)$ on MD5. The procedure searching M_3 such that $\mathrm{md5}(h, M_3)=\mathrm{md5}(h', M_3)$ is as follows:

1. Determine the bit position of the free bits.
   Since we will apply the bit-free-collision attacks on MD5 to improve the attacks on APOP, we locate the free bits in the last 8-bit positions of messages. Note that MD5 adopts little-endian. Therefore the free bits locate at the 8 MSB of m_{15} when M_3 is divided into $\{m_0, \ldots, m_{15}\}$ during the md5 computation.
2. Generate a random message M, where the intermediate chaining values at $1 \sim 15$ steps of $\mathrm{md5}(h, M)$ satisfy both the conditions of Den Boer *et al.*'s collision attack on md5 [3] and the conditions of Q9 tunnel [5]. Note that there is a contradiction between the conditions on the MSBs of b_9 and of b_{10}. We will set that $b_{9,32} = b_{10,32}$. Therefore the freedom for Q9 tunnel becomes 31 bits.
   Denote by $M^0, \ldots, M^{2^t-1}$ messages derived from M by setting the free bit to $0, 1, \ldots, 2^t - 1$ respectively. From the specification of the step function of md5, we can first randomly generate the intermediate chaining values at $1 \sim 15$ steps satisfying all conditions, and then calculate M:
   $$m_i = (b_{i+1} - b_i) \ggg s_i - a_i - f(b_i, c_i, d_i) - t.$$
3. Check whether the intermediate chaining variables at $16 \sim 24$ steps of $\mathrm{md5}(h, M^0)$, $\mathrm{md5}(h, M^1)$, …, $\mathrm{md5}(h, M^{2^t-1})$ can satisfy the conditions of Den Boer *et al.*'s attack. If no, go to step 2.
4. Run Q9 tunnel to find bit-free collision from 25-th step, which means to try all possible values of m_8. More details are shown in Sect. 2.1.4.
5. If the bit-free collision is not found after running the Q9 tunnel, go to step 2.

There are 9 conditions from the 16-th step until the 24-th step. By using the Q9 tunnel technique, if one message can satisfy all these 9 conditions, 2^{31} messages can be derived from it that each can satisfy these 9 conditions. There are 23 conditions from the 25-th step to the 64-th step. For a regular collision, roughly 2^{23} md5 computations are necessary. Hence, for t-bit-free collisions, roughly $2^{23\times 2^t}$ md5 computations are necessary. This means that we have to provide $2^{23}\times 2^t$ messages that can satisfy the conditions from the first step until the 24-th step. The complexity of such preparation is roughly $2^{9\times 2^t-31} \times 2^{23\times 2^t}$. Therefore the total complexity is $(1 + 2^{9\times 2^t-31}) \times 2^{23\times 2^t}$ md5 computations.

It seems that finding a 1-bit-free collision with a complexity roughly 2^{46} is practical. We implement the 1-bit-free collision search. Surprisingly, it takes only 12 hours by 12 computers on average to generate a 1-bit-free collision, which is much faster than the usual time for the 2^{46} md5 computations. One reason is that the complexity calculated by counting the number of the conditions is greater than the precise complexity. Moreover, due to the biased bit position of the conditions, since all the conditions are located in only MSB of the intermediate chaining values, the complexity should be less than the 2^{46} md5 computations.

One example of a 1-bit-free collision on MD5 is shown in Appendix.

3.3 Our Improved Attack on APOP

3.3.1 The Main Novelty

We will recover each password character part by part as follows: first locate the free bit in the target password character, then recover the 7-non-free bits by the guess-then-verify approach using the 1-bit-free collisions on MD5, and finally recover the free bit by the guess-then-verify approach using a regular collision on MD5.

The number of necessary chosen challenges will be half-reduced.

3.3.2 The Detailed Attack Procedure

The procedure of our attack to recover one password character is as follows. Moreover, we give a pseudo-source code in Table 3.

1. Locate the free bit in the targeted password character, which divides the password character into 7-non-free bits and one free bit.
2. Recover the 7-non-free bits first. The attacker adopts the exhaustively guess-then-verify approach: guess the value of 7-non-free bits, then generate a pair of challenges, which will lead to a t-bit-free collision after being concatenated with the guess value, and finally send the pair of challenges to the user to check whether the responses collide or not. If the responses collide, the guess value is true. Otherwise, the guess value is wrong. There are in total 2^7 possible candidates for the 7-non-free bits, so $(2^7 - 1)$ pairs of challenges are necessary in the worst case.
3. Recover the free bit by the guess-then-verify approach. The attacker guess the free bit is '0', then generate a pair of challenges that leads to a regular collision, and finally sends the pair of challenges to check whether the responses collide or not. If the responses collide, then the guess is correct, which means the free bit is 0. Otherwise, the guess value is wrong, which means the

Table 3 Our attack procedure.

Denote by p_r the password characters which have been recovered. Denote by p a password character which is going to be recovered. Note that the online work and offline work are *parallel* and *independent*.

```
Our procedure
Our attack utilizes 1-bit-free collision. Set the bit position of the free bit
in $p$, which will divide the $p$ into two parts: 1-free bit and 7-non-free
bits denoted as $p_f$ and $p_{nf}$, respectively. For simplicity, we assume the
1-free bit locates at MSB of $p$.

Stage 1: recover the value of $p_{nf}$.
    -Chosen challenge collection (offline)
        1. For $p_{nf}$ = 0000000 to 1111111 (7-non-free bits)
        2. Generate a pair of challenges $(C, C')$: $(C\|p_r\|p_f\|p_{nf}$,
           $C'\|p_r\|p_f\|p_{nf})$ is a 1-bit-free collision.
        3. Store $(C, C', p_{nf})$ to Table $\mathcal{T}$.
        4. End For
    -Impersonating as server (online)
        If $\mathcal{T}$ is not NULL, then
        1. Pick an element $(C, C', p_{nf})$ from $\mathcal{T}$.
        2. Erase $(C, C', p_{nf})$ from $\mathcal{T}$.
        3. Send $C$ to the user to obtain the response $R$.
        4. Send $C'$ to the user to obtain the response $R'$.
        5. If $R = R'$, then the current value $p_{nf}$ is the true 7-non-
           free bits of $p$. Goto Stage 2.
        6. If $R \neq R'$, continue to run Stage 1.
        Else, the attacker does not impersonate. Continue to run
        Stage 1.

Stage 2: recover the value of $p_f$.
    1. Guess the 1-free bit is 0.
    2. Generate a pair of challenges $(C, C')$ such that $(C\|p_r\|0\|p_{nf}$,
       $C'\|p_r\|0\|p_{nf})$ is a collision.
    3. Send $C$ to the user to obtain the response $R$.
    4. Send $C'$ to the user to obtain the response $R'$.
    5. If $R = R'$, the value of $p_f$ is 0. Otherwise, the value is 1.
    6. Halt the program.
```

free bit is 1. So the free bit will be recovered by one pair of challenges.

In the above procedure, the attacker will first recover the values of 7-non-free bits without the knowledge of the free bit, and then recover the value of the free bit. Totally $(2^7 - 1 + 1)$ pairs of challenges are necessary in the worst case. The total number of necessary challenges has been approximately half-reduced compared to the previous attacks on APOP.

4. New Results of NMAC-MD5

First of all, we will reveal a new weakness of NMAC-MD5. Then based on this weakness, we will propose a new outer-key recovery attack on NMAC-MD5, which leads to a full-key recovery attack on NMAC-MD5 by combining the previous inner-key recovery attack [4]. Finally we evaluate the complexity of our attack.

4.1 A New Weakness of NMAC-MD5

The input messages of the outer MD5 of NMAC-MD5 are the outputs of the inner MD5, which are 128-bit long and shorter than one block length (512 bits). Therefore the input messages of the outer MD5 function is only one message block after padded. Namely the outer MD5 is in fact md5$(k_1, \cdot)$. Recall md5 consists of a feed-forward operation: the output is the sum of initial value and the intermediate chaining value at the last step. As a result, the MAC value of NMAC-MD5 is the sum of the outer key k_1 and the intermediate chaining value at the 64-th step of the outer MD5. Hereafter we denote by ICV_{64} the intermediate chaining value at 64-th step of the outer MD5 of NMAC-MD5 for simplicity.

Based on the above observation, one trivial attack approach is obtaining both MAC value and ICV_{64} of a message, and then recovering the outer key k_1. The attacker can send any message to NAMC oracle to get the corresponding MAC value. However it is not trivial to obtain ICV_{64} for any message. We will solve this problem in the related-key setting: denote by MD5$(k_1$, MD5$(k_2, \cdot))$ the target NMAC oracle; and the attacker is given another NMAC oracle MD5$(k_1 \oplus \Delta^{MSB}$, MD5$(k_2, \cdot))$. The crucial idea is as follows:

Search a message M that causes a collision between the two NMAC oracles. According to Den Boer et al.'s collision attack on md5 [3], such a message can be obtained after 2^{46} *random messages are tried. Moreover, since the probability of a message randomly causing a collision between the two NMAC oracles is* 2^{-128}*, it is with an overwhelming probability that M satisfies all the conditions of Den Boer et al.'s collision attack. Therefore* ICV_{64} *of M satisfies* $a_{64,32} = c_{64,32}$.

Our outer-key recovery attack adopts the same strategy to obtain the inside information of ICV_{64} of some message, and then utilize it to recover the outer key.

4.2 Our Outer-Key Recovery Attack

Recall our attack is in the related-key setting: denote by MD5$(k_1$, MD5$(k_2, \cdot))$ the target NMAC oracle; the attacker is given another NMAC oracle MD5$(k_1 \oplus \Delta^{MSB}$, MD5$(k_2, \cdot))$. Divide k_1 into 32-bit values (k_a, k_b, k_c, k_d), ICV_{64} into $(a_{64}, b_{64}, c_{64}, d_{64})$, and MAC value into (h_a, h_b, h_c, h_d) respectively. We can get that $h_a = k_a + a_{64}$, $h_b = k_b + b_{64}$, $h_c = k_c + c_{64}$ and $h_d = k_d + d_{64}$. Our attack will recover $k_{a,i}$, $k_{c,i}$ $(1 \leq i \leq 31)$ and the value $k_{a,32} \oplus k_{c,32}$.

4.2.1 How to Recover $k_{a,32\sim31}$ and $k_{c,32\sim31}$

One observation. For the sum calculation $h_a = k_a + a_{64}$, fix the two bit values $h_{a,31}$ and $k_{a,31}$, and randomly change the other bit values without affecting the sum relation. The following property holds: if $h_{a,31} = k_{a,31}$, a bit carry from the 31-th bit to the 32-th bit happens with a probability $\frac{1}{2}$, and does not happen with a probability $\frac{1}{2}$; if $k_{a,31} \neq k_{a,31}$, a bit carry from the 31-th bit to the 32-th bit either happens with a probability 1 or does not happen with a probability 1. More precisely, if $k_{a,31} = 1$ and $h_{a,31} = 0$, the bit carry happens with a probability 1. If $k_{a,31} = 0$ and $h_{a,31} = 1$, the bit carry does not happen with a probability 1. The same observation exists during the calculation $h_c = k_c + c_{64}$.

Based on the above observation, we will illustrate our procedure of recovering $k_{a,32\sim31}$ and $k_{c,32\sim31}$.

1. Send random messages to the two NMAC oracles to check whether a collision happens. If a message causes collision, then save it into a set denoted as $\mathcal{M}$. For any $M \in \mathcal{M}$, ICV_{64} satisfies $a_{64,32} = c_{64,32}$.
2. Regroup $\mathcal{M}$ depending on the MAC bit values $h_{a,31}$ and $h_{c,31}$:

   $\mathcal{M}_0 : h_{a,31} = 0$ and $h_{c,31} = 0$;
   $\mathcal{M}_1 : h_{a,31} = 0$ and $h_{c,31} = 1$;
   $\mathcal{M}_2 : h_{a,31} = 1$ and $h_{c,31} = 0$;
   $\mathcal{M}_3 : h_{a,31} = 1$ and $h_{c,31} = 1$.

3. Pick out the group satisfying $h_{a,31} \neq k_{a,31}$ and $h_{c,31} \neq k_{c,31}$.
   From $h_a = k_a + a_{64}$ and $h_c = k_c + c_{64}$, the value $h_{a,32} \oplus h_{c,32}$ depends on the value $k_{a,32} \oplus k_{c,32}$, the value $a_{64,32} \oplus c_{64,32}$, and the XOR value of the bit carry values from the 31-th bit to the 32-th bit of the two addition computations. $k_{a,32} \oplus k_{c,32}$ is a constant. Any element from $\mathcal{M}$ satisfies $a_{64,32} \oplus c_{64,32} = 0$. Therefore, we can derive the XOR value of the bit carry values from the 31-th bit to the 32-th bit of $k_a + a_{64}$ and that of $k_c + c_{64}$ from $h_{a,32} \oplus h_{c,32}$. Based on the above observation, the bit carry values from the 31-th bit to the 32-th bit during either $k_a + a_{64}$ or $k_c + c_{64}$ is the same for any two elements of the target group with a probability 1, but differ from each other with a probability $\frac{1}{2}$ for the other groups. Therefore, the XOR values of the carry from the 31-th bit to the 32-th bit of $k_a + a_{64}$ and that of $k_c + c_{64}$ should be a constant for the target group, but change for the other groups. Namely we can pick out the target group.
4. Recover $k_{a,31}$ and $k_{c,31}$. Moreover, we can determine the value $k_{a,32} \oplus k_{c,32}$.
   Denote by $\mathcal{M}_j$ the group picked out in step 3. Utilizing $h_{a,31}$ and $h_{c,31}$ in $\mathcal{M}_j$, we get $k_{a,31} = 1 - h_{a,31}$ and $k_{c,31} = 1 - h_{c,31}$. For any element of $\mathcal{M}_j$, by detecting the values $k_{a,31}$ and $k_{c,31}$, we can determine whether bit carries happen during $k_a + a_{64}$ and $k_c + c_{64}$. For any element in $\mathcal{M}_j$, $a_{64,32} = c_{64,32}$. Therefore we can easily derive the value $k_{a,32} \oplus k_{c,32}$.

4.2.2 How to Recover $k_{a,i}$ and $k_{c,i}$ ($1 \leq i \leq 30$)

Since the procedure of recovering $k_{a,i}$ is the same with that of recovering $k_{c,i}$, here we will pick $k_{a,i}$ as an example to illustrate our attack procedure.

Our attack will recover the bit values of k_a from the most significant bit until the least significant bit. So we will recover $k_{a,30}$ first, then $k_{a,29}, \ldots$, and finally $k_{a,1}$. Suppose we have obtained the bit values $k_{a,30}, \ldots, k_{a,i+1}$, and are going to recover $k_{a,i}$. The detailed procedure is as follows.

1. Send random messages to the two oracles until one message M is obtained satisfying the following three conditions:
   a) M causes that the two NMAC oracles collide;
   b) $h_{c,30} \neq k_{c,30}$.
   c) $h_{a,j}=k_{a,j}$ ($i + 1 \leq \forall j \leq 30$);
2. Check whether a bit carry happens or not from the i-th bit to the $(i + 1)$-th bit during $k_a + a_{64}$.
   $h_{a,32} \oplus h_{c,32}$ is determined by $k_{a,32} \oplus k_{c,32}$, $a_{64,32} \oplus c_{64,32}$, and the XOR value of the bit carry value from the 31-th bit to the 32-th bit of $k_a + a_{64}$ and that of $k_c + c_{64}$. $k_{a,32} \oplus k_{c,32}$ has been recovered. From the condition a) in step 1, we can get $a_{64,32} = c_{64,32}$. From the condition b) in step 1, we can get whether the bit carry happens or not from 31-th bit to 32-th bit during $k_c + c_{64}$. So by detecting $h_{a,32} \oplus h_{c,32}$, we can derive whether the bit carry from the 31-th step to the 32-th step happens or not during $k_a + a_{64}$. Moreover, from condition c) in step 1, during $k_a + a_{64}$, whether the bit carry happens from the 31-th bit to the 32-th bit in fact depends on whether the bit carry happens from the i-th bit to the $(i + 1)$-th bit: if a bit carry happens from the i-th bit to the $(i + 1)$-th bit, then a bit carry will propagate to the 32-th bit.
3. If the result of step 2 is one of the following situations, we have recovered $k_{a,i}$. Otherwise, go to step 1.
   Situation 1: if the bit carry happens and $h_{a,i}$=1, then $k_{a,i}$=1;
   Situation 2: if the bit carry does not happen and $h_{a,i}$=0, then $k_{a,i}$=0.

4.3 A Full-Key Recovery Attack

Contini *et al.* proposed an inner-key recovery attack on NMAC-MD5 [4]. Combining their works, we can extend our attack to recover the full key of NMAC-MD5. The attack scenario is as follows.

1. Apply our attack to partially recover the outer key.
2. Adopt Contini *et al.*'s attack procedure to obtain the inner key.
3. Exhaustively search the unrecovered bit values of the outer key. At this step, the knowledge of the inner key is necessary.

4.4 Evaluating the Complexity

4.4.1 The Complexity of Recovering $k_{a,32\sim31}$ and $k_{a,32\sim31}$

Suppose we generate s elements for each group. At step 2 of the attack procedure in Sect. 4.2.1, any two elements of other groups differ from each other with probability $\frac{1}{2}$. The probability of failing to find two different elements in a group (not target group) is 2^{-s}. Since there are 3 other groups, the failure probability at step 2 is 3×2^{-s}. It needs roughly 2^{48} trials to find one element of a particular group since there are in total 48 conditions. To summarize, by $3 \times s \times 2^{48}$ queries, we can succeed with a probability $1-3\times2^{-s}$.

4.4.2 The Complexity of Recovering $k_{a,i}$ and $k_{c,i}$ ($1 \leq i \leq 30$)

At step 1 of the attack procedure in Sect. 4.2.2, it roughly needs 2^{77-i} queries, since 2^{46}, $2^{30-(i+1)+1}$ and 2 queries are necessary to find a message satisfying conditions a), b) and c) respectively. The results of the step 2 is one of the two situations of step 3 with a probability $\frac{1}{2}$. Therefore, the step 1 will be repeated with a probability $\frac{1}{2}$. Suppose step 1 is repeated q times. The complexity of recovering $k_{a,i}$ will be $q \times 2^{77-i}$ with a success probability $1 - 2^{-q}$. From 30-th bit until i-th bit, the total complexity is

$$\textstyle\sum_{t=i}^{t=30} q \times 2^{77-t} < q \times 2^{78-i}.$$

4.4.3 The Total Complexity

Finally we will evaluate the complexity of recovering full keys. The complexity of recovering $k_{a,32\sim31}$ and $k_{c,32\sim31}$ can be ignored compared with the complexity of recovering $k_{a,i}$ or $k_{c,i}$, where i is small. We will pay attention to the complexity of recovering $k_{a,i}$. Here we determine the value q to be 2 to obtain the low security bound of NMAC-MD5. Suppose we partially recover both k_a and k_c until i-th bit by online interacting with NMAC oracles. The number of the remaining unrecovered bit of k_1 is $65+2\times i$, which will be recovered by exhaustive offline computation. The offline complexity is $2^{65+2\times i}$ MD5 computations. Following the evaluation of the total complexity in [4], which is the sum of online queries and offline computations, the optimal bit position i should make the number of online queries equal to the number of offline MD5 computations:

$$2^{65+2\times i} = 2 \times 2^{78-i} + 2 \times 2^{78-i} \rightarrow i = 5.$$

Finally the complexity of our attack is 2^{75} online queries and 2^{75} offline MD5 computations.

5. Feedback to Practical Information Systems

The password recovery attack on APOP is practical. For example, the password has 6 characters. The attacker can obtain the first password character by using our attack procedure. One password character that most people use has 6 bits of entropy [7]. Following our attack, the number of necessary chosen challenges is just 32 in the worst case and 16 in the average case. The offline complexity of recovering the remaining 5 characters is at most 2^{40} MD5 computations, which takes less than a single day by using 10 PCs. APOP has been adopted by several practical mail systems such as Mozilla Thunderbird. An countermeasure that Mozailla Thunderbird adopts now is *restricting the challenges to be ASCII texts.* This countermeasure has two advantages: a) it is computationally infeasible to generate ASCII colliding messages on MD5 so far; b) the expense is little. However, with the increase of the computational power of the computers and the development of the attack techniques on MD5, the security of this countermeasure in the coming few years is very suspicious in our opinion. We suggest that those systems based on APOP should update their underlying MD5 to a more secure hash function. One candidate is SHA-2.

The key-recovery attack on NMAC-MD5 is still just a theoretical result. Therefore, the security of both NMAC-MD5 and HMAC-MD5 is not damaged. However, considering the increase of computational power, the hash digest of MD5, which is 128 bits, seems relatively short. We recommend that the related information systems should update HMAC-MD5 to HMAC-SHA-1, or HMAC based on more secure hash functions than SHA-1.

References

[1] M. Bellare, R. Canetti, and H. Krawczyk, "Keying hash functions for message authentication," CRYPTO 1996, LNCS, vol.1109, pp.1–15, 1996.

[2] M. Bellare, "New proofs for NMAC and HMAC: Security without collision-resistance," CRYPTO 2006, LNCS, vol.4117, pp.602–619, 2006.

[3] B. den Boer and A. Bosselaers, "Collisions for the compression function of MD5," EUROCRYPT 1993, LNCS, vol.765, pp.293–304, 1994.

[4] S. Contini and Y.L. Yin, "Forgery and partial key-recovery attacks on HMAC and NMAC Using hash collisions," ASIACRYPT 2006, LNCS, vol.4284, pp.37–53, 2006.

[5] V. Klima, "Tunnels in hash functions: MD5 collisions within a minute," Cryptology ePrint Archive, Report 2006/105. http://eprint.iacr.org/2006/105.pdf

[6] P. Fouque, G. Leurent, and P. Nguyen, "Full key-recovery attacks on HMAC/NMAC-MD4 and NMAC-MD5," CRYPTO 2007, LNCS vol.4622, pp.13–30, 2007.

[7] G. Leurent, "Message freedom in MD4 and MD5 collisions: application to APOP," FSE 2007, LNCS, vol.4593, pp.309–328, Springer-Verlag, 2007.

[8] J. Myers and M. Rose, "Post office protocol - version 3," RFC 1939 (Standard), May 1996. Updated by RFCs 1957, 2449. ftp://ftp.isi.edu/in-notes/rfc1939.txt

[9] Ronald L. Rivest, "The MD5 message digest algorithm," Request for Comments (RFC 1321), Network Working Group, 1992.

[10] Y. Sasaki, G. Yamamoto, and K. Aoki, "Practical password recovery on an MD5 challenge and response," Cryptology ePrint Archive, Report 2007/101.

[11] Y. Sasaki, L. Wang, K. Ohta, and N. Kunihiro, "Security of MD5 challenge and response: Extension of APOP password recovery attack," CT-RSA 2008, LNCS, vol.4964, pp.1–18, Springer, 2008.

[12] L. Wang, K. Ohta, and N. Kunihiro, "New key-recovery attacks on HMAC/NMAC-MD4 and NMAC-MD5," EUROCRYPT 2008, LNCS vol.4965, pp.237–253, Springer, 2008.

[13] L. Wang, Y. Sasaki, K, Sakiyama, and K. Ohta, "Bit-free collision: Application to APOP Attack," IWSEC 2009, LNCS, vol.5824, pp.3–21, Springer, 2009.

[14] X. Wang and H. Yu, "How to break MD5 and other hash functions," EUROCRYPT 2005, LNCS vol.3494, pp.19–35, 2005.

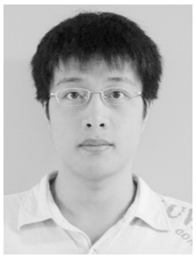
[15] X. Wang, Y.L. Yin, and H. Yu, "Finding collisions in the full SHA-1," CRYPTO 2005, LNCS vol.3621, pp.17–36, 2005.

Appendix: An Example of 1-bit-free Collision on MD5

Table A·1 1-bit-free collision on MD5. MD5($M_1 \| M_2 \| M_3$)=MD5($M_1 \| M'_2 \| M_3$), where the 1 free bit locates in M_3.

	IV bridge in [11]		
M_1	m_0=0x3938313c	m_1=0x37322d34	m_2=0x332d3635
	m_3=0x2e383933	m_4=0x37373936	m_5=0x3433302d
	m_6=0x38312d35	m_7=0x61704035	m_8=0x6f777373
	m_9=0x645f6472	m_{10}=0x63657465	m_{11}=0x5f726f74
	m_{12}=0x2e636264	m_{13}=0x6976746d	m_{14}=0x632e7765
	m_{15}=0x73752e61		
M_2	m_0=0x986e1da4	m_1=0x83707d06	m_2=0xa86e1ddd
	m_3=0xe264eedb	m_4=0xff68e19f	m_5=0x120ea5b3
	m_6=0x7437d3e2	m_7=0x600f543d	m_8=0x7c63c5ab
	m_9=0xe9ead9d9	m_{10}=0xa9b5c51e	m_{11}=0xc309f623
	m_{12}=0xfd534f1e	m_{13}=0xad33c7ad	m_{14}=0xfd0380c6
	m_{15}=0x7745f36a		
M'_2	m'_0=0x986e1da4	m'_1=0x83707d06	m'_2=0xa86e1ddd
	m'_3=0xe264eedb	m'_4=0xff68e19f	m'_5=0x120ea5b3
	m'_6=0x7437d3e2	m'_7=0x600f543d	m'_8=0x7c63c5ab
	m'_9=0xe9ead9d9	m'_{10}=0xa9b5c51e	m'_{11}=0x4309f623
	m'_{12}=0xfd534f1e	m'_{13}=0xad33c7ad	m'_{14}=0xfd0380c6
	m'_{15}=0x7745f36a		
h	0x**b**d7ade50, 0x**e**17a619d, 0x**8**e940937, 0x**f**d4af95f		
h'	0x**3**d7ade50, 0x**6**17a619d, 0x**0**e940937, 0x**7**d4af95f		

	searched message for 1-bit-free collision		
M_3	m_0=0xc0797ae2;	m_1=0xe95d42e6;	m_2=0x49fe29af;
	m_3=0x3329c9a9;	m_4=0xa790a55d;	m_5=0x0783e6d3;
	m_6=0xb906c7b1;	m_7=0x2d63e951;	m_8=0x9edac296;
	m_9=0x26afe101;	m_{10}=0xd4cfc4fb;	m_{11}=0xcb0d1667;
	m_{12}=0x77b75eab;	m_{13}=0xea993a34;	m_{14}=0x8c9868ae;
	m_{15}=0x7**e**ffffff;		
$m_{15,25}$ is the free bit: m_{15} = 0x7effffff or 0x7fffffff.			



Lei Wang was born in Hebei, China, on 2 October, 1983. He received the B.E. degree from Shang Hai Jiao Tong University in 2006, and the M.E. degree from the University of Electro-Communications in 2009. He is currently a Ph.D course student at Graduate School of Electro-Communications since 2009. He is presently engaged in research on cryptography. He was awarded the SCIS'08 paper prize.

Kazuo Ohta received his B.S., M.S., and Dr.S. degrees from Waseda University, Tokyo, Japan, in 1977, 1979, and 1990, respectively. He has been a professor at the University of Electro-Communications since 2001. He was a researcher at NTT Laboratories between 1979 and 2001. He is presently engaged in research on information security. Dr. Ohta is a member of IACR, IPSJ and IEEE.

Yu Sasaki received the B.E. and M.E. in Information and Communication Engineering from The University of Electro-Communications in 2005 and 2007. Since 2007, he has been a researcher at NTT Information Sharing Platform Laboratories, NTT Corporation. His current research interests are in cryptography. He was awarded the SCIS'07 paper prize.

Kazuo Sakiyama received the B.E. and M.E. degrees from Osaka University, Japan, in 1994 and 1996, respectively, the M.S. degree from the University of California, Los Angeles in 2003, and the Ph.D. degree in electrical engineering from the Katholieke Universiteit Leuven, Belgium, in 2007. From 1996 to 2004, he was with the Semiconductor and IC Division, Hitachi, Ltd., (now Renesas Technology Corp.). He is currently an Associate Professor at the University of Electro-Communications, Tokyo. His main research interest includes efficient and secure embedded system architectures and design methodologies for cryptographic applications. He is a member of the IACR, IPSJ and IEEE.

Noboru Kunihiro received his B.E., M.E. and Ph. D. in mathematical engineering and information physics from the University of Tokyo in 1994, 1996 and 2001, respectively. He is an Associate Professor of the University of Tokyo. He was a researcher of NTT Communication Science Laboratories from 1996 to 2002. He was a associate professor of the University of Electro-Communications from 2002 to 2008. His research interest includes cryptography, information security and quantum computations. He was awarded the SCIS'97 paper prize.