On the Security of NMAC and Its Variants

Fanbao Liu[1], Changxiang Shen[2], Tao Xie[3], Dengguo Feng[4]

[1] School of Computer, National University of Defense Technology, Changsha, 410073, Hunan, P. R. China
[2] School of Computer, Beijing University of Technology, Beijing, 100124, P. R. China
[3] The Center for Soft-Computing and Cryptology, NUDT, Changsha, 410073, Hunan, P. R. China
[4] State Key Lab of Information Security, Chinese Academy of Sciences, Beijing, P. R. China
liufanbao@yahoo.com.cn

Abstract. We first propose a general equivalent key recovery attack to a H^2-MAC variant NMAC_1, which is also provable secure, by applying a generalized birthday attack. Our result shows that NMAC_1, even instantiated with a secure Merkle-Damgård hash function, is not secure. We further show that this equivalent key recovery attack to NMAC_1 is also applicable to NMAC for recovering the equivalent inner key of NMAC, in a related key setting. We propose and analyze a series of NMAC variants with different secret approaches and key distributions, we find that a variant NMAC-E, with secret envelop approach, can withstand most of the known attacks in this paper. However, all variants including NMAC itself, are vulnerable to on-line birthday attack for verifiable forgery. Hence, the underlying cryptographic hash functions, based on Merkle-Damgård construction, should be re-evaluated seriously.

Keywords: NMAC, Keying Hash Function, Equivalent Key Recovery, Verifiable Forgery, Birthday Attack.

1 Introduction

HMAC [3, 2], a derivative of NMAC, is a practically and commonly used, widely standardized MAC construction nowadays. HMAC has two advantages. First, HMAC can make use of current hash functions, the most widely used ones are based on Merkle-Damgård construction [17, 6], without modification. Second, it is provable secure under two assumptions that the keyed compression function of the underlying hash function and the key derivation function in HMAC are pseudo random functions (PRFs) [2].

All in all, NMAC is the base of HMAC. For an iterated hash function H with Merkle-Damgård construction, NMAC is defined by

$$\text{NMAC}_{(K_{\text{out}},\,K_{\text{in}})}(M) = H(K_{\text{out}}, H(K_{\text{in}}, M))$$

where M is the input message, K_{in} and K_{out} are two random and independent secret n-bit keys.

After some prevalent iterated hash functions were broken [27, 12, 29, 31, 28], the security of NMAC and HMAC instantiated with those hash function were analysed [4, 8, 25, 30], which emphasized that NMAC and HMAC instantiated with broken hash functions are weak.

There are mainly three kinds of approaches to construct MAC algorithm by keying hash function in early days, which are secret prefix, secret suffix and secret envelop approaches [24], respectively. The secret prefix approach prepends a secret K to the message M before hashing computation, which is the basic design unit for NMAC and HMAC. The secret suffix approach appends a secret key K to the message M before hashing computation. The secret envelop approach, involving two keys, prepends a secret key K_1 and appends a secret key K_2 to the message M, respectively, before hashing computation. Based on these approaches and different key distributions, we propose some NMAC variants (also are HMAC variants), and analyze their security, by checking whether they are resistant for known attacks, for a better choice.

This paper analyses the security of the NMAC and its variants based on the assumption that the underlying hash function is secure (weak collision resistance), which is stronger than the origin assumption of that the underlying compression function is a PRF [2]. We find that NMAC is not secure enough to some extent, for example, its inner key is vulnerable to equivalent key recovery attack, which needs $O(2^{n/2})$ on-line queries and off-line computations, in the related key setting.

Our Contributions. We propose a general equivalent key recovery attack to NMAC_1 by a generalized on-line birthday attack, which needs about $2^{n/2}$ on-line MAC queries, and $2^{n/2}$ off-line MAC computations with any pre-set key. After a inner collision pair (M_0, M_0') is found, we get the exact value of intermediate chaining variable ICV_2 of the inner hashing of NMAC_1, which conduces to a selective forgery attack directly.

Based on the three earlier approaches to construct MAC algorithms and different key distributions, we propose a series of NMAC variants, we also analyse those variants in order to find a better and securer one. We find a variant of NMAC, named NMAC-E, with the modified version of the secret envelop approach, can withstand all known attacks to MAC algorithms.

Organization of this paper. This paper is divided into six sections. Section 2 recalls the related definitions and background. Section 3 proposes and crypt-analyses some NMAC variants with secret prefix approach. The security of NMAC is also discussed, based on these variants. Section 4 proposes and analyses the security of some NMAC variants with secret suffix approach. We present and analyse a better choice of NMAC variant with the modified version of the secret envelop approach, in section 5. We conclude the paper in the last section.

2 Preliminaries

We first explain some notations related to this paper, then present brief description of cryptographic hash function with Merkle-Damgård structure, some properties of thus function, and finally, we present a brief description of NMAC in this section.

2.1 Notations

n	The length of hash result
b	The length of a message block
H	A concrete hash function with n-bit result
$\tilde{H}$	A hash function without message padding
h	A compression function with a n-bit and a b-bit inputs, and a n-bit output
IV	The initial chaining variable of H
ICV_i	The intermediate chaining variables for the i-th iteration of H
K	A secret key with n bits
$k_{\texttt{in}}$	A padded key like k_{out}, with b bits
$K_{\texttt{in}}$	A secret key like K_{out}, with n bits
$x\|\|y$	The concatenation of two bit strings x and y
$\oplus$	The bit wise exclusive OR
$pad(M)$	The padding bits of M with length information
$padding$	The padding bits without length information, e.g. $1\|\|0^*$
$\|M\|$	The length of the string M

2.2 Brief Description of Merkle-Damgård Hash Function

Cryptographic hash functions with Merkle-Damgård structure compress message M of arbitrary length to a fixed length output $H(M)$. MD5 [23] and SHA-1 [7] are two typical Merkle-Damgård structure hash functions in use, which takes a variable-length message M (actually, $|M| < 2^{64}$) as input and outputs a 128-bit and 160-bit hash values, respectively.

M is first padded to be multiples of b bits, a '1', added at the tail of M, followed by '0's, until the bit length becomes $(b-64)$ on modulo b, and finally, the length of the unpadded message M is added to the last 64 bits. The padded M' is further divided into chunks of $(M_0, M_1, \ldots, M_{N-1})$, each is a b-bit block.

The compression function h takes a b-bit block M_i and a n-bit chaining variable ICV_i, initialized to IV, as input, and outputs ICV_{i+1}. For example, $ICV_1 = h(IV, M_0)$, and $H(M) = ICV_N = h(ICV_{N-1}, M_{N-1})$. For the details of the concrete compression functions, please refer [22, 23, 7].

Padding rule. For two arbitrary distinct messages M and M', if $|M| = |M'|$, then, the padding bits of these two messages are just the same. Since MD5 [23], MD4 [22] and SHA-1 [7] et al. share the same padding procedure, the padding rule is also applicable to them.

Extension Attack. Let $pad(M)$ denote the padding bits of M. For arbitrary unknown M_0, let $R = H(M_0)$, then for $M_1 = M_0||pad(M_0)||x$, where x is randomly generated. We can generate the hash of M_1 by computing $H(M_1) = h(R, x||pad(M_1))$, with no knowledge about M_0 except its length.

A Property of Hash Collision Pair. If two arbitrary distinct messages m, $m' \in \{0,1\}^*$ satisfy $H(m) = H(m')$, then m and m' are called a collision pair. Let $|m|$, the length of m, be multiples of message blocks, and $|m| = |m'|$ further, then for an arbitrary message x, $H(m||x) = H(m'||x)$ always holds.

Security Properties of Hash Functions Cryptographic hash functions need to satisfy the following security properties [16, 19]:

1. pre-image resistance: it should be computation infeasible to find a pre-image for a given hash result;
2. collision resistance: it should be computation infeasible to find two different inputs with the same hash result.

For an ideal hash function with n-bit result, finding a pre-image requires approximately 2^n hash operations. On the other hand, finding a collision requires only $2^{n/2}$ hash operations; this follows from the birthday paradox [9]. In this paper, we assume that the underlying hash functions of all MACs are secure, which means (weak) collision resistance.

2.3 Kerckhoffs' Principle

Kerckhoffs' Principle. The security of a crypt-system should depend solely on the secrecy of the key (password) [11].

The principle implies that the security of hash-based MAC should depend solely on the secrecy of the used key, not the MAC form nor the collision resistance of the underlying hash function. It should perform a pre-image attack against the underlying hash function, to break a MAC whose key length is equivalent to the underlying hash result.

2.4 Some Basic MAC Forms

There are three kinds of approaches to construct MAC in early days [24], which are secret prefix, secret suffix, secret envelop, respectively.

Secret Prefix Approach The secret prefix MAC M-P[1] is defined as:

$$\text{M-P}_K(M) = H(K, M)$$

where the IV of H is replaced with a secret key K before hashing computation. This approach is the basic design unit for NMAC and HMAC [3, 2]. However, the secret prefix M-P is vulnerable to the extension attack, which transforms to equivalent key recovery attack eventually [24]. M-P is also vulnerable to on-line

[1] This is the keyed chaining variable version, the origin version is keyed input.

birthday attack, which means the security of M-P is dependent on the weak collision resistance (WCR) of the underlying hash function.

Secret Suffix Approach The secret suffix MAC M-S is defined as:

$$\text{M-S}_K(M) = H(M||K)$$

where a secret key K is appended to the message M before hashing computation. This approach is vulnerable to off-line birthday attack [24], which means the security of M-S is solely dependent on the collision resistance (CR) of the underlying hash function.

Secret Envelop Approach The secret envelop MAC M-E is defined as:

$$\text{M-E}_{(K_1,K_2)}(M) = H(K_1, M||K_2)$$

where the IV of H is replaced with a secret key K_1, and then the other key K_2 is appended to the message M, before hashing computation. This approach is vulnerable to on-line birthday attack [24] (WCR dependent). M-E is also vulnerable to divide-and-conquer exhaustive-search key recovery [18, 20], which means that it needs about 2^{n+1} operations to recover both keys with n bits each, instead of claimed 2^{2n} operations [24].

2.5 NMAC

NMAC [3, 2] proposed by Bellare et al., is the basis of the most widely used cryptographic algorithms HMAC. NMAC is built from iterated hash function H, where the IV of H is replaced with a secret n-bit key K, the NMAC algorithm is defined as:

$$\text{NMAC}(M) = \text{NMAC}_{(K_{\text{out}},K_{\text{in}})}(M) = H(K_{\text{out}}, H(K_{\text{in}}, M))$$

where keys K_{in}, $K_{\text{out}} \in \{0,1\}^n$ in NMAC are to replace the IV of hash function H before further process. In practice, both keys are random and independently generated [3].

3 The security of Some Variants with Secret Prefix

NMAC applies two keys K_{in} and K_{out}, we first discuss the security of some variants of NMAC through different key deployments, then we analyse the security of NMAC.

3.1 The security of NMAC_1 (the keyed IV version of H^2-MAC)

We define NMAC_1 as:

$$\text{NMAC}_1(M) = H(H(K_{\text{in}}, M))$$

where the outer key K_{out} is omitted. A keyed input version of NMAC_1 was also proposed by Yasuda as H^2-MAC [33]. It was claimed that H^2-MAC gets rid of the disadvantage of the secret key management without losing the original advantage of HMAC[2]. This year, Wang announced an attack to recover the

[2] H^2-MAC is provable secure like HMAC [2]

equivalent key of H^2-MAC instantiated with the broken MD5 [29, 31], with about 2^{97} on-line operations [26]. However, we point out that the absence of the outer key is a real threat to the security of H^2-MAC [15], which does not exist in HMAC.

On-Line Birthday Attack for Verifiable Forgery Attack If we apply on-line birthday attack to NMAC_1 oracle, after about $2^{n/2}$ queries, we can get a collision pair (M, M'), which satisfies $\mathrm{NMAC}_1(M) = \mathrm{NMAC}_1(M')$. Then $\mathrm{NMAC}_1(M||pad(M)||x) = \mathrm{NMAC}_1(M'||pad(M')||x)$ always holds, for arbitrary message x. This means that we can generate verifiable forgery of NMAC_1, we first query the corresponding MAC value of $M||pad(M)||x$, and we get the very MAC value for $M'||pad(M')||x$, eventually.

After about $2^{n/2}$ on-line queries, any verifiable forgery to NMAC_1, based on the collision pair (M, M'), can be made with one additional on-line query.

Equivalent Key Recovery Attack to $\mathbf{NMAC_1}$ It seems that we can't get the value of $H(K_{\text{in}}, M)$ for the application of outer hashing. To find a way out, we apply the generalized birthday attack with two groups [9] to NMAC_1 and then recover its equivalent key $K_e = H(K_{\text{in}}, M_0)$.

Here, we first define the notation N^2 as $N^2 = H(H(C, M))$, where C is an n-bit constant or any parameter known to everybody, for example, the IV of H. Generally speaking, N^2 is the non-key version of NMAC_1.

We use different 1-block messages M_0s to generate the corresponding NMAC_1 values, and use different 1-block messages M_0's to generate the corresponding N^2 values. The overall strategy of equivalent key recovery attack to NMAC_1 is shown as follows.

1. Generate a group one G_1 with $r = 2^{n/2}$ elements, by computing the corresponding values of $H(H(C, M_0'))$ for r different Cs and M_0's, which can be randomly generated. Specifically, C can be set to the IV of H.
2. Generate a group two G_2 with $s = 2^{n/2}$ elements, by querying the corresponding values to NMAC_1 oracle with the secret key K_{in} for s different M_0s, where M_0s are randomly generated.
3. There is a collision pair (M_0, M_0') that not only satisfies $\mathrm{NMAC}_{1(K_{\text{in}})}(M_0) = N_C^2(M_0')$, but also satisfies $H(K, M_0) = H(C, M_0')$ (an inner collision between N^2 and NMAC_1 happens), with good probability [9].
4. Since $H(K, M_0) = H(C, M_0')$, and we know the value of C and M_0', we can compute the value of $K_e = H(K, M_0) = H(C, M_0')$.
5. Let pad_0 and pad_1 be the padding bits of M_0 and $M_0||pad_0||x$, respectively, for arbitrary message x. Hence, we generate the result of $H(K, M_0||pad_0||x)$ by computing $y = h(K_e, x||pad_1)$, then we compute $H(y)$ further, and finally we get the very value of $\mathrm{NMAC}_1(K, M_0||pad_0||x)$.

Success probability and Complexity. The probability $\Pr(|G_1 \cap G_2)| = 0$ that there are no distinct element in the intersection of the two groups is denoted by $P(2^n, r, s, 0)$. Let sp denote the success probability of the above attack (at least one collision pair exists), then we can get the value of sp by computing $sp = 1 - P(2^n, r, s, 0) \geq 0.632$ [15]. The elements of group G_1 computed by N^2 need $2^{n/2}$ off-line N^2 computations. The elements of group G_2 computed by NMAC_1 need $2^{n/2}$ on-line NMAC_1 queries. We can store the values of both group using hash table. Then the above algorithm will require $O(2^{n/2})$ time and space. For more details about this kind of attack and its optimization, please refer [15], which shows the equivalent key recovery attack to H^2-MAC.

After an inner collision pair (M_0, M_0') is found, we can apply $N_C^2(M_0')$ to compute the equivalent key of the NMAC_1. Finally, we can use the recovered equivalent key K_e to launch any selective forgery attack to NMAC_1 without on-line query, based on M_0, which claims that the security of NMAC_1 is broken.

Hence, we point out that the security of NMAC_1 is solely dependent on the (weak) collision resistance of the underlying hash function, not the strength of the used key.

However, it is interesting to notice that H^2-MAC, the keyed input version of NMAC_1, is provable secure under the assumption of that the underlying compression function h is a PRF-AX [33], which means that (weak) collision resistance of the underlying hash function can be dropped. Thus proof and assumption obvious violate our result.

3.2 The security of $\mathbf{NMAC_2}$

We define NMAC_2 as:

$$\mathrm{NMAC}_2(M) = H(K_{\mathrm{out}}, H(M))$$

where the inner key K_{in} is omitted. This variant NMAC_2 was also noted by Bellare et al. in [3].

The outer hashing only accepts $H(M)$ as legal input, which is an n-bit value. Though we can learn the value of $H(K_{\mathrm{out}}, H(M))$ easily, we can not use that information to launch the extension attack to NMAC_2.

Birthday Attack to $\mathbf{NMAC_2}$ We first apply off-line birthday attack to $H(M)$. After about $2^{n/2}$ off-line computations, we can get a collision pair (M, M'), which satisfies $H(M) = H(M')$, and $\mathrm{NMAC}_2(M) = \mathrm{NMAC}_2(M')$, eventually. Then, the equation of $\mathrm{NMAC}_2(M||pad(M)||x) = \mathrm{NMAC}_2(M'||pad(M')||x)$ always holds, for arbitrary message x. This means that we can generate verifiable forgery to NMAC_2, we first query for the MAC value of $M||pad(M)||x$, and we get the MAC value for $M'||pad(M')||x$, eventually.

After $2^{n/2}$ off-line computations, any verifiable forgery to NMAC_2, based on the collision pair (M, M'), can be made by additional one on-line query.

3.3 The security of $\mathbf{NMAC_3}$

We define NMAC_3 as:

$$\mathrm{NMAC}_3 = H(K_{\mathrm{io}}, H(K_{\mathrm{io}}, M))$$

where the inner key and outer are both set to K_{io}.

The on-line birthday attack for verifiable forgery applied to NMAC_1 is also applicable to NMAC_3. Here, we omit the details, for such verifiable forgery attack can be implemented without any modification.

Further, we point out that the off-line birthday attack to get verifiable forgery is also applicable to NMAC_3 after some optimization. We show the strategy as follows.

1. Query the corresponding MAC value of M_0 to the NMAC_3 oracle, which will answer $H(K_{\text{io}}, H(K_{\text{io}}, M_0))$.
2. Assume the unknown $H(K_{\text{io}}, M_0)$ be x_0, and pad_0 be the padding bits of x_0. We already know the corresponding value of $H(K_{\text{io}}, x_0)$ (an equivalent key of the inner hashing), which is $\text{NMAC}_3(M_0)$.
3. Based on the known $H(K_{\text{io}}, x_0)$, we launch an off-line birthday attack. We can find a collision pair (M_x, M'_x), which satisfies $H(K_{\text{io}}, x_0||pad_0||M_x) = H(K_{\text{io}}, x_0||pad_0||M'_x)$.
4. For arbitrary message x, we can launch a verifiable forgery attack.

However, since the value of $H(K_{\text{io}}, M_0)$ is unknown, how to use the above information to launch a verifiable forgery attack is still a open problem.

3.4 The security of NMAC

As pointed out by Bellare et al., the on-line birthday attack for verifiable forgery attack is also applicable to NMAC [2], here we omit the details. However, we further notice that we can generate verifiable forgery for NMAC, by off-line birthday attack, which is shown as the attack to NMAC_2, once the inner key K_{in} is leaked.

Related Key Attack to Recover the Equivalent Inner Key To recover the equivalent inner key K_e with n-bit, we have the following setting for our related-key attacks on NMAC.

There are two oracles $\text{NMAC}_{(K_{\text{out}}, K_{\text{in}})}$ and $\text{NMAC}_{(K'_{\text{out}}, K'_{\text{in}})}$. The relation between $(K_{\text{out}}, K_{\text{in}})$ and $(K'_{\text{out}}, K'_{\text{in}})$ is set as follows:

$$K_{\text{out}} = K'_{\text{out}} \quad \text{and} \quad K'_{\text{in}} \in \{Constants\}$$

where these two oracles share the same outer key, and the inner key of the oracle $\text{NMAC}_{(K'_{\text{out}}, K'_{\text{in}})}$ can be any known constant.

The overall strategy of the equivalent inner key recovery attack to NMAC is shown as follows.

1. Query $\text{NMAC}_{(K_{\text{out}}, K_{\text{in}})}$ oracle for the corresponding values of $2^{n/2}$ different M_0s, store their values in group one G_1.
2. Query $\text{NMAC}_{(K'_{\text{out}}, K'_{\text{in}})}$ oracle for the corresponding values of $2^{n/2}$ different M'_0s, store their values in group two G_2.

3. A pair (M_0, M_0') satisfies $\mathrm{NMAC}_{(K_{\mathrm{out}},K_{\mathrm{in}})}(M) = \mathrm{NMAC}_{(K'_{\mathrm{out}},K'_{\mathrm{in}})}(M)$ (the generalized birthday attack with two groups), and satisfies $H(K_{\mathrm{in}}, M_0) = H(K'_{\mathrm{in}}, M'_0)$ further (an inner collision happens).
4. Since $H(K_{\mathrm{in}}, M_0) = H(K'_{\mathrm{in}}, M'_0)$, and we know the value of K'_{in} and M'_0, hence we can calculate the very value of $K_e = H(K_{\mathrm{in}}, M_0) = H(K'_{\mathrm{in}}, M'_0)$.

We conclude that the equivalent inner key of NMAC is totally dependent on the generalized birthday attack (WCR), not the strength of the used inner key, in the related key setting.

However, if the outer key K_{out} of NMAC is leaked, then, it needs a generalized birthday attack to recover the equivalent inner key to break the entire system, shown as the attack to NMAC_1.

From these attacks, we claim that the security of NMAC is dependent on the secrecy of one of the keys, even if its both key are independently and randomly generated.

As pointed out by the editors of Cryptology ePrint Archive in our preliminary version of this paper [14], the equivalent key recovery attack to NMAC is not applicable to HMAC, since the HMAC keys are derived from a base key, and there exists no related keys.

4 The security of Some Variants with Secret Suffix

In this section, we discuss the security of some NMAC variants NMAC-S_i with secret suffix approach.

We first prove that the security of original secret suffix M-S is totally dependent on the collision resistance (CR) of the underlying hash function. We then discuss the security of some variants of NMAC with secret suffix approach.

4.1 The Security of $H(M||K)$

For an n-bit key K, we will prove as follows, the security of the secret suffix M-S is totally dependent on the collision resistance of the underlying hash function, instead of the pre-image resistance.

Theorem 1 *The security of $H(M||K)$ is totally dependent on the collision resistance of the underlying hash function H.*

We prove Theorem 1 by giving the complexity of the worst case of the key recovery attack and best case attack, respectively, which are all based on the assumption that the message M is multiples of bytes. The worst case of the key recovery attack is that we assume the collision attack of H has no control over the content of the collision pair (M, M'). The best case is that we assume the collision attack has full control over some bytes of the collision pair. We notice that the complexity of the collision attack is $2^{n/2}$ hash compressions by off-line birthday attack, for a hash function H with n-bit output. The attack is based on the "slice-by-slice" key recovery of trail key in secret envelop approach, proposed by Preneel et al. [18].

Proof. **The Best Case**. Since the collision attack has full control over some bits of the collision pair, to recover each byte of the key K, only $(2^8 - 1)$ collision pairs must be generated in the worst case. So we need to generate $(2^8-1)(n/16)$ collision pairs to recover the first $n/2$ bits of K, and we can recover the last $n/2$ bits of K through brute force attack, which needs $2^{n/2}$ hash compressions. So the total complexity of the full key recovery attack is $2^{n/2}\times(2^8-1)\times(n/16)+2^{n/2} < 2^{n/2+8+\log_2^{n/16}}$ hash compressions.

The Worst Case. Since the collision attack has no control over any bit of the collision pair, to recover the j-th ($1\leq j\leq n/8$) character of the key K, $2^{8\cdot j}$ collision pairs must be first generated. So we can recover the first $n/4$ bits of the key by generating $(2^8 + 2^{8\cdot 2} + \cdots + 2^{n/4})$ collision pairs, and we can recover the last $3\cdot n/4$ bits through brute force attack, which needs $2^{3\cdot n/4}$ hash compressions. The total complexity is $2^{n/2}\cdot(2^8 + 2^{8\cdot 2} + \cdots + 2^{n/4}) + 2^{3\cdot n/4} \approx 2^{n/2+n/4+1}$ hash compressions. □

Table 1. Complexity of Key Recovery Attack to Secret Suffix Approach

Cases	Bit by Bit	Byte by Byte	Word by Word	n-bit
The Best Case	$2^{n/2+\log_2^{n/2}}$	$2^{n/2+8+\log_2^{n/16}}$	$2^{n/2+32+\log_2^{n/64}}$	2^n
The Worst Case	$2^{n/2+n/4+1}$	$2^{n/2+n/4+1}$	$2^{n/2+n/4+1}$	2^n

All in all then, the complexity of the key recovery to $H(M||K)$ is range from $2^{n/2+8+\log_2^{n/16}}$ to $2^{n/2+n/4+1}$ hash compressions, which means that the security of M-S is dependent on the collision resistance of the underlying hash function H, instead of the pre-image resistance.

We list the complexity of key recovery attack to $H(M||K)$ in Table 1, with different limitations on the input message M. Word means that M must be multiples of 32-bit words. However, as shown in Table 1, we point out that both the best and worst case are exhaustive key search, if the message M is multiples of n bits.

4.2 The security of NMAC-S$_1$

We define NMAC-S$_1$ as:

$$\text{NMAC-S}_1 = H(H(M||K_{\text{in}}))$$

where the outer key K_{out} is omitted.

Birthday Attack to NMAC-S$_1$ After about $2^{n/2}$ off-line computations, an inner collision pair (M, M') will be found, where M and M' are multiples of blocks. Hence, we can construct a verifiable forgery for arbitrary x, which satisfies that $\text{NMAC-S}_1(M||x) = \text{NMAC-S}_1(M'||x)$.

Full Key Recovery Attack to NMAC-S_1 We can directly apply the full key recovery attack to $H(M||K_{\text{in}})$, since the outer hashing does not hide the inner collision. After that, we can fully recover the inner key of NMAC-S_1, and then can construct any verifiable forgery. The complexity of the key recovery attack to NMAC-S_1 can be shown Table 1.

4.3 The security of NMAC-S_2

We define NMAC-S_2 as:

$$\text{NMAC-S}_2 = H(H(M)||K_{\text{out}})$$

where the inner key K_{in} is omitted.

Birthday Attack to NMAC-S_2 Since the inner collision can't be hidden by the outer hashing with the key K_{out}, we can launch an off-line birthday attack to NMAC-S_2. After about $2^{n/2}$ off-line computations, an inner collision pair (M, M') will be found, where M and M' are multiples of blocks. Hence, we can construct a verifiable forgery for arbitrary x, which satisfies that NMAC-$S_2(M||x)$ = NMAC-$S_2(M'||x)$. We first query the corresponding MAC value of $M||x$ to the NMAC-S_2 oracle, then, we get the very result for $M'||x$.

However, it seems that no key recovery attack to NMAC-S_2 can be launched as NMAC-S_1. $H(M)$ is n bits long, and K_{out} is also n bits, which means that the concatenation of both are inside one block, so the slice-by-slice key recovery strategy can't be applied. Exhaustive search must be performed to break the outer key K_{out}, whose complexity is 2^n MAC computations.

4.4 The security of NMAC-S_3

We define NMAC-S_3 as:

$$\text{NMAC-S}_3 = H(H(M||K_{\text{io}})||K_{\text{io}})$$

where the inner and outer keys are equal.

Birthday Attack to NMAC-S_3 Since the inner collision can't be hidden by the outer hashing with the key K_{io}, we can launch an off-line birthday attack to NMAC-S_3. After about $2^{n/2}$ off-line computations, an inner collision pair (M, M') will be found, where M and M' are multiples of blocks. Hence, we can construct a verifiable forgery for arbitrary x, which satisfies that NMAC-$S_3(M||x)$ = NMAC-$S_3(M'||x)$. We first query the corresponding MAC value of $M||x$ to the NMAC-S_3 oracle, then, we get the very result for $M'||x$.

Key Recovery Attack to NMAC-S$_3$ We can directly apply the full key recovery attack to $H(M||K_{\text{io}})$, since the outer hashing does not hide the inner collision. After that, we can fully recover the inner key K_{io}, which is also the outer key, of NMAC-S$_3$. Finally, we can construct any verifiable forgery. The complexity of the key recovery attack to NMAC-S$_3$, which is analogous to NMAC-S$_1$, is also shown in Table 1.

4.5 The security of NMAC-S

We define NMAC-S as:

$$\text{NMAC-S} = H(H(M||K_{\text{in}})||K_{\text{out}})$$

where the inner and outer keys are different.

Birthday Attack to NMAC-S Since the inner collision can't be hidden by the outer hashing with the key K_{out}, we can launch an off-line birthday attack to HMAC-S. After about $2^{n/2}$ off-line computations, an inner collision pair (M, M') will be found, where M and M' are multiples of blocks. Hence, we can construct a verifiable forgery for arbitrary x, which satisfies that NMAC-S$(M||x) =$ NMAC-S$(M'||x)$. We first query the corresponding MAC value of $M||x$ to the HMAC-S oracle, then, we get the very result for $M'||x$.

Inner Key Recovery Attack to NMAC-S We can directly apply the full key recovery attack to $H(M||K_{\text{in}})$, since the outer hashing does not hide the appearance of the inner collision. After that, we can fully recover the inner key K_{in} of NMAC-S. However, with K_{in}, we can't directly construct any verifiable forgery, thanks to the outer hashing with the unknown K_{out}. The outer key K_{out} can't be recovered like K_{in}, which is also analysed in NMAC-S$_2$. It seems that we have to apply another off-line birthday attack to $H(M)$, for a meaningful verifiable forgery.

4.6 Counterpart for the Key Recovery Attack to NMAC-S Variants

To avoid the full key recovery attack to NMAC-S Variants, we modify the inner hashing form $H(M||Kin)$. Let *pad* be the padding bits of M, we re-define the inner hashing form as:

$$H(M||pad||K_{\text{in}})$$

where the inner key K_{in} resides on the last block, exactly on the first n bits of the entire b bits. Hence, slice-by-slice key recovery strategy can't be applied any more, for launching key recovery attack.

However, the NMAC-S Variants after modification are still vulnerable to off-line birthday attack for verifiable forgery attack.

5 The security of an NMAC Variant with Secret Envelop

In last two sections, we discuss the security of NMAC variants with secret prefix and secret suffix, respectively. In this section, we discuss the security of an NMAC variant, NMAC-E, with secret envelop approach.

5.1 A Modified Secret Envelop

We first propose a modified version of the secret envelop approach, which has the advantage of both equivalent key recovery resistance and slice-by-slice key recovery resistance. The modification is straightforward, we pad the input message M with *pad*, which can be some fixed constants, before appending the second key K_{i2}. We define the modified secret envelop M-E′ as:

$$\text{M-E}'_{(K_{\mathrm{i}1},K_{\mathrm{i}2})} = H(K_{\mathrm{i}1}, M||pad||K_{\mathrm{i}2})$$

where $K_{\mathrm{i}1}$ and $K_{\mathrm{i}2}$ are both n-bit keys. Both keys can be generated from a base key K, or generated randomly and independently. $|M||pad|$ is multiples of blocks long. $K_{\mathrm{i}2}$ resides on the first n bits of the last block.

Security Analysis. M-E′ is still vulnerable to the divide-and-conquer exhaustive search key recovery [18], hence, both keys can be generated from a base key K without security loss. However, off-line birthday attack can't be applied thanks to the secret "IV", the $K_{\mathrm{i}1}$, and equivalent key recovery attack can't be launched thanks to the appended key $K_{\mathrm{i}2}$. Further, slice-by-slice key recovery attack can be avoided for the *pad* of M, as a result, the appended key can't be split into any slice. Finally, M-E′ is vulnerable to on-line birthday attack for verifiable forgery, in fact, it seems that modifications must be made to the design criteria of the underlying hash function [19, 5, 10, 21, 32], in order to avoid this kind of attack.

5.2 The security of NMAC-E

We define NMAC-E as:

$$\text{NMAC-E} = \text{NMAC-E}_{(K_{\mathrm{o}1},K_{\mathrm{o}2},K_{\mathrm{i}1},K_{\mathrm{i}2})} = H(K_{\mathrm{o}1}, \text{M-E}'_{(K_{\mathrm{i}1},K_{\mathrm{i}2})}||K_{\mathrm{o}2})$$

where all of the four keys are b-bit keys, which are generated randomly and independently.

Off-Line Birthday Attack Resistance. NMAC-E is resistant to off-line birthday attack for verifiable forgery, thanks to the secret "IV", the $K_{\mathrm{i}1}$. Without any knowledge about the "IV", the off-line birthday attack to find a collision pair can't be launched.

Equivalent Key Recovery Attack Resistance. NMAC-E is resistant to equivalent key recovery attack, thanks to the appended key $K_{\mathrm{i}2}$. Even if the attacker can find out the result of M-E$(K_{\mathrm{i}1}, K_{\mathrm{i}2})$, no extension attack can be launched, hence, no equivalent key recovery attack happens.

Slice-by-Slice Key Recovery Attack Resistance. NMAC-E is also resistant to slice-by- slice key recovery attack. Since the key $K_{\mathrm{i}2}$ always resides on the first n bits of the last block of inner hashing, no splitting can be made to $K_{\mathrm{i}2}$, an exhaustive key search must be performed to break $K_{\mathrm{i}2}$. For the key $K_{\mathrm{o}2}$, it always resides on the position from $2b$-th bit to $(2b+n-1)$-bit of the only block of outer hashing, no splitting can be made to $K_{\mathrm{o}2}$, hence, an exhaustive key search must be performed to break $K_{\mathrm{o}2}$.

Divide-and-Conquer Exhaustive-Search Key Recovery. However, the divide-and-conquer exhaustive-search key recovery [18] can be applied to NMAC-E. To recover these four keys, about $2^n \cdot 4$ MAC operations must be performed. Hence, these four keys of NMAC-E can be generated by a key derivation function based on a origin n-bit key K.

On-Line Birthday Attack. The on-line birthday attack is applicable to NMAC-E, after about $2^{n/2}$ on-line MAC queries, a collision pair may be found that NMAC-E(M) = NMAC-E(M'). It means NMAC-E$(M_0||pad(M_0)||x)$ = NMAC-E$(M_0'||pad(M_0')||x)$ always holds, for arbitrary message x, which can lead to a verifiable forgery attack to NMAC-E.

We list the security properties of all NMAC variants discussed in this paper, in Table 2. **OFBAR** stands for off-line birthday attack resistance, **ONBAR** stands for on-line birthday attack resistance, **EKRAR** means equivalent key recovery attack resistance, **SSKRAR** means slice-by-slice key recovery attack resistance, **DCESKRR** stands for divide-and-conquer exhaustive- search key recovery resistance. ϕ means there only one key exists.

Table 2. Security Comparison between NMAC Variants

MAC	OFBAR	ONBAR	EKRAR	SSKRAR	DCESKRR
NMAC_1	Yes	No	No	Yes	ϕ
NMAC_2	Yes	No	No	Yes	ϕ
NMAC_3	Yes	No	No	Yes	No
NMAC	Yes	No	No	Yes	No
$\mathrm{NMAC\text{-}S}_1$	No	No	Yes	No	ϕ
$\mathrm{NMAC\text{-}S}_2$	No	No	Yes	No	ϕ
$\mathrm{NMAC\text{-}S}_3$	No	No	Yes	No	No
NMAC-S	No	No	Yes	No	No
NMAC-E	Yes	No	Yes	Yes	No

Performance Analysis of NMAC-E. NMAC-E introduces two extra keys and a padding process, compared to NMAC. However, since the padding happens at the tail of the message M, and the filling bits of *pad* are some constants, which aims to align the input block M' to be multiples of b bits, the cost of padding is negligible, especially for long message. The introduced extra two keys reside on the tail of the padding bits, the last precess block of the underlying hash function, in fact, each key replaces the former n padding bits of the Merkle-Damgård style, hence, both keys introduce no extra cost.

Further, we can easily prove that NMAC-E is a PRF (pseudorandom function) under the *sole* assumption that the underlying compression function h is a PRF (any PRF is a secure MAC). The proof is straightforward. First, $H(K_{i1}, M)$ is a pf-PRF (prefix free PRF) if the underlying compression function h is a PRF [1]. Second, $H(K_{i1}, M||pad)$ is also a pf-PRF, since the *pad* is some fixed constants. Third, $\text{M-E}'_{(K_{i1},K_{i2})} = H(K_{i1}, M||pad||K_{i2})$ is a PRF, if h is a PRF and $H(K_{i1}, M||pad)$ is a pf-PRF [1]. Since the outer hashing of NMAC can be done by applying the compression function h once, it is also a PRF. Finally, we conclude that NMAC is a PRF, if h is a PRF, a detailed version of this proof is shown in [13].

To utilize the advantage of NMAC-E and to employ the underlying hash functions as a black box like HMAC, we also propose a "HMAC" version of the NMAC-E, named HMAC-E.

We define HMAC-E as:

$$\text{HMAC-E} = \text{HMAC-E}(k_{o1}, K_{o2}, k_{i1}, K_{i2}) = H(K_{o1}||H(K_{i1}||M||pad||K_{i2})||K_{o2})$$

where k_{i1} and k_{i2} are b-bit keys, K_{i2} and K_{o2} are n-bit keys. The key derivation (KD) of HMAC-E is shown as follows, where C_i are pre-defined b-bit constants.

$$\text{KD}_{\text{HMAC-E}} = \begin{cases} K_{i1} = H(K \oplus C_1) \oplus C_1 \\ K_{i2} = H(K \oplus C_2) \\ K_{o1} = H(K \oplus C_3) \oplus C_3 \\ K_{o2} = H(K \oplus C_4) \end{cases}$$

Performance Analysis of HMAC-E. HMAC-E introduces four extra hashing for key derivation, it needs more time than HMAC for key preparation. However, it is negligible for long messages. The input message M must be padded first before being transferred to the underlying hash function, however, the padding content only depends on the message length, which needs negligible time to be accomplished. We point out that both keys K_{o1} and K_{o2} of outer hashing may be removed for simplifying key management, without security loss. A formal security proof for HMAC-E and some optimizations over HMAC-E are provided in [13].

6 Conclusion and Future Work

In this paper, we propose some variants of NMAC, and analyse their security, based on the assumption that the underlying hash functions are secure (WCR and CR). We first point out that NMAC_1, a keyed input version H^2-MAC proposed in [33], is vulnerable to equivalent key recovery attack with complexity about $2^{n/2}$ on-line queries. The security of NMAC_1 and H^2-MAC are totally dependent on the weak collision resistance of the underlying hash function, which directly violates the claimed provable security.

Further, we point out the inner key of NMAC is vulnerable to equivalent key recovery attack, in a related key setting. The security strength of NMAC depends on one of its two keys, even if its both keys are independently and randomly generated.

We also propose a securer variant NMAC-E, which has some advantages compared to NMAC, and HMAC-E. We notice that all kinds of NMAC variants are vulnerable to the on-line birthday attack for verifiable forgery. In fact, a pair (M_0, M_0'), which has the same MAC value after about $2^{n/2}$ on-line queries, is acceptable to some extent[3]. The only problem is that, there are so many collision pairs after the concatenation of arbitrary message x, once a collision pair is found. It implies that hash functions based on Merkle-Damgård construction must be re-fined.

Acknowledgement

We thank the editors of IACR Cryptology ePrint Archive for their valuable comments, for the preliminary version of this paper named "Cryptanalysis of HMAC and Its Variants". This work was partially supported by the program Core Electronic Devices, High-end General Purpose Chips and Basic Software Products in China (No. 2010ZX01037-001-001), and supported by the 973 program of China under contract 2007CB311202, and by National Science Foundation of China through the 61070228 project.

References

1. Bellare, M., Canetti, R., Krawczyk, H.: Pseudorandom functions revisited: the cascade construction and its concrete security. Foundations of Computer Science, Annual IEEE Symposium on 0, 514 (1996)
2. Bellare, M.: New Proofs for NMAC and HMAC: Security Without Collision-Resistance. In: Dwork, C. (ed.) Advances in Cryptology - CRYPTO 2006, Lecture Notes in Computer Science, vol. 4117, pp. 602–619. Springer Berlin / Heidelberg (2006)
3. Bellare, M., Canetti, R., Krawczyk, H.: Keying Hash Functions for Message Authentication. In: Koblitz, N. (ed.) Advances in Cryptology CRYPTO' 96, Lecture Notes in Computer Science, vol. 1109, pp. 1–15. Springer Berlin / Heidelberg (1996)
4. Contini, S., Yin, Y.: Forgery and Partial Key-Recovery Attacks on HMAC and NMAC Using Hash Collisions. In: Lai, X., Chen, K. (eds.) Advances in Cryptology ASIACRYPT 2006, Lecture Notes in Computer Science, vol. 4284, pp. 37–53. Springer Berlin / Heidelberg (2006)
5. Coron, J.S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-damgrd revisited: How to construct a hash function. In: Shoup, V. (ed.) Advances in Cryptology CRYPTO 2005, Lecture Notes in Computer Science, vol. 3621, pp. 430–448. Springer Berlin / Heidelberg (2005)
6. Damgård, I.: A Design Principle for Hash Functions. In: Brassard, G. (ed.) Advances in Cryptology CRYPTO' 89 Proceedings, Lecture Notes in Computer Science, vol. 435, pp. 416–427. Springer Berlin / Heidelberg (1990)

[3] It is not a forgery in this situation, since we have already queried the MAC oracle for their corresponding MAC results.

7. Eastlake, D.E., Jones, P.: US secure hash algorithm 1 (SHA1). RFC 3174, Internet Engineering Task Force (Sep 2001), http://www.rfc-editor.org/rfc/rfc3174.txt
8. Fouque, Pierre-Alain and Leurent, Gatan and Nguyen, Phong: Full key-recovery attacks on hmac/nmac-md4 and nmac-md5. In: Menezes, A. (ed.) Advances in Cryptology - CRYPTO 2007, Lecture Notes in Computer Science, vol. 4622, pp. 13–30. Springer Berlin / Heidelberg (2007)
9. Girault, M., Cohen, R., Campana, M.: A Generalized Birthday Attack. In: Barstow, D., Brauer, W., Brinch Hansen, P., Gries, D., Luckham, D., Moler, C., Pnueli, A., Seegmller, G., Stoer, J., Wirth, N., Gnther, C. (eds.) Advances in Cryptology EUROCRYPT 88, Lecture Notes in Computer Science, vol. 330, pp. 129–156. Springer Berlin / Heidelberg (1988)
10. Hirose, S., Park, J., Yun, A.: A simple variant of the merkle-damgård scheme with a permutation. In: Kurosawa, K. (ed.) Advances in Cryptology ASIACRYPT 2007, Lecture Notes in Computer Science, vol. 4833, pp. 113–129. Springer Berlin / Heidelberg (2007)
11. Kerckhoffs, A.: La cryptographie militaire. Journal des sciences militaires IX, 5–83 (Jan 1883)
12. Leurent, G.: MD4 is Not One-Way. In: Nyberg, K. (ed.) Fast Software Encryption, Lecture Notes in Computer Science, vol. 5086, pp. 412–428. Springer Berlin / Heidelberg (2008)
13. Liu, F., Shen, C., Xie, T.: A security proof for HMAC-E. unpublished (2011)
14. Liu, F., Shen, C., Xie, T., Feng, D.: Cryptanalysis of HMAC and Its Variants. unpublished (2011)
15. Liu, F., Xie, T., Shen, C.: Breaking H^2-MAC using Birthday Paradox. submission to Cryptology ePrint Archive (2011)
16. Menezes, A.J., Vanstone, S.A., Oorschot, P.C.V.: Handbook of Applied Cryptography. CRC Press, Inc., Boca Raton, FL, USA, 1st edn. (1996)
17. Merkle, R.: One Way Hash Functions and DES. In: Brassard, G. (ed.) Advances in Cryptology CRYPTO 89 Proceedings, Lecture Notes in Computer Science, vol. 435, pp. 428–446. Springer Berlin / Heidelberg (1990)
18. Preneel, B., Van Oorschot, P.: On the security of iterated message authentication codes. IEEE Transactions on Information Theory 45(1), 188 – 199 (1999), authentication codes;
19. Preneel, B.: Cryptographic Primitives for Information Authentication State of the Art. In: State of the Art in Applied Cryptography, Lecture Notes in Computer Science, vol. 1528, pp. 49–104. Springer Berlin / Heidelberg (1998)
20. Preneel, B., van Oorschot, P.: On the Security of Two MAC Algorithms. In: Maurer, U. (ed.) Advances in Cryptology EUROCRYPT 96. Lecture Notes in Computer Science, vol. 1070, pp. 19–32. Springer Berlin / Heidelberg (1996)
21. Ristenpart, T., Shrimpton, T.: How to build a hash function from any collision-resistant function. In: Kurosawa, K. (ed.) Advances in Cryptology ASIACRYPT 2007, Lecture Notes in Computer Science, vol. 4833, pp. 147–163. Springer Berlin / Heidelberg (2007)
22. Rivest, R.: The MD4 Message-Digest algorithm. RFC 1320, Internet Engineering Task Force (Apr 1992), http://www.rfc-editor.org/rfc/rfc1320.txt
23. Rivest, R.: The MD5 Message-Digest algorithm. RFC 1321, Internet Engineering Task Force (Apr 1992), http://www.rfc-editor.org/rfc/rfc1321.txt
24. Tsudik, G.: Message authentication with one-way hash functions. SIGCOMM Comput. Commun. Rev. 22, 29–38 (October 1992)

25. Wang, L., Ohta, K., Kunihiro, N.: New Key-Recovery Attacks on HMAC/NMAC-MD4 and NMAC-MD5. In: Smart, N. (ed.) Advances in Cryptology EUROCRYPT 2008, Lecture Notes in Computer Science, vol. 4965, pp. 237–253. Springer Berlin / Heidelberg (2008)
26. Wang, W.: Equivalent Key Recovery Attack on H^2-MAC Instantiated with MD5. In: Kim, T.h., Adeli, H., Robles, R.J., Balitanas, M. (eds.) Information Security and Assurance, Communications in Computer and Information Science, vol. 200, pp. 11–20. Springer Berlin Heidelberg (2011)
27. Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the Hash Functions MD4 and RIPEMD. In: Cramer, R. (ed.) Advances in Cryptology EUROCRYPT 2005, Lecture Notes in Computer Science, vol. 3494, pp. 551–551. Springer Berlin / Heidelberg (2005)
28. Wang, X., Yin, Y., Yu, H.: Finding Collisions in the Full SHA-1. In: Shoup, V. (ed.) Advances in Cryptology CRYPTO 2005, Lecture Notes in Computer Science, vol. 3621, pp. 17–36. Springer Berlin / Heidelberg (2005)
29. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer, R. (ed.) Advances in Cryptology EUROCRYPT 2005, Lecture Notes in Computer Science, vol. 3494, pp. 561–561. Springer Berlin / Heidelberg (2005)
30. Wang, X., Yu, H., Wang, W., Zhang, H., Zhan, T.: Cryptanalysis on HMAC/NMAC-MD5 and MD5-MAC. In: Joux, A. (ed.) Advances in Cryptology - EUROCRYPT 2009, Lecture Notes in Computer Science, vol. 5479, pp. 121–133. Springer Berlin / Heidelberg (2009)
31. Xie, T., Liu, F., Feng, D.: Could The 1-MSB Input Difference Be The Fastest Collision Attack For MD5?. Eurocrypt 2009, Poster Session, Cryptology ePrint Archive, Report 2008/391 (2008), http://eprint.iacr.org/
32. Yasuda, K.: How to fill up merkle-damgrd hash functions. In: Pieprzyk, J. (ed.) Advances in Cryptology - ASIACRYPT 2008, Lecture Notes in Computer Science, vol. 5350, pp. 272–289. Springer Berlin / Heidelberg (2008)
33. Yasuda, K.: HMAC without the “Second” Key. In: Samarati, P., Yung, M., Martinelli, F., Ardagna, C. (eds.) Information Security, Lecture Notes in Computer Science, vol. 5735, pp. 443–458. Springer Berlin / Heidelberg (2009)