Analysis of 3gpp-MAC and two-key 3gpp-MAC

Lars R. Knudsen [a] Chris J. Mitchell [b]

[a] *Technical University of Denmark, MAT, DK-2800 Kgs. Lyngby, Denmark,*
lars@ramkilde.com; http://www.ramkilde.com

[b] *ISG, Royal Holloway, University of London, Egham, Surrey TW20 0EX, UK,*
c.mitchell@rhul.ac.uk; http://www.isg.rhul.ac.uk/cjm

Abstract

Forgery and key recovery attacks are described on the 3gpp-MAC scheme, proposed for inclusion in the 3gpp specification. Three main classes of attack are given, all of which operate whether or not truncation is applied to the MAC value. Attacks in the first class use a large number of 'chosen MACs', those in the second class use a large number of 'known MACs', and those in the third class require a large number of MAC verifications, but very few known MACs and no chosen MACs. The first class yields both forgery and key recovery attacks, whereas the second and third classes are key recovery attacks only. Both single-key and two-key variants of 3gpp-MAC are considered; the forgery attacks are relevant to both variants, whereas the key recovery attacks are only relevant to the two-key variant.

1 Introduction

We are concerned here with the security of a CBC-MAC scheme proposed for inclusion in the 3gpp specification, see e.g., the home page of ETSI (the European Telecommunications Standards Institute), `www.etsi.org`. Specifically we consider both forgery and key recovery attacks, which provide upper bounds on the level of security offered by the scheme. We consider both the single key version (as employed in the 3gpp specifications) and the more general two key version.

The 3gpp-MAC scheme operates as follows. Suppose the underlying block cipher has n-bit blocks and uses a key of k bits. If X is an n-bit block then we write $e_K(X)$ for the encryption of X using key K. 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 scheme uses a pair of keys K, K', where, for 3gpp, K' is derived from K by

exoring K with a fixed mask consisting of alternate ones and zeros. The MAC computation is as follows.

$$\begin{aligned} H_1 &= e_K(D_1), \\ H_i &= e_K(D_i \oplus H_{i-1}), \quad (2 \le i \le q), \text{ and} \\ \text{MAC} &= e_{K'}(H_1 \oplus H_2 \oplus \cdots \oplus H_q). \end{aligned}$$

We refer to H_i as the chaining variable. For the purposes of this paper we assume that the padding method does not involve prefixing the data with a length block. Note that the MAC used will be truncated to the left-most m bits of the MAC value given in the above equation, where $m \le n$.

Following the approach used in [1], we use a four-tuple $[a, b, c, d]$ 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, c denotes the number of chosen data string/MAC pairs, and d denotes the number of on-line MAC verifications. Note c and d are distinguished because, in some environments, it may be easier for an attacker to obtain MAC verifications (i.e. to submit a data string/MAC pair and learn whether the MAC is valid) than to obtain the MAC for a chosen message.

In the analysis of MAC algorithms based on a block cipher with a k-bit key, it is standard to assume that the block cipher itself is secure, and hence a key recovery attack will require at least 2^k invocations of the block cipher. Observe that, for 3gpp-MAC, if the two k-bit MAC keys K, K' are related, i.e. one is derived from the other, or they are both derived from a single k-bit key, then a simple exhaustive search can be performed using a very small number of known MACs. The total complexity is $[(q+3)2^k, \lceil (k+1)/m \rceil, 0, 0]$, where q is the number of blocks in the shortest message for which a MAC is known. That is, from the point of view of key recovery attacks, using a single key version of this MAC scheme, as specified in the 3gpp documents, is not significantly better than the simplest of CBC-MAC schemes (e.g. those in [1]). The main purpose of 3gpp-MAC would therefore appear to be to achieve greater resistance to forgery attacks than the standardised CBC-MAC schemes, at minimal additional cost.

In the first part of Section 2 we exhibit a series of forgery attacks against 3gpp-MAC which show that the level of security obtained by this scheme is not as high as one might expect from an iterative MAC scheme employing a $2n$-bit internal chaining variable. Nevertheless the complexity compares well with the attack complexities for comparable standardised schemes from [1]. This analysis makes no assumptions about the keys, i.e. the keys are treated as being independent of one another. That is, the analysis applies regardless of whether the keys are chosen independently or one is derived from the other, as in 3gpp.

It is therefore arguable that the 3gpp-MAC scheme is a candidate for wider adoption as a CBC-MAC technique. This means that an analysis of the resistance of the two-key variant to key recovery attacks is also of interest, especially if the scheme is to be used with a block cipher such as DES with a relatively small key space. Thus, in the second part of Section 2 and in the remainder of the paper, we consider a variety of key recovery attacks against 3gpp-MAC in the two-key case. In the concluding section we tabulate the security of 3gpp-MAC against various types of attack; when considering 3gpp-MAC as a possible CBC-MAC technique for general applications, these tables should be compared with the comparable tables given in [1] for standardised CBC-MAC techniques.

2 Attacks using chosen MACs

Since the internal memory of 3gpp-MAC is $2n$ one might expect that a collision attack will require 2^n operations. But, as we will show, this is not the case. We first consider a chosen-text scenario, where we give two forgery attacks and then a key recovery attack derived from them.

Following [4], most of the attacks are based on finding collisions, i.e. two messages with the same MAC. We also refer to *internal collisions*, where this describes two messages with both the same MAC and the same final chaining variable, and *external collisions*, i.e. collisions that are not internal collisions.

A forgery attack

We distinguish between the cases where $m = n$ and $m < n$.

The case $m = n$. Collect $2^{(n+1)/2}$ messages of q blocks, where $q \geq 2$, and get the corresponding MAC values. (If the attacker has full control over q, he chooses $q = 2$.) One expects to find a single collision, that is, two different messages with the same MAC value; denote these by $D = D_1, \ldots, D_q$ and $D' = D'_1, \ldots, D'_q$. These messages collide in n of the $2n$ bits of internal memory. In the next step a collision is found also in the other n bits.

Choose $2^{n/2}$ messages of the form $D(i) = D_1, \ldots, D_q, E(i)$ and $2^{n/2}$ messages of the form $D'(j) = D'_1, \ldots, D'_q, E'(j)$, where $E(i)$ and $E'(j)$ are randomly chosen n-bit blocks, and get the corresponding MAC values. One expects to find a single internal collision, say $D(i)$ and $D'(j)$. Define $\Delta = E(i) \oplus E'(j)$.

Thus for any n-bit block Z the messages $D_1, \ldots, D_q, Z$ and $D'_1, \ldots, D'_q, Z \oplus \Delta$ will have the same MAC value. Therefore, one can forge the MAC of $D'_1, \ldots, D'_q, Z \oplus \Delta$ by observing or requesting the MAC of $D_1, \ldots, D_q, Z$. The

complexity of this attack is $[0, 2^{(n+1)/2}, 2^{n/2+1}, 0]$.

The case $m < n$; Attack 1. A first attack is a simple extension of the above attack, and applies when $m > n/2$. Collect $2^{(n+1)/2}$ messages and their corresponding MACs. One can expect to find 2^{n-m} pairs of messages with matching MACs, one of which will collide also in the $n - m$ discarded bits [3]. For each of these pairs, D and D' say, choose $2^{n/2}$ messages of the form $D(i) = D, E(i)$ and $2^{n/2}$ messages of the form $D'(j) = D', E'(j)$, where $E(i)$ and $E'(j)$ are randomly chosen n-bit blocks, and get the corresponding MAC values. From [3], one expects to find 2^{n-m} pairs of messages $D(i)$ and $D'(j)$ with matching MACs.

Define $\Delta_{i,j} = E(i) \oplus E'(j)$. It remains to test whether $\Delta_{i,j}$ can be used to forge MACs as in the case for $m = n$. Simply get the MACs of the messages D, Z and $D', Z \oplus \Delta_{i,j}$ for some value of Z, and check whether the MACs are equal. In almost all cases, one such test will suffice to distinguish external collisions and internal collisions. Thus, this attack requires $2^{n-m}(2^{n/2+1} + 2^{n-m})$ chosen messages, which for $m > n/2$ is approximately $2^{n-m}2^{n/2+1} = 2^{3n/2-m+1}$. The complexity of this attack is $[0, 0, 2^{3n/2-m+1}, 0]$, when $m > n/2$ (note that we ignore the $2^{(n+1)/2}$ MACs required in the first stage, since this number is dominated by $2^{3n/2-m+1}$).

The case $m < n$; Attack 2. A second attack is the following. The attacker first collects $2^{(m+n)/2}$ messages and their MAC values. Divide the $2^{(m+n)/2}$ known MACs into 2^m classes according to the value of the truncated MAC. It follows that at least one class will contain $2^{(n-m)/2}$ messages or more, all with the same MAC value. Two of these messages are expected to also collide in the discarded $n - m$ bits. Suppose the messages are $(E_1, E_2, \ldots, E_q)$ and $(F_1, F_2, \ldots, F_r)$ and that the internal part MACs for these two messages are H_q and H'_r.

For every message $(D_1, D_2, \ldots, D_s)$ in the class, get the MAC on message $(D_1, D_2, \ldots, D_s, X)$ for $2^{n/2}$ different randomly chosen values of X. This is a total of $2^{n/2}2^{(n-m)/2} = 2^{n-m/2}$ chosen MACs. By the usual birthday arguments, there is a good chance that MACs will have been computed for messages $(E_1, E_2, \ldots, E_q, X)$ and $(F_1, F_2, \ldots, F_r, Y)$ where $H_q \oplus X = H'_r \oplus Y$.

That is, somewhere amongst the set of $2^{n-m/2}$ chosen MACs is an internal collision. The problem remains of finding the internal collision from amongst the many other collisions, where the expected total number of messages involved in an 'external collision' is bounded above by 2^{2n-2m}. Note that once we have the internal collision, we can use it to forge MACs in much the same vein as in the case $m = n$.

If m is large relative to n (say $m > 2n/3$), then this is relatively straightforward

to achieve, since 2^{2n-2m} will be small relative to the total number of messages $(2^{n-m/2})$. For every message M which gives rise to a collision, obtain a further chosen MAC for the message M, C, where C is a fixed n-bit block. The number of additional MACs required will be less than $2^{n-m/2}$, since we only consider those messages which give rise to a collision. Hence we can ignore this number in assessing the attack complexity, since it is dominated by $2^{n-m/2}$. Hence, for this simple case, the total complexity of the attack is $[0, 2^{(m+n)/2}, 2^{n-m/2}, 0]$.

However, if m is smaller relative to n, say $m \leq 2n/3$, then the attack is a little more complex. In such a case, most of the messages $(D_1, D_2, \ldots, D_s, X)$ are involved in an external collision, and so the simple strategy will not work. Instead, we obtain further chosen MACs for every message (to avoid unnecessary sorting). The number of additional MACs, h, is set to

$$h = \lceil (2n - 2m)/m \rceil.$$

That is, for every $(D_1, D_2, \ldots, D_s, X)$, we obtain a set of h further chosen MACs for the messages $(D_1, D_2, \ldots, D_s, X, C_i)$, where $C_1, C_2, \ldots, C_h$ are a fixed set of n-bit blocks. That is we compute a total of $(h + 1)2^{n-m/2}$ chosen MACs. We thus associate with each choice for $(D_1, D_2, \ldots, D_s, X)$ an $(h + 1)$-tuple of MACs. It is simple to see that if $(D_1, D_2, \ldots, D_s, X)$ and $(D'_1, D'_2, \ldots, D'_{s'}, X')$ are two such choices which give rise to an internal collision, then all of the MACs in the respective $(h + 1)$-tuples will match. The value of h is chosen so that there is a very good chance that only the internal collision will give rise to a match (any additional matches can in any case be eliminated using a small additional number of chosen MACs, which will not affect the overall complexity estimate).

For this more complex case, the total complexity of the attack is therefore equal to $[0, 2^{(m+n)/2}, (\lceil (2n - 2m)/m \rceil + 1)2^{n-m/2}, 0]$. It follows that the number of known MACs increases and the number of chosen MACs decreases for increasing values of m.

In summary, if $m > 2n/3$ the first attack is faster, if $m < 2n/3$ the second attack is faster, and the complexities of the two attacks are similar if $m = 2n/3$. Note that these complexities compare favourably with the forgery attack complexities for most of the standardised CBC-MAC techniques, as detailed in an annex to [1]. That is, this technique might merit consideration for adoption in any future revision to the standard.

A key-recovery attack

The internal collisions of the previous section can also be used in key-recovery attacks. Take the pair of messages giving the internal collision from the forgery attack. For all values of the key K, compute the MACs of the two messages

and check for collisions in the $2n$ bits. For a wrong value of the key each of these collision checks will succeed with probability 2^{-2n}. If more than one key candidate passes this test, it is repeated for chosen MACs on messages constructed by adding a fixed block to the end of the two messages in the internal collision. Once K is known, K' can be found by exhaustive search.

The attack requires approximately $2^{k+1}q$ operations where q is the average length of the messages involved. In a chosen plaintext attack the attacker chooses $q = 2$ in which case the complexity is that of the forgery attack plus 2^{k+2} block cipher operations.

3 Key recovery attacks using known MACs

The 'basic' Preneel-van Oorschot attack

If $m = n$ then the attack of [4] applies with complexity $[s \times 2^k, 2^{n/2}, 0, 0]$, where s is a small positive integer approximately equal to $2q$, and q is the average block-length of the messages for which the MAC is known.

Briefly, when applied to this MAC scheme, the Preneel-van Oorschot attack is as follows. Consider arbitrary (padded) messages $D_1, D_2, \ldots, D_q$ and $E_1, E_2, \ldots, E_r$. Also let

$$
\begin{aligned}
H_1 &= e_K(D_1), \\
H_i &= e_K(D_i \oplus H_{i-1}), \quad (2 \le i \le q), \\
H'_1 &= e_K(E_1), \text{ and} \\
H'_i &= e_K(E_i \oplus H'_{i-1}), \quad (2 \le i \le r),
\end{aligned}
$$

The messages will give the same MAC if and only if $\bigoplus_{i=1}^{q} H_i = \bigoplus_{i=1}^{r} H'_i$. Hence a collision gives a means to eliminate candidates for K independently of the value of K', i.e. a collision can be used as the basis of an exhaustive search for K. A separate exhaustive search can be used to find K'. Finding the desired collision will require around $2^{n/2}$ known MACs, and the subsequent exhaustive search will require $q + r$ encryptions for each candidate key (where q and r are the numbers of blocks in the colliding messages). The attack complexity given above thus follows.

Unfortunately, if m is significantly less than n, then this attack becomes much more complex, since it becomes difficult to distinguish full n-bit collisions from m-bit ones. That is, there will be many external collisions that need to be eliminated in order to find the desired internal collision. Very approximately, $\max\{2^{n/2}, 2^{n-m}\}$ of the set of $2^{n/2}$ messages will be involved in one or more internal collisions. Hence, when the exhaustive search is performed, the number

of encryption operations necessary will grow to very approximately $2^{k+n-m}q$ if $m > n/2$ and $2^{k+n/2}q$ if $m \leq n/2$, where q is the average block-length of the messages. (Note that extra collisions and searches will be required to eliminate the remaining key candidates if $m \leq k$, although the numbers of calculations involved will not affect the overall complexity).

This is potentially a very large number of encryption operations. As a result, in the next section we consider a generalised version of the Preneel-van Oorschot attack which reduces the number of encryption operations at the cost of increasing the number of known MACs required.

A generalised Preneel-van Oorschot attack

The generalised attack operates in a series of three stages.

Stage 1: Processing the known MACs. Suppose the attacker has $2^{(n+m)/2}$ known MACs. The attacker divides these messages into 2^m classes according to the value of the MAC. Each class will contain approximately $2^{(n-m)/2}$ messages, all with the same MAC value, i.e. for which the first m bits of the 'untruncated MAC' are equal. Since there are 2^{n-m} possibilities for the $n-m$ bits discarded during MAC truncation, there is a good chance that, within any class, there is at least one pair of messages whose 'untruncated MACs' agree. To maximise this probability, we work with the classes containing the most messages.

Stage 2: Exhaustive search to find K. Choose the $\lceil k/m \rceil$ largest classes of messages (as constructed in Stage 1). For each such class perform an exhaustive search as follows.

For every key compute the value $\bigoplus_{i=1}^{q} H_i$ for each message in the class (where H_i is defined as above, and q is the number of blocks in the padded message). This involves approximately $2^k 2^{(n-m)/2} q$ encryption operations, where q is the average block-length of the messages. The values resulting are then examined to see if there is a collision. For the correct key K there is a good chance that there will be a collision, whereas for a randomly selected key the probability of a collision is equal to the probability that a randomly chosen set of $2^{(n-m)/2}$ n-bit values will contain two identical values. This is very approximately 2^{-m} (see [3]).

After this search has been performed for each of the message classes, each key will have an associated 'count', indicating the number of message classes for which a collision was found. A simple probabilistic analysis reveals that the key with the greatest count is likely to be the 'correct' value of K.

Note that the first exhaustive search (using the largest class) could be used to eliminate all but a small fraction of keys, by only proceeding with those keys for which a collision is found. This reduces the complexity of subsequent

searches dramatically, meaning that the overall cost of this stage is reduced to approximately $2^{k+(n-m)/2}q$ encryption operations. The downside of such an approach is that it would increase the risk that the 'correct' key is missed. In the complexity analysis we assume that such an approach is used.

Stage 3: Exhaustive search to find K'. The final stage is the simplest. Armed with the correct value of K, a set of any $\lceil (k+1)/m \rceil$ messages with known MACs can be used as the basis of an exhaustive search to find K'.

An analysis of the generalised attack

We start by considering the complexity of the attack. The costs of stage 1–3 are $2^{(n+m)/2}$ known MACs, $2^{k+(n-m)/2}q$ encryptions, and $2^k q$ encryptions respectively. This means that the total attack complexity is approximately $[2^{k+(n-m)/2}q, 2^{(n+m)/2}, 0, 0]$. Thus, by comparison with the 'basic' Preneel-van Oorschot attack, and assuming that $m \leq n/2$, the number of known MACs has increased from $2^{n/2}$ to $2^{(n+m)/2}$, whereas the number of off-line encryptions has decreased from $2^{k+n/2}q$ to $2^{k+(n-m)/2}q$.

Finally note that, if less then $2^{(n+m)/2}$ known MACs are available, then the attack still works. However, more classes of messages will need to be examined, i.e. there is a trade off between decreasing the number of known MACs and increasing the number of off-line encryption operations. In the limit, if only $2^{n/2}$ known MACs are available, all classes containing more than one message will need to be examined, and the attack becomes identical to the Preneel-van Oorschot attack discussed in the previous section. This justifies the use of the term 'generalised' in the name of the attack.

4 A key recovery attack using MAC verifications

We now consider a different type of key recovery attack, analogous to the Knudsen-Preneel attack, [2]. Instead of using a large number of known MACs, these attacks require only one (or at most a very small number of) known MAC(s) combined with a large number of MAC verifications.

The case $m = n$

First suppose $m = n$, i.e. there is no truncation, and suppose also that $k < n$. Now suppose the attacker knows the MAC for some message, i.e. the attacker knows that $\mathrm{MAC}(D_1, D_2, \ldots, D_q) = M$ for some (padded) sequence of message blocks $D_1, D_2, \ldots, D_q$.

The attacker now assembles the set of all 2^n $(q+1)$-block padded messages of the form $X, D_1, D_2, \ldots, D_q$, where X ranges over all possible n-bit blocks,

and for each finds out whether or not the valid MAC is M (thus requiring 2^n 'MAC verifications'). This will be true for the unique case where $e_K(X) = 0$ (the all-zero n-bit block), and also 'by chance' for approximately one value of X. Thus, if a unique value of X results from this test then the attacker can be sure that this is the desired value, i.e. the one for which $e_K(X) = 0$. If more than one value of X 'passes' the test, then a second known MAC can be used to eliminate the remaining false candidates by repeating the above test but with only the remaining candidate values of X.

Armed with the desired equation $e_K(X) = 0$, the attacker can do an exhaustive search for K, which will probably only yield the correct value of K since we assumed that $k < n$. Finally, once K is known, the known MAC can be used as the basis of an exhaustive search for K', which will again probably only yield the correct value of K' since $k < n$.

If the attacker is lucky, and only one known MAC is needed to find the desired X, the total complexity of the above attack is $[2^{k+1}, 1, 0, 2^n]$, since the exhaustive searches for K and K' will require 2^k encryptions each. If the attacker is unlucky and requires a second known MAC in the initial part of the attack then the complexity will increase to at most $[2^{k+1}, 2, 0, 2^n]$.

We deal with the case $k \geq n$ as part of the analysis of the case $m \leq n$.

The case $m \leq n$

If the MAC is truncated, i.e. if $m \leq n$, then a very similar attack approach will work, although a small number of additional known MACs are needed.

The case $k < n$. In this case the attacker starts the attack exactly as in the $m = n$ case, although, after the initial set of 2^n MAC verifications, a total of approximately $2^{n-m} + 1$ candidates for X will remain. The MAC verification step can be repeated with a second known MAC (and the set of remaining candidates for X), after which approximately $2^{n-2m} + 1$ candidates will remain. The process can be repeated as many times as necessary until a single candidate for X satisfying $e_K(X) = 0$ remains.

Thus, given a set of around $\lceil (n+1)/m \rceil$ known MACs, the required block X satisfying $e_K(X) = 0$ can be found. The total number of MAC verifications required will be approximately $2^n + 2^{n-m} + 2^{n-2m} + \cdots$, which is approximately the same as 2^n (given that, in practice, m will always be at least 8).

The exhaustive search for K will be exactly as in the previous case, and the search for K' may require use of up to $\lceil (k+1)/m \rceil$ known MACs (which can be the same as those used in the first stage of the attack). Again as previously, the search for K will probably yield a unique result because we assumed that $k < n$.

Given that we assumed that $k < n$, and hence $(k+1)/m < (n+1)/m$, the total complexity of the attack will thus be approximately $[2^{k+1}, \lceil (n+1)/m \rceil, 0, 2^n]$.

The case $k \geq n$. In this case the attacker proceeds as when $k < n$, and we suppose that, given around $\lceil (n+1)/m \rceil$ known MACs, the required block X satisfying $e_K(X) = 0$ can be found. The total number of MAC verifications required will be approximately 2^n. The complexity of this first part of the attack will therefore be approximately $[0, \lceil (n+1)/m \rceil, 0, 2^n]$.

However, the next stage is a little more complex in that, when the exhaustive search for K is performed using the equation $e_K(x) = 0$, approximately $2^{k-n}+1$ candidates for the key K will remain. It will thus be necessary to obtain further equations involving K to eliminate all but the correct candidate for K. Such equations can be obtained as follows. Suppose the attacker knows that $\mathrm{MAC}(D_1, D_2, \ldots, D_q) = M$ for some (padded) sequence of message blocks $D_1, D_2, \ldots, D_q$. (This can be the same as one of the 'known MACs' previously used). The attacker now assembles the set of all 2^n $(q+1)$-block padded messages of the form $D_1, D_2, \ldots, D_q, Y$, where Y ranges over all possible n-bit blocks, and for each finds out whether or not the valid MAC is M (thus requiring 2^n 'MAC verifications'). This will be true for the unique case where $H_q \oplus Y = X$, where H_q is the chaining variable as defined in Section 1, and also 'by chance' for approximately 2^{n-m} values of X. We will thus obtain a set of approximately $2^{n-m}+1$ candidate values Z for the solution to the equation $H_q = Z$.

This set of candidate values of Z can be used to eliminate a large fraction of the remaining candidates for K, i.e. after performing one such search we would expect the number of candidate values for K to be reduced from approximately $2^{k-n}+1$ to approximately $2^{k-n-m}+1$. The above procedure can be repeated as many times as necessary until just one candidate value for K remains; the number of iterations will be approximately $\lceil (k-n+1)/m \rceil$. Each iteration will require use of a single 'known MAC', 2^n MAC verifications, and $q2^x$ block cipher encryptions, where q is the length of the 'known MAC' message and 2^x is the number of candidate keys remaining at this stage of the attack. Note that it is reasonable to assume that $q2^x$ will be orders of magnitude smaller than 2^k, and hence we ignore the cost of these calculations.

Hence the total complexity of this stage of the attack will be

$$[2^k, \lceil (k-n+1)/m \rceil, 0, \lceil (k-n+1)/m \rceil 2^n]$$

where the set of $\lceil (k-n+1)/m \rceil$ known MACs can overlap with the set of known MACs used in the search for X.

Finally, as when $k < n$, the exhaustive search for K' will require use of ap-

proximately $\lceil (k+1)/m \rceil$ known MACs, i.e. the complexity of this last part of the attack will therefore be approximately $[2^k, \lceil (k+1)/m \rceil, 0, 0]$.

Thus the total attack complexity will be approximately

$$[2^{k+1}, \lceil (\max(k,n)+1)/m \rceil, 0, \lceil (k-n+m+1)/m \rceil 2^n].$$

Applying the attack to other MAC schemes

It should be clear that the above attacks (for $m < n$) will also work in almost exactly the same way against any of MAC algorithms 1, 2 or 3 from ISO/IEC 9797-1, [1], with the exception that the 'extra iterations' needed to deal with the case $k \geq n$ will need to be modified slightly, since for these schemes we know that $\mathrm{MAC}(D_1, D_2, \ldots, D_q) = \mathrm{MAC}(D_1, D_2, \ldots, D_q, Y)$ when $e_K(H_q \oplus Y) = H_q$ (as opposed to $H_q \oplus Y = X$).

Of course, for these algorithms the attack of Knudsen and Preneel, [2], also applies, and the two attacks have very similar complexities. The main difference is that the Knudsen-Preneel attack requires 2^k MAC verifications, whereas the attack described here requires $\lceil (k-n+m+1)/m \rceil 2^n$ MAC verifications. Hence, if $k \leq n$, the new attack is somewhat less efficient, although if $k > n+1$ the new attack requires less MAC verifications.

5 Conclusions

We have described three different types of key recovery attack on the 3gpp-MAC scheme. In doing so we have also described methods for forging 3gpp-MACs. Table 1 summarises the complexities of the various key-recovery attacks, and Table 2 summarises the complexities of the forgery attacks. In both tables k and n denote the size of the key and the block, respectively, of the underlying block cipher, m denotes the number of bits in the MAC, a, b, c, and d denote resources required for the attack (as in Section 1), and q denotes the number of blocks in the known messages of the particular attack.

Our results show that in the case where the MACs are not truncated the complexities of key-recovery attacks and forgery attacks are comparable to those of the standard CBC-MAC. Thus in this case the extra effort in the computation of 3gpp-MACs does not pay off in terms of increased security.

Finally note that for the 3gpp application we have $k = 128$, $n = 64$ and $m = 32$, which makes the key recovery attacks infeasible, and also means that the most efficient forgery attack requires around 3×2^{48} chosen messages. This is unlikely to pose a problem in practice.

Table 1
Key-recovery attack complexities

a	b	c	d	Condition	Section
2^{k+2}	$2^{(n+1)/2}$	$2^{n/2+1}$	0	$m = n$	2
2^{k+2}	0	$2^{3n/2-m+1}$	0	$n/2 < m < n$	2
2^{k+2}	$2^{(n+m)/2}$	$(\lceil (2n-2m)/m \rceil + 1)2^{n-m/2}$	0	$m < n$	2
$2^{k+(n-m)/2}q$	$2^{(n+m)/2}$	0	0		3
2^{k+1}	1	0	2^n	$m = n$, $k < n$	4
2^{k+1}	$\lceil (n+1)/m \rceil$	0	2^n	$k < n$	4
2^{k+1}	$\lceil (k+1)/m \rceil$	0	2^n	$k \geq n$	4

Table 2
Forgery attack complexities

a	b	c	d	Condition	Section
0	$2^{(n+1)/2}$	$2^{n/2+1}$	0	$m = n$	2
0	0	$2^{3n/2-m+1}$	0	$n/2 < m < n$	2
0	$2^{(n+m)/2}$	$(\lceil (2n-2m)/m \rceil + 1)2^{n-m/2}$	0	$m < n$	2

6 Acknowledgements

We would like to thank two anonymous referees for their valuable comments and corrections.

References

[1] 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.

[2] L.R. Knudsen and B. Preneel, *MacDES: MAC algorithm based on DES*, Electronics Letters **34** (1998), 871–873.

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

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