New CBC-MAC forgery attacks

Karl Brincat*
Visa International EU
PO Box 253
London W8 5TE, UK
brincatk@visa.com

Chris J. Mitchell
Information Security Group,
Royal Holloway, University of London,
Egham, Surrey TW20 0EX, UK
c.mitchell@rhbnc.ac.uk

15th October 2000 (m5)

Abstract

This paper is concerned with a particular type of attack against CBC-MACs, namely *forgery attacks*, i.e. attacks which enable an unauthorised party to obtain a MAC on a data string. Existing forgery attacks against CBC-MACs are briefly reviewed, together with the effectiveness of various countermeasures. This motivates the main part of the paper, where a family of new forgery attacks are described, which raise serious questions about the effectiveness of certain countermeasures.

Keywords: message authentication code, MAC, forgery, block cipher

1 Introduction

1.1 Use of MACs

MACs, i.e. *Message Authentication Codes*, are a widely used method for protecting the integrity and guaranteeing the origin of transmitted messages and stored files. To use a MAC it is necessary for the sender and recipient of a message (or the creator and verifier of a stored file) to share a secret key K, chosen from some (large) keyspace. The data string to be protected, D say, is input to a MAC function f, along with the secret key K, and the output is the MAC. We write MAC $= f_K(D)$. The MAC is then sent or stored with the message.

*The views expressed in this paper are personal to the author and not necessarily those of Visa International

1.2 A model for CBC-MACs

MACs are most commonly computed using a block cipher in a scheme known as a CBC-MAC (for *Cipher Block Chaining* MAC). This name derives from the CBC 'mode of operation' for block ciphers, and a CBC-MAC is computed using the same basic process. There are several variants of the CBC-MAC, although the following general model (see [4, 9]) covers most of these.

The computation of a CBC-MAC on a bit string D using a block cipher with block length n, uses the following six steps.

1. *Padding.* The data string D is subjected to a padding process, involving the addition of bits to D, the output of which (the *padded string*) is a bit string of length an integer multiple of n (say qn).

2. *Splitting.* The padded string is divided (or 'split') into a series of n-bit blocks, $D_1, D_2, \ldots, D_q$.

3. *Initial transformation.* Initial transformation I, which may be key-controlled, is applied to D_1 to give the first *chaining variable* H_1, i.e.

$$H_1 = I(D_1).$$

4. *Iteration.* Successive chaining variables are computed as

$$H_i = e_K(D_i \oplus H_{i-1})$$

for $i := 2, 3, \ldots, q$, where, as throughout, K is a block cipher key, $e_K(X)$ and $d_K(X)$ denote block cipher encryption and decryption of block X with key K, and $\oplus$ denotes bit-wise exclusive-or of blocks.

5. *Output transformation.* The n-bit *Output block* G is computed as

$$G = g(H_q)$$

where g is the output transformation (which may be key-controlled).

6. *Truncation.* The MAC is set equal to the leftmost m bits of G.

Most CBC-MACs adhere to this model, and such MACs will be the main focus of this paper.

1.3 Types of CBC-MAC scheme

The latest version of the relevant international standard, namely ISO/IEC 9797-1, [4], contains six different CBC-MAC variants. These are based on combinations of two Initial transformations and three Output transformations.

- Initial transformation 1 is defined as:

$$I(D_1) = e_K(D_1)$$

  where K is the same key as used in the Iteration step. I.e. Initial transformation 1 is the same as the Iteration step, and is the one used in both the original CBC-MAC, as defined in ANSI X9.9, [3], and CBC-MAC-Y (also known as the *ANSI Retail MAC*), standardised in ANSI X9.19, [2].

- Initial transformation 2 is defined as:

$$I(D_1) = e_{K''}(e_K(D_1))$$

  where K is the same key as used in the Iteration step, and K'' is a block cipher key distinct from K.

- Output transformation 1 is defined as:

$$g(H_q) = H_q,$$

  i.e. Output transformation 1 is the identity transformation, and is the one used in the original CBC-MAC, [3].

- Output transformation 2 is defined as:

$$g(H_q) = e_{K'}(H_q),$$

  where K' is a block cipher key distinct from K.

- Output transformation 3 is defined as:

$$g(H_q) = e_K(d_{K'}(H_q)),$$

  where K' is a block cipher key distinct from K. Output transformation 3 is the one used in CBC-MAC-Y, [2].

These options are combined in the ways described in Table 1 to yield four of the six different CBC-MAC schemes defined in ISO/IEC 9797-1, [4]. Note that algorithms 5 and 6 do not fit the general MAC model given above; as a result we do not consider these last two algorithms further in this paper.

Finally note that three Padding Methods are also defined in [4]. Padding Method 1 simply involves adding between 0 and $n-1$ zeros, as necessary, to the end of the data string. Padding Method 2 involves the addition of a single 1 bit at the end of the data string followed by between 0 and $n-1$ zeros. Padding Method 3 involves prefixing the data string with an n-bit block encoding the bit length of the data string, with the end of the data string padded as in Padding Method 1.

Table 1: CBC-MAC schemes defined in ISO/IEC 9797-1

Algorithm number	Input transformation	Output transformation	Notes
1	1	1	The 'original' CBC-MAC scheme.
2	1	2	K' may be derived from K.
3	1	3	CBC-MAC-Y. The values of K and K' shall be chosen independently.
4	2	2	K'' shall be derived from K' in such a way that $K' \neq K''$.

When using one of the six MAC algorithms it is necessary to choose one of the three padding methods, and the degree of truncation to be employed. All three Padding Methods can be deployed with all six MAC algorithms.

In the remainder of this paper the discussions primarily apply to MAC algorithms 1–4 from ISO/IEC 9797-1, used with Padding Methods 1–3. We also use the terminology of ISO/IEC 9797-1. In fact, these algorithms cover almost all CBC-MAC variants in common use today.

2 Attacks on CBC-MACs

There are two main types of attack on MAC schemes.

- In a *MAC forgery* attack [6], an unauthorised party is able to obtain a valid MAC on a message which has not been produced by the holders of the secret key. Typically the attacker will need a number of valid MACs and corresponding messages to use to obtain the forgery.
- A *key recovery* attack enables the attacker to obtain the secret key used to generate one or more MACs. Note that a successful key recovery attack enables the construction of arbitrary numbers of forgeries.

We introduce a simple way of quantifying the effectiveness of an attack. Following the approach used in [4], we do this by means of a four-tuple which specifies the size of the resources needed by the attacker. For each attack we specify the tuple $[a, b, c, d]$ where a denotes the number of off-line block cipher encipherments (or decipherments), b denotes the number of known data string/MAC pairs, c denotes the number of chosen data string/MAC pairs, and d denotes the number of on-line MAC verifications. The reason for distinguishing between the numbers c and d is that, in some environments, it may be easier for the attacker to obtain MAC verifications (i.e. to submit a data string/MAC pair and receive an answer indicating

whether or not the MAC is valid) than to obtain the genuine MAC value for a chosen message.

3 Simple MAC forgeries

We start by considering three 'simple' types of MAC forgery. All these forgery attacks apply regardless of the MAC algorithm in use.

- *MAC guessing.* The attacker selects a message and simply guesses the correct MAC value. The probability that the guess will be correct is 2^{-m}. Such attacks can be avoided by making m sufficiently large.
- *Verification forgery.* This is a simple development of the 'MAC guessing' technique. The attacker chooses a message, and then works through all possible MACs, submitting the chosen message combined with each MAC value for verification. This attack has complexity $[0, 0, 0, 2^m]$. Thus, even if an attacker only has access to a MAC verification function, selective verifiable forgeries are possible unless m is sufficiently large.
- *Trailing zeros forgery.* The third attack only applies when Padding Method 1 from [4] is in use. The attack works because of the observation that, if a padded message has final block D_q and the last '1' bit appears at position i (out of n) in D_q, then there are $n + 1 - i$ (unpadded) messages which, when padded, give the padded message. This means that, unless a message contains a multiple of n bits and ends in a '1' bit, given any message and MAC it is possible to discover other messages with the same MAC by deleting zeros from, or adding zeros to, the end of the message.

  The same general type of attack would apply to any scheme using a padding method where the mapping from messages to padded messages is not injective. Fortunately, Padding Methods 2 and 3 do not suffer from this problem — indeed, the main motivation for the design of Padding Method 2 was to avoid this problem.

4 More sophisticated forgeries

We now consider further attacks which apply only to particular variants of the CBC-MAC.

4.1 Simple cut and paste attack

Suppose that the MAC function in use is the 'original' MAC scheme, i.e. ISO/IEC 9797-1 MAC algorithm 1. Then, given two messages with valid MACs (computed using the same secret key K), we can compute a third 'composite' message with a valid MAC without knowing the key. For further details see, for example, [4]

4.2 Birthday attack

For this attack we suppose that Padding Method 3 is not being used. We also suppose that no truncation is employed, i.e. so that $m = n$. Suppose, by some means, an attacker discovers two messages with the same MAC. That is, suppose the attacker has found that the two messages with padded data strings $D_1, D_2, \ldots, D_q$ and $E_1, E_2, \ldots, E_r$ have the same MAC. Then it follows immediately that any pair of padded messages of the form $D_1, D_2, \ldots, D_q, X_1, X_2, \ldots, X_t$ and $E_1, E_2, \ldots, E_r, X_1, X_2, \ldots, X_t$ will also have the same MAC, regardless of the choice of $X_1, X_2, \ldots, X_t$.

By elementary probability theory relating to the so called 'Birthday Paradox' (see, for example, [9]), given a set of $2^{n/2}$ messages there is a good chance that two of them will have the same MAC. Thus, to find such a collision requires only approximately $2^{n/2}$ known message/MAC pairs. Armed with such a pair, the attacker now needs only persuade the user to generate a MAC on one more message to obtain a MAC forgery. Thus the total complexity of this attack is $[0,2^{n/2},1,0]$.

Finally note that this attack applies to all of ISO/IEC MAC algorithms 1–4, as long as Padding Method 3 is not used. Unfortunately, Padding Method 3 does not prevent another, slightly more sophisticated, forgery attack, described immediately below.

4.3 van Oorschot-Preneel attack

This attack is based on an observation of Preneel and van Oorschot, also independently made by Kaliski and Robshaw, which is summarised as Lemma 1 in [9]. The attack relies on finding an 'internal collision' for a pair of padded messages. That is, suppose $D_1, D_2, \ldots, D_q$ and $E_1, E_2, \ldots, E_r$ are two sequences of n-bit blocks obtained as a result of applying the padding and splitting processes to a pair of messages D and E. Suppose also that the chaining variables for the MAC computations for D and E are H_i, $(1 \leq i \leq q)$, and J_i, $(1 \leq i \leq r)$, respectively. Then an internal collision is where:

$$H_s = J_t$$

for some pair (s, t), where $s \leq q$ and $t \leq r$.

Given knowledge of an internal collision (by some means), the attacker immediately knows that the padded messages

$$D_1, D_2, \ldots, D_s, F_1, F_2, \ldots, F_u$$

and

$$E_1, E_2, \ldots, E_t, F_1, F_2, \ldots, F_u$$

will have the same MAC, regardless of $F_1, F_2, \ldots, F_u$. That is, given a known internal collision, a forgery requires only one chosen MAC. Note that, if Padding method 3 is used, then the above attack will work if and only if the values s, t and u satisfy $u = q - s = r - t$.

The problem remains of finding the internal collision. If Padding Method 3 is not in use then the attack works if we set $s = q$ and $t = r$ and look for MAC collisions amongst a large set of messages, i.e. the attack is the same as the Birthday Attack (see Section 4.2). However, when Padding Method 3 is in use, finding a 'useful' internal collision, i.e. one for which $s < q$, is a little more difficult albeit not impossible, as we now describe.

Suppose, that the attacker obtains the MACs for a set of $2^{n/2}$ messages, all of which agree in their final u n-bit blocks for some $u > 0$. As before, suppose also that $m = n$. Then, there is a good chance that two of these messages will have the same MAC. Since these two messages have their final u blocks the same, then we know that there will be an internal collision.

Specifically, suppose we know that the MACs for the sequences of blocks $D_1, D_2, \ldots, D_q$ and $E_1, E_2, \ldots, E_r$ are the same, and suppose we also have $D_i = E_{i+r-q}$ for $q - u + 1 \leq i \leq q$. If H_i and J_i denote chaining variables (as above), then we must have $H_s = J_t$ where $s = q - u$ and $t = r - u$.

If we regard the set of $2^{n/2}$ messages as chosen texts, then the attack has complexity [0,0,$2^{n/2}$,0], which, although large, is still more effective than the 'simple' MAC forgeries. However, it may be easier than this to obtain the desired MACs, bearing in mind that many messages are highly formatted. Thus it may be true 'by accident' that large numbers of messages for which a MAC is computed all end in the same way. If this is the case then the attack complexity might more reasonably be described as [0,$2^{n/2}$,1,0], i.e. the same as the Birthday forgery attack.

In any event it should be clear that Padding Method 3 does not protect against forgery attacks using internal or external collisions. This point is also made in Section III.B of [9]. Thus, to prevent such attacks, further countermeasures are needed. This is the subject of the remainder of this paper.

5 Countermeasures

Over the past few years a number of countermeasures to various forgery attacks have been proposed. Of course, there are certain forgery attacks which cannot be avoided, and serve as a baseline against which other attacks can be measured. We now review some of the proposed countermeasures.

- *Truncation.* Perhaps the most obvious countermeasure to the attacks described in Sections 4.2 and 4.3, is to choose the MAC length m such that $m < n$, i.e. to truncate the MAC. However, Knudsen, [5], has shown that, even when truncation is employed, the same attacks can still be made at the cost of a modest amount of additional effort. Moreover, if m is made smaller, then the MAC guessing and Verification forgery attacks (described in Section 3) become easier to mount.

- *Padding Methods 2 and 3.* Padding Method 2 was introduced specifically to deal with the Trailing zeros forgery (see Section 3). Padding Method 3 was introduced to counter certain key recovery attacks, and was also originally believed to counter Birthday forgeries (for further details see [9]). However, as we have seen, neither Padding Method is able, on its own, to prevent the attack described in Section 4.3.

- *Serial numbers.* A further countermeasure is briefly described in [4]. The idea is to prepend a unique serial number to data prior to computing a MAC. That is, every time a MAC is generated, the data to be MACed is prepended with a number which has never previously been used for this purpose (within the lifetime of the key).

  Although it is not stated explicitly in [4], it would seem that it is intended that the serial number should be prepended to the message prior to padding. Note also that it will be necessary to send the serial number with the message, so that the intended recipient can use it to help recompute the MAC (as is necessary to verify it).

  It is fairly simple to see why this approach foils the attacks of Sections 4.2 and 4.3. Both attacks require the forger to obtain the MAC for a chosen data string. However, because of the insertion of a serial number, the attacker is now no longer in a position to choose the data string. Thus it was believed that this countermeasure was effective against the non-trivial forgery attacks.

  However, as we will show below, serial numbers do not protect against 'shortcut' forgery attacks, even when combined with Padding Method 3. It is believed that this is the first time a forgery attack more efficient than the verification attack has been demonstrated against the serial number enhancement to CBC-MACs.

6 A new forgery attack

We now describe a new type of forgery attack. To simplify the presentation we start by describing the attack as applied to MAC algorithms 1, 2 or 3 with Padding Method 1 or 2 and no Serial Number prefix. Later we consider the scenario where Serial Numbers are used and lastly we consider the implications of this attack in the case where both Padding Method 3 and Serial Numbers are used.

6.1 The basic attack

We first consider the case where one of MAC algorithms 1, 2 or 3 is used together with Padding Method 1 or 2 and where the data is not prefixed with a Serial Number. As previously, we consider the case where there is no truncation and the size of the chaining variable is equal to the size of the final output, this common value being denoted by n. Assume that the attacker somehow obtains the corresponding MACs for approximately $2^{n/2}$ (padded) $(r+q+1)$-block messages $E'_1, E'_2, \ldots, E'_q, X, F_1, F_2, \ldots, F_r$ where $E'_1, E'_2, \ldots, E'_q$ are arbitrary n-bit blocks, $F_1, F_2, \ldots, F_r$ are arbitrary but fixed n-bit blocks, and X is an n-bit block that is different for each message. The attacker also obtains the corresponding MACs for approximately $2^{n/2}$ padded $(r+1)$-block messages of the form $Y, F_1, F_2, \ldots, F_r$, with the same fixed blocks F_i, $1 \leq i \leq r$, and a different n-bit block Y for each message.

Using an extension to the Birthday Paradox, [7, 8], given the number of MACs obtained there is a high probability that a MAC from the set of $(r+q+1)$-block messages is equal to a MAC from the set of $(r+1)$-block messages. In other words, $MAC(E_1, E_2, \ldots, E_q, X_0, F_1, F_2, \ldots, F_r) = MAC(Y_0, F_1, F_2, \ldots, F_r)$ for some particular known values of $E_1, \ldots, E_q, X_0$ and Y_0. Since the n-bit blocks $F_1, \ldots, F_r$ are the same for the two messages, it is the case that $MAC^*(E_1, E_2, \ldots, E_q, X_0) = MAC^*(Y_0)$, where $MAC^*(Z)$ denotes the computation of the MAC on the message Z without the Output Transformation. This final relation is equivalent to

$$MAC^*(E_1, E_2, \ldots, E_q) = X_0 \oplus Y_0.$$

As a result of this, if the attacker knows that the MAC for some (padded) message $Z, P_1, P_2, \ldots, P_t$ $(t \geq 1)$ is equal to M, then the attacker knows that the MAC for the message $E_1, E_2, \ldots, E_q, X_0 \oplus Y_0 \oplus Z, P_1, P_2, \ldots, P_t$ is also equal to M. This means that the complexity of this MAC forgery attack on a MAC algorithm with an n-bit output with no truncation is approximately $[0, 1, 2^{n/2+1}, 0]$. In the case of *DES* with no truncation, this is a forgery attack of complexity $[0, 1, 2^{33}, 0]$.

6.2 A forgery attack for the Serial Number case

Now consider the case where Serial Numbers are used with MAC algorithm 1, 2 or 3 and Padding Method 1 or 2. Note that serial numbers are meant to be prefixed to the messages to be MACed, that is, if $P_1, P_2, \ldots, P_t$ is a padded t-block message, then the MAC is calculated on the $(t+1)$-block message $S, P_1, P_2, \ldots, P_t$ where S is the serial number associated with the message. With this observation we point out that the above attack described for MAC algorithms not using Serial Numbers works *unchanged* for MAC algorithms using Serial Numbers. Only the interpretation of the first block of the various chosen texts used in the attack is different.

Note that for the $(r+q+1)$-block messages $E'_1, E'_2, \ldots, E'_q, X, F_1, F_2, \ldots, F_r$ described above, the E'_i's, $1 \leq i \leq q$ were arbitrary and not necessarily fixed. In the case of use of serial numbers, an attacker could submit $(r+q)$-block messages $E'_1, E'_2, \ldots, E'_{q-1}, X, F_1, F_2, \ldots, F_r$ to be MACed. The MAC algorithm returns the MAC for the string $S'_1, E'_1, E'_2, \ldots, E'_{q-1}, X, F_1, F_2, \ldots, F_r$ where S'_1 is the (unique) serial number selected by the MAC algorithm for the particular message. For verification of the MAC, the value of S'_1 has to be transmitted with the MAC of the message — if S'_1 is encrypted then this attack will not work. Hence the attacker is assumed to know the value of S'_1 for each of the $2^{n/2}$ $(r+q)$-block messages $E'_1, E'_2, \ldots, E'_{q-1}, X, F_1, F_2, \ldots, F_r$. Similarly, the attacker can submit the r-block message $F_1, F_2, \ldots, F_r$ $2^{n/2}$ times, each time obtaining a (different) MAC for the string $S'_2, F_1, F_2, \ldots, F_r$, for a known but different serial number S'_2.

As above, there is a non-trivial probability that an $(r+q)$-block message of the first type and one of the r-block submissions of the second type yield the same MAC, that is,

$$MAC(S_1, E_1, E_2, \ldots, E_{q-1}, X_0, F_1, F_2, \ldots, F_r) = MAC(S_2, F_1, F_2, \ldots, F_r),$$

for some known particular values of $S_1, S_2, E_1, \ldots, E_{q-1}$ and X_0. This means that $MAC^*(S_1, E_1, E_2, \ldots, E_{q-1}, X_0) = MAC^*(S_2)$ and therefore

$$MAC^*(S_1, E_1, E_2, \ldots, E_{q-1}) = X_0 \oplus S_2.$$

If the attacker knows that the MAC for a padded message $P_1, \ldots, P_t$ $(t \geq 1)$ using serial number S_3 is equal to M, he also knows that the MAC for the padded message $E_1, E_2, \ldots, E_{q-1}, X_0 \oplus S_2 \oplus S_3, P_1, P_2, \ldots, P_t$ using serial number S_1 is also equal to M. The complexity of this MAC forgery attack is the same as before, i.e. $[0, 1, 2^{n/2+1}, 0]$. The constructed block $X_0 \oplus S_2 \oplus S_3$ is the reason why the attack does not work if the serial numbers are not in the clear, since in this case the attacker does not know S_2 and S_3.

6.3 Combining Serial Numbers with Padding Method 3

The attack can be generalised to cover the case where Padding Method 3 and Serial Numbers are used in combination; there are two ways to combine these two features, and we describe attacks for both combinations.

Firstly, suppose they are combined as implied in [4], i.e. the serial number is prefixed before the message is padded, i.e. the length of the unpadded message is prefixed to the padded and serial numbered message. The attacker submits the r-block message $F_1, F_2, \ldots, F_r$ $2^{n/2}$ times, each time obtaining a (different) MAC for the string $L_2, S_2', F_1, F_2, \ldots, F_r$, for a varying serial number S_2'. Note that L_2 is the 'length-encoding block' for the message (it will be the same every time), as inserted by Padding Method 3.

The attacker also submits $2^{n/2}$ messages $E_1', E_2', \ldots, E_{q-2}', L_2, X, F_1, F_2, \ldots, F_r$ to be MACed, where L_2 is as above and X is different for each message. MACs are computed for strings $L_1, S_1', E_1', E_2', \ldots, E_{q-2}', L_2, X, F_1, F_2, \ldots, F_r$ where S_1' is the varying serial number, and L_1 is the length encoding block. As before we suppose that the attacker knows the values of S_1' and S_2' for each of the messages.

There is a good chance that an r-block message of the first type and an $r+q$-block message of the second type yield the same MAC, that is,

$$MAC(L_1, S_1, E_1, E_2, \ldots, E_{q-2}, L_2, X_0, F_1, F_2, \ldots, F_r) = MAC(L_2, S_2, F_1, F_2, \ldots, F_r),$$

for some particular values of $S_1, S_2, E_1, E_2, \ldots, E_{q-2}$ and X_0. This means that $MAC^*(L_1, S_1, E_1, E_2, \ldots, E_{q-2}, L_2, X_0) = MAC^*(L_2, S_2)$ and therefore

$$MAC^*(L_1, S_1, E_1, E_2, \ldots, E_{q-2}, L_2) = e_K(L_2) \oplus X_0 \oplus S_2.$$

Thus if an attacker knows the MAC for padded message $(L_2, S_3, P_1, P_2, \ldots, P_r)$ is equal to M, (where S_3 is any serial number), he knows that the MAC for the padded message $L_1, S_1, E_1, E_2, \ldots, E_{q-2}, L_2, X_0 \oplus S_2 \oplus S_3, P_1, P_2, \ldots, P_r$ is also equal to M. The complexity of this MAC forgery attack is the same as before, i.e. $[0, 1, 2^{n/2+1}, 0]$.

Secondly we consider the alternative way of combining serial numbers with Padding Method 3, i.e. where we first pad the message, then prefix the length of the unpadded message, and finally prefix the resulting string with the selected serial number. That is, for a (padded) message $P_1, P_2, \ldots, P_t$, the MAC algorithm is applied to the string $S, L, P_1, P_2, \ldots, P_t$, where S is the serial number block and L is the length block of the unpadded message. We describe yet another attack variant for this case.

Briefly, the attacker submits $2^{n/2}$ $(r+q)$-block padded messages of the form $E_1, E_2, \ldots, E_{q-2}, X, L_2, F_1, F_2, \ldots, F_r$ where $E_1, E_2, \ldots, E_{q-2}$ are arbitrary n-bit blocks, $F_1, F_2, \ldots, F_r$ are arbitrary but fixed n-bit blocks, L_2 is an

n-bit block representing the length of the unpadded string $F_1, F_2, \ldots, F_r$ (as required by Padding Method 3), and X is an n-bit block that is different for each message. The attacker obtains the corresponding MACs and the particular serial number S_1' used with each message. The attacker also submits the r-block padded string $F_1, F_2, \ldots, F_r$ $2^{n/2}$ times for MACing, obtaining the corresponding MACs and the different serial number S_2' used for each MAC obtained. There is a non-trivial probability that a MAC for one of the $(r+q)$-block messages is equal to a MAC for the r-block message for some serial numbers S_1 and S_2, that is,

$$MAC(S_1, L_1, E_1, \ldots, E_{q-2}, X_0, L_2, F_1, \ldots, F_r) = MAC(S_2, L_2, F_1, \ldots, F_r).$$

This means that $MAC^*(S_1, L_1, E_1, E_2, \ldots, E_{q-2}) = X_0 \oplus S_2$.

Suppose also that the attacker knows that the MAC for an r-block message $P_1, P_2, \ldots, P_r$, with unpadded length equal to L_2 and serial number S_3, is equal to M. Then he knows that the MAC for the $(r+q)$-block message $E_1, E_2, \ldots, E_{q-2}, X_0 \oplus S_2 \oplus S_3, L_2, P_1, P_2, \ldots, P_r$ of unpadded length L_1 and with serial number S_1 is also equal to M, or

$$MAC(S_1, L_1, E_1, E_2, \ldots, E_{q-2}, X_0 \oplus S_2 \oplus S_3, L_2, P_1, P_2, \ldots, P_r) = MAC(S_3, L_2, P_1, P_2, \ldots, P_r).$$

Note that the complexity of the attack is as before, i.e. it is $[0, 1, 2^{n/2+1}, 0]$.

6.4 Implications

The published version of ISO/IEC 9797-1, [4], indicates that forgery attacks can be avoided by using a combination of Padding Method 3 and Serial Numbers. However, the attacks described in Section 6.3 cast serious doubt on the value of serial numbers as a remedy to forgery attacks even when combined with Padding Method 3.

7 Summary and conclusions

In this paper we have surveyed some forgery attacks to which MAC algorithms may be subjected, including new attacks which can defeat some proposed countermeasures not successfully attacked before. In particular we have shown that combining Padding Method 3 and Serial Numbers is not as effective as was previously believed in defeating 'shortcut' forgery attacks.

Of course, in practice, other security features may prevent some or all of the described attacks from being a real threat. For example, in certain banking environments the security deployed in the access to a MAC algorithm is such that it is extremely difficult for an unauthorised user to obtain the MAC

corresponding to even one chosen text, let alone several. Also, if the MAC scheme is used in such a way that no key is used to compute more than a small number of MACs then certain attacks become impossible.

The use of Padding Method 1 is not automatically excluded because of the attack described in section 3. It is possible that in certain environments messages are highly formatted to the extent that the length of a message to be MACed is fixed or known from the context, and therefore a trailing zeroes forgery is not applicable.

In general it is important for users to carefully assess the significance of the various MAC attacks in the context of the environment in which the resulting MAC algorithm is to be used. There may be no benefit from using certain sophisticated MAC systems in an environment which has other security features in operation which make attacks against simpler MAC schemes impossible to carry out. On the other hand, care should be taken not to assume that a MAC which is secure in a certain environment is automatically secure in others. For example, a 32-bit MAC which may be safely used in a banking environment without any serious threat from a verification forgery, is possibly not safe if used on the Internet or any other environment where large numbers of verifications may be obtained in a short time.

The introduction and use of the AES algorithm [1], with a minimum 128-bit cipher block length, as the encryption function to be used in block-cipher based MACs means that all the attacks described here would become practically infeasible. However, this may not be the case if heavy truncation is used since, in this case, some of the attacks described in section 3 may still be possible.

In this paper we concentrated on block-cipher based MAC algorithms as described in [4]. It is possible that generalisations of the attacks described here may be also applicable to other (dedicated, hash function-based or proprietary) MAC algorithms which are based on iterated functions. Note that all practical MAC algorithms are iterated in construction.

Acknowledgements

The authors would like to thank anonymous referees who suggested the first attack variant described in Section 6.3.

References

[1] AES, a crypto algorithm for the twenty-first century, Advanced Encryption Standard (AES) development effort, 2000.

http://csrc.nist.gov/encryption/aes.

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

[3] American Bankers Association, Washington, DC. *ANSI X9.9–1986 (revised), Financial institution message authentication (wholesale)*, April 1986.

[4] 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.

[5] L.R. Knudsen. Chosen-text attack on CBC-MAC. *Electronics Letters*, **33**:48–49, 1997.

[6] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, Boca Raton, 1997.

[7] K. Nishimura and M. Sibuya. Occupancy with two types of balls. *Ann. Inst. Statist. Math.*, **40**:77–91, 1988.

[8] K. Nishimura and M. Sibuya. Probability to meet in the middle. *J. Cryptology*, **2**:13–22, 1990.

[9] B. Preneel and P.C. van Oorschot. On the security of iterated Message Authentication Codes. *IEEE Transactions on Information Theory*, **45**:188–199, 1999.