Cryptanalysis on HMAC/NMAC-MD5 and MD5-MAC*

Xiaoyun Wang[1,2], Hongbo Yu[1], Wei Wang[2],
Haina Zhang[2], and Tao Zhan[3]

[1] Center for Advanced Study, Tsinghua University, Beijing 100084, China
{xiaoyunwang,yuhongbo}@mail.tsinghua.edu.cn
[2] Key Laboratory of Cryptographic Technology and Information Security,
Ministry of Education, Shandong University, Jinan 250100, China
{wwang,hnzhang}@math.sdu.edu.cn
[3] Shandong University, Jinan 250100, China
zhantao@sdu.edu.cn

Abstract. In this paper, we present the first distinguishing attack on HMAC and NMAC based on MD5 without related keys, which distinguishes the HMAC/NMAC-MD5 from HMAC/NMAC with a random function. The attack needs 2^{97} queries, with a success probability 0.87, while the previous distinguishing attack on HMAC-MD5 reduced to 33 rounds takes $2^{126.1}$ messages with a success rate of 0.92. Furthermore, we give distinguishing and partial key recovery attacks on MDx-MAC based on MD5. The MDx-MAC was proposed by Preneel and van Oorschot in Crypto'95 which uses three subkeys derived from the initial key. We are able to recover one 128-bit subkey with 2^{97} queries.

Keywords: HMAC, NMAC, MDx-MAC, MD5, Distinguishing attack, Key recovery.

1 Introduction

Many cryptographic schemes and protocols use hash functions as primitives. In recent work [3,4,16,17,18,19], devastating collision attacks on hash functions from the MD4 family were discovered. Such attacks have undermined the confidence in the most popular hash functions such as MD5 or SHA-1, and raise the interest in reevaluating the actual security of the Message Authentication Code (MAC) algorithms based on them [7,6,9].

HMAC and NMAC are hash-based message authentication codes proposed by Bellare, Canetti and Krawczyk [1]. NMAC is the theoretical foundation of HMAC, and HMAC has been implemented in widely used protocols including SSL/TLS, SSH and IPsec. The security of NMAC and HMAC has been carefully analyzed in [1,2]. It was proved that NMAC is a pseudo-random function family (PRF) under the assumption that the compression function of the keyed hash function is a

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-01001-9_35

* Supported by the National Natural Science Foundation of China (NSFC Grant No. 60525201 and No. 90604036) and 973 Project (No.2007CB807902 and No.2007CB807903).

A. Joux (Ed.): EUROCRYPT 2009, LNCS 5479, pp. 121–133, 2009.

PRF. This proof can be extended to HMAC by an additional assumption: the key derivation function in HMAC is a PRF. However, if the underlying hash function is weak (such as MD5), the above proofs may not work.

There are three types of the attacks on HMAC/NMAC, namely, distinguishing attacks, existential forgery attacks and universal forgery attacks. Distinguishing attacks can be divided into *distinguishing-R* and *distinguishing-H* attacks [9], where distinguishing-R attack means distinguishing HMAC/NMAC from a random function, and distinguishing-H attack detects instantiated HMAC/NMAC (by existing hash functions) from HMAC/NMAC with a random function. A general distinguishing-R attack on HMAC using the birthday paradox was introduced by Preneel and van Oorschot [10]. This attack requires about $2^{\frac{l}{2}}$ messages and works with probability 0.63, where l is the length of the initial value.

In this paper, we focus on the distinguishing-H attack on HMAC/NMAC-MD5 that checks which cryptographic hash function is embedded in HMAC/NMAC. For simplicity, we call it distinguishing attack. In [9], Kim *et al.* introduced two kinds of distinguishers of the HMAC structure, the differential distinguisher and the rectangle distinguisher, and used them to analyze the security of HMAC based on HAVAL, MD4, MD5, SHA-0 and SHA-1. For MD5 reduced to 33 rounds, they described a distinguishing attack taking $2^{126.1}$ messages with a success probability 0.92. Using the pseudo-collisions found by den Boer and Bosselaers [5], Contini and Yin [6] proposed a related-key distinguishing attack on NMAC-MD5 with 2^{47} queries and a success probability of 0.25. They applied it to construct forgery and partial key-recovery attacks on NMAC-MD5. Fouque *et al.* [7] presented the first full-key recovery attack on HMAC/NMAC-MD4, and extended Contini and Yin's attack to the full key-recovery attack on NMAC-MD5. The latter was independently found by Rechberger and Rijmen [11,12] with better results than [7] by ignoring the conditions in the last 5 steps. They also proposed a full key-recovery attack in the related-key setting on NMAC with SHA-1 reduced to 34 steps, and improved the attack on HMAC instantiated with reduced SHA-1 variants of more steps in [12]. Recently, Wang *et al.* [15] suggested more efficient full key-recovery attacks on HMAC/NMAC-MD4 and NMAC-MD5 using near-collisions. However, all the attacks on NMAC-MD5 are in the related-key setting, hence these attacks can not be applied to the corresponding HMAC.

In this paper, we are able to get rid of related keys, and propose the first distinguishing attacks on HMAC/NMAC-MD5. Based on the dBB pseudo-collision [5], we search for a new kind of collision which is called a dBB collision. With the specific structure and high probability of a dBB collision, we can successfully distinguish a dBB collision from other random collisions found by the birthday attack. Once a dBB collision is detected, the distinguisher outputs HMAC/NMAC-MD5; otherwise, it outputs HMAC/NMAC with a random function. The attack needs 2^{97} queries in total, and the success rate is 0.87.

Another contribution of this paper is to introduce distinguishing and partial key recovery attacks on the MDx-MAC based on MD5 called the MD5-MAC. The MDx-MAC was proposed by Preneel and van Oorschot in Crypto'95 [10] which transforms any MD4-family hash function into a MAC algorithm. It uses three subkeys K_0, K_1 and K_2 which are derived from an original key K. The role of K_0 and K_2 is similar to the two secret keys in the envelope MAC construction, and four 32-bit words of K_1 are added to all the step operations of four rounds respectively. The dBB collision can be used not only to distinguish MD5-MAC directly, but also to capture the full subkey K_1 which is involved in the collision path. The number of queries in the attack is about 2^{97}.

This paper is organized as follows. Background and definitions are recalled in Section 2. In Section 3, we first introduce a distinguishing attack on the keyed IV MAC, which is an adaptive chosen message attack, and then extend it to distinguish HMAC/NMAC-MD5 from HMAC/NMAC with a random function. In Section 4, we present distinguishing and key recovery attacks on MD5-MAC. Finally, Section 5 concludes the paper.

2 Background and Definitions

2.1 Notations

H	:	a hash function
$\overline{H}$	:	a hash function without padding
h	:	a compression function
$x\|y$	:	concatenation of the two bitstrings x and y
$+$	:	addition modular 2^{32}
$\oplus$	:	bitwise exclusive OR
$\vee$	:	bitwise OR
$\wedge$	:	bitwise AND
$\lll s$	:	left-rotation by s-bit

2.2 Brief Description of MD5

MD5 [13] is a hash function proposed by Rivest as a strengthened version of MD4. It takes an arbitrary length message and produces a 128-bit hash value. First, the input message M is padded to be $\overline{M}$, a multiple of 512 bits. Suppose the length of M in bits is l. Append the bit “1” to the end of the message followed by k “0” bits, where k is the smallest non-negative integer such that $l + 1 + k = 448 \mod 512$. Then append the 64-bit block that is equal to the number l expressed using a binary representation. The padded message $\overline{M}$ is then divided into 512-bit message blocks, i.e., $\overline{M} = (M_0, \ldots, M_{n-1})$, and processed by Merkle-Damgård iterative structure. Each iteration invokes a compression function which takes a 128-bit chaining value and 512-bit message block as inputs, and outputs a 128-bit value as the hash value of this iteration.

The compression function has four rounds. Every round has 16 steps and employs a round function. For the padded message $\overline{M}$ with n blocks, the hash function is performed n iterations in total. The k-th iteration is the following:

- Input: 512-bit message $M_{k-1} = (m_0, m_1, \cdots, m_{15})$ and a 128-bit chaining value $(A_0, B_0, C_0, D_0) = CV_{k-1}$, where CV_{k-1} is the output of $(k-1)$-th iteration.
- Step update: For $0 \leq i \leq 15$,

$$\begin{aligned}
A_{i+1} &= B_i + (A_i + f(B_i, C_i, D_i) + w_{4i} + c_{4i}) \lll s_{4i}, \\
D_{i+1} &= A_{i+1} + (D_i + f(A_{i+1}, B_i, C_i) + w_{4i+1} + c_{4i+1}) \lll s_{4i+1}, \\
C_{i+1} &= D_{i+1} + (C_i + f(D_{i+1}, A_{i+1}, B_i) + w_{4i+2} + c_{4i+2}) \lll s_{4i+2}, \\
B_{i+1} &= C_{i+1} + (B_i + f(C_{i+1}, D_{i+1}, A_{i+1}) + w_{4i+3} + c_{4i+3}) \lll s_{4i+3},
\end{aligned}$$

  where for $0 \leq j \leq 63$, w_j is one of the 16 message words, c_j and s_j are step-dependent constants, f is a round-dependent Boolean function as follows:

$$\begin{array}{ll}
f(x, y, x) = (x \wedge y) \vee (\neg x \wedge z) & \text{if } 0 \leq i \leq 3, \\
f(x, y, x) = (x \wedge z) \vee (y \wedge \neg z) & \text{if } 4 \leq i \leq 7, \\
f(x, y, x) = x \oplus y \oplus z & \text{if } 8 \leq i \leq 11, \\
f(x, y, x) = y \oplus (x \vee \neg z) & \text{if } 12 \leq i \leq 15.
\end{array}$$

- Output: $CV_k = (A_0 + A_{16}, B_0 + B_{16}, C_0 + C_{16}, D_0 + D_{16})$.

$CV_n = H(M)$ is the hash value, and CV_0 is the initial value.

2.3 Pseudo-Collisions of MD5

Our attacks are based on the dBB pseudo-collisions found by den Boer and Bosselaers [5], which satisfy the following relations:

$$h(IV, M) = h(IV', M), \tag{1}$$

$$IV \oplus IV' = (2^{31}, 2^{31}, 2^{31}, 2^{31}) = \Delta^{\text{MSB}}, \tag{2}$$

$$\text{MSB}(B_0) = \text{MSB}(C_0) = \text{MSB}(D_0), \tag{3}$$

where M is a one-block message, and MSB means the most significant bit. The probability of the dBB pseudo-collision is 2^{-46}. The specific differential path for the dBB pseudo-collision is shown in Table 1. We call the relations (2) and (3) *dBB conditions*. Now we define an important type of collision as the dBB collision:

dBB collision: A collision of two-block messages $(x\|y, x'\|y)$ is called a dBB collision if

1. Let $CV = h(IV, x)$ and $CV' = h(IV, x')$. The pair (CV, CV') satisfies the dBB conditions.
2. (CV, y) and (CV', y) compose a dBB pseudo-collision, i.e., $h(CV, y) = h(CV', y)$.

2.4 Secret Prefix MAC, HMAC and NMAC

A MAC algorithm is a hash function with a secret key K as the secondary input. HMAC and NMAC are two popular MAC algorithms which are all derived from efficient hash functions. Another three earlier hash based MACs are constructed by the *Secret Prefix Method*, *Secret Suffix Method* and *Envelope Method.*

The secret prefix method was proposed in the 1980s, and was suggested for MD4 independently in [14] and [8]. The secret prefix MAC is constructed as:

$$\text{Secret-Prefix-MAC}_K(M) = H(K\|M).$$

If the key K (maybe padded) is a full block, secret prefix MAC is equivalent to a hash function with a secret IV, which is called the keyed IV MAC. We denote this kind of MAC construction based on MD5 as *KIMAC-MD5* which is the basic design unit for HMAC/NMAC-MD5.

The NMAC function, on input message M and a pair of 128-bit independent keys (K_1, K_2), is defined as:

$$\text{NMAC}_{(K_1,K_2)}(M) = H_{K_1}(H_{K_2}(M)).$$

In fact, the outer function acts on the output of the iterated hash function, and thus involves one iteration of the compression function. That is to say, the outer function is basically the compression function h_{K_1} acting on $H_{K_2}(M)$ which has been padded to a full block size.

Since the NMAC replaces the fixed IV in H with a secret key, this requires a modification of existing implementation of the underlying hash function. The construction of HMAC is motivated to avoid this problem, and still uses the usual fixed IV. On input message M and a single secret key K, HMAC is computed as:

$$\text{HMAC}_K(M) = H(\overline{K} \oplus \mathit{opad}\|H(\overline{K} \oplus \mathit{ipad}\|M)),$$

where $\overline{K}$ is the completion by adding "0"s of K to a full block of the iterated hash function, and *opad* and *ipad* are two one-block length constants.

Basically, HMAC_K is the same as $\text{NMAC}_{(h(\overline{K}\oplus opad),h(\overline{K}\oplus ipad))}$. For simplicity, we denote the $\text{HMAC}_K(M)$ by $H_{out}(H_{in}(M))$.

2.5 Description of MD5-MAC

The MDx-MAC was proposed by Preneel and van Oorschot in Crypto'95 [10], which converts MDx-family hash functions into MAC algorithms with a key K up to 128 bits. The underlying hash function can be any of MD5, RIPEMD, SHA, or other similar algorithms except MD4. For convenience, we denote the MDx-MAC based on MD5 as MD5-MAC.

Let $\overline{MD5}$ denote MD5 algorithm without padding. The 128-bit secret key K is expanded to three 128-bit subkeys K_0, K_1 and K_2 by the following procedure.

For $i = 0$ to 2, $K_i = \overline{MD5}(K||U_i||K)$, where U_0, U_1 and U_2 are three different 96-byte constants (See [10] for details). The MD5-MAC is then obtained from MD5 with the following modifications:

1. The initial value IV of MD5 is replaced by K_0.
2. The key K_1 is split into four 32-bit words denoted by $K_1[i]$ $(0 \leq i \leq 3)$ which are added to the constants used in round i of each MD5 iteration respectively.

3. Following the block containing the padding and appended length as defined by MD5, an additional 512-bit block of the following form
$$\overline{K_2} = K_2||K_2 \oplus T_0||K_2 \oplus T_1||K_2 \oplus T_2$$
is appended, where T_i $(0 \leq i \leq 2)$ are three 128-bit constants.
4. The MAC value is the leftmost m bits of the hash value.

In [10], the authors recommended $m = n/2$ for most applications. For our attack, we assume $m = n$.

3 Distinguishing Attacks on HMAC/NMAC-MD5

To describe the distinguishing attack on HMAC/NMAC-MD5, we start with a distinguishing attack on KIMAC-MD5 which is an adaptive chosen message attack.

3.1 Adaptive Chosen Message Attack on KIMAC-MD5

The core of the attack is to find a pair (x, x') whose output difference satisfies the dBB conditions, and to detect whether x and x' can lead to a dBB collision by appending 2^{47} y separately with a reasonable probability.

The adversary performs the following steps:

1. Generate a structure of 2^{66} random messages, and query the MACs of these messages. We assume that the MAC algorithm is either a KIMAC-MD5 or KIMAC with a random function (KIMAC-RF).
2. Use the birthday attack [20] to find two messages (x, x') where $(H_K(x), H_K(x'))$ satisfies the dBB conditions.
3. Let $pad(pad')$ be the padding for $x(x')$. Append 2^{47} different messages y to the messages $x\|pad$ and $x'\|pad'$ respectively, and query the MACs with the two sets of 2^{47} messages.
4. If a collision $(x\|pad\|y, x'\|pad'\|y)$ is found, output the MAC as KIMAC-MD5. Otherwise, the MAC is KIMAC-RF.

The data complexity of the attack is $2^{66} + 2 \cdot 2^{47} \approx 2^{66}$ chosen messages. Since we can use the birthday attack to search pairs satisfying dBB-conditions, the time complexity is dominated by the size of the structure in Step 1 (the data collection phase), which is about 2^{66} queries, i.e., 2^{66} MAC computations.

Now let us analyze the success probability of this attack. From the above process, we observe that, when a specific message pair (x, x') is found in step 2, our attack succeeds in the following cases. If the KIMAC is based on MD5, a message y such that $H_K(x\|pad\|y) = H_K(x'\|pad'\|y)$ is searched. Otherwise, if the KIMAC is KIMAC-RF, no collision is found. The detailed computation of the probability is as follows.

For two random messages x and x', the output difference $H_K(x) \oplus H_K(x')$ satisfies the dBB conditions with probability:

$$\frac{1}{2^{128}} \times \frac{1}{4} = \frac{1}{2^{130}}.$$

According to the birthday paradox and Taylor series expansion, no matter what kind of oracle MAC is, among the 2^{66} messages, we can find a message pair (x, x') satisfying the dBB conditions with probability

$$q \approx 1 - (1 - \frac{1}{2^{130}})^{C^2_{2^{66}}} \approx 1 - e^{-2} \approx 0.86.$$

For KIMAC-MD5, the collision in step 4 happens with higher probability 2^{-46} instead of the average probability 2^{-128}. So, when the KIMAC is based on MD5, we can find a collision among 2^{47} adaptive chosen messages in Step 4 with probability $p_1 = 1 - (1 - \frac{1}{2^{46}})^{2^{47}} \approx 0.86$. Otherwise, a collision occurs for KIMAC-RF with a low probability $p_2 = 1 - (1 - \frac{1}{2^{128}})^{2^{47}} \approx 0$. Hence, the success rate of this attack is

$$\begin{aligned} & q \times [\frac{p_1}{2} + (\frac{1-p_2}{2})] \\ & \approx 0.86 \times (0.86 \times \frac{1}{2} + \frac{1}{2}) \\ & \approx 0.80. \end{aligned}$$

The success rate can be improved by repeating the attack several times.

3.2 Adaptive Chosen Message Attack on HMAC-MD5

The above attack cannot be applied to HMAC-MD5 directly due to the fact that the dBB collision of H_{in} is concealed by the outer level hashing H_{out}. However, we can discard all other collisions by some concrete detective techniques, and save the dBB collisions.

Suppose that we get a collision of HMAC which has the form $(x\|y, x'\|y)$. Denote $\overline{H}_{in}(x)$ as CV, and $\overline{H}_{in}(x')$ as CV', for simplicity. Let $\Delta CV = CV \oplus CV'$. The key of our attack is to distinguish the dBB collisions according to the relation of $\overline{H}_{in}(x)$ and $\overline{H}_{in}(x')$:

1. *Internal collision*: If $\Delta CV = 0$, $(x\|y, x'\|y)$ is called an internal collision.
2. *External collision*: If $\Delta CV \neq 0$, $(x\|y, x'\|y)$ is an external collision. Furthermore, when ΔCV satisfies the dBB conditions, and (CV, y) and (CV', y) compose a dBB pseudo-collision, $(x\|y, x'\|y)$ is a dBB collision. Otherwise, the collision is a non-dBB external collision.

The adversary performs as follows:

1. Generate 2^{89} one-block messages x randomly, and append a fixed 447-bit message y (taking padding into consideration) to each x. Query all the messages $x\|y$ to get their MACs.

2. Find all the collided messages $(x\|y, x'\|y)$ satisfying $\mathrm{HMAC}_K(x\|y) = \mathrm{HMAC}_K\ (x'\|y)$. Note that on average, there are 2^{49} internal collisions, 2 dBB collisions and 2^{50} non-dBB external collisions.
3. For all the collisions $(x\|y, x'\|y)$ collected in step 2 , we append $y' \neq y$ to x and x', query $(x\|y', x'\|y')$, and check if they collide. This way, the internal collisions can be detected. In the next step, we only need to distinguish the dBB collisions from the non-dBB external collisions.
4. For the remaining collisions, append 2^{47} different $y' \neq y$ to x and x', respectively, query the MACs for $x\|y'$ and $x'\|y'$, and check whether a collision occurs. Once a collision $(x\|y', x'\|y')$ is found, we conclude that the original collision $(x\|y, x'\|y)$ is a dBB collision, and output the MAC is HMAC-MD5. Otherwise, the MAC is a HMAC-RF.

Complexity evaluation

There are at most 2^{177} pairs produced by 2^{89} messages, so the expected number of internal collisions is $2^{177-128} = 2^{49}$. Similarly, the expectation of non-dBB external collisions is $2^{49} + 2^{49} = 2^{50}$ where 2^{49} collisions occur after H_{in} and other 2^{49} collisions occur after H_{out}. For two messages x and x', the output difference $h_K(x) \oplus h_K(x')$ satisfies the dBB conditions with probability 2^{-130}. Consequently, there are $2^{177-130} = 2^{47}$ pairs satisfying the dBB conditions, and about $2^{47-46} = 2$ of them are dBB collisions.

In step 1, the data complexity is 2^{89}. We keep a table of 2^{89} entries in step 2, finding $2^{49} + 2^{50} + 2$ collisions needs about 2^{89} table lookups. In step 3, the time complexity is about $2^{49} + 2^{50} + 2 \approx 2^{50.58}$ MAC computations. In step 4 , both the data and time complexity are about $(2^{50} + 2) \times 2^{47} \approx 2^{97}$.

Therefore, the total time complexity of attack is about 2^{97} MAC computations and 2^{89} table lookups, and data complexity is about 2^{97} chosen messages.

Success rate

As analyzed in section 3.1, we divide the success rate into two parts:

- If the MAC is HMAC-MD5, the attack succeeds when a dBB collision is found among $2^{50.58}$ collisions.

  The probability that there exists a dBB collision among $2^{50.58}$ collisions is

  $$1 - (1 - \frac{1}{2^{130+46}})^{2^{177}} \approx 1 - e^{-2} \approx 0.86.$$

  The probability that the dBB collision can be detected in step 4 is about

  $$1 - (1 - \frac{1}{2^{46}})^{2^{47}} \approx 1 - e^{-2} \approx 0.86.$$

  Thus, if the MAC is HMAC-MD5, the attack can find a dBB collision with probability $0.86 \times 0.86 = 0.74$.
- If the MAC is HMAC-RF, the attack succeeds when no dBB collision is detected. The success probability is about

  $$((1 - \frac{1}{2^{128}})^{2^{47}})^{2^{50}} \approx 1.$$

Therefore, the success rate of the whole attack is about

$$\frac{1}{2} \times 0.74 + \frac{1}{2} \times 1 = 0.87.$$

3.3 Chosen Message Attack on HMAC-MD5

In this subsection, we relax the adaptive chosen message attack to a chosen message attack. The data complexity will increase up to 2^{113}, but the table size is reduced from original 2^{89} to 2^{66} entries.

The chosen message distinguishing attack on HMAC-MD5 is described as follows:

1. Select a set of 2^{66} one-block messages x at random. Append a chosen 447-bit message y to all the x, and form a structure of 2^{66} messages $x\|y$. Choose 2^{47} different messages y to produce 2^{47} structures. Make 2^{113} MAC queries for all of the structures.
2. For each structure, fulfill the birthday attack [20] to find all the collisions $(x\|y, x'\|y)$ satisfying $\mathrm{HMAC}_K(x\|y) = \mathrm{HMAC}_K(x'\|y)$.
3. For each collision $(x\|y, x'\|y)$ found in step 2, we determine the type of the collision.
   - Check whether all the pairs $(x\|y', x'\|y')$ in other structures are collisions. If all other pairs $(x\|y', x'\|y')$ collide, then $(x\|y, x'\|y)$ is an internal collision.
   - Check whether there exists at least one y' such that $(x\|y', x'\|y')$ is a collision in another structure. If so, we conclude that $(x\|y, x'\|y)$ is a dBB collision, and the MAC is HMAC-MD5. If there is no dBB collision, the MAC is HMAC-RF.

It is clear that the attack needs about 2^{113} chosen messages. For each structure, the expectation is 8 internal collisions and 16 external collisions. So the total number of collisions in all 2^{47} structures is about $24 \times 2^{47} < 2^{52}$. For each collision, 2^{47} table lookups are needed. Therefore the time complexity is less than $2^{52} \times 2^{47} = 2^{99}$ table lookups, and the table size is 2^{66} entries.

The computation of success rate is the same as in subsection 3.2.

Application to NMAC: NMAC is a generalized version of HMAC as introduced in subsection 2.3. Since the above attack on HMAC-MD5 has no relation with the secret key, hence it can be applied to NMAC-MD5 directly.

4 Partial Key Recovery Attack on the MD5-MAC

Obviously, the distinguishing attack on HMAC/NMAC-MD5 described in Section 3 is also applicable to distinguish the MD5-MAC from the MDx-MAC based on a random functions.

It should be noted that our distinguishing attack on HMAC/NMAC-MD5 can not be extended to recover the inner keys. Even though the partial bits of intermediate states of the second block y can be recovered using the method in [6], we can not derive the inner keys of HMAC/NMAC by the backward computation because of the existence of the first block x. For the MDx-MAC, the situation is different since its secret keys are not only involved in the beginning (IV) and the end, but also in every iteration. We are able to recover the second secret key K_1 involved in every iteration.

This process can be divided into three phases. The first phase is to find a dBB collision. Note that by the techniques described in section 3, it's easy to find a dBB collision $(x\|y, x'\|y)$ with the complexity of 2^{97} MAC computations. The second phase is to recover some bits of the intermediate states by the given dBB collision $(x\|y, x'\|y)$. The third phase is to recover the secret key K_1.

4.1 Recovering Some Bits of the Intermediate States

We can use the bit carry method of [6] to recover 255 bits of the intermediate chaining variables of the second block y. Let $y = y[0]y[1]...y[15]$ where each $y[i]$ is a 32-bit words. Table 1 lists the dBB differential path. The 6-th column of the table contains sufficient conditions that guarantee the dBB differential holds, and the last column lists the recovered 255 bits of the first round. The complexity of this part is less than $2^{46} \times 255 \approx 2^{54}$ MAC computations.

4.2 Recovering the 128-Bit Subkey K_1

We implement the *divided and conquer* attack to recover the 32-bit subkeys $K_1[0]$, $K_1[1]$, $K_1[2]$ and $K_1[3]$ separately.

1. **Recovering the 32-bit subkey** $K_1[0]$
   From Table 1, 95 bits in the first five variables A_1, D_1, C_1, B_1 and A_2 can be recovered. We guess the other 65 unknown bits, and for each guess, we compute $K_1[0]$, D_2, C_2, B_2, A_3 and D_3 successively.

$$\begin{aligned}
K_1[0] &= (A_2 - B_1) \ggg s_0 - A_1 - f_1(B_1, C_1, D_1) - y'[0] - C_0\\
D_2 &= A_2 + (D_1 + f(A_2, B_1, C_1) + y[1] + c_1 + K_1[0]) \lll s_1\\
C_2 &= D_2 + (C_1 + f(D_2, A_2, B_1) + y[2] + c_2 + K_1[0]) \lll s_2\\
B_2 &= C_2 + (B_1 + f(C_2, D_2, A_2) + y[3] + c_3 + K_1[0]) \lll s_3\\
A_3 &= B_2 + (A_2 + f(B_2, C_2, D_2) + y[4] + c_4 + K_1[0]) \lll s_4\\
D_3 &= A_3 + (D_2 + f(A_3, B_2, C_2) + y[5] + c_5 + K_1[0]) \lll s_5
\end{aligned}$$

   If the 90 bits of D_2, C_2, B_2, A_3 and D_3 are consistent with the corresponding recovered bits in Table 1, we get the right secret key $K_1[0]$ and A_1, D_1, C_1, B_1 and A_2. In this way, we can compute all the chaining variable values A_i, B_i, C_i and D_i $(1 \leq i \leq 4)$ in the first round.

2. **Recovering the 32-bit subkey** $K_1[1]$
   Guess the 32-bit key $K_1[1]$. For each candidate $K_1[1]$, we compute A_i, B_i, C_i and D_i $(5 \leq i \leq 8)$ using the known values (A_4, D_4, C_4, B_4) and message

y. If $K_1[1]$ is the right key, then A_i, B_i, C_i, and D_i $(5 \leq i \leq 8)$ will satisfy all the 15 conditions in steps 17-33 of Table 1 with probability 1. Otherwise, A_i, B_i, C_i, and D_i $(5 \leq i \leq 8)$ will satisfy the 15 conditions with probability 2^{-15}. In this way, there are about $2^{32} \cdot 2^{-15} = 2^{17}$ candidates $K_1[1]$ left. It only needs two other dBB collisions $(x\|y', x'\|y')$ to discard the wrong $K_1[1]$, and capture the right one from the 2^{17} candidates. To find two other dBB collisions takes about 2^{47} MAC computations.

3. **Recover the 32-bit subkeys $K_1[2]$ and $K_1[3]$**
   From Table 1, we know that there is only one condition in the third round. This means that at most 33 dBB collisions (colliding pairs have the common first block (x, x')) are needed to filter the wrong keys from a 2^{32} key space and obtain the right key $K_1[2]$. Similarly, as there are 15 conditions in the 4-th round, 3 dBB collisions are required to determine the right $K_1[3]$.

Overall, the complexity of the key recovery is dominated by 2^{97} queries, which is the complexity of finding a dBB collision.

5 Conclusions

In this paper, we utilize the dBB pseudo-collisions to construct dBB collisions which have the dBB specific structure and differential path with high probability. The specific structure can be used to construct a distinguishing attack on HMAC/NMAC-MD5, with 2^{97} queries and 2^{97} MAC computations under adaptive chosen message attack. Under chosen message attacks, the complexities is up to 2^{113} queries and 2^{99} table lookups. For MD5-MAC, the specific differential path can be used to recover the subkey involved in the differential path with complexity of 2^{97} queries.

Acknowledgements. We would like to thank Christian Rechberger and three reviewers for their very helpful comments on the paper. We also hope to thank Guangwu Xu for revising the paper during his stay in Tsinghua University.

References

1. Bellare, M., Canetti, R., Krawczyk, H.: Keying hash functions for message authentication. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 1–15. Springer, Heidelberg (1996)
2. Bellare, M.: New Proofs for NMAC and HMAC: Security without collision-resistance. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 602–619. Springer, Heidelberg (2006)
3. Biham, E., Chen, R.: Near-collisions of SHA-0. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 290–305. Springer, Heidelberg (2004)
4. Biham, E., Chen, R., Joux, A., Carribault, P., Lemuet, C., Jalby, W.: Collisions of SHA-0 and reduced SHA-1. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 36–57. Springer, Heidelberg (2005)

5. den Boer, B., Bosselaers, A.: Collisions for the compression function of MD5. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 293–304. Springer, Heidelberg (1994)
6. Contini, S., Yin, Y.L.: Forgery and partial key-recovery attacks on HMAC and NMAC using hash collisions. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 37–53. Springer, Heidelberg (2006)
7. Fouque, P.-A., Leurent, G., Nguyen, P.Q.: Full key-recovery attacks on HMAC/NMAC-MD4 and NMAC-MD5. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 13–30. Springer, Heidelberg (2007)
8. Galvin, J.M., McCloghrie, K., Davin, J.R.: Secure management of SNMP networks. Integrated Network Management II, 703–714 (1991)
9. Kim, J.-S., Biryukov, A., Preneel, B., Hong, S.H.: On the security of HMAC and NMAC based on HAVAL, MD4, MD5, SHA-0 and SHA-1 (Extended abstract). In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 242–256. Springer, Heidelberg (2006)
10. Preneel, B., van Oorschot, P.: MDx-MAC and building fast MACs from hash functions. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 1–14. Springer, Heidelberg (1995)
11. Rechberger, C., Rijmen, V.: On authentication with HMAC and non-random properties. In: Dietrich, S., Dhamija, R. (eds.) FC 2007. LNCS, vol. 4886, pp. 119–133. Springer, Heidelberg (2007)
12. Rechberger, C., Rijmen, V.: New results on NMAC/HMAC when instantiated with popular hash functions. Journal of Universal Computer Science 14(3), 347–376 (2008)
13. Rivest, R.L.: The MD5 message digest algorithm. Request for Comments (RFC 1321), Network Working Group (1992)
14. Tsudik, G.: Message authentication with one-way hash functions. ACM Comput. Commun. Rev. 22(5), 29–38 (1992)
15. Wang, L., Ohta, K., Kunihiro, N.: New key-recovery attacks on HMAC/NMAC-MD4 and NMAC-MD5. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 237–253. Springer, Heidelberg (2008)
16. Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the hash functions MD4 and RIPEMD. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 1–18. Springer, Heidelberg (2005)
17. Wang, X., Yu, H.: How to break MD5 and other hash functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)
18. Wang, X., Yu, H., Yin, Y.L.: Efficient collision search attacks on SHA-0. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 1–16. Springer, Heidelberg (2005)
19. Wang, X., Yin, Y.L., Yu, H.: Finding collisions in the full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)
20. Yuval, G.: How to swindle rabin. Cryptologia 3, 187–190 (1979)

Appendix

Table 1. The dBB differential path and its corresponding sufficient conditions

step	m_i	CV	shift s_i	Output difference	Sufficient conditions	Recovered bits
-4		A_0		32		
-3		D_0		32		
-2		C_0		32	$C_{0,32} = D_{0,32}$	
-1		B_0		32	$B_{0,32} = C_{0,32}$	
1	m_0	A_1	7	32	$A_{1,32} = B_{0,32}$	32, 31, 30, ..., 8
2	m_1	D_1	12	32	$D_{1,32} = A_{1,32}$	32, 31, 30, ..., 13
3	m_2	C_1	17	32	$C_{1,32} = D_{1,32}$	32, 31, 30, ..., 18
4	m_3	B_1	22	32	$B_{1,32} = C_{1,32}$	32, 31, 30, ..., 23
5	m_4	A_2	7	32	$A_{2,32} = B_{1,32}$	32, 31, 30, ..., 8
6	m_5	D_2	12	32	$D_{2,32} = A_{2,32}$	32, 31, 30, ..., 13
7	m_6	C_2	17	32	$C_{2,32} = D_{2,32}$	32, 31, 30, ..., 18
8	m_7	B_2	22	32	$B_{2,32} = C_{2,32}$	32, 31, 30, ..., 23
9	m_8	A_3	7	32	$A_{3,32} = B_{2,32}$	32, 31, 30, ..., 8
10	m_9	D_3	12	32	$D_{3,32} = A_{3,32}$	32, 31, 30, ..., 13
11	m_{10}	C_3	17	32	$C_{3,32} = D_{3,32}$	32, 31, 30, ..., 18
12	m_{11}	B_3	22	32	$B_{3,32} = C_{3,32}$	32, 31, 30, ..., 23
13	m_{12}	A_4	7	32	$A_{4,32} = B_{3,32}$	32, 31, 30, ..., 8
14	m_{13}	D_4	12	32	$D_{4,32} = A_{4,32}$	32, 31, 30, ..., 13
15	m_{14}	C_4	17	32		
16	m_{15}	B_4	22	32	$B_{4,32} = C_{4,32}$	
17	m_1	A_5	5	32	$A_{5,32} = B_{4,32}$	
18	m_6	D_5	9	32	$D_{5,32} = A_{5,32}$	
19	m_{11}	C_5	14	32	$C_{5,32} = D_{5,32}$	
20	m_0	B_5	20	32	$B_{5,32} = C_{5,32}$	
21	m_5	A_6	5	32	$A_{6,32} = B_{5,32}$	
22	m_{10}	D_6	9	32	$D_{6,32} = A_{6,32}$	
23	m_{15}	C_6	14	32	$C_{6,32} = D_{6,32}$	
24	m_4	B_6	20	32	$B_{6,32} = C_{6,32}$	
25	m_9	A_7	5	32	$A_{7,32} = B_{6,32}$	
26	m_{14}	D_7	9	32	$D_{7,32} = A_{7,32}$	
27	m_3	C_7	14	32	$C_{7,32} = D_{7,32}$	
28	m_8	B_7	20	32	$B_{7,32} = C_{7,32}$	
29	m_{13}	A_8	5	32	$A_{8,32} = B_{7,32}$	
30	m_2	D_8	9	32	$D_{8,32} = A_{8,32}$	
31	m_7	C_8	14	32	$C_{8,32} = D_{8,32}$	
32	m_{12}	B_8	20	32		
33	m_5	A_9	4	32		
34	m_8	D_9	11	32		
35	m_{11}	C_9	16	32		
36	m_{14}	B_9	23	32		
37	m_1	A_{10}	4	32		
38	m_4	D_{10}	11	32		
39	m_7	C_{10}	16	32		
40	m_{10}	B_{10}	23	32		
41	m_{13}	A_{11}	4	32		
42	m_0	D_{11}	11	32		
43	m_3	C_{11}	16	32		
44	m_6	B_{11}	23	32		
45	m_9	A_{12}	4	32		
46	m_{12}	D_{12}	11	32		
47	m_{15}	C_{12}	16	32		
48	m_2	B_{12}	23	32	$B_{12,32} = D_{12,32}$	
49	m_0	A_{13}	6	32	$A_{13,32} = C_{12,32}$	
50	m_7	D_{13}	10	32	$D_{13,32} = B_{12,32}$	
51	m_{14}	C_{13}	15	32	$C_{13,32} = A_{13,32}$	
52	m_5	B_{13}	21	32	$B_{13,32} = D_{13,32}$	
53	m_{12}	A_{14}	6	32	$A_{14,32} = C_{13,32}$	
54	m_3	D_{14}	10	32	$D_{14,32} = B_{13,32}$	
55	m_{10}	C_{14}	15	32	$C_{14,32} = A_{14,32}$	
56	m_1	B_{14}	21	32	$B_{14,32} = D_{14,32}$	
57	m_8	A_{15}	6	32	$A_{15,32} = C_{14,32}$	
58	m_15	D_{15}	10	32	$D_{15,32} = B_{14,32}$	
59	m_6	C_{15}	15	32	$C_{15,32} = A_{15,32}$	
60	m_{13}	B_{15}	21	32	$B_{15,32} = D_{15,32}$	
61	m_4	A_{16}	6	32	$A_{16,32} = C_{15,32}$	
62	m_{11}	D_{16}	10	32	$D_{16,32} = B_{15,32}$	
63	m_2	C_{16}	15	32	$C_{16,32} = A_{16,32}$	
64	m_9	B_{16}	21	32		