Forgery and Key Recovery Attacks on PMAC and Mitchell's TMAC Variant

Changhoon Lee¹, Jongsung Kim^{1,2}, Jaechul Sung³,
Seokhie Hong¹, Sangjin Lee¹

¹ Center for Information Security Technologies(CIST),
Korea University, Anam Dong, Sungbuk Gu, Seoul, Korea
{joshep,crypto77,hsh,sangjin}@cist.korea.ac.kr

² Katholieke Universiteit Leuven, ESAT/SCD-COSIC, Belgium
Kim.Jongsung@esat.kuleuven.be

³ Department of Mathematics, University of Seoul, 90
Cheonnong Dong, Dongdaemun Gu, Seoul, Korea
jcsung@uos.ac.kr

Abstract. In this paper we discuss the security of PMAC, a provably secure and parallelizable MAC scheme proposed by Black and Rogaway, and Mitchell's TMAC variant, proposed to improve the security of TMAC. We show how to devise forgery attacks on PMAC and compare the success rate of our forgery attacks with their security bound. We also present forgery attacks on TMAC variant and show the security of TMAC variant is not improved in the sense of the forgery attack. Furthermore, key recovery attacks on PMAC and TMAC variant are presented in various parameters. Our results imply they have no significant advantage in comparison with other well-established MAC schemes.

Keywords : MAC (Message Authentication Code), Forgery Attacks, Key Recovery Attacks, CBC-MAC, PMAC, TMAC Variant.

1 Introduction

A Message Authentication Code (MAC) scheme is a symmetric-key cryptosystem that is extensively used to protect a message from unauthorized alteration in internet security protocols and in banking applications. Generally, a MAC scheme consists of a tag generation algorithm (for a sender) and a tag verification algorithm (for a receiver). The tag generation algorithm takes as input a message M , a secret key K and a nonce IV , and returns a tag T . The verification algorithm takes as input M , T , K and IV , and returns a binary value indication whether or not the message/tag/nonce tuple is valid.

In the standard model of evaluating MAC security, there are two main kinds of practical attacks on MAC algorithms. One is a forgery attack in which an attacker tries to produce a message with a valid MAC without the knowledge of the secret key K . The other is a key recovery attack in which an attacker tries to obtain the secret key used to generate MAC values. A successful key recovery attack also enables the construction of arbitrary numbers of forgeries.

PMAC [2], which uses a k -bit key and an arbitrary-bit length message, and outputs a τ -bit tag, was proposed as a fully parallelizable alternative to CBC-MAC. It is provably secure under the assumption that the underlying n -bit block cipher is a pseudorandom permutation. More precisely, the upper bound on the advantage of a forgery attacker for PMAC is $\frac{(\sigma+1)^2}{2^{n-1}}$, where $\sigma = \sum_{i=1}^q \lceil \frac{|M_i|}{n} \rceil$ and M_i are message queries. However, there is no known cryptanalytic result on PMAC, which quantifies the number of message queries with which the forgery attacker can break PMAC.

TMAC [11], which requires two keys, a k -bit key and a n -bit key, was proposed by Kurosawa and Iwata. It is a refinement of XCBC-MAC [4] with three different keys and attains a provable security as the EMAC [19] does. Recently, some researchers [21, 15, 8, 16] evaluated the security of TMAC. Among them, Mitchell [15, 16] presented various attacks on TMAC and proposed a new TMAC variant to improve the security of TMAC.

In this paper we discuss the security of PMAC and TMAC variant against the forgery and the key recovery attacks. We first show forgery attacks on PMAC, which provide a lower bound on the attacker's advantage together with a required number of message queries. Our attack on PMAC where truncation is performed requires about $2^{\frac{n}{2}+1}$ known messages and $2^{n-\tau}$ MAC verifications where τ is the bit-size of the MAC, and the attack on PMAC where no truncation is performed requires about $2^{\frac{n}{2}+1}$ known messages. All these attacks work with a success rate of 0.63. The latter attack shows that there exists a forgery attack on PMAC with $2^{\frac{n}{2}+1}$ message queries (all are one-block messages) such that the lower bound on the advantage of the attacker for PMAC is $L_b = \frac{(\sigma+1)^2}{2^{n+3}}$, since $\sigma = 2^{\frac{n}{2}+1}$ implies $L_b \approx 0.5 (< 0.63)$ (more details will be given in Sect. 3.1). Note that it does not contradict the security proof of PMAC in [2] and $L_b = \frac{1}{16} \cdot U_b$ for the upper bound $U_b = \frac{(\sigma+1)^2}{2^{n-1}}$. We also give a forgery attack on the TMAC variant whose complexity is equal to that of the forgery attack on TMAC, $2^{\frac{n}{2}+1}$ known messages. This fact implies that the security of TMAC variant is not improved in the sense of the forgery attack. Furthermore, key recovery attacks on PMAC and TMAC variant are also presented in various parameters.

This paper is organized as follows: The descriptions of PMAC, TMAC variant are presented in Sect. 2. In Sect. 3 and Sect. 4, forgery and key recovery attacks on PMAC and TMAC variant are presented, respectively. Finally, we summarize our attacks and compare them with previous attacks on other MAC schemes in Sect. 5.

2 Descriptions of PMAC and TMAC Variant

The most common block-cipher based MAC is CBC-MAC [5]. This scheme operates as follows: Suppose the underlying block cipher E has an n -bit block and uses a k -bit key K . We write $E_K(x)$ for the encryption of x using key K , where x is an n -bit block. A padded message M is first split into $M[1], \dots, M[r]$ where each block $M[i]$ is of size n -bit and then the CBC-MAC value of the M is computed as $\text{CBC-MAC}_K(M) = H_r$, where $H_0 = 0$, $H_i = E_K(H_{i-1} \oplus M[i])$, $1 \leq i \leq r$.

2.1 PMAC (Parallelizable Message Authentication Code)

Unlike popular MAC algorithms such as CBC-MAC and Hashed MAC, PMAC proposed by Black and Rogaway is not inherently sequential but parallelizable. It also uses a key K and an additional key information $L = E_K(0^n)$ which is defined from K . Furthermore, it can handle messages which are not multiples of the block length without the need for obligatory padding, which would increase the number of block cipher calls. PMAC operates as follows (See Fig. 1):

Algorithm $\text{PMAC}_K(M)$

1. $L \leftarrow E_K(0^n)$
2. if $|M| > n2^n$ then return 0^τ
3. Partition M into $M[1] \cdots M[r]$
4. for $i \leftarrow 1$ to $r - 1$ do
5. $X[i] \leftarrow M[i] \oplus \gamma_i \cdot L$
6. $Y[i] \leftarrow E_K(X[i])$
7. $\Sigma \leftarrow Y[1] \oplus \cdots \oplus Y[r - 1] \oplus \text{pad}(M[r])$
8. if $|M[r]| = n$ then $X[r] = \Sigma \oplus L \cdot x^{-1}$
 else $X[r] \leftarrow \Sigma$
9. $MAC = \text{truncation of first } \tau \text{ bits of } (E_K(X[r]))$
10. return MAC

PMAC : Constants γ_i are Gray codes which are polynomial multiplication operators over $GF(2^n)$. The $\text{pad}(A)$ means the string $A||10^{n-|A|-1}$ in the case $|A| < n$.

2.2 TMAC Variant

TMAC [11], which requires two keys, a k -bit K and an n -bit K' , was proposed by Kurosawa and Iwata. It is a refinement of XCBC [4] using $(k + 2n)$ -bit keys, that is, a key triple (K_1, K_2, K_3) used in XCBC is replaced by $(K, K' \cdot u, K')$, where u is a constant which is a multiplication operator over $GF(2^n)$ defined in [11]. The usage of the $(K, K' \cdot u, K')$ depends on whether or not padding has been necessary. If the length of the input message M is a multiple of n , then TMAC operates exactly the same as CBC-MAC using the key K , except for XORing an n -bit key $K' \cdot u$ before encrypting the last block. Otherwise, after padding is applied, TMAC operates CBC-MAC using the key K , but this time an n -bit key K' is XORed before the last block is encrypted.

Recently, several attacks on TMAC was presented in [21, 15, 8, 16]. In particular, Mitchell [15, 16] considered various attacks (including forgery attacks, partial key attacks and key recovery attacks) against TMAC and proposed a new TMAC variant to remove a simple algebraic relationship between $K' \cdot u$ and K' which is used as the basis of the key recovery attacks. The only difference between this variant and TMAC is to use a key triple, i.e., $(K, E_{K'}(S_2), E_{K'}(S_3))$ substitutes for $(K, K' \cdot u, K')$ in the TMAC variant, where S_2 and S_3 are different fixed n -bit strings. For convenience we call this scheme TMAC-V. Fig. 2 represents the TMAC-V scheme.

Fig. 1. The PMAC scheme

Fig. 2. The TMAC-V scheme

3 Forgery and Key Recovery Attacks on PMAC

In this section we present forgery and key recovery attacks on PMAC where truncation is used or not.

3.1 Forgery Attacks on PMAC

For clarification we separately describe our forgery attacks on PMAC with no truncation and on PMAC with truncation.

PMAC where no truncation is performed ($\tau = n$): We first consider the case where no truncation is performed. Assume that the attacker obtains the corresponding MAC values for approximately $2^{n/2}$ different $(q + 1)$ -block messages $M(1)^i = (M[1], \dots, M[q], X^i)$ where $(M[1], \dots, M[q])$ is any fixed sequence of n -bit blocks and the last blocks X^i ($1 \leq i \leq 2^{n/2}$) whose sizes are less than n bits are pairwise-distinct, and q is an arbitrary positive integer. Note that all these messages require padding. Assume that the attacker also obtains the corresponding MAC values for approximately $2^{n/2}$ different $(q + 1)$ -block messages $M(2)^j = (M[1], \dots, M[q], Z^j)$ where $(M[1], \dots, M[q])$ is the same sequence as that in $M(1)^i$ and the last blocks Z^j ($1 \leq j \leq 2^{n/2}$) whose sizes are equal to n bits are pairwise-distinct.

Using the birthday paradox arguments [17] we can expect to find at least one external collision with a probability of $0.63 (\approx 1 - e^{-1})$. In other words, one of the MAC values from the first set of messages $M(1)^i$ will equal to one of the MAC values from the second set of messages $M(2)^j$ with a probability of 0.63. Denote by $M(1)^* = (M[1], \dots, M[q], X^*)$ and $M(2)^* = (M[1], \dots, M[q], Z^*)$ the pair of messages which cause this collision. Since the n -bit blocks $(M[1], \dots, M[q])$ are the same for the two messages, $\sum_{i=1}^q (Y[i])$ are also the same for the two messages where each $Y[i]$ means the output of the i -th E cipher (see Fig. 1). We then have the following equation for a collision.

$$E_K((X^* || padding) \oplus \sum_{i=1}^q (Y[i]) \oplus 0^n) = E_K(Z^* \oplus \sum_{i=1}^q (Y[i]) \oplus L \cdot x^{-1}). \quad (1)$$

Since $E_K(\cdot)$ is a permutation on the set of all n -bit blocks, we get

$$L \cdot x^{-1} = (X^* || padding) \oplus Z^*. \quad (2)$$

Thus, the attacker can compute the key information L from Eq. (2) with no knowledge of K and can use it to forge the MAC of a new message. The attacker chooses a message $M(1)^k = (M[1], \dots, M[q], X^k)$ in the first set of messages $M(1)^i$ which does not cause a collision and gets the corresponding MAC value as follows.

$$E_K((X^k || padding) \oplus \sum_{i=1}^q (Y[i]) \oplus 0^n). \quad (3)$$

The attacker then also knows the corresponding MAC value for a message $(M[1], \dots, M[q], U)$ is the same as the MAC value (Eq. (3)) of the $M(1)^k$, where $U = (X^k || \text{padding}) \oplus L \cdot x^{-1}$ and U is n -bit. That is, one can forge the MAC of $(M[1], \dots, M[q], (X^k || \text{padding}) \oplus L \cdot x^{-1})$ which does not belong to the second set of messages by our assumption. The complexity of this attack is thus $[0, 0, 2^{n/2+1}, 0]$. Note that the tuple $[a, b, c, d]$ introduced in [5] is used to quantify the resources needed for an attack where a denotes the number of off-line block cipher encipherments (or decipherments), b denotes the number of known message string/MAC pairs, c denotes the number of chosen message string/MAC pairs, and d denotes the number of on-line MAC verifications (i.e., to submit a message string/MAC pair and receive an answer indicating whether or not the MAC is valid than to obtain the genuine MAC value for a message).

Taking into account one-block message sets $M(1)^i = X^i$ and $M(2)^j = Z^j$ in the above two message sets, i.e., $q = 0$, we can convert this chosen message attack into a known message attack with a $[0, 2^{n/2+1}, 0, 0]$ complexity. It follows that we can construct a forgery attack with $2^{n/2+1}$ message queries such that the lower bound on the advantage of the forgery attacker for PMAC is $L_b = \frac{(\sigma+1)^2}{2^{n+3}}$, since $\sigma = 2^{n/2+1}$ implies $L_b \approx 0.5 (< 0.63)$ where $\sigma = \sum_{i=1}^q \frac{\lceil |M_i| \rceil}{n}$ and M_i are message queries. Note that if we perform this attack with a random function instead of PMAC, the probability of producing a forgery is 2^{-n} , so this probability can be ignored in the computation of the advantage of the forgery attacker. Since $L_b = \frac{1}{16} \cdot U_b$ for the upper bound $U_b = \frac{(\sigma+1)^2}{2^{n-1}}$ in [2], our attack and [2] show that there exists a forgery attacker $\mathcal{A}$ for PMAC such that $\frac{1}{16} \cdot U_b \leq \text{Adv}(\mathcal{A}) \leq U_b$, where $\text{Adv}(\mathcal{A})$ is the advantage of the $\mathcal{A}$. This does not contradict the security bound shown by Black and Rogaway [2], but shows the tightness of their security bound.

PMAC where truncation is performed ($\tau < n$): Now we consider the case where truncation is used in the PMAC algorithm. As like the foregoing assumptions we suppose that the attacker obtains two sets which are composed of $2^{n/2}$ message/MAC pairs each, denoted $M(1)^i = (M[1], \dots, M[q], X^i)$ and $M(2)^j = (M[1], \dots, M[q], Z^j)$ where the last blocks X^i and Z^j have the same conditions as those of the above attack.

Similarly, we can expect to find about $2^{n-\tau}$ external collisions (i.e., messages between two message sets with matching MAC values). Let these messages which cause external collisions denote respectively $M(1)^*(l) = (M[1], \dots, M[q], X^*(l))$ and $M(2)^*(l) = (M[1], \dots, M[q], Z^*(l))$ for $1 \leq l \leq 2^{n-\tau}$. The problem remains to find the internal collision from amongst the many external collisions. If $M(1)^*(k) = (M[1], \dots, M[q], X^*(k))$ and $M(2)^*(k) = (M[1], \dots, M[q], Z^*(k))$ for some k ($1 \leq k \leq 2^{n-\tau}$) are the internal collision, as like the previous subsection, we get the following equation.

$$E_K((X^*(k) || \text{padding}) \oplus \sum_{i=1}^q Y[i] \oplus 0^n) = E_K(Z^*(k) \oplus \sum_{i=1}^q Y[i] \oplus L \cdot x^{-1}).$$

So we get

$$L \cdot x^{-1} = (X^*(k) || padding) \oplus Z^*(k). \quad (4)$$

In order to find the index k , i.e., the internal collision, the attacker can perform the following process: First, the attacker computes the candidates of $L \cdot x^{-1}$, denoted $L \cdot x^{-1}(l)$, where $L \cdot x^{-1}(l) = (X^*(l) || padding) \oplus Z^*(l)$, $1 \leq l \leq 2^{n-\tau}$ (from the above $M(1)^*(l)$ and $M(2)^*(l)$). Second, the attacker chooses a message $(M[1], \dots, M[q], X^S)$ in the first set of messages $M(1)^i$. Third, the attacker requires the MAC verifications for the $2^{n-\tau}$ messages $(M[1], \dots, M[q], U(l))$ with the MAC value of $(M[1], \dots, M[q], X^S)$ where $U(l) = (X^S || padding) \oplus L \cdot x^{-1}(l)$. That is, the attacker checks whether or not the MAC values for $(M[1], \dots, M[q], U(l))$ are the same as the MAC value of $(M[1], \dots, M[q], X^S)$ by the MAC verifications. If a message $(M[1], \dots, M[q], U(l))$ passes the above test, $L \cdot x^{-1}(l)$ is a candidate of $L \cdot x^{-1}$. By this process, the attacker can reduce the $2^{n-\tau}$ candidates of internal collisions to $2^{n-2\tau}$. If the attacker repeats this process by $\lceil n/\tau \rceil$ times, then the attacker can find the desired internal collision and from this collision the attacker can also obtain $L \cdot x^{-1}$ with no knowledge of K . Thus, the attacker can forge the MAC of a new message using the obtained value of $L \cdot x^{-1}$ as in the previous attack. This attack requires about $2^{n/2+1}$ chosen messages and $2^{n-\tau} + 2^{n-2\tau} + \dots + 2^{n-\lceil n/\tau \rceil \cdot \tau} \approx 2^{n-\tau}$ MAC verifications. That is, the complexity of this attack is $[0, 0, 2^{n/2+1}, 2^{n-\tau}]$. Similarly, this attack can be converted into a known message attack with a $[0, 2^{n/2+1}, 0, 2^{n-\tau}]$ complexity (by considering $q = 0$).

3.2 Key Recovery Attacks on PMAC

In order to devise key recovery attacks on PMAC, we ask for MAC values of $\lceil \frac{k}{\tau} \rceil$ one-block messages M_i whose bit-lengths are all less than n , i.e., with the known message attack we obtain $(M_1, T_1), (M_2, T_2), \dots, (M_{\lceil \frac{k}{\tau} \rceil}, T_{\lceil \frac{k}{\tau} \rceil})$, where each T_i is the first τ bits of $E_K(pad(M_i))$. We use the obtained message/MAC pairs to do an exhaustive search for the key. If τ is larger than k , it requires about 2^{k-1} block cipher E encryptions on average to recover the key. Otherwise, it requires about $2^k + 2^{k-\tau} + \dots + 2^{k-(\lceil \frac{k}{\tau} \rceil - 1)\tau} \approx 2^k$ block cipher E encryptions to recover the key since each message/MAC pair offers a τ -bit restriction. Hence, if $\tau > k$ then the complexity of the attack is $[2^{k-1}, 1, 0, 0]$, otherwise, $[2^k, \lceil \frac{k}{\tau} \rceil, 0, 0]$.

4 Forgery and Key Recovery Attacks on TMAC-V

In this section we describe forgery and key recovery attacks on a CBC-MAC variant, TMAC-V, with known or chosen message queries.

4.1 Forgery Attacks on TMAC-V

Our TMAC-V attacks also start from collecting enough message/MAC pairs to get a collision. With the known message attack the attacker obtains the corresponding MACs for approximately $2^{n/2}$ different one-block messages $X^i (1 \leq i \leq$

$2^{n/2}$) where X^i whose sizes are less than n bits are pairwise-distinct. Note that all these messages require padding. The attacker also obtains the corresponding MACs for approximately $2^{n/2}$ different one-block messages $Z^j (1 \leq j \leq 2^{n/2})$ where Z^j whose sizes are equal to n bits are pairwise distinct.

Using the birthday paradox arguments [17], with a high probability, approximately 0.63, the attacker expects to find at least one external collision between two message sets. Suppose the pair of messages which cause a collision are respectively X^* and Z^* where $|X^*| < n$ and $|Z^*| = n$. Then, by the definition of TMAC-V, the attacker has the following equation.

$$E_K((X^*||padding) \oplus E_{K'}(S_3)) = E_K(Z^* \oplus E_{K'}(S_2)). \quad (5)$$

Since $E_K(\cdot)$ is a permutation,

$$(X^*||padding) \oplus Z^* = E_{K'}(S_2) \oplus E_{K'}(S_3). \quad (6)$$

The attacker can use Eq. (6) to forge the MAC of a new message as follows: The attacker chooses a message X^S in the first set of messages which does not cause a collision. By the definition of TMAC-V, the MAC value of the X^S is $E_K((X^S||padding) \oplus E_{K'}(S_3))$. The attacker then also knows that the corresponding MAC value for a message $((X^S||padding) \oplus ((X^*||padding) \oplus Z^*))$ whose size is n bits is the same as the MAC value of X^S . Thus, the attacker can forge the MAC of a new message $((X^S||padding) \oplus ((X^*||padding) \oplus Z^*))$ using $2^{n/2+1}$ known messages. Therefore, the complexity of this attack is $[0, 2^{n/2+1}, 0, 0]$. It is easy to see that we are able to use multi-block message queries (as in the PMAC attacks) to devise a chosen message attack on TMAC-V with a $[0, 0, 2^{n/2+1}, 0]$ complexity.

4.2 Key Recovery Attacks on TMAC-V

We present here two attacks to recover the whole key of TMAC-V; one uses Eq. (5) and Eq. (6) derived from the above forgery attacks on TMAC-V and the other uses the meet-in-the-middle technique.

As stated above, we use $2^{n/2+1}$ known messages to get a collision, i.e., Eq. (5) and Eq. (6). If $n > k$, it requires about 2^k block cipher E encryptions on average to recover the key K' since Eq. (6) asks two encryptions of E for each candidate of the key K' . Once we recover the key K' , we again use Eq. (5) to recover the remaining key K . Since we know the input and output pair of E_K , it requires about 2^{k-1} block cipher E encryptions on average to recover the key K . Hence, the complexity of this attack is $[3 \cdot 2^{k-1}, 2^{n/2+1}, 0, 0]$. If $n < k$, in order to recover the key K' from Eq. (6) we should require $\lceil k/n \rceil$ collisions satisfying Eq. (6). So the data complexity of this attack is $\lceil \sqrt{\lceil k/n \rceil} \rceil \cdot 2^{n/2+1}$ known messages from which we expect more than $\lceil k/n \rceil$ collisions (the cardinality of the first message set is the same as that of the second message set, $\lceil \sqrt{\lceil k/n \rceil} \rceil \cdot 2^{n/2}$). Similarly, we apply these collisions to Eq. (6) and Eq. (5) for finding K' and K , respectively. Since it requires $2 \cdot (2^k + 2^{k-n} + \dots + 2^{k-(\lceil k/n \rceil - 1) \cdot n}) \approx 2^{k+1}$ E encryptions to

recover the key K' and $2^k + 2^{k-n} + \dots + 2^{k-(\lceil k/n \rceil - 1) \cdot n} \approx 2^k$ E encryptions to recover the key K , the complexity of this attack is $[3 \cdot 2^k, \lceil \sqrt{\lceil k/n \rceil} \rceil \cdot 2^{n/2+1}, 0, 0]$.

The meet-in-the-middle technique also allows to recover the key of TMAC-V. First, we encrypt S_2 through E using all candidates of K' and keep all the encrypted values with respect to key candidates in a table, called $\mathcal{S}$ (this step requires 2^k n -bit blocks and 2^k k -bit blocks of memory together with 2^k E encryptions). Second, we ask for MAC values of $\lceil 2k/n \rceil$ one-block messages whose bit-lengths are all less than n , i.e., with the known message attack we obtain message/MAC pairs $(M_1, T_1), (M_2, T_2), \dots, (M_{\lceil 2k/n \rceil}, T_{\lceil 2k/n \rceil})$. Third, we decrypt T_1 through E using all candidates of K and keep all the decrypted values with respect to key candidates in a table, called $\mathcal{T}$ (this step also requires 2^k n -bit blocks and 2^k k -bit blocks of memory together with 2^k E decryptions) and we update $\mathcal{S}$ by adding M_1 with each of the stored values in $\mathcal{S}$. In this step we discard keys which do not match between $\mathcal{S}$ and $\mathcal{T}$ (this step can efficiently be done by sorting the tables $\mathcal{S}$ and $\mathcal{T}$). We expect about 2^{2k-n} keys to be remained after this step. Fourth, we also decrypt T_2 through E using the candidates of K and update $\mathcal{T}$ with all the decrypted values with respect to key candidates K , and we again update $\mathcal{S}$ by adding $M_1 \oplus M_2$ with each of the stored values in $\mathcal{S}$. We then expect 2^{2k-2n} keys to be remained after this step. We repeatedly do this step for all the remaining message/MAC pairs one by one, then after all steps we recover K and K' since the expectation of the number of remaining keys is $2^{2k-\lceil 2k/n \rceil \cdot n} < 1$ (the right key is not discarded). The total time complexity is less than $(\lceil 2k/n \rceil + 1) \cdot 2^k$ E encryptions and thus the complexity of this attack is approximately $[(\lceil 2k/n \rceil + 1) \cdot 2^k, \lceil 2k/n \rceil, 0, 0]$ with 2^{k+2} storage.

5 Conclusion

PMAC is a fully parallelizable alternative to the CBC-MAC, and Michell's TMAC variant, TMAC-V, is an improvement of TMAC. In this paper we have studied the security of PMAC and TMAC-V against the forgery and the key recovery attacks. Tables 1 and 2 summarize our attacks together with previous attacks on other MAC algorithms. From the tables, we clearly know that, in terms of security, PMAC (which is similar in functionality to the OMAC) and TMAC-V do not offer significant advantages in comparison with XCBC, TMAC, OMAC and EMAC. In particular, TMAC-V does not have a good advantage over TMAC against the forgery and the full key recovery attacks although it removes a simple algebraic relationship between two keys to improve TMAC, and our attacks on PMAC do not contradict the security proof but establish the tightness of the security bound.

6 Acknowledgments

We would like to thank the anonymous referees and C. Mitchell for helpful comments about this work. This research was supported by the MIC(Ministry of

Table 1. Forgery Attack Complexities

Scheme	[a, b, c, d]	Condition	Paper
PMAC	$[0, 2^{n/2+1}, 0, 0]$	no truncation	This paper
PMAC	$[0, 2^{n/2+1}, 0, 2^{n-\tau}]$	τ -bit truncation	This paper
TMAC-V	$[0, 2^{n/2+1}, 0, 0]$	no truncation	This paper
XCBC	$[0, 2^{n/2+1}, 0, 0]$	no truncation	[15, 16]
TMAC	$[0, 2^{n/2+1}, 0, 0]$	no truncation	[15, 16]
OMAC	$[0, 2^{n/2+1}, 0, 0]$	no truncation	[15, 16]
EMAC	$[0, 2^{n/2}, 1, 0]$	no truncation	[5]

Table 2. Key Recovery Attack Complexities

Scheme	[a, b, c, d]	Condition	Paper
PMAC	$[2^{k-1}, 1, 0, 0]$	$\tau > n$	This paper
PMAC	$[2^k, \lceil \frac{k}{\tau} \rceil, 0, 0]$	$\tau \leq n$	This paper
TMAC-V	$[3 \cdot 2^{k-1}, 2^{n/2+1}, 0, 0]$	P-O attack($n > k$)	This paper
TMAC-V	$[3 \cdot 2^k, \lceil \sqrt{\lceil \frac{k}{n} \rceil} \cdot 2^{n/2+1}, 0, 0]$	P-O attack($n \leq k$)	This paper
TMAC-V	$[(\lceil \frac{2k}{n} \rceil + 1) \cdot 2^k, \lceil \frac{2k}{n} \rceil, 0, 0]$	MIMA(2^{k+2} storage)	This paper
XCBC	$[2^k, 2^{n/2+1}, 0, 0]$	P-O attack	[15, 16]
XCBC	$[2^{k+1}, \lceil \frac{k+n}{n} \rceil, 0, 0]$	MIMA(negligible storage)	[15, 16]
TMAC	$[2^k, 2^{n/2+1}, 0, 0]$	P-O attack	[15, 16]
TMAC	$[2^{k+1}, \lceil \frac{k+n}{n} \rceil, 0, 0]$	MIMA(negligible storage)	[15, 16]
OMAC	$[2^k, 2^{n/2+1}, 0, 0]$	P-O attack	[15, 16]
OMAC	$[2^{k+1}, \lceil \frac{k}{n} \rceil, 0, 0]$	MIMA(negligible storage)	[15, 16]
EMAC	$[2^{k+1}, 2^{n/2}, 0, 0]$	P-O attack	[5]
EMAC	$[s \cdot 2^k, \lceil \frac{2k}{n} \rceil, 0, 0]$	MIMA($O(2^k)$ storage)	[5]

¹ MIMA: the meet-in-middle attack² P-O attack: the attack based on the Preneel-van Oorschot attack [20]

Information and Communication), Korea, under the ITRC(Information Technology Research Center) support program supervised by the IITA(Institute of Information Technology Assessment). The second author was financed by a Ph.D. grant of the Katholieke Universiteit Leuven and by the Korea Research Foundation Grant funded by the Korean Government(MOEHRD) (KRF-2005-213-D00077) and supported by the Concerted Research Action (GOA) Ambiorics 2005/11 of the Flemish Government and by the European Commission through the IST Programme under Contract IST2002507932 ECRYPT.

References

1. K. Brincat and C. Mitchell, *New CBC-MAC Forgery Attack*, ACISP 2001, LNCS 2119, pp. 3-14, Springer-Verlag, 2001.
2. J. Black and P. Rogaway, *A Block-Cipher Mode of Operation for Parallelizable Message Authentication*, Advances in Cryptology – EUROCRYPT 2002 , LNCS 2332, pp. 384-397, Springer-Verlag, 2002.

3. M. Bellare, J. Kilian and P. Rogaway, *The Security of the Cipher Block Chaining Message Authentication Code*, Advances in Cryptology – CRYPTO 1994, LNCS 839, pp. 341-358, Springer-Verlag, 1994.
4. J. Black and P. Rogaway, *CBC-MACs for Arbitrary-Length Messages : The Three Key Construction*, Advances in Cryptology – CRYPTO 2000, LNCS 1880, pp. 197-215, Springer-Verlag, 2000.
5. ISO/IEC 9797-1, *Information technology – Security techniques – Message Authentication Codes (MACs)– Part 1 : Mechanisms using a block cipher*, International Organization for Standardization, Geneva, Switzerland, 1999.
6. T. Iwata and K. Kurosawa, *OMAC: One-Key CBC MAC*, FSE 2003, LNCS 2887, pp. 137-162, Springer-Verlag, 2003.
7. T. Iwata and K. Kurosawa, *On the Security of Two New Omac Variants*, ICISC 2003, LNCS 2971, pp. 67-78, Springer-Verlag, 2004.
8. T. Iwata and K. Kurosawa, *Strong security bounds for OMAC, TMAC, XCBC*, Indocrypt 2003, LNCS 2904, pp. 402-415, Springer-Verlag, 2003.
9. E. Jaulmes, A. Joux and F. Valette, *On the Security of Randomized CBC-MAC beyond the Birthday Limit: A New Construction*, FSE 2002, LNCS 2365, pp. 237-251, Springer-Verlag, 2002.
10. T. Kohno, *Related-Key and Key-Collision Attacks against RMAC*, Cryptology ePrint Archive, 2002. Available at <http://eprint.iacr.org>. Also available at <http://csrc.nist.gov/CryptoToolkit/modes/comments>.
11. K. Kurosawa and T. Iwata, *TMAC: Two-Key CBC-MAC*, Topics in Cryptology – CT-RSA 2003, LNCS 2612, pp. 33-49, Springer-Verlag, 2003.
12. L. Knudsen and C. Mitchell, *Analysis of 3GPP-MAC and two-key 3GPP-MAC*, Discrete Applied Mathematics, pp. 181-191, Elsevier Science, 2003.
13. L. Knudsen and C. Mitchell, *Partial key recovery attack against RMAC*, Journal of Cryptology, Vol. 18, No. 4, pp. 375-389, 2005.
14. L. Knudsen and T. Khono, *Analysis of RMAC*, FSE 2003, LNCS 2887, pp. 191-200, Springer-Verlag, 2003.
15. C. Mitchell, *On the security of XCBC, TMAC, and OMAC*, Technical Report RHCL-MA-2003-4, Aug. 2003. Available at <http://www.rhul.ac.uk/mathematics/techreports> or <http://csrc.nist.gov/CryptoToolkit/modes/comments>.
16. C. Mitchell, *Partial Key Recovery Attacks on XCBC, TMAC and OMAC*, Cryptography and Coding, 10th IMA International Conference – CCC 2005, LNCS 3796, pp. 155-167, Springer-Verlag, 2005.
17. A. Menezes, P.C. van Oorschot and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Boer Raton, 1997.
18. National Institute of Standards and Technology (NIST), Gaithersburg, MD. *NIST Special Publication 800-38B, Draft Recommendation for Block Cipher Modes of Operation: the RMAC Authentication*, November, 2002.
19. E. Petrank and C. Rackoff, *CBC MAC real-time data sources*, Journal of Cryptology, Vol. 13, No. 3, pp. 315-348, Springer-Verlag, 2000.
20. B. Preneel and P.C. van Oorschot, *On the security of iterated Message Authentication Codes*, IEEE Transactions on Information Theory, Vol. 45, No. 1, pp. 188-199, 1999.
21. J. Sung, D. Hong and S. Lee, *Key Recovery Attacks on the RMAC, TMAC, and IACBC*, ACISP 2003, LNCS 2727, pp. 265-273, Springer-Verlag, 2003.