Analysis of RMAC

Lars R. Knudsen
DTU, Denmark

November 10, 2002

Abstract

In this paper the newly proposed RMAC system is analysed and a generic attack is presented. The attack can be used to find one of the two keys in the system faster than by an exhaustive search. Also, a serious attack on RMAC used with triple-DES is presented.

1 Introduction

RMAC [1] is an authentication system based on a block cipher. The block cipher algorithms currently approved to be used in RMAC are the AES and triple-DES.

RMAC is based on a block cipher with b-bit blocks and k-bit keys. RMAC takes as inputs: a message M of an arbitrary number of bits, two keys $K1, K2$ each of k bits and a salt R of r bits, where $r \leq k$. It produces an m-bit MAC value, where $m \leq b$. The method is as follows. Encrypt M using the block cipher in CBC mode using the key $K1$. The last ciphertext block is then encrypted with the key $K3 = K2 + R$ where '+' is addition modulo 2. The resulting ciphertext is then truncated to m bits to form the MAC. The two keys $K1, K2$ may be generated from one k-bit key in a standard way [1].

There are five parameter sets in [1] for each of two block sizes.

Parameter Set	$b = 128$ (r, m)	$b = 64$ (r, m)
I	$(0, 32)$	$(0, 32)$
II	$(0, 64)$	$(64, 64)$
III	$(16, 80)$	n/a
IV	$(64, 96)$	n/a
V	$(128, 128)$	n/a

In Appendix A of [1] it is noted that for RMAC with two independent keys $K1$ and $K2$ 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. For RMAC with parameter sets II and V, however, 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 $K2$ requires only an expected number of 2^k decryptions of the block cipher [2]. Given a message M and the MAC using the salt R, request the MAC of M 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 $K2$ can be found after about 2^k decryption operations. Subsequently, $K1$ can be found in roughly the same time.

The rest of this paper is organised as follows. In §2 an attack on RMAC used with three-key triple-DES is presented. The attack finds all three DES keys in time roughly that of three times an exhaustive search for a DES key using only a few MACs. §3 presents an attack on RMAC used with any block cipher. The attack finds one of the two keys in the system faster than by an exhaustive search.

2 Attack on RMAC with three-key triple DES

One of the block cipher algorithms approved to be used in RMAC is triple-DES with 168-bit keys. Consider RMAC with parameter set II, that is with 64-bit MACs and a 64-bit salt. The key for the final encryption is then $K3 = K2 + (R \mid 0^{104})$. However, it is not specified in [1] how the three DES keys are derived from $K3$. Assume that the first DES key is taken as the rightmost 56 bits of $K2 + (R \mid 0^{104})$, the second DES as the middle 56 bits, and the third DES as the leftmost 56 bits. Assume an attacker is given two MACs of the same message M but using two different values, R and R' of the salt. Assume that the rightmost eight bits of both R and R' are zeros. 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 2^{56} operations and derive one of the three DES keys. This attack has a probability of success of 2^{-16}. 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 M using two different values of the salt. Decrypt through the final DES component with the third DES key. With a high probability 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 about 2^{56} operations. Subsequently, the final DES key can be found in about 2^{56} operations and the scheme is broken.

3 A generic attack

In this section we present an attack on the RMAC system with parameter set II for $b = 64$ and RMAC with parameter set V for $b = 128$. The attack finds the value of $K2$ after which RMAC reduces to a simple CBC-MAC for which it is well-known that simple forgeries can be found. 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 shall make use of the following lemma which is easily proved.

Lemma 1 *Let A, B, and C be boolean variables. Then*

$$A \Rightarrow B \quad \Leftrightarrow \quad not(B) \Rightarrow not(A),$$

and

$$A \Rightarrow (B \ AND \ C) \quad \Leftrightarrow \quad not(B) \ OR \ not(C) \Rightarrow not(A).$$

Let D be some message (with an arbitrary no. of blocks). Then the MAC of D, $\mathrm{MAC}_{K1,K2}(D, R)$, is the last block from the CBC-encryption using $K1$, encrypted once again using the key $K2 + R$, where R is the salt. 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 $R_0, \ldots, R_{t-1}$ and denote the common MAC value by M'. For simplicity denote $K2 + R_0$ by K, and $K2 + R_i$ by $K + a_{i-1}$ for $i = 1, \ldots, t-1$. 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 $i = 0, \ldots, t-1$, that if $L = K$ or $L = K + a_i$ then $d_L(M') = d_{L+a_i}(M')$. Using Lemma 1 one gets 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 $K2$ 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 $d_L(M') = d_{L+a_0}(M') = d_{L+a_1}(M') = ...$ after which all false alarms

are expected to be detected. The expected number of false alarms is $t + \binom{t-1}{2}$.

Let us show the case of a 3-collision in more details. Assume that the random numbers, the 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 $K2 + R_0, K2 + R_1$, and $K2 + R_2$ all decrypt M' to the same (unknown) message z, thus

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

where $K = K2 + R_0$, $a_0 = R_0 + R_1$ 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 1 enables us to rewrite the above implications as follows.

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

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 generalises 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 $K2$, one question is by how much. 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 $K2$ 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 b-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 7/8 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^{b-3} values of L, does $4 \times 2^{b-3} = 2^{b-1}$ encryptions, and one can check $7 \times 2^{b-3}$ keys. In the next phase of the attack one chooses 2^{b-6} values of L, does $4 \times 2^{b-6} = 2^{b-4}$ encryptions, and one can check $7 \times 2^{b-6}$ keys. At this point, a total of $7 \times 2^{b-3} + 7 \times 2^{b-6} = 2^b - 2^{b-3} - 2^{b-6}$ keys have been checked at the cost of about $2^{b-1} + 2^{b-4}$ encryptions. In total, an exhaustive search for $K2$ 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 [3].

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

From Stirling's formula $n! = \sqrt{2\pi n}(n/e)^n(1 + \Theta(\frac{1}{n}))$, one gets that $(t!)^{1/t} \approx t/e$ for large t. Thus, with $s = (t/e)2^{(t-1)b/t}$ one expects to get a t-collision. Table 2 lists the complexities of finding t-collisions depending on the block size b.

There are many variants of this attack depending on how many chosen texts the attacker has access to. Table 3 lists the complexities of some instantiations of the attacks, where for triple-DES the number of chosen texts has been chosen to be less than 2^{64} (since the salt can be a maximum of 64 bits) and for AES the time complexity and the number of chosen texts needed have been made comparable. In both cases an exhaustive search for the key has been reduced by a factor of eight, so the correct value of the key can be expected trying half of that number of values. 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.

Table 2: The estimated number of texts needed to find a t-collision.

t	#texts needed	
	$b = 64$	$b = 128$
3	2^{44}	2^{86}
4	2^{49}	2^{97}
5	2^{53}	2^{104}
6	2^{55}	2^{108}
7	2^{57}	2^{112}
8	2^{58}	2^{114}
9	2^{59}	2^{116}
10	2^{60}	2^{118}
17	2^{63}	2^{123}

Table 3: Expected running times and chosen texts of attacks finding $K2$ of RMAC.

Algorithm	k	b	Parameter sets	t	Expected running time	# chosen texts
3-DES	112	64	II	12	2^{108}	2^{63}
AES	128	128	V	20	2^{124}	2^{123}

References

[1] NIST. DRAFT Recommendation for Block Cipher Modes of Operation: the RMAC Authentication Mode. NIST Special Publication 800-38B. October 18, 2002.

[2] Chris Mitchell. Private communication.

[3] R. Rivest and A. Shamir. Payword and Micromint: Two simple micropayment schemes. *Cryptobytes*, 2(1):7–11, 1996.