On the security of XCBC, TMAC and OMAC

Chris J. Mitchell

Technical Report
RHUL–MA–2003–4
19 August 2003

Information Security Group
Department of Mathematics
Royal Holloway, University of London
Egham, Surrey TW20 0EX, England
http://www.rhul.ac.uk/mathematics/techreports

Abstract

The security provided by the XCBC, TMAC and OMAC schemes is analysed and compared with other MAC schemes. The results imply that there is relatively little to be gained practically through the introduction of these schemes by comparison with other well-established MAC functions. Moreover, TMAC and OMAC possess design weaknesses which enable part of the secret key to be recovered much more easily than would ideally be the case — design changes are suggested which alleviate this problem. Whether or not the proofs of security are retrievable for the modified designs remains an open question, although the need for change would appear to be clear.

1 Introduction

In this paper the security of three related methods for computing Message Authentication Codes (MACs) is analysed and compared with the level of security provided by other, more well-established, MACing techniques. The security analysis is given in terms of the most efficient (known) forgery and key recovery attacks that can be launched against the schemes.

The three MAC schemes considered here are known as XCBC [3], OMAC [8] and TMAC [7] (see also [6]). These three schemes are all examples of CBC-MACs, i.e. they are all based on the use of a block cipher in Cipher Block Chaining Mode — see, for example, [9]. Various types of CBC-MAC scheme have been in wide use for many years for protecting the integrity and guaranteeing the origin of data.

Note that all three of these schemes have been specifically designed for use with messages of variable length. Hence we compare these MAC schemes with two other schemes also designed for messages of arbitrary length, namely EMAC [2] and the ANSI retail MAC [1], otherwise known as MAC algorithms 2 and 3 (respectively) from ISO/IEC 9797-1 [5].

2 Key recovery and forgery attacks

There are two main classes of attack on a MAC scheme, namely *key recovery* attacks, in which an attacker is able to discover the secret key used to compute the MACs, and *forgery attacks* in which an attacker is able to determine the correct MAC for a message (without a legitimate key holder having generated it). Key recovery attacks are clearly more powerful than forgery attacks since once the key is known arbitrary forgeries are possible. We also consider *partial key recovery attacks* in which an attacker is able to obtain part of the secret key.

Using a simplified version of the approach used in [5], we use a three-tuple $[a, b, c]$ 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 data string/MAC pairs, and c denotes the number of chosen data string/MAC pairs.

3 XCBC and some simple attacks

The XCBC scheme was originally proposed by Black and Rogaway in 2000 [3], with the objective of providing a provably secure CBC-MAC scheme which minimises the number of block cipher encryptions and decryptions.

3.1 Definition

The XCBC scheme operates as follows. First (as throughout) suppose that the underlying block cipher transforms an n-bit block of plaintext into an n-bit block of ciphertext (i.e. it is an n-bit block cipher), and that it uses a key of k bits. If X is an n-bit block then we write $e_K(X)$ (or $d_K(X)$) for the block cipher encryption (or decryption) of the n-bit block X using key K.

The XCBC MAC scheme uses a triple of keys (K_1, K_2, K_3) where K_1 is a block cipher key, i.e. it contains k bits, and K_2, K_3 are both n-bit strings. The XCBC MAC computation is as follows.

A message D is first padded and split into a sequence of q n-bit blocks: $D_1, D_2, \ldots, D_q$. Note that there are two possibilities for the padding process. If the bit-length of the message is already an integer multiple of n then no padding is performed. However, if the bit-length of the message is not a multiple of n then the padded message consists of the message concatenated with a single one bit followed by the minimal number of zeros necessary to make the bit-length of the padded message a multiple of n. (Note that this padding strategy is not a 1-1 mapping of messages to padded messages; however, problems are avoided by the use of two different MAC computation strategies, as described immediately below).

The computation of the MAC depends on whether or not padding has been necessary. In the first case, i.e. where no padding is necessary, the MAC computation is as follows:

$$\begin{aligned} H_1 &= e_{K_1}(D_1), \\ H_i &= e_{K_1}(D_i \oplus H_{i-1}), \quad (2 \leq i \leq q-1), \text{ and} \\ \text{MAC} &= e_{K_1}(D_q \oplus H_{q-1} \oplus K_2). \end{aligned}$$

In the second case, i.e. where padding is applied, the MAC computation is as follows:

$$\begin{aligned} H_1 &= e_{K_1}(D_1), \\ H_i &= e_{K_1}(D_i \oplus H_{i-1}), \quad (2 \leq i \leq q-1), \text{ and} \\ \text{MAC} &= e_{K_1}(D_q \oplus H_{q-1} \oplus K_3). \end{aligned}$$

That is, the keys K_2 and K_3 are ex-ored with the final plaintext block depending on whether or not padding is necessary.

Note that the MAC used will be truncated to the left-most m bits of the MAC value given in the above equation, where $m \leq n$. In this paper we only consider the case where $m = n$, i.e. where no truncation is performed.

Before proceeding we observe that other authors have also considered the security of XCBC and related schemes. In particular, Furuya and Sakurai [4] have considered various attacks against 2-key variants of XCBC. However, some of the previous work (including that in [4]) has focussed on weaknesses arising from particular choices for the underlying block cipher. This contrasts with the approach followed in this paper which considers attacks independent of the block cipher.

3.2 Forgery attacks on XCBC

Suppose a fixed key triple (K_1, K_2, K_3) is in use for computing XCBC-MACs. Let $D_1, D_2, \ldots, D_q$ be any sequence of n-bit blocks, where $q \geq 0$ is arbitrary. Suppose (by some means) an attacker has the MACs for $2^{n/2}$ different messages which, after padding and splitting into a sequence of n-bit blocks, all have the form $D_1, D_2, \ldots, D_q, X$, for some n-bit block X. Because padding has been applied the MACs will all be computed using the key K_3. (Note that, since padding is applied, all of these messages must have unpadded bit-length ℓ satisfying $qn < \ell < (q+1)n$).

Suppose that the attacker also has the MACs for a further $2^{n/2}$ different messages to which padding is not applied and which, after division into a sequence of n-bit blocks, have the form $D_1, D_2, \ldots, D_q, Y$, for some n-bit block

Y. Because padding has not been applied, the MACs will all be computed using the key K_2. Note that, since no padding is applied, all these messages will have length precisely $(n+1)q$ bits.

The total number of message/MAC pairs required is clearly $2^{n/2+1}$. In the discussion below we 'cheat' slightly and refer to these as 'known MACs' rather than 'chosen MACs'. The justification for this is that if $q = 0$ then we do not impose any conditions on the messages for which MACs are required (except for their lengths). Also, there may be applications where the first part of a message is fixed, and only the last block is variable — again in such a case the required message/MAC pairs can be obtained without choosing the messages.

By the 'usual' birthday paradox probability arguments (see, for example, [9]), with a probability of approximately 0.63 one of the MACs from the first set of messages will equal one of the MACs from the second set. Suppose the pair of messages concerned are respectively $D_1, D_2, \ldots, D_q, X^*$ and $D_1, D_2, \ldots, D_q, Y^*$ for some n-bit blocks X^* and Y^*.

Before proceeding, suppose that Q is the 'simple' CBC-MAC for the q-block message $D_1, D_2, \ldots, D_q$, i.e. if

$$\begin{aligned} H_1 &= e_{K_1}(D_1), \text{ and} \\ H_i &= e_{K_1}(D_i \oplus H_{i-1}), \quad (2 \leq i \leq q) \end{aligned}$$

then $Q = H_q$.

Then, by definition, we immediately have that

$$e_{K_1}(Q \oplus X^* \oplus K_3) = e_{K_1}(Q \oplus Y^* \oplus K_2).$$

Hence, since encryption with a fixed key is a permutation of the set of all n-bit blocks, we have $Q \oplus X^* \oplus K_3 = Q \oplus Y^* \oplus K_2$, i.e. $X^* \oplus Y^* = K_2 \oplus K_3$. That is, the attacker has learnt the value of $K_2 \oplus K_3$. Knowledge of this value immediately enables forgeries to be computed.

Specifically, suppose $(D_1, D_2, \ldots, D_q)$ is the padded version of a message (of unpadded length ℓ satisfying $(q-1)n < \ell < qn$) for which the MAC M is known. Then the **unpadded** message $(D_1, D_2, \ldots, D_q \oplus K_2 \oplus K_3)$ also has MAC M. The overall complexity of this forgery attack is [0, $2^{n/2+1}$, 0].

Note that the above is, in some sense, also a partial key recovery attack, since the attacker has reduced the number of unknown key bits from $k + 2n$ to $k + n$. However, to simplify the presentation below we do not consider this further here.

3.3 Key recovery attacks on XCBC

We describe two main types of key recovery attack. The first attack (essentially based on the Preneel-van Oorschot attack [11, 12]) requires a significant number of known MACs and 'only' 2^k block cipher operations. The second attack (a 'meet-in-the-middle' attack) requires minimal numbers of known MACs, but potentially larger numbers of block cipher operations (and more storage).

The first attack is as follows. Suppose an attacker knows the MACs for $2^{n/2}$ different messages of length less than n bits. Thus, after padding and division into n-bit blocks, all these messages will consist of one block. Suppose the attacker also knows the MACs for a further $2^{n/2}$ different messages of bit-length ℓ satisfying $n < \ell < 2n$, i.e. messages which, after padding, contain two blocks.

Exactly as above, there is a good chance (probability $\simeq 0.63$) that a message from the first set will have the same MAC as a message from the second set. Suppose that the one-block and two-block messages concerned are X and (Y, Z) respectively. Since both messages involve padding, key K_3 is used in both cases. Then we know that

$$e_{K_1}(K_3 \oplus X) = e_{K_1}(K_3 \oplus Z \oplus e_{K_1}(Y)).$$

Hence, since e_{K_1} is a permutation on the set of all n-bit blocks, we have $K_3 \oplus X = K_3 \oplus Z \oplus e_{K_1}(Y)$, i.e. $X \oplus Z = e_{K_1}(Y)$. It is now possible (at least in principle) to perform an exhaustive search through all possible values for K_1, and as long as $k < n$ it is likely that only the correct value will satisfy this equation. If $k \geq n$ then a number of 'false' matches will be found — however, these can be eliminated in the next stage with minimal effort.

Given a candidate for K_1, any of the known MACs for one-block messages can be decrypted using this value of K_1 to reveal the value of K_3. This candidate key pair can then be tested on a further known MAC, and all false keys can quickly be eliminated (the complexity of this step does not affect the overall attack complexity since it is only conducted when a candidate for K_1 is found, which will only happen occasionally). The total expected complexity of this first key recovery attack is thus $[2^{k-1}, 2^{n/2+1}, 0]$, since the correct key should be found (on average) halfway through the exhaustive search of the key space.

For the second attack we present just one variant (many other meet-in-the-middle variants exist). Suppose that the attacker has access to $\lceil (k+n)/n \rceil$ known single-block message/MAC pairs all of which involve no padding (and hence K_2 is used), together with a single known message/MAC pair for which padding is applied, i.e. a total of $\lceil (k+2n)/n \rceil$ known MACs. One interesting point regarding the attack we now describe is that negligible storage is required, unlike similar attacks on EMAC (although this would no longer be true if the messages contained more than one block).

The attacker chooses one of the single-block messages for which padding is not used — suppose the one message block is D and the MAC is M. The attacker first goes through all 2^k possible values for the key K_1 and computes $d_{K_1^*}(M) \oplus D = K_2^*$ for each candidate value K_1^*. The pair $(K_1^*,\ K_2^*)$ is then tested as a candidate for $(K_1,\ K_2)$ using a second known message/MAC pair for which padding was not used. This will require a single block cipher operation, and almost all incorrect candidate key pairs will be eliminated. By means of further tests against known message/MAC pairs (from the set of $\lceil (k+n)/n \rceil$), with high probability all but the correct key pair can be eliminated. The single remaining known MAC can be used to derive K_3. The total complexity of this attack is thus $[2^{k+1},\ \lceil (k+2n)/n \rceil,\ 0]$. (Note that this attack requires only four times as many block cipher operations as the previous attack, and requires only a handful of known MACs compared to a very large number for the previous attack).

4 TMAC and its security

The TMAC scheme, a simple variant of XCBC, was proposed by Kurosawa and Iwata [8] with the goal of reducing the number of required keys from three to two.

4.1 Definition of TMAC

The TMAC scheme operates in exactly the same way as XCBC except that it only uses a key pair $(K,\ K')$ instead of a key triple, where K is a k-bit block cipher key and K' contains n bits. A key triple $(K_1,\ K_2,\ K_3)$, as used by XCBC, is then derived from $(K,\ K')$ by setting $K_1 = K$, $K_2 = u.K'$ and $K_3 = K'$, where u is a constant (defined in [8]) and multiplication by u takes place in a specific representation of the finite field of 2^n elements (also specified in [8]).

4.2 Forgery attacks on TMAC

Clearly the forgery attack on XCBC described in Section 3.2 will also apply to TMAC. There does not appear to be any obvious way in which to take advantage of the added structure in TMAC to make such an attack more efficient.

4.3 Key recovery attacks on TMAC

Again, both the key recovery attacks on XCBC described in Section 3.3 will also apply to TMAC.

There also exists a *partial* key recovery attack which will yield the key K' rather more simply than the entire key can be obtained — most importantly this attack does not require a search through the entire key space.

Suppose that the attacker performs the forgery attack described in Section 3.2. Then, the attacker will learn the value of $K_2 \oplus K_3 = S$, say. However, in the case of TMAC, we also know that $K_2 = u.K_3$, where multiplication by the public constant u is defined over the finite field of 2^n elements. Thus $K_3 = S.(u+1)^{-1}$, and hence the attacker can learn the values of both K_2 and K_3. The total complexity of this partial key recovery attack is thus the same as that of the forgery attack, i.e. [0, $2^{n/2+1}$, 0]. (Note that this yields another full key recovery attack with complexity [2^{k-1}, $2^{n/2+1}$, 0]).

Before proceeding we consider the implications of knowledge of K_2 and K_3. At first glance it is not obvious that this is any worse than knowing the value of $K_2 \oplus K_3$, which already enables simple forgeries. However, it is more serious since it enables a far wider range of forgeries to be performed. For example, suppose the (unpadded) message $D_1, D_2, \ldots, D_q$ has MAC M, i.e. $M = e_{K_1}(K_2 \oplus D_q \oplus H_{q-1})$, where H_{q-1} is defined as above. Then, if message $E_1, E_2, \ldots, E_r$ has MAC N, it is not hard to see that the message

$$D_1, D_2, \ldots, D_{q-1}, D_q \oplus K_2, E_1 \oplus M, E_2, E_3, \ldots, E_r$$

also has MAC N.

Finally note that this partial key recovery attack against TMAC has previously been described by Sung, Hong and Lee [13].

4.4 Improving TMAC

The main reason that TMAC is significantly weaker than XCBC is the fact that a simple algebraic relationship exists between K_2 and K_3. This not only enables K_2 to be trivially deduced from K_3 (and vice versa), it also enables a second linear equation in K_2 and K_3 to be used to deduce both K_2 and K_3.

However, there is no reason for such a simple relationship to exist between K_2 and K_3. One way of avoiding this would be to cryptographically derive both K_2 and K_3 from the single key K'. One way in which this could be done would be to define two different fixed n-bit strings, S_2 and S_3 say, and to put $K_2 = e_{K'}(S_2)$ and $K_3 = e_{K'}(S_3)$. With this definition, knowledge of one of K_2 (or K_3) will not enable K_3 (or K_2) to be deduced, as long as the block cipher e resists known ciphertext attacks. Also, knowledge of $K_2 \oplus K_3$ will also not enable K_2 and K_3 to be deduced (again assuming that e resists known ciphertext attacks). This change would, however, mean that K' contains k rather than n bits.

Of course, this change invalidates the security proof for TMAC. It is not immediately clear how easy it will be to repair the proof for a revised TMAC of this general type.

5 The security of OMAC

The OMAC scheme, a further simple variant of XCBC, was proposed by Iwata and Kurosawa [7] with the goal of further reducing the number of required keys from three to one.

5.1 Definition of OMAC

The OMAC scheme operates in exactly the same way as XCBC except that it only uses a single key K instead of a key triple, where K is a k-bit block cipher key. A key triple (K_1, K_2, K_3), as used by XCBC, is then derived from (K, K') by setting $L = e_K(0^n)$, $K_1 = K$, $K_2 = u.L$ and $K_3 = u^2.L$, where 0^n is the n-bit block of all zeros, and u is a constant (defined in [7]) and multiplication by u and u^2 takes place in a specific representation of the finite field of 2^n elements (also specified in [7]).

Note that there are, in fact, two different variants of OMAC, known as OMAC1 and OMAC2. The version defined above is OMAC1, and is the one analysed here. However the analysis is almost identical for OMAC2, which is identical to OMAC1 except that $K_3 = u^{-1}.L$.

5.2 Forgery attacks on OMAC

Just as for TMAC, the forgery attack on XCBC described in Section 3.2 will also apply to OMAC. There does not appear to be any obvious way in which to take advantage of the added structure in OMAC to make such an attack more efficient.

5.3 Key recovery attacks on OMAC

Both the key recovery attacks on XCBC described in Section 3.3 will also apply to OMAC, as will a simple variant of the partial key recovery attack described in Section 4.3. In this case however, a second partial key recovery attack exists, which we now describe. This attack is designed to enable L to be determined, knowledge of which immediately enables both K_2 and K_3 to be determined. The attack is similar to that described in Section 3.3.

Suppose an attacker knows the MACs for $2^{n/2}$ different messages of length less than n bits. Thus, after padding and division into n-bit blocks, all these messages will consist of one block. Suppose the attacker also knows the MACs for a further $2^{n/2}$ different messages of bit-length ℓ satisfying $n < \ell < 2n$, i.e. messages which, after padding, contain two blocks, and for which the first n bits are all zero.

Exactly as above, there is a good chance (probability $\simeq 0.63$) that a message from the first set will have the same MAC as a message from the second set. Suppose that the one-block and two-block messages concerned are X and (0^n, Z) respectively (recall that the second message must begin with n zeros). Since both messages involve padding, key K_3 is used in both cases. Then we know that

$$e_{K_1}(K_3 \oplus X) = e_{K_1}(K_3 \oplus Z \oplus e_{K_1}(0^n)).$$

Hence, since e_{K_1} is a permutation on the set of all n-bit blocks, we have $K_3 \oplus X = K_3 \oplus Z \oplus e_{K_1}(Y)$, i.e. $X \oplus Z = e_{K_1}(0^n)$. But $K_1 = K$, and thus we know that $L = X \oplus Z$. Thus L, and hence K_2 and K_3, are immediately available to the attacker. This latter attack has complexity [0, $2^{n/2}$, $2^{n/2}$].

5.4 Improving OMAC

Analogously to the proposed improvements to TMAC (given in Section 4.4), it is suggested that K_2 and K_3 are derived from K using the following process: put $K_2 = e_K(S_2)$ and $K_3 = e_K(S_3)$, where S_2 and S_3 are fixed and distinct n-bit strings. This will avoid the attack described in Section 4.3.

In order to avoid the OMAC-specific attack described in Section 5.3, it is suggested that the key K_1 used in MAC computations is not the same as the key used to derive K_2 and K_3, to prevent MAC computations accidentally revealing K_2 and/or K_3. This is simple to achieve by setting $K_1 = K \oplus S_1$ for a fixed k-bit string S_1.

Again, whether or not a security proof can be devised for this modified OMAC is an interesting open question.

6 Benchmark results and comparisons

We next consider the security provided by two well-known and standardised CBC-MAC schemes, namely EMAC and the ANSI retail MAC. Note that, unlike XCBC, TMAC and OMAC, these schemes operate independently of whether or not a message is padded and how padding is performed.

6.1 EMAC

EMAC is standardised as MAC algorithm 2 in ISO/IEC 9797-1 [5], and has been proven secure by Petrank and Rackoff [10]. EMAC uses a pair of keys (K_1, K_2) where K_1 and K_2 are both block cipher keys, i.e. they contain k bits. A message D is first padded and split into a sequence of q n-bit blocks: $D_1, D_2, \ldots, D_q$.

The EMAC computation, which essentially involves double encrypting the final block, is as follows:

$$\begin{aligned} H_1 &= e_{K_1}(D_1), \\ H_i &= e_{K_1}(D_i \oplus H_{i-1}), \quad (2 \leq i \leq q-1), \text{ and} \\ \text{MAC} &= e_{K_2}(e_{K_1}(D_q \oplus H_{q-1})). \end{aligned}$$

As summarised in [5], the most effective (known) forgery attack against EMAC has complexity [0, $2^{n/2}$, 1] and the best key recovery attacks have complexity either [2^{k+1}, $2^{n/2}$, 0] or [$s.2^k$, $\lceil 2k/n \rceil$, 0] (for some small value of

s), where the second attack requires $O(2^k)$ storage. Note that the second attack is a meet-in-the-middle attack.

6.2 ANSI retail MAC

The ANSI retail MAC (abbreviated as ARMAC below) is standardised as MAC algorithm 3 in ISO/IEC 9797-1 [5]. Note that this scheme is widely used with the block cipher DES (which has $n = 64$ and $k = 56$) in environments where obtaining $2^{n/2} = 2^{32}$ message/MAC pairs is deemed infeasible. However, it seems that a security proof for this scheme does not exist. Nevertheless, since it closely resembles EMAC, heuristically one might expect a similar level of provable security.

This MAC scheme again uses a pair of keys $(K_1,\ K_2)$ where K_1 and K_2 are both block cipher keys, i.e. they contain k bits. A message D is first padded and split into a sequence of q n-bit blocks: $D_1, D_2, \ldots, D_q$. The MAC computation is as follows:

$$\begin{aligned} H_1 &= e_{K_1}(D_1), \\ H_i &= e_{K_1}(D_i \oplus H_{i-1}), \quad (2 \leq i \leq q-1), \text{ and} \\ \text{MAC} &= e_{K_1}(d_{K_2}(e_{K_1}(D_q \oplus H_{q-1}))). \end{aligned}$$

As summarised in [5], the best known forgery attack against the ANSI retail MAC has complexity [0, $2^{n/2}$, 1] and the best-known key recovery attack has complexity [2^{k+1}, $2^{n/2}$, 0] — note that one attraction of ARMAC is that it does not appear to be subject to meet-in-the-middle attacks.

6.3 Comparisons

We compare the three ‘new’ MAC algorithms, i.e. XCBC, OMAC and TMAC, with the two longer-established schemes with respect to two different criteria: efficiency and security.

Efficiency can be further sub-divided into two different categories: key length, and the number of block cipher operations required to compute the MAC for a message. The key lengths for the five MAC schemes considered here are given in Table 1.

The number of block cipher operations (encryptions or decryptions) required to compute the MAC for a message is specified in Table 2. Note that it is assumed that EMAC and ARMAC are used with the “always add a ‘1’ and

Table 1: Key lengths

XCBC	TMAC	OMAC	EMAC	ARMAC
$k+2n$	$k+n$	k	$2k$	$2k$

then as many zeros as necessary" padding method, which is standardised as padding method 2 in ISO/IEC 9797-1 [5].

Table 2: Computational complexity (block cipher operations)

No. of data bits (ℓ)	XCBC	TMAC	OMAC	EMAC	ARMAC
$(t-1)n < \ell < tn$	t	t	t	$t+1$	$t+2$
$\ell = tn$	t	t	t	$t+2$	$t+3$

From Table 2 it should be clear that XCBC, TMAC and OMAC all offer workload advantages over EMAC and ARMAC. This workload advantage is slightly increased by the fact that XCBC, TMAC and OMAC only require one block cipher key 'set up' per MAC computation, whereas EMAC and OMAC require two — the difference this makes depends on the block cipher in use. However, these advantages are probably insignificant for messages that are more than a few blocks long, and even for short messages they are unlikely to be a major issue; banking networks have been using ARMAC with a relatively slow block cipher such as DES for many years for very large numbers of messages, using relatively primitive hardware.

We sub-divide the security comparison into three sub-categories, covering forgery attacks, key recovery attacks and partial key recovery attacks. The complexities of forgery attacks against the five MAC schemes considered here are specified in Table 3.

Table 3: Forgery attack complexities

XCBC	TMAC	OMAC	EMAC	ARMAC
[0,$2^{n/2+1}$,0]	[0,$2^{n/2+1}$,0]	[0,$2^{n/2+1}$,0]	[0,$2^{n/2}$,1]	[0,$2^{n/2}$,1]

The complexities of key recovery attacks are specified in Table 4. Note that this table does not take account of the fact that the complexities of the second attacks for XCBC, TMAC and OMAC require no significant storage, whereas the second attack against EMAC requires around $O(2^k)$ storage.

Table 4: Key recovery attack complexities

XCBC	TMAC	OMAC	EMAC	ARMAC
[2^{k-1},$2^{n/2+1}$,0]	[2^{k-1},$2^{n/2+1}$,0]	[2^{k-1},$2^{n/2+1}$,0]	[2^{k+1},$2^{n/2}$,0]	[2^{k+1},$2^{n/2}$,0]
[2^{k+1},$\lceil(k+2n)/n\rceil$,0]	[2^{k+1},$\lceil(k+n)/n\rceil$,0]	[2^{k+1},$\lceil k/n\rceil$,0]	[$s.2^k$,$\lceil 2k/n\rceil$,0]	

Finally, the complexities of partial key recovery attacks (where they exist) are specified in Table 5.

Table 5: Partial key recovery attack complexities

XCBC	TMAC	OMAC	EMAC	ARMAC
—	[0,$2^{n/2+1}$,0]	[0,$2^{n/2+1}$,0] [0,2^n,2^n]	—	—

7 Conclusions

It should be clear from the analysis above that, in terms of security, XCBC, TMAC and OMAC offer no significant advantage by comparison with EMAC and ARMAC. Moreover, in some cases, they would appear to be weaker, although the most significant weaknesses of TMAC and OMAC can be avoided by changing the key derivation procedure. Unfortunately, these changes would appear to invalidate the proofs of security for these schemes, and it would be interesting to see whether proofs for these revised schemes can be devised.

Nevertheless, XCBC, TMAC and OMAC do offer a small practical advantage in terms of a modest reduction in the number of block cipher operations, although this is unlikely to be significant in most applications. If a proof of security for the revised OMAC can be devised, it would be true that this 'new OMAC' would be at least as attractive as any of the currently standardised schemes.

In summary, there does not appear to be a compelling case for standardising these new CBC-MAC schemes, and certainly OMAC and TMAC should not be adopted in their current form.

Acknowledgements

The author would like to gratefully acknowledge a number of helpful comments and suggestions from Kenny Paterson and Matt Robshaw, which have considerably improved this paper.

References

[1] American Bankers Association, Washington, DC, *ANSI X9.19, financial institution retail message authentication*, August 1986.

[2] A. Berendschot, B. den Boer, J.-P. Boly, A. Bosselaers, J. Brandt, D. Chaum, I. Damgard, M. Dichtl, W. Fumy, M. van der Ham, C. J. A. Jansen, P. Landrock, B. Preneel, G. Roelofsen, P. de Rooij, and J. Vandewalle, *Integrity primitives for secure information systems*, Lecture Notes in Computer Science, vol. 1007, Springer-Verlag, Berlin, 1995.

[3] J. Black and P. Rogaway, *CBC-MACs for arbitrary length messages: The three-key constructions*, Advances in Cryptology — Crypto 2000 (M. Bellare, ed.), Lecture Notes in Computer Science, vol. 1880, Springer-Verlag, Berlin, 2000, pp. 197–215.

[4] S. Furuya and K. Sakurai, *Risks with raw-key masking — The security evaluation of 2-key XCBC*, Information and Communications Security, 4th International Conference, ICICS 2002 (R. H. Deng, S. Qing, F. Bao, and J. Zhou, eds.), Lecture Notes in Computer Science, vol. 2513, Springer-Verlag, Berlin, 2002, pp. 327–341.

[5] International Organization for Standardization, Genève, Switzerland, *ISO/IEC 9797–1, Information technology — Security techniques — Message Authentication Codes (MACs) — Part 1: Mechanisms using a block cipher*, 1999.

[6] T. Iwata and K. Kurosawa, *Stronger security bounds for OMAC, TMAC and XCBC*, 2003, Department of Computer and Information Sciences, Ibaraki University, Japan.

[7] ———, *OMAC: One-key CBC MAC*, Proceedings of FSE 2003, Lecture Notes in Computer Science, Springer-Verlag, Berlin, to appear.

[8] K. Kurosawa and T. Iwata, *TMAC: Two-key CBC MAC*, Topics in Cryptology — CT-RSA 2003 (M. Joye, ed.), Lecture Notes in Computer Science, vol. 2612, Springer-Verlag, Berlin, 2003, pp. 33–49.

[9] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of applied cryptography*, CRC Press, Boca Raton, 1997.

[10] E. Petrank and C. Rackoff, *CBC MAC for real-time data sources*, Journal of Cryptology **13** (2000), 315–338.

[11] B. Preneel and P.C. van Oorschot, *A key recovery attack on the ANSI X9.19 retail MAC*, Electronics Letters **32** (1996), 1568–1569.

[12] ———, *On the security of iterated Message Authentication Codes*, IEEE Transactions on Information Theory **45** (1999), 188–199.

[13] J. Sung, D. Hong, and S. Lee, *Key recovery attacks on the RMAC, TMAC, and IACBC*, ACISP 2003 (R. Safavi-Naini and J. Seberry, eds.), Lecture Notes in Computer Science, vol. 2727, Springer-Verlag, Berlin, 2003, pp. 265–273.