J. Cryptology (2005) 18: 375–389
DOI: 10.1007/s00145-004-0324-7

Journal of
CRYPTOLOGY

© 2005 International Association for Cryptologic Research

Partial Key Recovery Attack Against RMAC*

Lars R. Knudsen
Department of Mathematics, Technical University of Denmark,
DK-2800 Lyngby, Denmark
lars@ramkilde.com

Chris J. Mitchell
Royal Holloway, University of London,
Egham, Surrey TW20 OEX, England
c.mitchell@rhul.ac.uk

Communicated by Ronald Cramer

Received 14 April 2003 and revised 1 September 2003
Online publication 24 February 2005

Abstract. In this paper new "partial" key recovery attacks against the RMAC block cipher based Message Authentication Code scheme are described. That is we describe attacks that, in some cases, recover one of the two RMAC keys much more efficiently than previously described attacks. Although all attacks, but one, are of no major threat in practice, in some cases there is reason for concern. In particular, the recovery of the second RMAC key (of k bits) may only require around $2^{k/2}$ block cipher operations (encryptions or decryptions). The RMAC implementation using triple DES proposed by NIST is shown to be very weak.

Key words. Message Authentication Codes, RMAC, AES, Triple DES.

1. Introduction

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, along with the secret key K, and the output is the MAC. The MAC is then sent or stored with the message.

Most common message authentication algorithms today are iterated MAC algorithms. The MAC input D is padded to a multiple of the block size, and is then divided into q blocks denoted D_1–D_q. The m-bit MAC involves an initial value $IV = H_0$, a *compres-*

* The first author was supported by the Danish National Science Research Council Grant No. 21-02-0093.

sion function h, an *output transformation* g, and an n-bit ($n \geq m$) *chaining variable* H_i between stage $i-1$ and stage i:

$$\begin{aligned} H_i &= h(D_i, H_{i-1}), \qquad 1 \leq i \leq q, \\ \mathrm{MAC}_K(D) &= g(H_q)\,. \end{aligned}$$

The secret key may be employed in h, and/or in g.

ISO/IEC 9797-1 [5] lists three possible padding rules for iterated MACs. If the message string is $D = D_1, D_2, \ldots, D_q$, then these rules are:

1. append additional zero bits, such that $|D_q| = n$;
2. if $|D_q| = n$ append an extra block D_{q+1} consisting of one one-bit followed by zero bits, such that $|D_{q+1}| = n$, otherwise append a one-bit then zero bits, such that $|D_q| = n$;
3. apply rule 1 then prefix the data with one n-bit block D_0 containing q.

One very widely used class of iterated MAC schemes are called CBC-MACs, where (typically) $h(x, y) = e_K(x \oplus y)$, and where $e_K(z)$ denotes the block cipher encryption of z using the key K. To avoid some simple forgery attacks [13, Section 9.5], g is most often a function which outputs a certain subset of bits of the input, or consists of one or more encryption operations, or is a combination of the two. In this paper we are concerned with a particular example of a CBC-MAC scheme known as RMAC [6], [17].

There are two main classes of attack on a MAC scheme, namely *key recovery* attacks, in which an attacker is able to discover the secret key used to compute the MACs, and *forgery attacks* in which an attacker is able to determine the correct MAC for a message (without the legitimate key holder having generated it).

Forgery attack: This attack consists of predicting the value of $\mathrm{MAC}_K(D)$ for a message D without initial knowledge of K. If the adversary can do this for a single message, he is said to be capable of *existential forgery*. If the adversary is able to determine the MAC for a message of his choice, he is said to be capable of *selective forgery*. A very simple "attack" is to choose an arbitrary fraudulent message, and to append a randomly chosen MAC value. Ideally, the probability that this MAC value is correct is equal to $1/2^m$, where m is the number of bits of the MAC value. Practical attacks often require that a forgery is *verifiable*, i.e., that the forged MAC is known to be correct beforehand with probability near 1.

Key recovery attack: This attack consists of finding the key K itself from a number of message/MAC pairs. Such an attack is more powerful than a forgery, since it allows for arbitrary selective forgeries. Ideally, any attack allowing key recovery requires about 2^k operations (here k denotes the bit-length of K). If m is the size of the MAC and if one assumes that $\mathrm{MAC}_K(D)$ is a random function from the key to the MAC, then verification of such an attack requires about $\lceil k/m \rceil$ text/MAC pairs. To see this, note that the expected number of keys which will take the message D to a certain given MAC value is 2^{k-m}. Extending this argument, the expected number of keys which will take $\lceil k/m \rceil$ messages to certain given MAC values is $2^{k-(m\lceil k/m \rceil)} \leq 1$.

Partial key recovery attacks are those in which only some $k' < k$ bits of the secret key are retrieved.

Preneel and van Oorschot [20] present a general forgery attack that applies to all iterated MACs. Its feasibility depends on the bit sizes n of the chaining variable and m of the MAC result and on the nature of the output transformation g. The basic attack requires several known texts, but only a single chosen text. However, in some cases restrictions are imposed on the known texts; for example, if the padding method used is the third above, all messages must have equal length. For an input pair (x, x') with $\mathrm{MAC}_K(x) = g(H_q)$ and $\mathrm{MAC}_K(x') = g(H'_q)$, a collision is said to occur if $\mathrm{MAC}_K(x) = \mathrm{MAC}_K(x')$. This collision is called an *internal* collision if $H_q = H'_q$, and an *external* collision if $H_q \neq H'_q$ but $g(H_q) = g(H'_q)$. The attack is based on the following simple observation:

Lemma 1 [20]. *An internal collision for an iterated MAC algorithm allows a verifiable MAC forgery, through a chosen-text attack requiring a single chosen text.*

This follows since, for an internal collision (x, x'), $\mathrm{MAC}_K(x \parallel y) = \mathrm{MAC}_K(x' \parallel y)$ for any single block y; thus a requested MAC on the chosen text $x \parallel y$ provides a forged MAC (i.e., the same MAC) for $x' \parallel y$ (here $\parallel$ denotes concatenation). Note this assumes that the MAC algorithm is deterministic. Also, the forged message is of a special form, which may limit the practical impact of the attack.

The next propositions show the complexities of finding an internal collision for a given MAC algorithm.

Proposition 1 [20]. *Let* MAC() *be an iterated MAC function with an n-bit chaining variable and m-bit result, and an output transformation g that is a permutation. An internal collision for MAC can be found using an expected number of $u = \sqrt{2 \cdot 2^{n/2}}$ known text/MAC pairs of at least $t = 2$ blocks each.*

Proposition 2 [20]. *Let* MAC() *be an iterated MAC function with an n-bit chaining variable and m-bit result, and output transformation g which is a random function. An internal collision for h can be found using u known text/MAC pairs of at least $t = 2$ blocks each and v chosen texts of at least three blocks. The expected values for u and v are as follows: $u = \sqrt{2} \cdot 2^{n/2}$ and v is approximately 2^{n-m}.*

A widely used type of CBC-MAC (when using DES) is where the output transformation g is a two-key triple encryption (with keys $K_1 = K$ and K_2); this is commonly known as the ANSI retail MAC, since it first appeared in [1]:

$$g(H_q) = e_{K_1}(d_{K_2}(H_q)) = e_{K_1}(d_{K_2}(e_{K_1}(D_q \oplus H_{q-1}))).$$

The scheme is designed for use with DES as the underlying block cipher to compensate for its short key length. However, it has been shown that the above generic forgery attack, requiring about $2^{n/2}$ known text/MAC pairs, can be extended to an attack which recovers the entire key in time $3 \cdot 2^k$ encryptions, compared with 2^{2k} encryptions for an exhaustive search [19], [20]. Other key recovery attacks, also requiring a small multiple of 2^k encryption operations, need only a few known texts, but require many MAC verifications [10], [11], [14].

A variant of the ANSI retail MAC with the same computational complexity was proposed in [11], where a double block cipher encryption is used in the first and last iteration:

$$H_1 = e_{K_2'}(e_{K_1}(D_1)) \quad \text{and} \quad g(H_q) = e_{K_2}(H_q).$$

Here K_2' is derived from K_2. However, it has been shown that there are again key recovery attacks of time complexity approximately only that of an exhaustive search over one of the keys involved [3], [4]. Also, it has recently been shown [7] that for all schemes but one in ISO/IEC 9797-1 [5], key recovery attacks exist requiring about $2^{n/2}$ chosen texts and 2^k block cipher operations.

Bellare et al. [2] provide a proof of security for the "basic" CBC-MAC where $g(H_q) = H_q$, i.e., a lower bound on the complexity of breaking the system given certain assumptions about the underlying block cipher. The proof assumes that the messages authenticated are all of the same length. DMAC is a CBC-MAC scheme with $g(H_q) = e_{K'}(H_q)$, where K' is a key different from K [5], [18], [21]. There is a also proof of security for DMAC, similar to the aforementioned one for the "basic" CBC-MAC, but which holds for messages of varying length [18].

Following the approach used in [5], 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. Moreover, as is the case for RMAC, it is possible to learn different information from a MAC verification than from a chosen MAC, since we assume that the attacker has no control over the random salt used for a chosen MAC.

This paper is organized as follows. Section 2 presents the RMAC scheme. In Section 3 several key-recovery attacks on RMAC are presented, all of which recover one of the two RMAC keys much more efficiently than previously described attacks. Section 4 presents an efficient attack on RMAC used with triple-DES as proposed by NIST [17], and Section 5 gives some concluding remarks.[1]

2. The RMAC scheme

The RMAC message authentication code was proposed by Jaulmes et al. [6] in 2002. The scheme operates as follows. First suppose the underlying block cipher has n-bit blocks and uses a key of k bits. The MAC scheme uses a pair of keys K_1, K_2. The MAC computation is as follows.

A message D is first padded and split into a sequence of q n-bit blocks: $D_1, D_2, \ldots, D_q$. The entity computing the MAC generates a k-bit random salt R, and then makes the

[1] The attacks of Sections 3.5 and 4 also appear in [9], together with key recovery attacks in multi-user settings.

following computations:

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

The salt R must be stored or sent with the message and MAC.

For the purposes of this paper we assume that the padding method does not involve prefixing the data with a length block. (NIST [17] specifies that padding rule 2 should be used with RMAC.) Note that the MAC used will be truncated to the leftmost m bits of the MAC value given in the above equation, where $m \le n$.

RMAC was designed as an alternative to the DMAC scheme with better resistance against the generic forgery attacks mentioned in the Introduction [6]. However, this element of increased security comes at a price. The attacks presented in this paper recover one of the two k-bit keys of RMAC faster than by brute force. Once this key has been found the scheme is vulnerable to some simple forgery attacks, see, e.g., Section 9.5 of [13]. We remark that none of the attacks presented in this paper would apply to DMAC [5], [18], [21], for which the best known attacks recovering either of the two keys require 2^k operations, where k is the size of the involved keys.

3. Partial Key Recovery Attacks on RMAC

In this section we outline several partial key recovery attacks on RMAC. The attacks are applied to general block ciphers of length n using a k-bit key and do not exploit intrinsic properties of the underlying block cipher.

3.1. *The Basis for a Partial Key Recovery Attack*

Suppose an attacker knows, by some means, the value H_q computed during the RMAC computation for the padded message $D_1, D_2, \ldots, D_q$. Suppose the attacker also knows a MAC for this message (M say) and the corresponding random salt R.

The attacker first chooses an integer s ($0 \le s \le k$) and then, by some means, obtains the MACs $M_1, M_2, \ldots, M_{2^s}$ for this same padded message $D_1, D_2, \ldots, D_q$ for a total of 2^s different (random) salts $R_1, R_2, \ldots, R_{2^s}$. The set of MACs $\{M_1, M_2, \ldots, M_{2^s}\}$ (and corresponding salt values) is sorted to simplify comparisons.

The attacker next computes $C_j = e_{L_j}(H_q)$ for a series of 2^{k-s} arbitrary distinct keys L_j, $j = 1, 2, \ldots, 2^{k-s}$. After each computation the value of C_j is compared with the set of MACs $\{M_1, M_2, \ldots, M_{2^s}\}$. Then with a probability of $1 - (1 - 2^{-k})^{2^k} \simeq 0.63$ (for $k \ge 5$), for at least one pair of values of i, j ($1 \le i \le 2^s$, $1 \le j \le 2^{k-s}$) the equation

$$K_2 \oplus R_i = L_j$$

will hold, where K_2 is one of the two secret keys used to compute the RMACs. If this equation holds then $C_j = M_i$. Of course, this latter equation may hold by chance (with probability 2^{-m}). Thus, during the course of computing the 2^{k-s} values C_j, there will be a total of around 2^{k-m} "false" matches. Eliminating a false match is simple—it requires one decryption of a different MAC value M_{i^*} (using the appropriate offset of the key L_j).

After elimination of all false matches the key K_2 will have been recovered. Hence the total complexity for finding the key K_2 is as follows:

- 2^s chosen MACs, and
- $2^{k-s} + 2^{k-m}$ encryption or decryption operations.

Of course, this all depends on the attacker having a means to find the value H_q. Later in this paper we describe two ways in which this might occur. However, first we consider a simple variation on the above procedure.

3.2. *A Variant Partial Key Recovery Attack*

Observe that the m-bit MAC for an arbitrary message with an arbitrary salt R can be found using an expected $2^{m-1} + (2^{m-1} - 1)/2^m \simeq 2^{m-1}$ MAC verifications. Thus, instead of requiring 2^s chosen MACs, the first part of the attack described in Section 3.1 can be performed using 2^{s+m-1} MAC verifications. Thus, in this case, the total complexity for finding the key K_2 is:

- 2^{s+m-1} MAC verifications, and
- $2^{k-s} + 2^{k-m}$ encryption or decryption operations.

Rather interestingly, this variant of the attack can be made deterministic instead of probabilistic without increasing the complexity. This is because the attacker is able to choose the 2^s salt values $R_1, R_2, \ldots, R_{2^s}$ as well as the 2^{k-s} values $L_1, L_2, \ldots, L_{2^{k-s}}$. Since the attack depends on finding an R_i and L_j such that $R_i \oplus K_2 = L_j$, it becomes possible to choose the values of $R_1, R_2, \ldots, R_{2^s}$ and $L_1, L_2, \ldots, L_{2^{k-s}}$ to try to ensure that such an equation will hold (for some i and j) whatever the value of K_2.

Suppose the attacker chooses the sets $\mathcal{R} = \{R_1, R_2, \ldots, R_{2^s}\}$ and $\mathcal{L} = \{L_1, L_2, \ldots, L_{2^{k-s}}\}$ such that $\mathcal{R}$ and $\mathcal{L}$ are mutually orthogonal subspaces, of dimensions s and $k - s$, respectively, of the k-dimensional vector space over GF(2) with elements the binary k-tuples. This can, for example, readily be achieved by letting $\mathcal{R}$ be the set of all k-tuples whose final $k - s$ digits are zeros, and letting $\mathcal{L}$ be the set of all k-tuples whose first s digits are zeros. Then, for any k-tuple K_2, there exists a unique pair $(R_i, L_j) \in \mathcal{R} \times \mathcal{L}$ such that $K_2 = R_i \oplus L_j$. This holds since the union of a basis for $\mathcal{R}$ with a basis for $\mathcal{L}$ will be a basis for the entire k-dimensional space. Hence, for any k-tuple K_2, there exists a unique pair $(R_i, L_j) \in \mathcal{R} \times \mathcal{L}$ such that $R_i \oplus K_2 = L_j$.

Thus, in this case, if $\mathcal{R}$ and $\mathcal{L}$ are chosen appropriately, the desired "match" will be found with probability 1.

3.3. *A Simple Method for Finding H_q*

The first of the two methods we describe for finding H_q applies only in the case where $m = n$, i.e., where there is no MAC truncation. Suppose the attacker knows the MAC M (and corresponding random salt R) for a $(q + 1)$-block padded message $D_1, D_2, \ldots, D_q, D_{q+1}$. Then, for a sequence of distinct values E (where E is an n-bit block), the attacker conducts a MAC verification for the triple (E, M, R), where E is the one-block padded message, M is the MAC, and R is the salt. (Observe that any MAC verification operation will always involve such a triple of inputs.) This procedure con-

tinues until the MAC verification gives a positive result. The simplest way to implement this would be to use an n-bit counter to generate successive values of E.

Now suppose that E^* is the value for which MAC verification succeeds—thus we know that

$$e_{K_2 \oplus R}(e_{K_1}(E^*)) = M.$$

However, given that M is also the MAC for the $(q+1)$-block padded message $D_1, D_2, \ldots, D_q, D_{q+1}$, again using the salt R, we also know that

$$e_{K_2 \oplus R}(e_{K_1}(D_{q+1} \oplus H_q)) = M.$$

Thus, since the block cipher encryption function must be one-to-one (for any fixed key), we know that

$$e_{K_1}(E^*) = e_{K_1}(D_{q+1} \oplus H_q),$$

hence

$$E^* = D_{q+1} \oplus H_q,$$

and thus

$$H_q = D_{q+1} \oplus E^*.$$

It remains for the attacker to learn a MAC for the q-block padded message $D_1, D_2, \ldots, D_q$, which requires one chosen MAC (the value of the random salt is not significant), and the attacker will now know a padded message $D_1, D_2, \ldots, D_q$, the value H_q, and a MAC (with its corresponding salt).

We conclude by considering the complexity of the above procedure. It requires

- one known MAC,
- 2^{n-1} MAC verifications (this is the expected number before the MAC verification succeeds), and
- one chosen MAC.

Thus, in this case (where $m = n$), the total complexity of the attack to find K_2 is either

- $[2^{k-s} + 2^{k-n}, 1, 2^s, 2^{n-1}]$, when the approach described in Section 3.1 is followed, or
- $[2^{k-s} + 2^{k-n}, 1, 0, 2^{s+n-1} + 2^{n-1}]$, when the method in Section 3.2 is employed.

In the case where two-key triple DES [13], [15] is being employed (with $k = 112$ and $n = 64$) this gives a complexity of either $[2^{112-s} + 2^{48}, 1, 2^s, 2^{63}]$ or $[2^{112-s} + 2^{48}, 1, 0, 2^{63} + 2^{63+s}]$. Hence, for example, if $s = 56$ the attack complexities are $[2^{56}, 1, 2^{56}, 2^{63}]$ and $[2^{56}, 1, 0, 2^{119}]$, and if $s = 20$ the complexities are $[2^{92}, 1, 2^{20}, 2^{63}]$ and $[2^{92}, 1, 0, 2^{83}]$.

More generally, where a block cipher with a 128-bit key and a 64-bit block size is used, e.g., IDEA [13] or MISTY1 [12], the attack complexity is either $[2^{128-s} + 2^{64}, 1, 2^s, 2^{63}]$ or $[2^{128-s} + 2^{64}, 1, 0, 2^{63} + 2^{63+s}]$. Putting $s = 64$ in the first formulation and

$s = 32$ in the second, we get attack complexities of $[2^{65}, 1, 2^{64}, 2^{63}]$ or $[2^{96}, 1, 0, 2^{95}]$ which are lower than might be expected.

Finally note that if AES [16] (with a 128-bit key) is used, then the attack complexity is either $[2^{128-s}, 1, 2^s, 2^{127}]$ or $[2^{128-s}, 1, 0, 2^{127} + 2^{127+s}]$. Hence, in this case, the attacks are unlikely to be of much practical significance.

3.4. *Another Method for Finding H_q*

The second method we describe for finding H_q applies regardless of the value of m, although it is less efficient than the previously described method when $m = n$.

The attacker first chooses a sequence of $\lceil n/m \rceil$ salts $R_1, R_2, \ldots, R_{\lceil n/m \rceil}$. The attacker next uses a series of MAC verifications to find a set S_1 of $2^{(n-m)/2}$ messages (each at least two n-bit blocks long), for which the m-bit MAC is equal to all zeros when the salt R_1 is used.

The attacker next uses a series of MAC verifications to learn the MACs of all the messages in S_1 for each of the other salts $R_2, R_3, \ldots, R_{\lceil n/m \rceil}$. For each message and for each salt the attacker submits each possible m-bit MAC in turn for a MAC verification until the correct MAC is found. As a result the attacker will know a sequence of $\lceil n/m \rceil$ MACs for each of the $2^{(n-m)/2}$ messages in S_1, and moreover the first MAC in each case will be all zeros.

The attacker now follows exactly the same procedure to generate a second set of messages S_2 of size $2^{(n-m)/2}$ (using exactly the same sequence of salt values $R_1, R_2, \ldots, R_{\lceil n/m \rceil}$). Each message in this set shall be precisely one block long. As a result the attacker will know a sequence of $\lceil n/m \rceil$ MACs for each of the $2^{(n-m)/2}$ one-block messages, again with the property that the first MAC in every case will equal all zeros.

By the usual birthday paradox arguments [13], with probability $\simeq 0.63$, for one padded message $D_1, D_2, \ldots, D_{q+1}$ in set S_1, the "partial MAC value" H_{q+1} will equal the value of $e_{K_1}(E)$ for a one-block message E in the set S_2. This will be evident because the corresponding tuples of MAC values will be equal. (There may be a small number of "false alarms", i.e., pairs of tuples of MAC values which agree, even though the partial MAC values disagree. These can be eliminated using a number of additional MAC verifications.)

Given such a pair of messages, the remainder of the procedure is exactly the same as in the previous section.

The complexity of the above process is simple to compute. A total of

$$2^{(n-m)/2} \times 2$$

messages need to be found which have a MAC of all zeros. Finding one such message will require 2^m MAC verifications on average, and hence finding the messages will require a total of

$$2^{(n+m+2)/2}$$

MAC verifications. A further

$$\lceil n/m - 1 \rceil \times 2 \times 2^{(n-m)/2}$$

MACs need to be computed. Computing such a MAC requires on average 2^{m-1} MAC verifications, and hence computing these MACs requires a total of

$$\lceil n/m - 1 \rceil \times 2^{(n+m)/2}$$

MAC verifications. This gives a total of

$$\lceil n/m + 1 \rceil \times 2^{(n+m)/2}$$

MAC verifications.

Thus, in this case, the total complexity of the attack to find K_2 is equal to either

- $[2^{k-s} + 2^{k-m}, 0, 2^s, \lceil n/m + 1 \rceil \times 2^{(n+m)/2}]$, when the approach described in Section 3.1 is followed, or
- $[2^{k-s} + 2^{k-m}, 0, 0, \lceil n/m + 1 \rceil \times 2^{(n+m)/2} + 2^{s+m-1}]$, when the method in Section 3.2 is employed.

We note three cases where this attack appears significant:

- Suppose two-key triple DES [15] is used with $m = 16$; the second attack variant with $s = 20$ gives an attack complexity of $[2^{96}, 0, 0, 2^{42}]$.
- If two-key triple DES is employed with $m = 32$ then the complexity of the second attack variant (with $s = 18$) is $[2^{94}, 0, 0, 2^{50}]$.
- Finally, if AES [16] (with a 128-bit key) is used with $m = 32$, then the second attack variant (with $s = 40$) has a complexity of $[2^{96}, 0, 0, 2^{82}]$.

3.5. *A Partial Key-Recovery Attack for the Case $m = n$*

In this section an attack is presented which, in some cases where $m = n$, is more efficient than the attacks of the previous sections. In the following, let $d_K(x)$ denote the decryption of x using the key K for the underlying block cipher.

The attack is based on multiple collisions.

Definition 1. A t-collision for a MAC is a set of t messages all producing the same MAC value.

We make use of the following lemma which is easily proved.

Lemma 2. *Let A, B, and C be boolean variables. Then*

$$\begin{aligned} A \Rightarrow B \quad &\Leftrightarrow \quad not(B) \Rightarrow not(A), \quad and \\ A \Rightarrow (B\ AND\ C) \quad &\Leftrightarrow \quad not(B)\ OR\ not(C) \Rightarrow not(A). \end{aligned}$$

Let D be some message (with an arbitrary number of blocks). The attack goes as follows. Request the MACs of D for s different values of the salt R. Assume that the attacker finds a t-collision, where the salts are $R_0, R_1, \ldots, R_{t-1}$ and denote the common MAC value by M'. For simplicity denote $K_2 + R_0$ by K, and $K_2 + R_i$ by $K + a_{i-1}$ for $i = 1, 2, \ldots, t - 1$. The attacker guesses a key value L and computes the decryptions of the MAC value M' using the keys $L, L + a_0, \ldots, L + a_{t-1}$. Then it

holds for $i = 0, 1, \ldots, t-1$, that if $L = K$ or $L = K + a_i$ then $d_L(M') = d_{L+a_i}(M')$. Using Lemma 2 it follows that if $d_L(M') \neq d_{L+a_i}(M')$ then $L \neq K$ and $L \neq K + a_i$ for $0 \leq i < t$. Similarly, if $d_{L+a_i}(M') \neq d_{L+a_j}(M')$ then $L \neq K + a_i + a_j$ for $0 \leq i \neq j < t$. In this way an exhaustive search for K_2 can be made faster than brute force.

In some rare cases one gets equal values in the inequality tests. As an example, if $d_L(M') = d_{L+a_i}(M')$ for some i, then one needs to check if $d_L(M') = d_{L+a_0}(M') = d_{L+a_1}(M') = \cdots$ after which all false alarms are expected to be detected. The expected number of false alarms is $t + \binom{t-1}{2}$.

We show the case of a 3-collision in more detail. Assume that the random salts used are R_0, R_1, and R_2 (which are known to the attacker). Since the messages are the same for all MACs and since the MACs are equal, say M', one knows that the keys $K_2 + R_0$, $K_2 + R_1$, and $K_2 + R_2$ all decrypt M' to the same (unknown) message z, thus

$$d_K(M') = d_{K+a_0}(M') = d_{K+a_1}(M'),$$

where $K = K_2 + R_0$, $a_0 = R_0 + R_1$, and $a_1 = R_0 + R_2$.

The following implications are immediate:

$$\begin{array}{lcll}
L = K & \Rightarrow & d_L(M') = d_{L+a_0}(M') & \text{AND } d_{L+a_0}(M') = d_{L+a_1}(M'), \\
L = K + a_0 & \Rightarrow & d_{L+a_0}(M') = d_L(M') & \text{AND } d_L(M') = d_{L+a_0+a_1}(M'), \\
L = K + a_1 & \Rightarrow & d_{L+a_1}(M') = d_{L+a_0+a_1}(M') & \text{AND } d_{L+a_1}(M') = d_L(M'), \\
L = K + a_0 + a_1 & \Rightarrow & d_{L+a_0+a_1}(M') = d_{L+a_1}(M') & \text{AND } d_{L+a_1}(M') = d_{L+a_0}(M').
\end{array}$$

Lemma 2 enables us to rewrite the above implications as follows:

$$\begin{array}{lcl}
d_L(M') \neq d_{L+a_0}(M') & \Rightarrow & L \neq K, \\
d_{L+a_0}(M') \neq d_L(M') & \Rightarrow & L \neq K + a_0, \\
d_{L+a_1}(M') \neq d_L(M') & \Rightarrow & L \neq K + a_1, \\
d_{L+a_1}(M') \neq d_{L+a_0}(M') & \Rightarrow & L \neq K + a_0 + a_1.
\end{array}$$

Take (guess) a key value, L, and compute $d_L(M')$, $d_{L+a_0}(M')$, and $d_{L+a_1}(M')$. If $d_L(M') \neq d_{L+a_0}(M')$, then $L \neq K$ and $L \neq K + a_0$, if $d_{L+a_0}(M') \neq d_{L+a_1}(M')$, then $L \neq K + a_0 + a_1$, and if $d_L(M') \neq d_{L+a_1}(M')$, then $L \neq K + a_1$.

Summing up, with a 3-collision (provided a_0, a_1 are different) one can check the values of four keys from three decryption operations.

Let us next assume that there is a 4-collision. Let the four keys in the 4-collision be K, $K + a_0$, $K + a_1$, $K + a_2$. Then from the results of $d_L(M')$, $d_{L+a_0}(M')$, $d_{L+a_1}(M')$, and $d_{L+a_2}(M')$, one can check the validity of four keys. Moreover, by arguments similar to the case of a 3-collision, from the four decryptions, one can check the values of all keys of the form $K + a_i + a_j$, where $0 \leq i \neq j \leq 2$. Thus from four decryption operations one can check

$$4 + \binom{3}{2} = 7$$

keys.

Table 1

t	$u = t + \binom{t-1}{2}$	u/t
3	4	1.3
4	7	1.8
5	11	2.2
6	16	2.7
7	22	3.1
8	29	3.6
9	37	4.1
10	46	4.6
17	136	8.0

This generalizes to the following result. With a t-collision one can check the values of

$$u = t + \binom{t-1}{2}$$

keys from t decryption operations. Table 1 lists values of t, u, and u/t. It should be clear that t-collisions can be used to reduce a search for the key K_2; the question is by how much? That is, how many values of L need to be tested before the sets of keys $\{L, L+a_0, \ldots, L+a_{t-1}, L+a_0+a_1, \ldots, L+a_{t-2}+a_{t-1}\}$ cover the entire key space?

Consider the case $t = 3$. One can assume $a_0 \neq a_1$ (otherwise there is no collision), and that with a high probability there are two bit positions where $a_0 \neq a_1$. Without loss of generality assume that these are the two most significant bits and that these bits are "01" for a_0 and "10" for a_1. Then a strategy is the following: Let L run through all keys where the most significant two bits are "00". Then clearly the sets

$$\{L, L + a_0, L + a_1, L + a_0 + a_1\}$$

cover the entire key space and an exhaustive search for K_2 is reduced by a factor of $\frac{4}{3}$, since in the attack one can check the value of four keys at the cost of three decryptions.

Consider the case $t = 4$. With a high probability the k-bit vectors a_0, a_1, and a_2 are pairwise different. Also, with a high probability there are three bit positions where a_0, a_1, and a_2 are linearly independent (viewed as three-bit vectors). Without loss of generality assume that the bits are the three most significant bits and that these are "001" for a_0, "010" for a_1 and "100" for a_2. Then a strategy is the following: Let L run through all keys where the most significant three bits are "000". Then clearly the sets

$$\{L, L + a_0, L + a_1, L + a_2, L + a_0 + a_1, L + a_0 + a_2, L + a_1 + a_2\}$$

cover seven-eights of the key space. Next fix the most significant three bits of L to "111", find other bit positions where a_0, a_1, and a_2 are different and repeat the strategy. Thus, in the first phase of the attack one chooses 2^{k-3} values of L, does $4 \times 2^{k-3} = 2^{k-1}$ encryptions, and checks $7 \times 2^{k-3}$ keys. In the next phase of the attack one chooses 2^{k-6} values of L, does $4 \times 2^{k-6} = 2^{k-4}$ encryptions, and checks $7 \times 2^{k-6}$ keys. At this point,

a total of $7 \times 2^{k-3} + 7 \times 2^{k-6} = 2^k - 2^{k-3} - 2^{k-6}$ keys have been checked at the cost of about $2^{k-1} + 2^{k-4}$ encryptions. In total, an exhaustive search for K_2 is reduced by a factor of almost two.

For higher values of t the attacker's strategy becomes more complex. We claim that with a high probability ("good" values of a_i) the factor saved in an exhaustive search for the key is close to the value of u/t (see Table 1).

The following result shows the complexity of finding t-collisions [22].

Lemma 3. *Consider a set of s randomly chosen b-bit values. With $s = c2^{(t-1)b/t}$ one expects to get one t-collision, where $c \approx (t!)^{1/t}$.*

If it is assumed for a fixed message D and a (randomly chosen) salt R that the resulting MAC is a random m-bit value, one can apply the lemma to estimate the number of texts needed to find a t-collision.

Consider a few examples. With $s = 2^{(n+1)/2}$ one expects to get one pair of colliding MACs, that is, one (2-)collision. With $s = (1.8)2^{2n/3}$ one expects to get a 3-collision, that is, three MACs with equal values ($6^{1/3} \approx 1.8$). With $s = (2.2)2^{3n/4}$ one expects to get one 4-collision ($24^{1/4} \approx 2.2$).

From Stirling's formula $n! = \sqrt{2\pi n}(n/e)^n(1 + \Theta(1/n))$, one gets that $(t!)^{1/t} \approx t/e$ for large t, where e is the base of the natural logarithm. Thus, with $s \approx (t/e)2^{(t-1)n/t}$ one expects to get a t-collision.

The total complexity of the attack to find K_2 is equal to

- $[2^{k-1}/(u/t),\ 0,\ (t/e)2^{(t-1)nt},\ 0]$, where

$$u = t + \binom{t-1}{2}.$$

The attack is therefore more efficient that those of the previous sections in the cases where $n = m = k$. As an example, using AES [16], $n = m = k = 128$, with $t = 17$, the complexity of the attack is $[2^{124}, 0, 0, 2^{123}]$.

As a final remark, note that the message D in the attack need not be chosen nor known by the attacker. Therefore one can argue that this attack is stronger than a traditional "chosen-text" attack.

4. The NIST RMAC draft proposal

In this section we comment on the implementation of RMAC as proposed by NIST [17] on November 5, 2002.

In Appendix A of [17] it is noted that for RMAC with two independent keys K_1 and K_2 an exhaustive search for the keys is expected to require the generation of 2^{2k-1} MACs, where k is the size of one key. However, for the cases with $m = n$ this can be done much faster under a chosen message attack with just one known message and one chosen message. Independently of how the two keys are generated, an exhaustive search for the key K_2 requires only an expected number of 2^k decryptions of the block cipher. Given a message D and the MAC using the salt R, request the MAC of D again. With a high probability this MAC is computed with a salt R', such that $R' \neq R$. For these

two MACs, the values just before the final encryption will be equal and K_2 can be found after about 2^k decryption operations. Subsequently, K_1 can be found in roughly the same time.

4.1. *Partial Key-Recovery Attack for RMAC Used with 3DES*

One of the block cipher algorithms approved to be used in RMAC is triple-DES with 168-bit keys. Consider RMAC using triple-DES with $n = m = 64$ using a 64-bit salt R (which is one option in [17]). The key for the final encryption is then $K_3 = K_2 + (R \mid 0^{104})$. However, it is not specified in [17] how the three DES keys are derived from K_3.

Assume that the first DES key is taken as the rightmost 56 bits of $K_2 + (R \mid 0^{104})$, the second DES key as the middle 56 bits, and the third DES key as the leftmost 56 bits. Assume an attacker is given two MACs of the same message D but using two different values, R and R', of the salt. Assume that the rightmost eight bits of both R and R' are equal. Then the encryption of the last same block for the two MACs is done using triple-DES where for one MAC the key used is (a, b, c), and where for the other MAC the key used is $(a, b, c \oplus d)$. Since the attacker knows d, he can decrypt through a single DES operation, find c in the expected time of 2^{56} operations, and derive one of the three DES keys [8].

This attack has a probability of success of 2^{-8}. If the attack fails, it is repeated for other values of D, R, and/or R'. With 2^8 iterations one finds the single DES key with high probability. After the third DES key has been found, it is possible to find the second DES key with similar complexity. Note that eight bits of the salt affect the second DES key. Request the MAC of a message D_2 using two different values of the salt. Decrypt through the final DES component with the third DES key. With a probability of $1 - 2^{-8}$ the two second DES keys in the final encryption will be different as a result of different salt values. Since the salts are known by the attacker, one finds the second DES in the expected time of 2^{56} operations. Thus, the attack which finds two of three DES keys has complexity $[2^{64}, 2^8, 2^8, 0]$.

Subsequently, the final DES key can be found using 2^{56} MAC verifications as follows. Assume one is given the MACs, M_1 and M_2, of two different messages D_1 and D_2, each consisting of an arbitrary number of 64-bit blocks. Request the MAC, M_3, of the message $D_1 \mid E$, where E is a one-block message. Let x_1, x_2, and x_3 be the values just before the final triple DES encryptions in the computations of M_1, M_2, and M_3. Given the value of the final single-DES key of K_2 one can compute also the MAC of the message $D_2 \mid (E \oplus x_1 \oplus x_2)$. Note that the value just before the final triple DES encryptions in this case is x_3. Also note that the attacker has full control over the key bits which are modified using the (random) salts. Therefore this last part of the attack works regardless of how the salts are chosen, as long as the attacker knows them.

In total the complexity of the attack which finds all of K_2 (i.e., all three DES keys) is $[2^{64}, 2^8, 2^8, 2^{56}]$, which is much less than expected.

5. Conclusion

In this paper several partial key recovery attacks on RMAC are presented. Apart from the attack on RMAC used with triple DES as proposed by NIST, none of the attacks is

likely to be a major concern in practice; however, it is important to have an accurate idea of the precise level of security offered by any scheme, and the attacks described help provide more accurate upper bounds on the security level of RMAC. Moreover, since security proofs tend to focus on the security of the entire key, the existence of partial key recovery attacks such as those described here may not be revealed by theoretical results.

Since this type of attack does not reveal the entire key, it cannot be claimed that these attacks threaten the security of the scheme. However, apart from the fact that determining part of the key rather more easily than the rest is a rather disconcerting property for any cryptographic scheme to have, two further possible ramifications are as follows:

- First, note that in cases where both the RMAC keys are derived from a single key, this attack suggests that it is extremely important to ensure that knowledge of one of the two keys does not threaten the secrecy of the other key. Fortunately, the scheme proposed in the NIST draft standard [17] derives the two keys by using the single key to encrypt different fixed strings. In such a case, knowledge of one of the keys does not pose a major threat to the secrecy of the other key.
- Second, once the second of the two RMAC keys has been compromised, a huge variety of forgery attacks becomes possible. Essentially, the scheme becomes equivalent to the simplest of CBC-MACs, where only a single block cipher key is used and where no special processing is applied to the final message block.

Acknowledgments

The authors thank Tadayoshi Kohno and an anonymous referee for helpful discussions.

References

[1] ANSI X9.19. *Financial Institution Retail Message Authentication*. American Bankers Association, August 13, 1986.

[2] M. Bellare, J. Kilian, and P. Rogaway. The security of cipher block chaining. In Y.G. Desmedt, editor, *Advances in Cryptology - CRYPTO* '94, pages 341–358. LNCS 839. Springer-Verlag, Berlin, 1994.

[3] D. Coppersmith, L.R. Knudsen, and C.J. Mitchell. Key recovery and forgery attacks on the MacDES MAC algorithm. In M. Bellare editor, *Advances in Cryptology - CRYPTO* 2000, pages 184–196. LNCS 1880. Springer-Verlag, Berlin, 2000.

[4] D. Coppersmith and C.J. Mitchell, Attacks on MacDES MAC algorithm. *Electronics Letters*, vol. 35, no. 19 (1999), pp. 1626–1627.

[5] International Organization for Standardization. *ISO/IEC* 9797–1, *Information Technology—Security Techniques—Message Authentication Codes* (*MACs*)*—Part* 1: *Mechanisms Using a Block Cipher*. 105 Genève, Switzerland 1999.

[6] E. Jaulmes, A. Joux, and F. Valette. On the security of randomized CBC-MAC beyond the birthday paradox limit: a new construction. In J. Daemen and V. Rijmen, editors, *Fast Software Encryption* 2002, pages 237–251. LNCS 2365. Springer-Verlag, Berlin, 2002.

[7] A. Joux, G. Poupard and J. Stern. New attacks against standardized MACs. In T. Johansson, editor, *Fast Software Encryption*, 10*th International Workshop*, *FSE* 2003, *Lund*, *Sweden*, *February* 2003, *Revised Papers*, pages 170–181. LNCS 2887. Springer-Verlag, Berlin, 2003.

[8] J. Kelsey, B. Schneier, and D. Wagner. Key-schedule cryptanalysis of IDEA, G-DES, GOST, SAFER, and triple-DES. In Neal Koblitz, editor, *Advances in Cryptology*: *CRYPTO* '96, pages 237–251. LNCS 1109. Springer-Verlag, Berlin, 1996.

[9] L.R. Knudsen and T. Kohno. An analysis of RMAC. In T. Johansson, editor, *Fast Software Encryption*, 10*th International Workshop*, *FSE* 2003, *Lund*, *Sweden*, *February* 2003, *Revised Papers*, pages 182–191. LNCS 2887. Springer-Verlag, Berlin, 2003.

[10] L.R. Knudsen, and C.J. Mitchell. Analysis of 3gpp-MAC and two-key 3gpp-MAC. *Discrete App. Math.*, vol. 128, no. 1 (2003), pp. 181–191.

[11] L.R. Knudsen and B. Preneel. MacDES: a new MAC algorithm based on DES. *Electron. Lett.*, vol. 34, no. 9 (April 1998), pp. 871–873.

[12] M. Matsui. New block encryption algorithm MISTY. In E. Biham, editor, *Fast Software Encryption*, 4*th International Workshop*, *FSE '*97, *Haifa*, *Israel*, *January* 20–22, 1997, *Proceedings*, pages 54–68. LNCS 1267. Springer-Verlag, Berlin, 1997.

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

[14] C.J. Mitchell, Key recovery attack on ANSI retail MAC. *Electron. Lett.*, vol. 39, no. 14 (2003), pp. 361–362.

[15] National Institute of Standards and Technology. *Federal Information Processing Standards Publication* 46-3 (*FIPS PUB* 46-3): *Data Encryption Standard* (*DES*). NIST, Gaithersburg, MD, October 1999.

[16] National Institute of Standards and Technology. *Federal Information Processing Standards Publication* 197 (*FIPS PUB* 197): *Specification for the Advanced Encryption Standard* (*AES*). NIST, Gaithersburg, MD, November 2001.

[17] National Institute of Standards and Technology. *NIST Special Publication* 800-38*B*, *Draft Recommendation for Block Cipher Modes of Operation*: *the RMAC Authentication Mode*. NIST, Gaithersburg, MD, November 2002.

[18] E. Petrank and C. Rackoff. CBC MAC for real-time data sources. *J. Cryptology*, vol. 13, no. 3 (2000), pp. 315–338.

[19] B. Preneel and P.C. van Oorschot, A key recovery attack on the ANSI X9.19 retail MAC, *Electron. Lett.*, vol. 32, no. 17 (1996), pp. 1568–1569.

[20] B. Preneel and P.C. van Oorschot. On the security of iterated message authentication codes. *IEEE Trans. Inform. Theory*, vol. 45, no. 1 (1999), 188–199.

[21] RIPE. A. Bosselaers and B. Preneel, editors, *Integrity Primitives for Secure Information Systems*. *Final Report of RACE Integrity Primitives Evaluation* (*RIPE-RACE* 1040). LNCS 1007. Springer-Verlag, Berlin, 1995.

[22] R. Rivest and A. Shamir. Payword and Micromint: two simple micropayment schemes. *Cryptobytes*, vol. 2, no. 1 (1996), pp. 7–11.