Key recovery and forgery attacks on the MacDES MAC algorithm

Don Coppersmith[1], Lars R. Knudsen[2], and Chris J. Mitchell[3]

[1] IBM Research, T.J. Watson Research Center, Yorktown Heights, NY 10598, USA
`copper@watson.ibm.com`

[2] Department of Informatics, University of Bergen, N-5020, Bergen, Norway
`lars.knudsen@ii.uib.no`

[3] Information Security Group, Royal Holloway, University of London, Egham, Surrey TW20 0EX, UK
`c.mitchell@rhbnc.ac.uk`

Abstract. We describe a series of new attacks on a CBC-MAC algorithm due to Knudsen and Preneel including two key recovery attacks and a forgery attack. Unlike previous attacks, these techniques will work when the MAC calculation involves prefixing the data to be MACed with a 'length block'. These attack methods provide new (tighter) upper bounds on the level of security offered by the MacDES technique.

Key words. Message Authentication Codes. Cryptanalysis. CBC-MAC.

1 Introduction

CBC-MACs, i.e. Message Authentication Codes (MACs) based on a block cipher in Cipher Block Chaining (CBC) mode, have been in wide use for many years for protecting the integrity and origin of messages. A variety of minor modifications to the 'basic' CBC-MAC have been devised and adopted over the years, in response to various cryptanalytic attacks (for a survey see [7]). The latest version of the international standard for CBC-MACs, ISO/IEC 9797–1, [4], which was recently published, contains a total of six different MAC algorithms.

This paper is concerned with one of these algorithms, namely MAC Algorithm 4. This algorithm has only recently been added to the draft international standard, and was intended to offer a higher degree of security than previous schemes at a comparable computational cost. It was originally proposed by Knudsen and Preneel, [6] and, when used with the DES block cipher, was given the name 'MacDES'.

Some key recovery attacks against this scheme have recently been described, [3], but these do not work when 'Padding method 3' is used, which involves prefixing the data to be MACed with a length block. The key recovery and forgery attacks described below are designed specifically to work in this case.

2 Preliminaries

MAC algorithm 4 uses three block cipher keys, K, K' and K'', where either K'' is derived from K', or K' and K'' are both derived from a single key. However, for the attacks here we make no assumptions about how K' and K'' are related. We assume that the block cipher uses k-bit keys. We denote the block cipher encryption operation by $Y = e_K(X)$, where Y is the n-bit ciphertext block corresponding to the n-bit plaintext block X, and K is the k-bit key. We denote the corresponding decryption operation by $X = d_K(Y)$.

The MAC is computed on a data string by first padding the data string so that it contains an integer multiple of n bits, and then breaking it into a series of n-bit blocks. If the n-bit blocks derived from the padded data string are $D_1, D_2, \ldots, D_q$, then the MAC computation is as follows.

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

for some $H_1, H_2, \ldots, H_{q-1}$. Finally, M is truncated as necessary to form the m-bit MAC.

ISO/IEC FDIS 9797–1 provides three different padding methods. 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.

When using one of the six MAC algorithms from ISO/IEC FDIS 9797–1, it is necessary to choose one of the three padding methods, and the degree of truncation to be employed. We consider the case where Padding Method 3 is used, and where there is no truncation (as already noted, the case where either of the other two Padding Methods is used is dealt with in [3]). Hence, given that the block cipher in use has an n-bit block length, the MAC has $m = n$ bits. E.g., in the case of DES we have $m = n = 64$ and $k = 56$.

Finally note that we assume that all MACs are computed and verified using one particular triple of keys (K, K', K''). The two key recovery attacks described below are designed to discover these keys. The forgery attack also described in this paper enables the observer of messages and corresponding MACs to obtain a new (valid) message/MAC pair which had not been observed. Of course, in general, a successful key recovery attack enables arbitrary numbers of forgeries to be constructed in a trivial way.

3 A key recovery attack

We start by describing a chosen plaintext attack for key recovery which is more efficient than any previously known attacks on this MAC scheme. In a later section we describe a still more efficient key recovery attack.

3.1 Outline of attack

The attack operates in two stages. In stage 1 we find a pair of n-bit block-pairs: (D_1, D_2) and (D_1', D_2'), which should be thought of as the first pair of blocks of longer padded messages, with the property that the 'partial MACs' for these two pairs are equal, i.e. so that if

$$\begin{aligned} H_1 &= e_{K''}(e_K(D_1)), \\ H_2 &= e_K(D_2 \oplus H_1), \\ H_1' &= e_{K''}(e_K(D_1')), \quad \text{and} \\ H_2' &= e_K(D_2' \oplus H_1'), \end{aligned}$$

then $H_2 = H_2'$. This is what is usually referred to as an 'internal collision', although we call this particular case a 'hidden internal collision' since it will not be evident from a complete MAC calculation.

Given that D_1 and D_1' are to be thought of as the first blocks of padded messages (and given we are assuming Padding Method 3 is in use), D_1 and D_1' will be encodings of the length of the unpadded messages. The attack relies on D_1' being an encoding of a message for which the padded version has $q+1$ blocks ($q \geq 4$) and D_1 being an encoding of a message for which the padded version has q blocks.

In stage 2 we will use this pair of block-pairs to launch an attack which is essentially the same as Attack 1 of [3].

3.2 Stage 1 — Finding the hidden internal collision

We first choose D_1 and D_1'. Typically one might choose D_1 to be an encoding of $3n$ and D_1' to be an encoding of $4n$, which will mean that D_1 will be the first block of a 4-block padded message and D_1' will be the first block of a 5-block padded message. For the purposes of the discussion here we suppose that D_1 encodes a bit-length resulting in a q-block padded message ($q \geq 4$).

The attacker chooses (arbitrary) values for n-bit blocks labeled D_2 and $D_5, D_6, \ldots, D_q$. The attacker then generates $2^{n/2}$ messages which, when padded using Padding Method 3, will have the form

$$D_1, D_2, X, Y, D_5, D_6, \ldots, D_q$$

where X and Y are arbitrary n-bit blocks, and, by some means, obtains the MACs for these messages. By routine probabilistic arguments (called the 'birthday attack', see [7]), there is a good chance that two of the messages will have the same MAC.

Suppose the two pairs of blocks (X, Y) involved are (X_1, Y_1) and (X_2, Y_2). Then if

$$\begin{aligned} H_1 &= e_{K''}(e_K(D_1)), \\ H_2 &= e_K(D_2 \oplus H_1), \\ H_3 &= e_K(X_1 \oplus H_2), \\ H_4 &= e_K(Y_1 \oplus H_3), \\ H_3' &= e_K(X_2 \oplus H_2), \quad \text{and} \\ H_4' &= e_K(Y_2 \oplus H_3'), \end{aligned}$$

we know that $H_4 = H_4'$. We call the two pairs (X_1, Y_1) and (X_2, Y_2) a 'diagnostic pair'. These can be used to find the desired hidden internal collision.

The attacker now constructs 2^n message pairs. Each pair will, when padded using Padding Method 3, have the form:

$$D_1', W, X_1, Y_1, D_5', D_6', \ldots, D_{q+1}'$$

and

$$D_1', W, X_2, Y_2, D_5', D_6', \ldots, D_{q+1}'$$

where W varies over all possible n-bit blocks, (X_1, Y_1) and (X_2, Y_2) are as above, and $D_5', D_6', \ldots, D_{q+1}'$ are arbitrary (if desired, they could be different for each message pair). The attacker now, by some means, discovers whether or not the two messages within each pair have the same MAC; this will typically require 2^n chosen MACs, together with 2^n MAC verifications.

First consider the case where the 'partial MAC' from the block-pair (D_1', W) is the same as the partial MAC from the block pair (D_1, D_2). That is, if H_2 is as above, and if

$$\begin{aligned} H_1' &= e_{K''}(e_K(D_1')), \quad \text{and} \\ H_2' &= e_K(W \oplus H_1'), \end{aligned}$$

then suppose that $H_2 = H_2'$. Note that there will always exist a value W giving this property, since as $W \oplus H_1'$ ranges over all possible n-bit blocks, then so will H_2'. But, given the discussion above regarding the diagnostic pairs (X_1, Y_1) and (X_2, Y_2), this immediately means that the pair of messages involving this value of W will yield the same MACs as each other.

Second consider the pairs of messages for all the other $2^n - 1$ values of W. In these cases we know that $H_2 \neq H_2'$. There will be at least one 'false positive', namely when $H_2' = H_2 \oplus X_1 \oplus X_2$. For the remaining cases, assuming that the block cipher behaves in a random way, the probability of the MACs from the message pair being identical is 2^{-n}.

Hence, as the search proceeds through the entire set of message pairs, we would expect approximately three 'positives' — one corresponding to the case we desire, i.e. where $H_2 = H_2'$, one where $H_2' = H_2 \oplus X_1 \oplus X_2$, and one random 'false positive'. If a second diagnostic pair is available then this can be used to immediately rule out any false positives. If not then we can proceed to stage 2 with all the 'positives' from stage 1, and all but the genuine positive will yield inconsistent results.

Note that the cost of this part of the attack is $2^{n/2}$ 'chosen MACs' (to find the diagnostic pair), and 2^n chosen MAC calculations and 2^n MAC verifications to find the hidden internal collision.

3.3 Stage 2 — Recovering the key

From the previous stage we have a pair of n-bit block-pairs: (D_1, D_2) and (D'_1, D'_2), with the property that the 'partial MACs' for these two pairs are equal, i.e. so that if

$$\begin{aligned} H_1 &= e_{K''}(e_K(D_1)),\\ H_2 &= e_K(D_2 \oplus H_1),\\ H'_1 &= e_{K''}(e_K(D'_1)), \quad \text{and}\\ H'_2 &= e_K(D'_2 \oplus H'_1), \end{aligned}$$

then $H_2 = H'_2$. In addition D'_1 encodes the length of a message, which, when padded, contains $(q+1)$ blocks, and D_1 encodes the length of a message, which, when padded, contains q blocks.

The attacker now, by some means, obtains the MAC for a set of $2^{n/2}$ padded messages of the form

$$D'_1, D'_2, E_3, E_4, \ldots, E_{q+1}$$

where $E_3, E_4, \ldots, E_{q+1}$ are arbitrary. By the usual 'birthday attack' arguments, there is a good chance that two of the messages will have the same MAC. Suppose the two padded strings $D'_1, D'_2, E_3, E_4, \ldots, E_{q+1}$ and $D'_1, D'_2, E'_3, E'_4, \ldots, E'_{q+1}$ yield the same MAC, and suppose that the common MAC is M. Note that the cost of this part of the attack is $2^{n/2}$ 'chosen MACs'. (Note also that we need to assume that $E_{q+1} \neq E'_{q+1}$).

Next submit two chosen padded strings for MACing, namely

$$D_1, D_2, E_3, E_4, \ldots, E_q$$

and

$$D_1, D_2, E'_3, E'_4, \ldots, E'_q$$

namely the strings one obtains by deleting the last block from each of the above two messages and replacing the first two blocks by D_1 and D_2 (these remain 'valid' padded messages because D_1 encodes the length of a message which, when padded, contain q blocks). If we suppose that the MACs obtained are M' and M'' respectively, then we know immediately that

$$d_{K'}(M') \oplus E_{q+1} = d_K(d_{K'}(M)) = d_{K'}(M'') \oplus E'_{q+1}$$

since (D_1, D_2) and (D'_1, D'_2) yield the same 'partial MAC'.

Now run through all possibilities L for the unknown key K', and set $x(L) = d_L(M')$ and $y(L) = d_L(M'')$. For the correct guess $L = K'$ we will have $x(L) = d_{K'}(M')$ and $y(L) = d_{K'}(M'')$, and hence $E_{q+1} \oplus x(L) = E'_{q+1} \oplus y(L)$. This will hold for $L = K'$ and probably not for any other value of L, given that $k < n$ (if $k \geq n$ then either a second 'collision' or a larger brute force search will probably be required).

Having recovered K', we do an exhaustive search for K using the relation $d_{K'}(M') \oplus E_{q+1} = d_K(d_{K'}(M))$ (which requires 2^k block cipher encryptions). Finally we can recover K'' by exhaustive search on any known text/MAC pair, e.g. from the set of $2^{n/2}$, which again will require 2^k block cipher encryptions.

3.4 Complexity of the attack

We start by introducing a simple way of quantifying the effectiveness of an attack. Following the approach used in [4], we use a four-tuple $[a, b, c, d]$ to specify the size of the resources needed by the attacker, 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.

Using this notation, the complexity of Stage 1 of the attack is $[0, 0, 2^n, 2^n]$ and the complexity of stage 2 is $[2^{k+2}, 0, 2^{n/2}, 0]$ (note that we ignore lower order terms). Hence the overall attack complexity is $[2^{k+2}, 0, 2^n, 2^n]$, i.e. in the case of DES the attack complexity is $[2^{58}, 0, 2^{64}, 2^{64}]$.

This is sufficiently high to rule out most practically conceivable attacks. However it is substantially less than the complexity of the best previously known attack as given in [6], which was $[2^{89}, 0, 2^{65}, 2^{55}]$.

4 A more efficient key recovery attack

We now present a second key recovery attack which is considerably more efficient than the attack just described. The attack is in two stages. The second stage is identical to the second stage of the first attack; the improvement is in the first stage. We find a pair of n-bit blocks (D_1, D'_1) with the following properties.

- D'_1 encodes the length of a message which, when padded, contains $q + 1$ blocks.
- D_1 encodes the length of a message which, when padded, contains q blocks.

– $e_{K''}(e_K(D_1)) \oplus e_{K''}(e_K(D_1')) = V$, for some *known* n-bit block V.

Then choosing $D_2' = D_2 \oplus V$ yields a pair of n-bit block-pairs (D_1, D_2) and (D_1', D_2') for which the partial MACs are equal.

For fixed values $D_1, D_4, D_5, \ldots, D_q$, a total of $2^{n/2}$ different values of X, and a set of t different values of Y, by some means obtain the MACs of the padded messages $(D_1, X, Y, D_4, D_5, \ldots, D_q)$. Choose the $2^{n/2}$ values of X to cover all the n-bit blocks with most significant $n/2$ bits set to zero (for simplicity we assume n is even). If t is sufficiently large, then for most values of X there will exist at least one tuple (Y, X', Y') such that

$$\mathrm{MAC}(D_1, X, Y, D_4, D_5, \ldots, D_q) = \mathrm{MAC}(D_1, X', Y', D_4, D_5, \ldots, D_q).$$

In such a case we know that

$$e_K(X \oplus e_{K''}(e_K(D_1))) \oplus e_K(X' \oplus e_{K''}(e_K(D_1))) = Y \oplus Y'. \tag{1}$$

If the Y values are fixed for every value of X, each such match results in an additional match, since a match for messages with blocks X, Y and X', Y' also gives a match for messages with blocks X, Y' and X', Y. To avoid this (it doesn't help us) for each value of X we choose the Y values randomly depending on X.

Now consider the graph G whose vertices are the messages X, and with an edge between X and X' when a relationship of the type (1) is known. This graph has $2^{n/2}$ vertices and a number of edges dependent on t. With $t = 2^{n/4}t'$ we have a total of $T = 2^{3n/4}t'$ messages and about $T^2/2 = 2^{3n/2}t'^2/2$ pairs of messages. Assuming that the underlying block cipher behaves randomly this results in about $2^{n/2}t'^2/2$ pairs of messages with colliding MAC values. Thus the graph G will have $2^{n/2}$ vertices and approximately $2^{n/2}t'^2/2$ edges. If we view the graph as a random graph [9], then a "giant component" will arise when the number of edges is sufficiently larger than half the number of vertices. With $t' = 2$ it can be shown that with high probability there is a component containing 98% of all vertices.[1]

We know the value

$$e_K(X \oplus e_{K''}(e_K(D_1))) \oplus e_K(X' \oplus e_{K''}(e_K(D_1)))$$

whenever X and X' are in the same connected component, by adding the appropriate equations together. So for most values of X we know the value

$$f(X) = e_K(X \oplus e_{K''}(e_K(D_1))) \oplus e_K((X \oplus 1) \oplus e_{K''}(e_K(D_1)))$$

[1] With s vertices and $cs/2$ randomly placed edges, with $c > 1$, there is a single "giant component" whose size is almost exactly $(1 - t(c))s$, where [1]

$$t(c) = (1/c) \sum_{k=1}^{\infty} (k^{k-1}(ce^{-c})^k)/k!.$$

For $c = 4$, $1 - t(c) = 0.98$.

where 1 denotes the n-bit block with least significant bit 1 and all other bits set to zero.

Now repeat the above process but for a set of padded messages with $q+1$ rather than q blocks. In this case label the fixed values $D_1', D_4', D_5', \ldots, D_{q+1}'$, and obtain the MACs for messages

$$D_1', Z, Y, D_4', D_5', \ldots, D_{q+1}'$$

for $2^{n/2+1}$ values of Z and t values of Y, where the values of Z cover all n-bit blocks whose least significant $n/2$ bits are forced to 0 (except the single least significant bit which covers both values). As previously, if t is sufficiently large then for many (most) values of Z we will have an equation of the form

$$g(Z) = e_K(Z \oplus e_{K''}(e_K(D_1'))) \oplus e_K((Z \oplus 1) \oplus e_{K''}(e_K(D_1'))).$$

Now find values X, Z such that $f(X) = g(Z)$. Then we know with a high probability that either

$$X \oplus e_{K''}(e_K(D_1)) = Z \oplus e_{K''}(e_K(D_1')),$$

or

$$X \oplus e_{K''}(e_K(D_1)) = Z \oplus 1 \oplus e_{K''}(e_K(D_1')).$$

This (almost) gives us the desired relationship between D_1 and D_1'. In fact the next stage of the attack can be carried out for both possible relationships. This will not significantly affect the overall attack complexity, since the complexity of the second stage is much less than that of the first stage.

Complexity of attack

It should be clear that finding the relationship between the desired pair of blocks (D_1, D_1') requires $3 \times 2^{n/2} \times t$ 'chosen MACs', where $t = 2^{n/4}t'$. Hence the complexity of the first stage of the attack is $[0, 0, 3t' \times 2^{3n/4}, 0]$ for small $t' \geq 1$. The complexity of the second stage of the attack is $[2^{k+2}, 2^{n/2}, 0, 0]$. Thus, assuming that the second stage of the attack is performed twice, we get an overall attack complexity of $[2^{k+3}, 2^{n/2+1}, 3t' \times 2^{3n/4}, 0]$. In the case of DES this gives an attack complexity of $[2^{59}, 2^{33}, s \times 2^{48}, 0]$ for a small $s \geq 3$.

5 A MAC forgery attack

We next consider a forgery attack against the same MAC scheme, and which uses a similar method of attack. What is particularly significant about this attack is that, analogously to the attack in [6], it is based almost exclusively on 'MAC verifications' rather than 'chosen MACs'. As mentioned above, in certain circumstances it may be substantially 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. The attack also requires almost no memory.

5.1 Details of attack

By some means suppose the attacker obtains the MAC, M, for the padded message

$$D_1, D_2, D_3, ..., D_q$$

(where D_1 encodes the length of a $(q-1)n$-bit message). Suppose next that the attacker submits the 2^n messages

$$D_1', W, D_2, D_3, ..., D_q$$

for MAC verification with candidate MAC M, where W takes on all possible values, and where D_1' encodes the length of a qn-bit message. Precisely one of the 2^n messages will have valid MAC M.

Armed with the correct W it is now possible to forge the MAC of any padded message of q blocks by requesting the MAC of a padded message of $q+1$ blocks or vice versa. This is because we know that

$$\mathrm{MAC}(D_1', W, E_2, ..., E_q) = \mathrm{MAC}(D_1, E_2, ..., E_q)$$

for any blocks $E_2, E_3, \ldots, E_q$.

There are variants of this attack which allow the block W to be inserted between any pair of blocks. Also the attack is only dependent on the block length, and will also work against Triple DES CBC-MAC and other iterated MAC schemes of similar structure.

5.2 Complexity

It is straightforward to see that the complexity of the above attack is simply $[0, 0, 1, 2^n]$. In addition, once the 2^n verifications have been performed, each additional MAC forgery only requires one 'chosen MAC'.

This compares with the best previously known forgery attack for this MAC scheme, namely the Preneel-van Oorschot attack, [8], which has complexity $[0, 0, 2^{n/2}, 0]$.

6 Preventing the attacks

Before considering countermeasures against these attacks it is first important to note that if k (the bit length of the key) and n (the cipher block length) are chosen to be sufficiently large, then all these attacks become infeasible. In particular, with the lengths envisaged for the emerging Advanced Encryption Standard (AES) these attacks are of academic interest only. However, for the time being many systems are reliant on ciphers such as DES, for which k and n are both uncomfortably small. Thus, finding countermeasures which do not involve too many additional encryption operations remains of practical importance.

6.1 Using Serial Numbers

Probably the simplest way of preventing all the attacks described above is to use *Serial Numbers*, as described in [4]. The basic idea is to prepend a unique serial number to a data string prior to computing a MAC. That is, every time a MAC is generated by a device, that device ensures that the data to be MACed is prepended with a number which has never previously been used for this purpose (at least 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, and this is the assumption we make here. 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 described in this paper. All attacks require the forger to obtain the MAC for a *chosen* data string. However, the attacker is now no longer in a position to choose the data string, since the MAC generator will insert a serial number (previously unused) as part of the MAC computation process. Note that an attacker can still verify MACs on particular messages using serial numbers of his own choice.

Note that the effectiveness of Serial numbers against forgery attacks on the MAC scheme considered here is discussed in more detail in [2].

6.2 A further MacDES variant

Another possible way to defeat the attacks described previously is to modify the MAC scheme to introduce an extra key. The key recovery attacks exploit the fact that one key of the CBC-chaining equals the first of the two keys in the final double encryption. We can therefore preclude such attacks by using the key K only in the middle step of the MAC calculation and not in the first and final steps. I.e. we can introduce a fourth key, K''', which could be derived from K (with K'' derived from K'), and put:

$$\begin{aligned} H_1 &= e_{K''}(e_{K'''}(D_1)), \\ H_i &= e_K(D_i \oplus H_{i-1}), (2 \leq i \leq q-1), \text{ and} \\ M &= e_{K'}(e_{K'''}(D_q \oplus H_{q-1})). \end{aligned}$$

However, this modified scheme can still be attacked with about 2^{65} chosen plaintexts and 2^{64} work. Note that although such an attack is not practical, it is much faster than an exhaustive search for the key. We now sketch the attack.

Select 2^{60} values of X_i that are 0 in the most significant 4 bits (say), and 2^5 arbitrary values of Y_j. Fix words D_1 and D_4 corresponding to a padded message of eventual length 4×64 bits. (Thus D_1 encodes a message length of 3×64 bits.) By some means obtain MACs for the 2^{65} messages

$$(D_1, X_i, Y_j, D_4).$$

We are guaranteed to have at least 2^{64} coincidences of the form

$$\mathrm{MAC}(D_1, X_i, Y_j, D_4) = \mathrm{MAC}(D_1, X_k, Y_m, D_4).$$

Recalling that H_1 is constant throughout this exercise (but unknown), each coincidence gives the knowledge that

$$e_K(X_i \oplus H_1) \oplus Y_j = e_K(X_k \oplus H_1) \oplus Y_m = d_K(H_3),$$

whence

$$e_K(X_i \oplus H_1) \oplus e_K(X_k \oplus H_1) = Y_j \oplus Y_m.$$

Construct a graph of 2^{60} vertices, each vertex representing an allowable value of X_i. An edge joins two vertices when we have knowledge of the type given in the last equation. That is, vertices X_i and X_k are joined when we know the value of

$$e_K(X_i \oplus H_1) \oplus e_K(X_k \oplus H_1).$$

Because we have 2^{64} edges and 2^{60} vertices, routine arguments about random graphs (see, for example [9]) predicts that we will have one 'giant connected component' in this graph, which contains most of the vertices. If two vertices X_i, X_p lie in the same connected component, we can (by chasing edges and adding the corresponding equations) find the corresponding sum

$$e_K(X_i \oplus H_1) \oplus e_K(X_p \oplus H_1).$$

In particular for most 'acceptable' values of X_i we know the sum

$$f(X_i) = e_K(X_i \oplus H_1) \oplus e_K((X_i \oplus 1) \oplus H_1).$$

However, we still do not know K or H_1.

Now for each guess k for K, and each of 2^4 choices of z, compute $e_k(z) \oplus e_k(z \oplus 1)$ and see whether this equals $f(X_i)$ for some value of X_i. If so, compute also $e_k(z \oplus 2) \oplus e_k(z \oplus 3)$ and check whether that matches $f(X_i \oplus 2)$ for the same X_i. If so, then K and H_1 can be determined. This last step takes time $2^{56} \times 2^4 \times 4 = 2^{62}$.

Having recovered K and H_1, it is then necessary to break double-DES to recover K' and K''' (complexity approximately [2^{56+t},0,0,0] with 2^{56-2t} space [10]). This also involves doing the above attack twice, since two pairs of input and output of double-DES are need to determine the secret keys. Finally, break single DES to find K''.

6.3 Truncation

Perhaps the most obvious countermeasure to the attacks described here is to choose the MAC length m such that $m < n$, i.e. to truncate the Output Block of the MAC calculation. However, Knudsen [5] has shown that, even when truncation is employed, in some cases the same attacks can still be made at the cost of a modest amount of additional effort. Moreover, if m is made smaller, then certain trivial forgery attacks become easier to mount (see, for example, [2]).

We now consider how truncation affects the key recovery and forgery attacks described above.

Key recovery attacks The key recovery attacks no longer work as described. However, as we now sketch, a 2^{64} chosen text attack can still work, even with truncated (32-bit) MACs and even with the four-key variant (as described in Section 6.2).

In fact for most of the attack it is possible to make a trade-off between chosen texts and computation. Also, it is possible to recover all keys, or to build a 'dictionary'. The attack complexity is approximately 2^{64+3-p} chosen texts and 2^{56+2p} computation, for $0 < p < 32$, and with full key-recovery we need an additional amount of 2^{56+t} computation using 2^{56-2t} space and about 2^{57} MAC verifications.

Pick a block of $x = 2^{64-2p}$ words X, with the low order $2p$ bits set to 0. For each X, pick $y = 2^p$ words Y. Optimally these depend on X in a random fashion (to avoid some duplications). Pick four messages Z, fixed throughout. Fix the initial block D_1. Obtain the MACs of the $2^{64-2p} \times 2^p \times 4$ blocks (D_1, X, Y, Z). Let $h(X, Y)$ be the concatenation of the four MACs of

$$(D_1, X, Y, Z_1), (D_1, X, Y, Z_2), (D_1, X, Y, Z_3), (D_1, X, Y, Z_4).$$

The coincidence $h(X, Y) = h(X', Y')$ is (essentially) equivalent to $H_3(D_1, X, Y) = H_3(D_1, X', Y')$ (the equality after three rounds).

Now use the same random graph idea as previously, with about x edges among the x vertices. Throw in a slop factor so that there are $2x$ edges. Then nearly everything is in the giant component. Evaluate, for each X, the function

$$g(X) = H_2(D_1, X) \oplus H_2(D_1, X \oplus 1) = e_K(H_1 \oplus X) \oplus e_K(H_1 \oplus X \oplus 1).$$

Now for each of $w = 2^{2p}$ trial values W, and each of 2^{56} trial values k for K, evaluate $f(k, W) = e_k(W) \oplus e_k(W \oplus 1)$. The Ws are 0 in the high order $64 - 2p$ bits. For each false positive $g(X) = f(k, W)$, do a bit more sleuthing. Eventually you find the right setting: $K = k$, $H_1 = X \oplus W$.

Then (if $p = 0$ and we had initially 2^{67} chosen texts) we have enough information to get the truncated 32 bits of $e_{K'}(e_{K'''}(U))$ for each 64-bit input U, although we don't have enough information to easily get K' and K''' themselves. If p is larger than 0 then we will only build a partial dictionary.

Alternatively, once K and H_1 are recovered, one can break double-DES to recover K'' and K'''. At this point, K, K'', and K''' are known and one finds K' using about 2^{57} MAC verifications.

Forgery attacks Truncating the MAC values does not substantially increase the complexity of the forgery attack described in Section 5. For example, if the MAC length $m = 32$, two known-text MACs (of equal lengths) will be required. In the verification step, a check is performed on the second message (only) if the first verification succeeds.

7 Conclusions

We have seen that the most effective key recovery attack against the MacDES scheme (with Padding Method 3) has complexity $[2^{59}, 2^{33}, s \times 2^{48}, 0]$ for a small

$s \geq 3$. This compares with the previously best known attack which has complexity $[2^{89}, 0, 2^{65}, 2^{55}]$. This means that this scheme is still better than the ANSI retail MAC, i.e. MAC algorithm 3 from [4], but not as much as previously thought. In addition a new forgery attack against this scheme (and others) has been described, requiring just one 'chosen MAC'. The use of MAC truncation makes the attacks considerably more difficult. As an example, when used with DES and a MAC value of 32 bits we outlined a key-recovery attack of complexity $[2^{64}, 0, 2^{63}, 2^{57}]$ (with possible trade-offs between chosen texts and computation). If Serial Numbers are employed, then the attacks appear to become even more infeasible.

References

1. B. Bollobás. *Random graphs.* Academic Press, 1985.
2. K. Brincat and C.J. Mitchell. A taxonomy of CBC-MAC forgery attacks. Submitted, January 2000.
3. D. Coppersmith and C.J. Mitchell. Attacks on MacDES MAC algorithm. *Electronics Letters*, **35**:1626–1627, 1999.
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*, December 1999.
5. L.R. Knudsen. Chosen-text attack on CBC-MAC. *Electronics Letters*, **33**:48–49, 1997.
6. L.R. Knudsen and B. Preneel. MacDES: MAC algorithm based on DES. *Electronics Letters*, **34**:871–873, 1998.
7. A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, Boca Raton, 1997.
8. B. Preneel and P.C. van Oorschot. On the security of iterated Message Authentication Codes. *IEEE Transactions on Information Theory*, **45**:188–199, 1999.
9. J. Spencer. *Ten lectures on the probabilistic method.* Society for Industrial and Applied Mathematics, Philadelphia, PA, second edition, 1994.
10. P.C. van Oorschot and M.J. Wiener. Parallel collision search with cryptanalytic applications. *Journal of Cryptology*, 12(1):1–28, 1999.