TELKOMNIKA, Vol.11, No.2, June 2013, pp. 381~392
ISSN: 1693-6930
accredited by DGHE (DIKTI), Decree No: 51/Dikti/Kep/2010

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
e-mail: liufanbao@yahoo.com.cn[1]

Abstrak

Berdasarkan pada tiga pendekatan konstruksi awal MAC(Message Authentication Code), Kami mengusulkan dan melakukan analisis terhadap beberapa varian dari NMAC. Kami mengusulkan beberapa pemulihan serangan kunci pada varian NMAC tersebut, sebagai contoh, kita dapat memulihkan kunci dalam yang ekuivalen dari NMAC pada sejumlah $O(2^{n/2})$ operasi MAC, padasetting kunci yang berhubungan. Kami mengusulkan NMAC-E, suatu varian NMAC dengan secret envelope, untuk mencapai proses yang lebih efisien dan tanpa kehilangan pada sisi sekuriti, yang hanya membutuhkan satu panggilan pada fungsi hash yang mendasari, bukan pada dua sperti yang ada pada HMAC.

***Kata kunci**: NMAC, keying hash function, equivalent key recovery, MAC forgery, birthday attack*

Abstract

Based on the three earlier MAC (Message Authentication Code) construction approaches, we propose and analyze some variants of NMAC. We propose some key recovery attacks to these NMAC variants, for example, we can recover the equivalent inner key of NMAC in about $O(2^{n/2})$ MAC operations, in a related key setting. We propose NMAC-E, a variant of NMAC with secret envelop, to achieve more process efficiency and no loss of security, which needs only one call to the underlying hash function, instead of two invocations in HMAC.

***Keywords**: NMAC, keying hash function, equivalent key recovery, MAC forgery, birthday attack*

1. Introduction

HMAC (Hash-based Message Authentication Code) [2][3], a derivative of NMAC (Nested Message Authentication Code), is a practically and commonly used, widely standardized message authentication code (MAC) construction. HMAC has two advantages. First, HMAC can make use of current hash functions, the most widely used ones are based on Merkle- Damg˚ard construction [5][14], 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].

After some prevalent iterated hash functions were broken [10][23][24][25][27], the security of NMAC and HMAC instantiated with those hash functions were analyzed [4][7][21][26], which emphasized that NMAC and HMAC instantiated with broken hash functions are weak.

There are mainly three kinds of approaches to construct MAC algorithms by keying hash functions in early days: secret prefix, secret suffix and secret envelop approaches [20]. The secret prefix approach prepends a secret key K to the message M before hashing computation, which is the basic design unit of 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 to known attacks, for a better choice.

Received December 30, 2012; Revised March 12, 2013; Accepted March 25, 2013

This paper, however, analyzes the security of NMAC and its variants based on the assumption that the underlying hash functions are secure (collision resistance, CR), instead of that instantiated with broken hash functions. We also point out that the assumption of CR is a stronger notion than the origin assumption of that the underlying compression function is a PRF [2]. We then 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 a related key setting.

In this paper, we propose some variants of NMAC, and analyze their security, based on the assumption that the underlying hash functions are secure. We first point out that $NMAC_1$ like the keyed input version H^2-MAC proposed in [31], 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 collision resistance of the underlying hash function, instead of the PRF property, 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 it's both keys are independently and randomly generated. We also propose a more secure variant NMAC-E, which has some advantages compared to NMAC, and HMAC-E.

This paper is divided into seven sections. Section 2 recalls the related definitions and background. Section 3 proposes and crypt analyzes some NMAC variants including NMAC with secret prefix approach. Section 4 proposes and analyzes the security of some NMAC variants with secret suffix approach. We present and analyze a better choice of NMAC variant with the modified version of the secret envelop approach, in section 5. Section six presents some related work. We conclude the paper in the last section.

2. Preliminaries

2.1 Notations

Let h be a compression function mapping $\{0, 1\}^n \times \{0, 1\}^b \rightarrow \{0, 1\}^n$, and let H be a concrete hash function mapping $\{0, 1\}^* \rightarrow \{0, 1\}^n$. Let IV be the initial chaining variable of H. Let k denote a secret key with b bits and K denote a secret key with n bits, respectively. x||y denotes the concatenation of two bit strings x and y. |G| denotes the number of elements of the set G. pad(M) denotes the padding bits of M in Merkle-Damg°ard style.

2.2 NMAC

NMAC [2] [3], proposed by Bellare et al., is the basis of the most widely used cryptographic algorithm 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:

$$NMAC_{(Kout,Kin)}(M) = H(K_{out}, H(K_{in}, M)) \quad (1)$$

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

2.3 Security Notions of MAC

A universal forgery attack results in the ability to forge MACs for any message. A selective forgery attack results in a MAC tag on a message of the adversary's choice. An existential forgery merely results in some valid message/MAC pair not already known to the adversary.

3. The security of Some Variants with Secret Prefix

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

We define $NMAC_1$ through keyed IV approach as:

$$(NMAC_1)_{Kin}(M) = H(H(K_{in}, M)) \quad (2)$$

Where the IV of the outer hashing of $NMAC_1$ is not replaced with any secret key. A

keyed input version of $NMAC_1$ was also proposed by Yasuda as H^2-MAC [31], which is shown as (3). It was claimed that H^2-MAC gets rid of the disadvantage of the secret key management without losing the original advantage of HMAC. Wang announced an attack to recover the equivalent key of H^2-MAC instantiated with the broken MD5 [25][27], with about 297 on-line MAC operations [22]. However, Liu et al. pointed out that the absence of the outer key is a real threat to the security of H^2-MAC [12], they could recover the equivalent key using birthday paradox with complexity of about $O(2^{n/2})$ MAC operations.

$$H^2\text{-}MAC_K(M) = H(H(K||pad||M)) \quad (3)$$

On-Line Birthday Attack for Existential Forgery Attack.

If we apply on-line birth- day attack to $NMAC_1$ oracle, after about $2^{n/2}$ queries, we can get a collision pair (M, M') with the same length, which satisfies $NMAC_1(M) = NMAC_1(M')$. Then for arbitrary message x, the equation $NMAC_1(M||pad(M)||x) = NMAC_1(M'||pad(M')||x)$ always holds. 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.

This kind of attack is applicable to all MAC algorithms instantiated with Merkle-Damg°ard hash functions, also noticed by Yasuda [29]. Hence, the rest of the paper will not discuss the specified attack again.

Equivalent Key Recovery Attack to $NMAC_1$.

We use the same technologies to recover the equivalent key of $NMAC_1$ as in [12] of H^2-MAC, with slight modifications to achieve more efficiency. We generate the group one G_1 using H(x), instead of $H(H(C, M_i))$ in [12], which can reduce at least half of the space and time. We apply the generalized birthday attack with two groups [8] to $NMAC_1$ and then recover its equivalent key $K_e = H(K_{in}, M_j)$.

Here, We first define the notation N^2 as $N^2 = H(x)$, where x is an n-bit input (key). x can be viewed as $x = H(C, M)$, where C is a constant and M is the input message. Generally, N^2 is the non-key version of $NMAC_1$. We use different n-bit input messages x_is $(0 \le i \le 2^n - 1)$ to generate the corresponding N^2 values, and use different 1-block messages M_js $(0 \le j \le 2^n - 1)$ to generate the corresponding $NMAC_1$ 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 $H(x_i)$ for r different x_is, which can be randomly generated.
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_js, where M_js are also randomly generated.
3. There will be some pairs (x_i, M_j) that satisfies $(NMAC_1)_{Kin}(M_j) = N^2(x_i)$, with good probability [8].
4. However, we cannot know that whether $x_j = H(K_{in}, M_j)$ further holds, we need to kick out the unsatisfied pairs, which will be discussed later in key selection. After that, we have a pair that satisfies $x_i = H(K_{in}, M_j)$ and $(NMAC_1)_{Kin}(M_j) = N(x_i)$. So we find out the equivalent key of $NMAC_1$ of $K_e = H(K_{in}, M_j) = x_i$.
5. Let pad_0 and pad_1 be the padding bits of M_j and $M_j||pad_0||x$, respectively, for arbitrary message x. We generate the intermediate value of $H(K_{in}, M_j||pad_0||x)$ by computing $y = h(K_e, x||pad_1)$, and calculate H(y) further, and get $NMAC_1(K_{in}, M_j||pad_0||x)$, eventually.

Key Selection

To select a pair that satisfies $x_i = H(K_{in}, M_j)$, we always assume that each pair we have is the right pair. To confirm the assumption, we first randomly generate an arbitrary message α; and then we generate the padding bits pad of the $M_j||pad_0||\alpha$; third, we compute $N^2(\alpha) = h(x_i, \alpha||pad)$ and query the corresponding result θ of $M_j||pad_0||\alpha$ to $NMAC_1$ oracle, we note that θ may be computed as follows.

$$\theta = NMAC_1(K_{in}, M_j||pad_0||\alpha) = H(h(H(K_{in}, M_j), \alpha||pad)) \quad (4)$$

Finally, we check if $N^2(\alpha) = \theta$ holds, if so, (x_i, M_j) is the right pair. Otherwise, discard that pair.

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) ≥ 0.632 [12]. The elements of group G_1 computed by N^2 need $2^{n/2}$ off-line N^2 computations (N^2 just consists of one hash computation). 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 groups using hash tables. Then the above algorithm will require $O(2^{n/2})$ time and space to complete.

We can use the recovered equivalent key K_e to launch any selective forgery attack to $NMAC_1$ without additional on-line query, 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 collision resistance of the underlying hash function, not the strength of the used key.

3.2 The security of $NMAC_2$

We define $NMAC_2$ as:

$$(NMAC_2)_{Kout}(M) = H(K_{out}, H(M)) \quad (5)$$

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 $H(K_{out}, H(M))$ easily, we cannot use that information to launch the extension attack to $NMAC_2$.

Off-Line Birthday Attack to $NMAC_2$

We first apply an 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* $NMAC_2(M) = NMAC_2(M')$, eventually. Then, $NMAC_2(M||pad(M)||x) = NMAC_2(M'||pad(M')||x)$ always holds, for arbitrary message x. It means that we can generate verifiable forgery to $NMAC_2$, we first query for the MAC value of $M||pad(M)||x$, and get the MAC value for $M'||pad(M')||x$, eventually.

3.3 The security of $NMAC_3$

We define $NMAC_3$ as:

$$(NMAC_2)_{Kio}(M) = H(K_{io}, H(K_{io},M)) \quad (6)$$

Where the inner and outer keys are both set to K_{io}.

The on-line birthday attack for existential forgery applied to $NMAC_1$ is also applicable to $NMAC_3$ with any modification. Further, we point out that the off-line birthday attack to get existential 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 $H(K_{io}, H(K_{io}, M_0))$.
2. Assume the unknown $H(K_{io}, M_0)$ *be* x_0, and pad_0 be the padding bits of x_0. We already know the corresponding value of $H(K_{io}, x_0)$ (an equivalent key of the inner hashing), which is $NMAC_3(M_0)$.
3. Based on the known $H(K_{io}, x_0)$, we launch an off-line birthday attack to find a collision pair (M_x, M_x') satisfying $H(K_{io}, x_0||pad_0||M_x) = H(K_{io}, x_0||pad_0||M_x')$.
4. For arbitrary message x, we can launch a verifiable forgery attack.

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

3.4 The security of NMAC

As pointed out by Bellare et al., the on-line birthday attack for existential forgery attack is also applicable to NMAC [2], here we omit the details. However, we further notice that we can generate existential forgery for NMAC, by an 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 $NMAC_{(Kout, Kin)}$ and $NMAC_{(Kout', Kin')}$. We set the relation between (K_{out}, K_{in}) and (K_{out}', K_{in}') as follows:

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

Where these two oracles share the same outer key, and the inner key of $NMAC_{(Kout', Kin')}$ can be any known n-bit Constants, such as the IV of H.

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

1. Query $NMAC_{(Kout, Kin)}$ oracle for the corresponding values of $2^{n/2}$ different M_is, store their values in group one G_1.
2. Query $NMAC_{(Kout', Kin')}$ oracle for the corresponding values of $2^{n/2}$ different M_j's, store their values in group two G2.
3. A pair (M_i, M_j') satisfies $NMAC_{(Kout, Kin)}(M_i) = NMAC_{(Kout', Kin')}(M_j')$ (the generalized birthday attack with two groups), and further satisfies $H(K_{in}, M_i) = H(K_{in}', M_j')$ (an inner collision happens).
4. Since $H(K_{in}, M_i) = H(K_{in}', M_j')$, and we know the value of K_{in}' and M_j', hence we can calculate the very value of $K_e = H(K_{in}, M_i) = H(K_{in}', M_j')$.

We conclude that the equivalent inner key of NMAC is totally dependent on the generalized birthday attack, 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 it's both key are independently and randomly generated. As pointed out by the editors of Cryptology ePrint Archive in our preliminary version of this paper, the equivalent key recovery attack to NMAC is not applicable to the practical HMAC, since the HMAC keys are derived from a base key, and there exists no related key.

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 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 strengthen of the key.

Theorem 1

The security of H(M||K) is totally dependent on the collision resistance of the underlying hash function H, instead of the strengthen of the used key.

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 an 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 [16].

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 ($1 \le j \le n/8$) 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) ranges 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 strength of the key. Here, we assume that the underlying hash function is secure, in fact, for some applications with broken hash functions; the situation is totally in danger. For example, APOP (Authentication Post Office Protocol), which is instantiated with broken MD5, applies secret suffix approach; an attacker can recover the password as long as 352 bits in practical time [11].

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 cases 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:

$$(NMAC\text{-}S_1)_{Kin}(M) = H(H(M||K_{in})) \quad (7)$$

Where the outer key K_{out} is omitted. The off-line birthday attack can be applied to NMAC- S_1. Full Key Recovery Attack to NMAC-S_1. We can directly apply the full key recovery attack to $H(M||K_{in})$, 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:

$$(NMAC\text{-}S_2)_{Kout}(M) = H(H(M)||K_{out}) \quad (8)$$

Where the inner key K_{in} is omitted. The off-line birthday attack can be applied to NMAC-S_2 . 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:

$$(NMAC\text{-}S_3)_{Kio}(M) = H(H(M||K_{io})||K_{io}) \quad (9)$$

Where the inner and outer keys are equal. The off-line birthday attack can be applied to NMAC-S_3.

Key Recovery Attack to NMAC-S_3

We can directly apply the full key recovery attack to $H(M||K_{io})$, 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:

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

Where the inner and outer keys are different. The off-line birthday attack can be applied to NMAC-S.

Inner Key Recovery Attack to NMAC-S.

We can directly apply the full key recovery attack to $H(M||K_{in})$, 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 analyzed in NMAC-S_2. It seems that we have to apply additional off-line birthday attack to H(M), for a meaningful existential 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 $H(M||K_{in})$. We always assume that n|b, which means that b is the multiples of n. Let pad_n $(1||0^*)$ be the padding bits of M, pad_n is defined as

$$n|(|M| + |pad_n|) \quad (10)$$

We re-define the inner hashing form as:

$$H(M||pad_n||K_{in})$$

Where the inner key K_{in} resides as a whole part on the input block. We have the following theorem for the key recovery attack.

Theorem 2

Slice-by-slice key recovery strategy cannot be applied to $H(M||pad_n||K_{in})$, for launching key recovery attack.

Proof. Since $n|b$ and $n|(|M| + |pad_n|)$, and $|K_{in}| = n$, *then* $n|(|M||pad_n||K_{in}|)$, hence, no slice can be made to the key K_{in}.

However, the NMAC-S Variants after modification are still vulnerable to off-line birthday attack for existential 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 NMAC-E with Modified Secret Envelop

We propose NMAC-E with 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_n, which can be some fixed constants, before appending the outer key K . We define NMAC-E as:

$$NMAC\text{-}E_{(K)} = H(K, M||pad_n||K)$$

Where K is a randomly generated n-bit key. $M||pad_n$ is multiples of n bits.

5.2 The security of NMAC-E

Off-Line Birthday Attack Resistance.

NMAC-E is resistant to off-line birthday attack for existential forgery, thanks to the secret "IV", the key K. 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. Even if the attacker can find out the result of NMAC-$E_{(K)}$ easily, 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 as proven in Theorem 2.

Divide-and-Conquer Exhaustive-Search Key Recovery.

The divide-and-conquer exhaustive-search key recovery [16] cannot be applied to NMAC-E, since our scheme use one key, and a brute force attack should be performed to find out the key. The attacks performed to NMAC also show that it is not necessary to bind two keys to strengthen the MAC scheme.

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 $NMAC\text{-}E(M) = NMAC\text{-}E(M')$.

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. ¤ 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	¤
$NMAC_2$	Yes	No	No	Yes	¤
$NMAC_3$	Yes	No	No	Yes	No
NMAC	Yes	No	No	Yes	No
NMAC-S_1	No	No	Yes	No	¤
NMAC-S_2	No	No	Yes	No	¤
NMAC-S_3	No	No	Yes	No	No
NMAC-S	No	No	Yes	No	No
NMAC-E	Yes	No	Yes	Yes	¤

Performance Analysis of NMAC-E.

NMAC-E uses only one call of the underlying hash function, but introduces a message 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. Moreover, NMAC-E processes only one more block, compared to the inner hashing of NMAC. Hence, the NMAC-E is efficient than NMAC.

5.3 Security Proof for NMAC-E

We first recall the Theorem 3.1 of [1]; it says that H^* is a pf-VI-PRF, if the underlying compression function h is an FI-PRF. we list the detail of the Theorem as follow, where we change the notions to suitable this paper.

Theorem 3.1 of [1].

Let h be a function family with Dom(h) = $\{0, 1\}^b$, Range(h) = $\{0, 1\}^n$, and key length n. Suppose h is (t, q, 1, ε)-secure and let l ≥ 1. Then h^* is (t, q, l, ε)-secure against prefix-free distinguishers, where

$$t = t' - cq.(l + n + b).(Time(h) + \log q)$$
$$\varepsilon = ql\varepsilon'$$

Here, c is a specific, small constant whose value can be determined from the proof. For the detail of this proof, please refer [1].

In [1], a construction called F^{acsc} was proposed to construct VI-PRF family $_{\delta-A}F^*$, based on an FI-PRF. $_{\delta-A}F$ is constructed as follows. Given a family F with key length n + δ, having key (K, d) where $K \in \{0, 1\}^n$ and $d \in \{0, 1\}^\delta$, let $_{\delta-A}F_{K,d}(x) = F(x||d)$ for all b-bit x. Finally, $_{\delta-A}F^*$ is a VI-PRF, which is provided by Corollary 4.2 of [1].

Corollary 4.2 of [1].

Let h be a function family with Dom(h) = $\{0, 1\}^b$, Range(h) = $\{0, 1\}^n$, and key length n. Suppose h is (t, q, 1, ε)-secure and let l ≥ 1. Then $_{\delta\text{-}A}h^*$ is (t, q, l, ε)-secure, where

$$t = t' - cq.(n + b + (l + \log \delta).Time(h) + (n + l + b).\log q)$$
$$\epsilon = l_{q\epsilon}' + blq2^{-\alpha}.$$

However, we can't get the conclusion that NMAC-E is a VI-PRF directly, based on the both conclusions of [1], since another part padding is inserted between the message M and the "δ" key Ki2 in NMAC-E.

We notice that NMAC-E(K) = H ∗(K, M ||padn ||K), where M is first padded with some fixed bits, and then transferred to be processed by H. We divide the proof of that; NMAC-E is a PRF if the underlying h is a dual PRF, into two parts. First, we prove that H ∗(K, M ||pad_n) is a pf-VI-PRF. Second, we prove that NMAC-E is a PRF under the sole assumption that the underlying compression function h is a PRF.

Theorem 3.

If the compression function h of the underlying hash function H is a PRF, then $H^*(K, M||pad_n)$, where pad_n is the padding bits of M without length information, is a pf-VI-PRF.

Proof. From the Theorem 3.1 of [1], we know that H^*(K, M) is a pf-PRF if the underlying compression function h is a PRF. Since the content of pad_n (the number of '0's to be filled) is totally determined by the length of M, which means it is obviously known. Hence, M||pad_n is not prefix-free, we can directly get the conclusion that $H^*(K, M||pad_n)$ is a pf-VI-PRF, if the underlying h is a PRF.

Based on the Theorem 3 and the corollary 4.2 of [1], we can get the conclusion that NMAC-E is a PRF under the sole assumption that the underlying compression function h is a PRF.

Theorem 4.
If the compression function h of the underlying hash function H is a PRF, then NMAC-$E_{(K)}$ construction $H^*(K, M||pad_n||K)$, where pad_n is the padding bits of M without length information, is a VI-PRF.
Proof. This theorem can be conducted directly based on the Theorem 3 and the Corollary 4.2 of [1].

5.4 HMAC-E

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:

$$HMAC\text{-}E = HMAC\text{-}E_{(K)} = H(K||M||pad_n||K)$$

Where K is an n-bit key.

HMAC-E is a PRF under the sole assumption that the underlying hash function is a dual PRF, the proof is similar to the security proof of NMAC-E, for the lack of space, we omit the details. HMAC-E calls the underlying hash function H only once, whereas HMAC requires two invocations of H, and HMAC-E involves no key derivation. HMAC-E can achieve more process efficiency, compared to HMAC, and without loss of security.

6. Related Work

The ENMAC algorithm [15] increases efficiency over HMAC using a secret-prefix approach for short messages. The MDP construction [9] operates as a secret-prefix MAC algorithm for messages of any length by applying a permutation. The Sandwich construction [30] is similar to our proposed NMAC-E, but it suffers low efficiency over short messages compared to our scheme. The L-Lane HMAC construction [29] was proposed to avoid the general birthday attack to HMAC. The BNMAC algorithm [28] aim to improve efficiency over HMAC, using single key approach. The H^2-MAC construction [31], omitting the outer hash of HMAC, tends to improve efficiency over HMAC with provable secure, but recent research shows that it is vulnerable to equivalent key recovery attack [12] based on the assumption that the underlying hash function is (weak) collision resistance.

7. Conclusion and Future Work

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

We notice that all kinds of NMAC variants, based on Merkle-Damg°ard construct hash functions, are vulnerable to the on-line birthday attack for verifiable forgery. In fact, a pair (M_i, M_j'), which has the same MAC value after about $2^{n/2}$ on-line queries, is acceptable to some extent. It is not a forgery in this situation, since we have already queried the MAC oracle for their corresponding MAC results. The only problem is that, there are so many collision pairs after the concatenation of arbitrary message x, once a collision pair is found.

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. 1996; 514.
[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. Heidelberg: Springer Berlin. 2006: 602–619.
[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. Heidelberg: Springer Berlin. 1996: 1–15.
[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. Heidelberg: Springer Berlin. 2006: 37–53.
[5] Damg°ard I. A Design Principle for Hash Functions. In: Brassard, G. (ed.) Advances in Cryptology CRYPTO' 89 Proceedings, Lecture Notes in Computer Science, vol. 435. Heidelberg: Springer Berlin. 1990: 416–427.
[6] Eastlake DE, Jones P. US secure hash algorithm 1 (SHA1). RFC 3174, Internet Engineering Task Force (Sep 2001), http://www.rfc-editor.org/rfc/rfc3174.txt.
[7] Fouque, Pierre-Alain, Leurent, Gatan, 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. Heidelberg: Springer Berlin. 2007: 13–30.
[8] 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. Heidelberg: Springer Berlin. 1988: 129–156.
[9] Hirose S, Park J, Yun A. A simple variant of the merkle-damg°ard scheme with a permutation. In: Kurosawa K. (ed.) Advances in Cryptology ASIACRYPT 2007, Lecture Notes in Computer Science, vol. 4833. Heidelberg: Springer Berlin. 2007: 113–129.
[10] Leurent G. MD4 is not One-Way. In: Nyberg K (ed.) Fast Software Encryption, Lecture Notes in Computer Science, vol. 5086. Heidelberg: Springer Berlin. 2008: 412–428.
[11] Liu F, Liu Y, Xie T, Feng D, Feng Y. Fast Password Recovery Attack: Application to apop. *Journal Intelligent Manufacturing*. 2012: 1–11.
[12] Liu F, Xie T, Shen C. Breaking H^2-MAC Using Birthday Paradoc. Cryptology eprint Archive, Report 2011/647 (2011), http://eprint.iacr.org/
[13] Menezes AJ, Vanstone SA, Oorschot PCV. Handbook of Applied Cryptography. CRC Press, Inc., Boca Raton, FL, USA, 1st edn. 1996.
[14] 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.Heidelberg: Springer Berlin. 1990: 428–446.
[15] Patel S. An efficient mac for short messages. In: Nyberg K., Heys H. (eds.) Selected Areas in Cryptography, Lecture Notes in Computer Science, vol. 2595. Heidelberg: Springer Berlin. 2003: 353–368.
[16] Preneel B, Van Oorschot P. On the security of iterated message authentication codes. *IEEE Transactions on Information Theory*. 1999; 45(1): 188 – 199.
[17] 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. Heidelberg: Springer Berlin. 1998: 49–104.
[18] 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. Heidelberg: Springer Berlin. 1996: 19–32.
[19] Rivest R. The MD5 Message-Digest algorithm. RFC 1321, Internet Engineering Task Force (Apr 1992), http://www.rfc-editor.org/rfc/rfc1321.txt.
[20] Tsudik G. Message authentication with one-way hash functions. *SIGCOMM Comput Commun Rev.* 1992; 22: 29–38.
[21] 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. Heidelberg: Springer Berlin. 2008: 237–253.
[22] Wang W. Equivalent Key Recovery Attack on H2-MAC Instantiated with MD5. In: Kim Th, Adeli H, Robles RJ, Balitanas M (eds.) Information Security and Assurance, Communications in Computer and Information Science, vol. 200. Heidelberg: Springer Berlin. 2011: 11–20.
[23] 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. Heidelberg: Springer Berlin. 2005: 551–551.
[24] 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. Heidelberg: Springer Berlin. 2005: 17–36.
[25] 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. Heidelberg: Springer Berlin. 2005: 561–561.
[26] 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. Heidelberg: Springer Berlin. 2009: 121–133.
[27] 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/.
[28] Yasuda K. Boosting merkle-damg°ard hashing for message authentication. In: Kurosawa, K. (ed.) Advances in Cryptology ASIACRYPT 2007, Lecture Notes in Computer Science, vol. 4833. Heidelberg: Springer Berlin. 2007: 216–231.
[29] Yasuda K. Multilane hmac-security beyond the birthday limit. In: Srinathan K, Rangan C, Yung M. (eds.) Progress in Cryptology INDOCRYPT 2007, Lecture Notes in Computer Science, vol. 4859. Heidelberg: Springer Berlin. 2007: 18–32.
[30] Yasuda K. "sandwich" is indeed secure: How to authenticate a message with just one hashing. In: Pieprzyk J, Ghodosi H, Dawson E. (eds.) Information Security and Privacy, Lecture Notes in Computer Science, vol. 4586. Heidelberg: Springer Berlin. 2007: 355–369.
[31] 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. Heidelberg: Springer Berlin. 2009: 443–458.