New Key-Recovery Attacks on HMAC/NMAC-MD4 and NMAC-MD5

Lei Wang, Kazuo Ohta, and Noboru Kunihiro

The University of Electro-Communications,
1-5-1 Chofugaoka, Chofu-shi, Tokyo, 182-8585, Japan
{wanglei, ota, kunihiro}@ice.uec.ac.jp

Abstract. At Crypto '07, Fouque, Leurent and Nguyen presented full key-recovery attacks on HMAC/NMAC-MD4 and NMAC-MD5, by extending the partial key-recovery attacks of Contini and Yin from Asiacrypt '06. Such attacks are based on collision attacks on the underlying hash function, and the most expensive stage is the recovery of the so-called outer key. In this paper, we show that the outer key can be recovered with near-collisions instead of collisions: near-collisions can be easier to find and can disclose more information. This improves the complexity of the FLN attack on HMAC/NMAC-MD4: the number of MAC queries decreases from 2^{88} to 2^{72}, and the number of MD4 computations decreases from 2^{95} to 2^{77}. We also improved the total complexity of the related-key attack on NMAC-MD5. Moreover, our attack on NMAC-MD5 can partially recover the outer key without the knowledge of the inner key, which might be of independent interest.

keywords: HMAC, NMAC, key-recovery, MD4, MD5, differential attack, near-collision.

1 Introduction

Many cryptographic schemes and protocols use hash functions. Their actual security might need to be reassessed, in light of the seminal work by Wang *et al.* [12–15] on finding collisions on hash functions from the MD4 family. This paper deals with key-recovery attacks on HMAC and NMAC using differential techniques. HMAC and NMAC are hash-based message authentication codes proposed by Bellare, Canetti and Krawczyk [1]. HMAC has been implemented in widely used protocols including SSL, TLS, SSH, and IPsec. The construction of HMAC/NMAC is based on a keyed hash function. Let H be an iterated Merkle-Damgård hash function, which defines a keyed hash function H_k by replacing the IV with the key k. Then HMAC and NMAC are defined as:

$$\mathrm{HMAC}_k(M) = H(\bar{k} \oplus opad || H(\bar{k} \oplus ipad || M));$$
$$\mathrm{NMAC}_{k_1,k_2}(M) = H_{k_1}(H_{k_2}(M)),$$

where M is the input message, k and (k_1, k_2) are the secret keys of HMAC and NMAC respectively, $\bar{k}$ means k padded to a single block, $||$ means concatenation, and *opad* and *ipad* are two one-block length constants. NMAC is the theoretical foundation of HMAC: HMAC_k is essentially the same as $\text{NMAC}_{H(\bar{k}\oplus opad),H(\bar{k}\oplus ipad)}$, except with a change in the length value included in the padding. In [1, 2], the security proof was first given for NMAC, and then extended to HMAC. Attacks on NMAC can usually be adapted to HMAC, except in the related-key setting. Hereafter, k_1 and k_2 (for HMAC: $H(\bar{k} \oplus opad)$ and $H(\bar{k} \oplus ipad)$ with the appropriate changes in the padding) are referred to as the outer key and the inner key, respectively. The corresponding hash functions of k_1 and k_2 are referred to as the outer hash function and the inner hash function, respectively.

The security of HMAC and NMAC

The security of HMAC /NMAC has been carefully analyzed by its designers [1, 2]. It has been proved that NMAC is a pseudo-random function family (PRF) under a single assumption: (1) compression function of the keyed hash function is a PRF. The proof for NMAC has been extended to HMAC by an additional assumption: (2) the key derivation function in HMAC is a PRF. However, if the underlying hash function is weak (such as MD4 and MD5), the above proofs may not apply.

There are three types of attacks [4–6, 8, 9] on HMAC/NMAC:

- *Distinguishing attacks*: distinguish HMAC/NMAC from a random function.
- *Existential forgery attacks*: compute a valid MAC for a random message.
- *Universal forgery attacks:* compute a valid MAC for any given message.

We focus on universal forgery attacks, by trying to recover the secret keys k_1 and k_2, like in previous work [4, 5, 9]. Contini and Yin [4] proposed partial key-recovery attacks on HMAC/NMAC instantiated with MD4, MD5[1], SHA-0 and step-reduced SHA-1. Their attacks can only recover the inner key k_2, which is insufficient for a universal forgery attack. Fouque, Leurent and Nguyen [5] presented the first full-key attack on HMAC/NMAC-MD4, by proposing an outer-key recovery attack. They also extended the attack of [4] into a full key-recovery attack on NMAC-MD5 in the related-key setting: this attack was independently found by Rechberger and Rijmen [9], who also proposed a full key-recovery attack in the related-key setting on NMAC with SHA-1 reduced to 34 steps. These full key-recovery attacks first apply the attack of [4] to recover the inner key k_2, then use additional MAC queries to derive several bits of the outer key k_1, and finally the rest of the outer key is obtained by the exhaustive search using offline hash computations. Recovering the outer key is so far the most expensive stage.

Our contributions

We propose new outer-key recovery attacks on HMAC/NMAC-MD4 and NMAC-MD5[2], which leads to full key-recovery attacks by using the inner-key attacks of [4]. Compared to previous work by Fouque *et al.* [5], the main novelty is the use of *near-collisions* instead of collisions. Recall that a near-collision is a pair of distinct messages whose hash values are almost the same, differing only by

[1] The attack on NMAC-MD5 is a related-key attack, and therefore does not apply to HMAC-MD5.

[2] Our attack on NMAC-MD5 is in the related-key setting, like [5, 9].

a few bits (see [7]): our near-collisions are based on a local collision at some intermediate step of the compression function, which significantly simplifies the difference propagation in the last few steps. Our attacks can be sketched as follows. We call the MAC oracle on exponentially many messages chosen in such a way that we can expect to find near-collisions in the outer hash function. By observing the shape of the near-collisions obtained, we are able to derive certain bits of the final values of the four 32-bit intermediate values a, b, c, d of the outer hash function. This discloses a few bits of the outer key k_1, since each 128-bit MAC value is exactly $(k_a+a, k_b+b, k_c+c, k_d+d)$ because MD4 and MD5 use the Davies-Meyer mode, where k_1 is decomposed as four 32-bit variables k_a, k_b, k_c and k_d.

The cost of our attacks is summarized in Table 1. In the case of HMAC/NMAC-MD4, near-collisions are easier to find and disclose more information, which allows to considerably improve the FLN attack [5] in both the number of MAC queries and the number of offline MD4 computations. In the case of NMAC-MD5, compared to the FLN-RR attack [5, 9], total complexity is decreased. Moreover, we note that our attack can partially recover the outer key without the knowledge of the inner key k_2, which might be of independent interest.

Table 1. Comparison with previous work

Universal forgery attack		previous result	our new result
HMAC-MD4 NMAC-MD4	Online queries	2^{88} [5]	2^{72}
	Offline MD4 computations	2^{95} [5]	2^{77}
	Total complexity	2^{95}	2^{77}
NMAC-MD5 related-key setting	Online queries	2^{51} [5, 9]	2^{75}
	Offline MD5 computations	2^{100} [5, 9]	2^{75}
	Total complexity	2^{100}	2^{76}

Organization of the paper

Section 2 reviews background and related work. In Section 3, we explain the advantages of our attacks compared to previous work. In Sections 4 and 5, we present in details our attacks on HMAC/NMAC-MD4 and NMAC-MD5. Finally, we conclude and give open problems in Section 6.

2 Background and Notation

2.1 Description of MD5 and MD4

There is no standard notation for the description of MD5 and MD4. In this paper, we adopt a notation similar to that of [4].

MD5 and MD4 have the Merkle-Damgård structure and output a 128-bit hash value. First, the input message is padded to be the multiple of 512 bits: add '1' in the tail of the input message; add '0's until the bit length becomes 448 modulo 512; add the length of input message (before padding) to the last 64 bits. Then the padded message M is divided into 512-bit messages $M =$

$(M_0, M_1, \ldots, M_{n-1})$. The 128-bit IV is represented as H_0 (which is the secret key in the keyed hash function). The compression function is first applied on M_0 and H_0 as input, which outputs a 128-bit value H_1. By iterating over all the message blocks M_i, we obtain a final 128-bit value H_n, which is defined to be the hash value of M.

Compression function of MD5

The compression function takes a 512-bit message block m and a 128-bit value H as input. First, m is divided into sixteen 32-bit values $(m_0, \ldots, m_{15})$, and H is divided into four 32-bit variables (a_0, b_0, c_0, d_0). The compression function consists of 64 steps, regrouped into four 16-step rounds. Each step is defined as follows:

$$a_i = d_{i-1}, c_i = b_{i-1}, d_i = c_{i-1},$$
$$b_i = b_{i-1} + (a_{i-1} + f(b_{i-1}, c_{i-1}, d_{i-1}) + m_k + t) \lll s_i,$$

where m_k is one of $(m_0, \ldots, m_{15})$, the index k being given by a permutation of $\{0, \ldots, 15\}$ depending on the round, t is a constant defined in each round, $\lll s_i$ means a left-rotation by s_i bits, and f is a Boolean function depending on the round.

1R: $f(X, Y, Z) = (X \wedge Y) \vee (\neg X \wedge Z)$
2R: $f(X, Y, Z) = (X \wedge Z) \vee (Y \wedge \neg Z)$
3R: $f(X, Y, Z) = X \oplus Y \oplus Z$
4R: $f(X, Y, Z) = (X \vee \neg Z) \oplus Y$

The final output is $(a_0 + a_{64}, b_0 + b_{64}, c_0 + c_{64}, d_0 + d_{64})$, which means that MD5 uses the Davies-Meyer mode.

Compression function of MD4

The differences between MD5 and MD4 are the following:

- MD4 consists of 48 steps regrouped into three 16-step rounds.
- Each step is defined as: $b_i = (a_{i-1} + f(b_{i-1}, c_{i-1}, d_{i-1}) + m_k + t) \lll s_i$, where m_k is given by different round permutations.
- In the 2nd round: $f(X, Y, Z) = (X \wedge Y) \vee (Y \wedge Z) \vee (X \wedge Z)$.

2.2 Pseudo-collision of MD5

In [3], den Boer and Bosselaers found a pseudo-collision on the compression function of MD5 of the following form:

$$\text{MD5}(IV,\ M) = \text{MD5}(IV',\ M)$$

Here, the one-block message M is the same, and only the IVs are different. The total probability of their pseudo-collision is 2^{-46}, provided that IV and IV' satisfy the following relations:

- $\Delta IV = (IV \oplus IV') = (0x80000000, 0x80000000, 0x80000000, 0x80000000)$;
- If we decompose the IV as four 32-bit variables (a_0, b_0, c_0, d_0), then the MSBs of b_0, c_0 and d_0 must be the same.

In the rest of this paper, the difference ΔIV of their pseudo-collision will be denoted by Δ^{MSB}, and this pseudo-collision will be referred to as the dBB pseudo-collision.

2.3 Recovering the inner key of HMAC/NMAC-MD4

We recall the differential attack of Contini and Yin [4] to recover the inner key:

1. Determine a message difference ΔM and a differential path DP for a collision attack on MD4. Let n be the number of sufficient conditions.
2. Generate a random one-block message M, and send both M and $M+\Delta M$ to the HMAC/NMAC oracle until one pair of messages $(M_1, M_1+\Delta M)$ collides. Since the number of sufficient conditions is n, such a pair $(M_1, M_1 + \Delta M)$ will be obtained after roughly 2^n pairs of messages are queried.
3. Recover the intermediate chaining variables (ICV) in step t of 1R of $H(k_2, M_1)$. This technique is one main contribution of the inner-key recovery attack of Contini and Yin [4]. For details, please refer to [4].
4. Derive the inner key k_2 by inverse calculation from the obtained ICV. This is easy since each step of MD4 is invertible. For instance, with MD4, if m_{t-1} and ICV in step t are known, ICV in step $t-1$ can be calculated as follows.

$$
\begin{aligned}
&b_{t-1} = c_t, c_{t-1} = d_t, d_{t-1} = a_t,\\
&a_{t-1} = (b_t \ggg s_0) - m_{t-1} - f(c_t, d_t, a_t).
\end{aligned}
$$

The related-key attack on NMAC-MD5 [4] is based on the same ideas. The attack exploits the freedom over the input messages, which explains why this attack is the most efficient attack known to recover the inner key k_2. However, for the outer hash function of HMAC/NMAC, the input message is the output of the inner hash function, for which there is much less freedom. This attack is therefore not well-suited to recover the outer key.

2.4 Recovering the outer key of HMAC/NMAC-MD4

We recall the differential attack of Fouque, Leurent and Nguyen [5] to recover the outer key:

1. Determine a message difference ΔM and a differential path DP for a collision attack on MD4 in such a way that the differential path has one sufficient condition depending on one bit of k_1. Let n be the number of sufficient conditions without counting the one on k_1.
2. Generate pairs of messages (M, M') satisfying $H_{k_2}(M') = H_{k_2}(M) + \Delta M$. This technique is detailed in Appendix A, which will be utilized in our own attack.
3. Send M and M' to the HMAC/NMAC oracle. Once roughly 2^n pairs of messages (M, M') are queried, if a collision is obtained, the outer key k_1 satisfies the sufficient condition. Otherwise, k_1 is very unlikely to satisfy the sufficient condition. So with 2^{n+1} queries, we will recover one bit of k_1.
4. Change ΔM and DP, and recover other bits of k_1.

The first two steps are the most important steps of the attack [5]. The main idea is to find a differential path with one sufficient condition on the outer key k_1. If k_1 satisfies the condition, a collision will be found with a suitable number of queries. Otherwise, no collision is likely to be found after the same number of queries. This will disclose bits of k_1.

However, if we divide the outer key k_1 as (k_a, k_b, k_c, k_d) for the computation of the outer MD4, then it turns out that such conditions can only be set on k_b and k_c, so the attack can not recover any of the bits of k_a and k_d.

3 Attacks on HMAC/NMAC with Near-Collisions

In this section, we give an overview of our new attacks on HMAC/NMAC based on near-collisions. A detailed description of the attacks will be given in respectively Section 4 for the MD4 case, and Section 5 for the MD5 case.

3.1 Overview

We first give an overview in the case of MD4. Thanks to [4], we can already assume that we know the inner key k_2 of HMAC/NMAC-MD4, and we want to recover the outer key k_1, which will be decomposed as four 32-bit variables k_a, k_b, k_c and k_d. Because MD4 uses the Davies-Meyer mode, we know that the 128-bit value of HMAC/NMAC-MD4 is exactly $(k_a + a, k_b + b, k_c + c, k_d + d)$, where a, b, c, d denote the final values of the four 32-bit intermediate values of the outer MD4.

The FLN attack [5] used an IV-dependent differential path for MD4 collisions, and derived bits of k_1 by observing whether or not collisions for the outer MD4 occurred. We will use a differential path for MD4 near-collisions which is independent of the IV, and we will collect near-collisions. These near-collisions are based on a local collision at some intermediate step of the MD4 compression function. Thanks to special properties of our differential path, we will be able to extract certain bits of (a, b, c, d), depending on the shape of the near-collision. Because of the Davies-Meyer mode, this will disclose certain bits of k_1.

Thus, the structure of our attack on HMAC/NMAC-MD4 is the following:

1. Determine a message difference ΔM and a differential path DP for a near-collision attack on MD4. Let n be the number of sufficient conditions.
2. Generate pairs of messages (M, M') satisfying $H_{k_2}(M') = H_{k_2}(M) + \Delta M$. We can use the FLN technique [5], described in Appendix A.
3. Send M and M' to the HMAC/NMAC-MD4 oracle. Once roughly 2^n pairs of messages (M, M') are queried, we obtain a near-collision.
4. Once a near-collision with (M, M') is obtained, we look at the shape of the near-collision: due to choice of our differential path, we know that certain shapes of near-collisions can only arise if certain bits of (a, b, c, d) are equal to 1 at the end of the computation of NMAC-MD4(M). This discloses bits of k_1 thanks to the Davies-Meyer mode.
5. Change ΔM and DP, and recover other bits of k_1.

Our related-key attack on NMAC-MD5 is based on similar ideas. We use the differential path of [3] associated to the dBB pseudo-collision. This differential path also gives rise to near-pseudo-collisions, that is, MD5(IV, M)

and MD5(IV', M) only differ by a few bits. Of course, instead of calling the NMAC-MD5 oracle on random messages M and M' such that $H_{k_2}(M') = H_{k_2}(M) + \Delta M$, we will call the NMAC-MD5 oracle on a randomly chosen M with two related keys corresponding to Δ^{MSB}. Because this does not use the inner key k_2, we will thus be able to recover bits of k_1 without knowing k_2.

3.2 Features

We summarize the main features of our attacks, compared to [5, 9]:

The HMAC/NMAC-MD4 case:

- Generating a near-collision requires much less queries than a collision. Compared to the FLN attack [5], the number of MAC queries is reduced to 2^{72} from 2^{88},
- Our MD4 near-collisions disclose more information than collisions. Indeed, we can recover bits of k_b, k_c and k_d, rather than just bits of k_b and k_c. Compared to the FLN attack [5], this discloses 51 bits of the outer key k_1, instead of only 22 bits. Hence, the number of offline MD4 computations is reduced to 2^{77} from 2^{95} (FLN attack decreased their offline complexity to 2^{95} from 2^{106} using some speeding up technique. Please refer to [5] for details.).

The NMAC-MD5 case:

- our attack does not require any control over the input messages, so our attack can partially recover the outer key k_1 without knowing the inner key k_2, unlike previous work. This might be of independent interest.
  We increase the number of online queries, but we can derive more information on the outer key: 63 bits of k_1 can be recovered, instead of only 28 bits [5, 9]. There is no standard calculation method of the total complexity. We will follow that of [9]: the sum of the online complexity and the offline complexity. Finally we recovered 53 bits of k_1 in order to make the online and the offline complexity be equal: 2^{75}. The total complexity of MD5 computations is reduced to 2^{76} from 2^{100}.

4 New Key Recovery Attack on HMAC/NMAC-MD4

We now precisely describe our new outer-key recovery attack on HMAC/NMAC-MD4. Recall that the outer key k_1 is decomposed as (k_a, k_b, k_c, k_d).

Denote the final values (after 48 steps) of the 32-bit intermediate values of the outer MD4 as (a_{48}, b_{48}, c_{48}, d_{48}). Then the output of HMAC/NMAC-MD4 is: $(h_a, h_b, h_c, h_d)=(k_a+a_{48}, k_b+b_{48}, k_c+c_{48}, k_d+d_{48})$. So we have the following relations when comparing two outputs of HMAC/NMAC-MD4:

$$\Delta h_a=\Delta a_{48},\ \Delta h_b=\Delta b_{48},\ \Delta h_c=\Delta c_{48} \text{ and } \Delta h_d=\Delta d_{48}.^3$$

[3] If two values differ at the MSBs, there will exist error probability. We will ignore such situations because they do not happen in our attack.

As a result, we can detect the difference propagation in the last four steps of the outer MD4 from the final output values of HMAC/NMAC. Based on this weakness of HMAC/NMAC-MD4 due to the Davies-Meyer mode, we will obtain bit-values of a_{48}, c_{48} and d_{48}. This, in turn, will disclose bits of k_a, k_c and k_d.

Our attack has both online work and offline work. We will first describe our near-collision on MD4. Then, we will explain details of online work and offline work.

4.1 Near-collisions on MD4

The main contribution of this paper is the use of near-collisions. Our near-collisions on MD4 are based on a local collision at step 29. We determine the message differences ΔM as $\Delta m_3{=}2^i$, that is, the messages only differ in m_3. The corresponding differential path is given in Appendix D. This differential path works for the cases $i = 3 \sim 5, 7 \sim 17, 20 \sim 25$: other values of i fail because of carry expansion.

The above near-collisions have the following properties:

- m_3 is used in step 45 of 3R. If the local collision in step 29 happens, the differences propagation in the last four steps will be significantly simplified.
- Because we use a local collision in step 29, we only need to consider the differential path until step 29. This reduces the number of sufficient conditions, and therefore the number of queries to obtain a near-collision.

4.2 Online work: obtaining bit-values of a_{48}, c_{48} and d_{48}

The procedure is as follows, where the message difference ΔM is $\Delta m_3 = 2^i$:

1. Generate pairs of messages (M, M') such that MD4(k_2, M')=MD4$(k_2, M)+\Delta M$. We adapt the technique proposed in [5], which is given in Appendix A.
2. Send such messages M and M' to the HMAC/NMAC-MD4 oracle to obtain any of the following three kinds of near-collisions:
   - Pairs $(M_a^i,\ M_a^{i}{}')$ such that $\Delta h_a{=}2^{i+3}$, $\Delta h_d{=}{*}2^{i+12}$ and $\Delta h_c{=}{*}2^{i+23} \pm 2^{i+14} \pm 2^{i+15}$; [4]
   - Pairs $(M_c^i, M_c^{i}{}')$ such that $\Delta h_a{=}2^{i+3}$, $\Delta h_d{=}{*}2^{i+12}$, $\Delta h_c{=}{*}2^{i+23} * 2^{i+14}$, and expected Δh_b;[5]
   - Pairs $(M_d^i,\ M_d^{i}{}')$ such that: $\Delta h_a{=}2^{i+3}$, $\Delta h_d{=}{*}2^{i+12}$ and $\Delta h_c{=}{*}2^{i+14} \pm 2^{i+23} \pm 2^{i+23}$.
3. Change the index i, and repeat steps 1 and 2 until all values of i are used.

First, let us observe that the above near-collisions are very likely to come from our differential path. Indeed, the shape of our near-collisions impose fixed differences on three 32-bit words, so a pair (M, M') chosen uniformly at random would

[4] $*$ means that the sign does not matter, and $\pm 2^{i+14} \pm 2^{i+15}$ means that the signs of these two differences are the same.

[5] Δh_b consists of $\pm 2^{i+6} \pm 2^{i+7}$

give such a near-collision with probability 2^{-96}. However, our pairs (M, M') chosen in step 1 have a much higher probability 2^{-64} to near-collide.[6]

We now claim that the messages obtained above with near-collisions satisfy the following conditions on the final values of the intermediate values of the outer MD4:

$$M_a^i\text{: } a_{48,i+3} = 1;\ M_c^i\text{: } c_{48,i+3} = 1;\ M_d^i\text{: } d_{48,i+3} = 1.$$

For instance, consider the case of M_a^i. Because of the near-collision, the difference propagation in 3R only exists in the last four steps. At step 47, the variable generated is c_{48}. And input differences only exist in a_{48} and d_{48}: $\Delta a_{48} = 2^{i+3}$ and $\Delta d_{48} = *2^{i+12}$. Since the number of the bits of the left rotation is 11, both $\pm 2^{i+14}$ and $\pm 2^{i+15}$ of Δc_{48} must be caused by 2^{i+3} of Δa_{48}. Such a difference propagation can not happen if there does not exist a carry during the calculation $a_{48}+2^{i+3}$, so the probability of $a_{48,i+3}$=1 is 1. With a similar reasoning, the messages M_c^i and M_d^i satisfy $c_{48,i+3} = 1$ and $d_{48,i+3} = 1$, respectively.

Finally, we can obtained near-colliding messages M_a^i such that $a_{48,i+3}$=1 for i=3 $\sim$ 5, 7 $\sim$ 15, 20 $\sim$ 25: other values of i fail because of carry expansion. In total, there are 18 near-colliding messages M_a^i, which can disclose values of $k_{a,i+3}$. Details are shown in section 4.3. So we can recover 18 bit-values of k_a by online work. Similarly, k_c and k_d are also partially recovered by online work. Near-colliding messages M_c^i and M_d^i are obtained for $i = 3 \sim 5$, $9 \sim 17$, $20 \sim 23$ and $i = 3 \sim 5$, $9 \sim 17$, $21 \sim 25$ respectively. So 16 bit-values of k_c and 17 bit-values of k_d, corresponding $k_{c,i+3}$ and $k_{d,i+3}$ of M_c^i and M_d^i respectively, can be recovered. In total, 51 bits of the outer key k_1 are recovered by the online work.

4.3 Offline work: recovering k_a, k_c and k_d

The way to recover k_a, k_c and k_d is the same. We will pick k_a as an example to explain the details:

1. Guess the values of $k_{a,i}$ for $i = 0 \sim 5$, 9, $19 \sim 22$, $29 \sim 31$: the index i that we fail obtaining M_a^{i-3}. These bit-values of k_a will be recovered by the offline exhaustive search. The total number of possibilities is 2^{14}.
2. Calculate other bits of k_a from the least significant to the most significant bits using M_a^i. First, the 6-th bit of k_a will be calculated using M_a^3.

   -Recovering $k_{a,6}$: compare $k_{a,5\sim 0}$ with $h_{a,5\sim 0}$. If $k_{a,5\sim 0} > h_{a,5\sim 0}$, there exists a carry from bit 5 to 6 during the computation of k_a+a_{48}. Otherwise, there will be no carry from bit 5 to 6 during the computation of k_a+a_{48}. Since $a_{48,6}$=1, the carry influence is known, and the value $h_{a,6}$ is known, so the value $k_{a,6}$ can be calculated.

   Then, the 7-th bit will be derived from M_a^4. Then the 8-th bit, and so on. Finally, all other bits of k_a will be recovered.

By a similar process, all the bits of k_c and k_d will be recovered.

[6] Details are shown in section 4.4.

4.4 Complexity analysis

As explained in section 4.2, we can obtain 18 bits, 17 bits and 16 bits of k_a, k_d and k_c using M_a^i, M_d^i and M_c^i, respectively. Totally 51 bits of k_1 are recovered by the online work, so the complexity of the offline exhaustive search is 2^{77} (2^{128-51}) MD4 computations.

Now we analyze the complexity of online work. This depends on the probability of the specified shape of near-collision, which can be regarded as two parts: probability of near-collision and that of specified difference propagation in the last four steps. The probability of our near-collisions is 2^{-60} since there are in total 60 conditions of differential path. The probabilities of difference propagation in the last four steps of outer MD4 are shown in Appendix B. One pair (M_a^i, M_a^i'), (M_c^i, M_c^i'), and (M_d^i, M_d^i') can be obtained with a probability $2^{-60} \times 1 \times \frac{2}{3} \times \frac{1}{9}$ (greater than 2^{-64}), $2^{-60} \times \frac{2}{3} \times \frac{4}{9} \times \frac{1}{4}$ (greater than 2^{-64}), and $2^{-60} \times \frac{2}{3} \times \frac{1}{9}$ (greater than 2^{-64}) respectively: one above pair can be obtained with roughly 2^{66} queries. As a result, the total online complexity is 51×2^{66} (less than 2^{72}) queries.

Experiment
It is impossible to carry out the real experiment. Instead, we separate the experiment to two parts:

-Confirm the correctness of DP: an example is shown in Appendix C.

-Confirm the correctness of key recovery technique by only focusing on the last four steps of outer MD4: the intermediate variables at step 44 and the message m_3 are randomly generated.

5 New Key Recovery Attack on NMAC-MD5

Similarly with MD4 case, we can detect the difference propagation in the last four steps of the outer MD5 from the final output values of HMAC/NMAC-MD5. It seems that our near-collision attack can be extended to HMAC/NMAC-MD5. However, we have not found suitable message difference and differential path for near-collision on MD5. Thanks to dBB pseudo-collision, where the difference propagation in the last four steps of the outer MD5 is very simple, we will be able to obtain bit-values of the intermediate values (after 64 steps) in the outer MD5 by detecting the shape of near-pseudo-collision or pseudo-collision. This, in turn, will disclose the outer key k_1.

In this section, we will explain the details of our outer-key recovery attack on NMAC-MD5 in the related-key setting: the attacker obtains $\mathrm{MD5}_{k_1}(\mathrm{MD5}_{k_2}(M))$ and $\mathrm{MD5}_{k_1'}(\mathrm{MD5}_{k_2}(M))$ denoted as NMAC and NMAC' respectively hereafter; k_1 and k_1' satisfy Δ^{MSB} defined in section 2.2. Recall that k_1 is decomposed as (k_a, k_b, k_c, k_d). Denote the intermediate variables (after 64 steps) in the outer MD5 as $(a_{64}, b_{64}, c_{64}, d_{64})$. Then the output of NMAC-MD5 is: $(h_a, h_b, h_c, h_d) = (k_a + a_{64}, k_b + b_{64}, k_c + c_{64}, k_d + d_{64})$.

Our new outer-key recovery attack consists of online work and offline work. The online work partially recovers k_a and k_c without knowledge of the inner key k_2, which might be of independent interest. The offline work is just the exhaustive search, where the inner key is necessary. We will first describe near-pseudo-collision on MD5. Then we will explain details of the online work. Since the offline work is just the exhaustive search, we will omit it.

5.1 Near-pseudo-collision on MD5

According to dBB pseudo-collision, once a local collision happens at step 63, the shape of near-pseudo-collision will depend on $a_{64,31}$ and $c_{64,31}$:

- if $a_{64,31} = c_{64,31}$: collision happens;
- if $a_{64,31} \neq c_{64,31}$: the final output differences are $\Delta h_a = 0$, $\Delta h_b = \pm 2^{20}$, $\Delta h_c = 0$ and $\Delta h_d = 0$.

So we can obtain the relation between $a_{64,31}$ and $c_{64,31}$ by detecting the shape of near-pseudo-collision. [7]

5.2 Online work: recovering $k_{a,31\sim 30}$ and $k_{c,31\sim 30}$

The procedure is as follows:

1. Generate messages randomly and send them to NMAC and NMAC' to obtain near-pseudo-colliding messages $\{M\}$, regrouped depending on the values of $h_{a,30}$ and $h_{c,30}$:
   - $\{M_0\}$: $h_{a,30} = 0$ and $h_{c,30} = 0$;
   - $\{M_1\}$: $h_{a,30} = 0$ and $h_{c,30} = 1$;
   - $\{M_2\}$: $h_{a,30} = 1$ and $h_{c,30} = 0$;
   - $\{M_3\}$: $h_{a,30} = 1$ and $h_{c,30} = 1$.
2. Determine relation between $k_{a,31}$ and $k_{c,31}$ based on each element of each sub-group utilizing the following tool:

   Tool: during $k_a + a_{64}/k_c + c_{64}$, if $h_{a,30}/h_{c,30} = 0$, there exists a carry from bit 30 to 31. Otherwise, there does not exist a carry from bit 30 to 31.

3. Check the results of step 2 for each sub-group. There should be only one sub-group that all elements have the same result, which will disclose $k_{a,31\sim 30}$ and $k_{c,31\sim 30}$ as follows:

   the result of step 2 is the real relation between $k_{a,31}$ and $k_{c,31}$;
   $k_{a,30} = 1 - h_{a,30}$; $k_{c,30} = 1 - h_{c,30}$.

First we will explain why the relation between $k_{a,31}$ and $k_{c,31}$ can be determined at step 2: the above tool determines the carry influence from bit 30 to 31 during $k_a + a_{64}/k_c + c_{64}$; the shapes of near-pseudo-collisions show the relation between $a_{64,31}$ and $c_{64,31}$; the relation between $h_{a,31}$ and $h_{c,31}$ is easy to check. Pick one pseudo-colliding element $m \in \{M_0\}$ as an example. We can obtain that $a_{64,31} = c_{64,31}$; there exists a carry from bit 30 to 31 during $k_a + a_{64}/k_c + c_{64}$. Consequently, the relation between $k_{a,31}$ and $k_{c,31}$ is determined as follows:

[7] Hereafter, we regard pseudo-collision as a special kind of near-pseudo-collision just for simplicity.

$h_{a,31} = h_{c,31} \Rightarrow k_{a,31} = k_{c,31}$;
$h_{a,31} \neq h_{c,31} \Rightarrow k_{a,31} \neq k_{c,31}$.

Then we will explain why only one sub-group does not have different results at step 2. This is because of the utilized tool. The error probability of the tool depends on the relation between $k_{a/c,30}$ and $h_{a/c,30}$.

- $k_{a/c,30} = h_{a/c,30}$: error probability is $\frac{1}{2}$. For example, if both values are 0, according to the tool, we will assume that there is always a carry from bit 30 to 31. However, in fact the carry influence depends on the value $a/c_{64,31}$: carry exists if $a/c_{64,30} = 1$, and no carry if $a/c_{64,30} = 0$. Since the value of $a/c_{64,30}$ is random, the error probability is $\frac{1}{2}$.
- $k_{a/c,30} \neq h_{a/c,30}$: error probability is 0. For example, if $k_{a/c,30} = 0$ and $h_{a/c,30} = 1$, we can obtain that $k_{a/c,30\sim0} < h_{a/c,30\sim0}$, so there will be no carry with probability 1, which is the same with the tool.

So only the sub-group satisfying $k_{a/c,30} \neq h_{a/c,30}$ should be without error. In other words, all elements of this sub-group have the same result at step 2. This also explains the way we recover $k_{a/c,31\sim30}$ at step 3.

5.3 Online work: recovering other bits of k_a and k_c

Since the way of recovering k_a is exactly the same with that of recovering k_c, we will pick k_a as an example in this section. The value of k_a is recovered from the most significant to the least significant bit. Suppose bits $k_{a,30\sim(i+1)}$ $(0 \leq i \leq 29)$ have been already obtained. The following procedure shows how to recover $k_{a,i}$.

1. Randomly generate messages and send them to the two NMACs until one message M_1 obtained satisfying the following three conditions:
   a) near-pseudo-collision happens;
   b) $h_{a,j}$=$k_{a,j}$ $(i + 1 \leq j \leq 30)$;
   c) $h_{c,30} \neq k_{c,30}$.
2. Determine the carry influence from bit i to $i + 1$ during $k_a + a_{64}$, where a_{64} is the intermediate value (after 64 steps) of the outer MD5 of $\text{MD5}_{k_1}($ $\text{MD5}_{k_2}(M_1))$.
3. Determine the value of $k_{a,i}$ by the result of step 2.
   -Carry: $h_{a,i}$=1 $\Rightarrow$ $k_{a,i}$=1;
   $h_{a,i}$=0 $\Rightarrow$ repeat steps 1 and 2.
   -No carry: $h_{a,i}$=0 $\Rightarrow$ $k_{a,i}$=0;
   $h_{a,i}$=1 $\Rightarrow$ repeat steps 1 and 2.

First, we can easily obtain the carry influence from bit 30 to 31 during $k_a + a_{64}$ based on conditions a) and c): condition a) guarantees that the relation between $a_{64,31}$ and $c_{64,31}$ can be determined; condition c) guarantees that the carry influence from bit 30 to 31 can be determined during $k_c + c_{64}$.

Then, we will obtain the carry influence from bit i to $i+1$ based on condition b): condition b) guarantees that the carry influence from bit i to $i + 1$ and that from bit 30 to 31 are the same during $k_a + a_{64}$.

Finally, we will recover the value of $k_{a,i}$: if there exists a carry from bit i to $i+1$ and $h_{a,i} = 1$, then $k_{a,i} = 1$with probability 1; if there does not exist a carry from bit i to $i + 1$ and $h_{a,i} = 0$, then $k_{a,i} = 0$with probability 1;

5.4 Complexity analysis

Near-pseudo-collision is with a rough probability 2^{-45} since there are in total 45 conditions until step 63 according to dBB pseudo-collision on MD5.

Complexity of recovering $k_{a,31\sim 30}$ and $k_{c,31\sim 30}$
As explained in section 5.2, the error probability of other sub-groups is $\frac{1}{2}$. So we need to generate four elements for each sub-group. To guarantee the attack will succeed, we will totally generate 32 elements for $\{M\}$. The complexity will be $32 \times 2^{46} = 2^{51}$ queries.

Complexity of recovering $k_{a,i}$ and $k_{c,i}$ $(0 \le i \le 29)$
Considering the complexity of recovering $k_{a,i}$ is the same with that of recovering $k_{c,i}$, we will pick $k_{a,i}$ as an example.

In section 5.3, it needs $2^{46} \times 2^{30-(i+1)+1} \times 2 = 2^{77-i}$ queries to obtain one message satisfying conditions a), b) and c) in step 1. According to steps 2 and 3, we might repeat step 1 twice. So totally the complexity is $2 \times 2^{77-i} = 2^{78-i}$ queries.

There is no standard calculation method of the total complexity. We will follow that of [9], which is the sum of the online and the offline complexity. If we will recover bits of $k_{a,30\sim i}$ and $k_{c,30\sim i}$, with roughly 2^{80-i} queries, the value of i should make the online and the offline complexity be equal: $2^{80-i} = 2^{128-(31-i)\times 2-1} \Rightarrow i = 5$. As a result, we will recover $k_{a,30\sim 5}$, $k_{c,30\sim 5}$ and the relation between $k_{a,31}$ and $k_{c,31}$. The online complexity is less than 2^{75} queries, and the offline complexity is 2^{75} MD5 computations.[8]

Experiment
It is impossible to carry out the real experiment. Similarly with HMAC/NMAC-MD4 case, we only focus on the last 4 steps of outer MD5, so we will randomly generate the intermediate variables at step 60 and messages m_2 and m_4.

6 Conclusion

This paper proposed new outer-key recovery attacks on HMAC/NMAC-MD4 and NMAC-MD5 (with related-key setting).

So far, no key-recovery attack has been published on HMAC/NMAC-MD5 without related-key setting. There are two reasons: (1) the inner-key recovery attack of Contini and Yin [4] can not succeed because all differential paths published so far have more than 128 sufficient conditions; (2) Wang *et al.*'s collision attack on MD5, multi-block collision, can no be used for the outer-key recovery attack, because the input message of the outer MD5 is the hash values of the inner MD5, just one-block length.

Our near-collisions may solve the second problem, since our near-collisions are only one-block length. Here we focus on the outer-key recovery attack, and assume that the inner key has been obtained. Moreover, our near-collisions are

[8] For the offline MD5 computations, we will assume the inner key k_2 has been obtained by the inner-key recovery attack of Contini and Yin [4].

easier to be obtained than collisions, only counting sufficient conditions until some intermediate step where a local collision happens.

As explained above, once the number of sufficient conditions of near-collision is less than 128, outer-key recovery attack might be a real attack on HMAC/NMAC-MD5 without related-key setting.

Acknowledgements

We would like to thank Phong Q.Nguyen for improving our paper and anonymous reviewers for helpful comments.

References

1. Mihir Bellare, Ran Canetti and Hugo Krawczyk. “Keying hash functions for message authentication.” CRYPTO 1996, LNCS, vol. 1109, pp. 1–15, 1996.
2. Mihir Bellare. “New Proofs for NMAC and HMAC: Security without Collision-Resistance.” CRYPTO 2006, LNCS, vol. 4117, pp. 602–619, 2006.
3. Bert den Boer and Antoon Bosselaers. “Collisions for the Compression Function of MD5.” EUROCRYPT 1993, LNCS, vol. 765, pp. 293–304, 1994.
4. Scott Contini and Yiqun Lisa Yin. “Forgery and partial key-recovery attacks on HMAC and NMAC using hash collisions.” ASIACRYPT 2006, LNCS, vol. 4284, pp. 37–53, 2006.
5. Pierre-Alain Fouque, Gaëtan Leurent and Phong Q.Nguyen, “Full Key-Recovery Attacks on HMAC/NMAC-MD4 and NMAC-MD5.” CRYPTO 2007, LNCS vol. 4622, pp. 13–30, 2007.
6. Jongsung Kim, Alex Biryukov, Bart Preneel and Seokhie Hong, “On the Security of HMAC and NMAC Based on HAVAL, MD4, MD5, SHA-0, and SHA-1.” SCN 2006, LNCS vol. 4116, pp. 242–256, 2006.
7. Alfred Menezes, Paul van Oorschot and Scott Vanstone, ”Handbook of Applied Cryptography.” CRC Press, 1997.
8. Christian Rechberger and Vincent Rijmen “Note on Distinguishing, Forgery and Second Preimage Attacks on HMAC-SHA-1 and a Method to Reduce the Key Entropy of NMAC.” Cryptology ePrint Archive, Report, 2006/290 (2006).
9. Christian Rechberger and Vincent Rijmen “On Authentication with HMAC and Non-Random Properties.” Financial Cryptography 2007, LNCS vol. 4886, pp. 39–57, 2007.
10. Ronald L. Rivest. “The MD4 Message-Digest Algorithm.” CRYPTO 1990, LNCS vol. 537, pp. 303–311, 1991.
11. Ronald L. Rivest “The MD5 Message Digest Algorithm.” Request for Comments (RFC 1321), Network Working Group, 1992.
12. Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen and Xiuyuan Yu. “Cryptanalysis of the Hash Functions MD4 and RIPEMD.” EUROCRYPT 2005, LNCS vol. 3494, pp. 1–18, 2005.
13. Xiaoyun Wang and Hongbo Yu. “How to Break MD5 and Other Hash Functions.” EUROCRYPT 2005, LNCS vol. 3494, pp. 19–35, 2005.
14. Xiaoyun Wang, Hongbo Yu and Yiqun Lisa Yin. “Efficient Collision Search Attacks on SHA-0.” CRYPTO 2005, LNCS vol. 3621, pp. 1–16, 2005.
15. Xiaoyun Wang, Yiqun Lisa Yin and Hongbo Yu. “Finding Collisions in the Full SHA-1.” CRYPTO 2005, LNCS vol. 3621, pp. 17–36, 2005.

A FLN attack: generating pairs of messages (M, M') that $H_{k_2}(M')$=$H_{k_2}(M)$+ΔM efficiently

In [5], Fouque *et al.* proposed an efficient way to generate pairs of messages (M, M') satisfying $H_{k_2}(M')$=$H_{k_2}(M)$+ΔM.[9] This technique works on hash functions that have the Merkle-Damgård structure. The procedure is as follows:

1. Generate one pair of one-block length messages (M_1, M_1') satisfying $H_{k_2}(M') = H_{k_2}(M) + \Delta M$ by birthday attack, where padding is not considered. Since the output of MD4 is 128-bit length, (M_1, M_1') will be obtained after roughly 2^{64} MD4 computation.
2. (M_1, M_1') will be extended to a family of two-block pair messages such that $H_{k_2}(M_1||M_2) = H_{k_2}(M_1'||M_2') + \Delta M$. The length of M_2 and M_2' must be no longer than 447 bits because of the padding rule.

   Selecting M_2 and M_2'
   Denote $H_{k_2}(M)$ and $H_{k_2}(M')$ as h_1 and h_1', respectively. we will obtain that $H_{k_2}(M_1||M_2) = H_{h_1}(M_2)$ and $H_{k_2}(M_1'||M_2') = H_{h_1'}(M_2')$. Denote intermediate chaining variables after 48 steps as ICV_{48}. $\mathrm{MD4}_{h_1}(M_2)$= h_1 + ICV_{48}. Similarly, $\mathrm{MD4}_{h_1'}(M_2')$= h_1' + ICV'_{48}. Since h_1'=h_1+ΔM, if $\mathrm{ICV}'_{48} = \mathrm{ICV}_{48}$, $\mathrm{MD4}_{h_1'}(M_2')$=$\mathrm{MD4}_{h_1}(M_2) + \Delta M$, so $\mathrm{MD4}_{k_2}(M_1||M_2)$=$\mathrm{MD4}_{k_2}(M_1'||M_2') + \Delta M$. As explained above, M_2 and M_2' should satisfy that $\mathrm{ICV}_{48} = \mathrm{ICV}'_{48}$. Such pair M_2 and M_2' can be obtained utilizing Wang *et al.*' collision attack on MD4. Please refer to [5] for more details.

B Probabilities of difference propagation in 3R

If near-collision happens, and the message difference ΔM is $\Delta m_3 = 2^i$.

-Δa_{48}=2^{i+3}: the probability is 1 except that bit i or $i+3$is MSB. During our attack, $i \leq 25$, so $i+3 \leq 28$.

-Δd_{48}=$*2^{i+12}$: the probability can be regarded as $\frac{2}{3}$. Δd_{48} depends on the bit carry expansion of Δa_{48} because f works bit-independently. f is XOR.

No carry with probability $\frac{1}{2}$: Δd_{48}=$*2^{i+12}$ with probability 1.
1-bit carry with probability $\frac{1}{4}$: Δd_{48}=$*2^{i+12}$ with probability $\frac{1}{2}$.
2-bit carries with probability $\frac{1}{8}$: Δd_{48}=$*2^{i+12}$ with probability $\frac{1}{4}$.

$$\vdots$$

So the probability is almost $\frac{1/2}{1-1/4}$=$\frac{2}{3}$.

Δc_{48}=$*2^{i+23} \pm 2^{i+14} \pm 2^{i+15}$: Similarly with analysis above, $*2^{23}$ of Δc_{48} is with probability $\frac{2}{3}$. $\pm 2^{i+14} \pm 2^{i+15}$ of Δc_{48} is with probability $\frac{1}{6}$. Totally, the probability is $\frac{2}{3} \times \frac{1}{6} = \frac{1}{9}$.

Δc_{48}=$*2^{i+23} * 2^{i+24}$: similarly with analysis above, the probability is $\frac{2}{3} \times \frac{2}{3}$=$\frac{4}{9}$.

[9] As shown in section 2.4, ΔM is determined differences of inner hash values instead of $M' - M$.

Δc_{48}=$*2^{i+14} \pm 2^{i+23} \pm 2^{i+24}$: similarly with analysis above, the probability is $\frac{2}{3} \times \frac{1}{6} = \frac{1}{9}$.

$\pm 2^{i+6} \pm 2^{i+7}$ of Δb_{48}: the probability of Δc_{48} with a carry is $\frac{1}{2}$, and the probability that Δf consists of $\pm 2^{i+23} \pm 2^{i+24}$ is $\frac{1}{2}$. Totally, the probability is $\frac{1}{4}$.

C An example of near-collision on HMAC/NMAC-MD4

In order to confirm the correctness of our differential path of near-collision on MD4, we will provide an example in Table 2. The messge difference is $\Delta m_3 = 2^3$.

Table 2. An example of near-collision

Outer key k_2	$k_a = 0xae23667d$; $k_b = 0x9ae8ba3c$; $k_c = 0x3775447e$; $k_d = 0x9614f6dc$
Near-colliding messages	$m_0 = 0x4bb5f397$; $m_1 = 0x9a645f8a$; $m_2 = 0x7f3529c4$; $m_3 = 0x1e7b831\bar{7}$
(output of the inner MD4)	$m'_0 = 0x4bb5f397$; $m'_1 = 0x9a645f8a$; $m'_2 = 0x7f3529c4$; $m'_3 = 0x1e7b831\bar{f}$
Step 29 of the outer MD4	$a_{29} = 0x84f021a1$; $b_{29} = 0x89f4c2d8$; $c_{29} = 0x62dbbc57$; $d_{29} = 0x76bdb3a6$

D DP and SCs of near-collision on MD4

The shown DP and SC is for Δm_3=2^3. DP and SC for other cases can be derived from this one by rotating all the bit differences and bit conditions. Cases $i = 3 \sim 5$, $7 \sim 17$, $20 \sim 25$ succeeds.

Table 3. DP and SCs

Step	Shift		Δb_i	
i	s_i	Δm_{i-1}	Numerical difference	Sufficient conditions
1	3			
2	7			
3	11			$b_{3,22} = b_{2,22}$
4	19	2^3	2^{22}	$b_{4,22} = 0$
5	3			$b_{5,22} = 0$
6	7			$b_{6,22} = 1$
7	11			$b_{7,9} = b_{6,9}$
8	19		2^9	$b_{8,9} = 0$
9	3			$b_{9,9} = 0$
10	7			$b_{10,9} = 1$
11	11			$b_{11,28} = b_{10,28}$
12	19		2^{28}	$b_{12,28} = 0$
13	3			$b_{13,28} = 0$
14	7			$b_{14,28} = 1$
15	11			$b_{15,15} = b_{14,15}$
16	19		2^{15}	$b_{16,15} = 0$
17	3			$b_{17,15} = b_{15,15}$
18	5			$b_{18,15} = b_{17,15}$
19	9			$b_{19,28} = b_{18,28}$, $b_{19,29} \neq b_{18,29}$, $b_{19,30} = b_{18,30}$
20	13		2^{28} $(28 \sim 30)$	$b_{20,0} = b_{19,0}$, $b_{20,28\sim29} = 1$, $b_{20,30} = 0$
21	3		-2^0	$b_{21,0} = 1$, $b_{21,28\sim30} = b_{19,28\sim30}$
22	5			$b_{22,0} = b_{20,0}$, $b_{22,28\sim30} = b_{21,28\sim30}$
23	9			$b_{23,0} = b_{22,0}$, $b_{23,9} = b_{22,9}$
24	13		2^9	$b_{24,3\sim8} = b_{23,3\sim8}$, $b_{24,9} = 0$
25	3		-2^3 $(3 \sim 9)$	$b_{25,3\sim8} = 0$, $b_{24,9} = 1$
26	5			$b_{26,3\sim8} = b_{24,3\sim8}$
27	9			$b_{27,3\sim8} = b_{26,3\sim8}$, $b_{27,9} \neq b_{26,9}$
28	13			
29	3	2^3		
30	5			

The symbol $i \sim j$ for numerical difference means difference propagates from bit i to j. The symbol $i \sim j$ for sufficient conditions means all bits from i to j.