RMAC
A randomized MAC beyond the birthday paradox limit

Éliane Jaulmes, Antoine Joux and Frédéric Valette

DCSSI Crypto Lab
18, rue du Dr. Zamenhof
F-92131 Issy-Les-Moulineaux.
Tel: (33) 1.41.46.37.20
Fax: (33) 1.41.46.37.01
email: `eliane.jaulmes@wanadoo.fr`
`Antoine.Joux@ens.f`
`fred.valette@wanadoo.f`

Abstract. RMAC is a randomized message authentication code based on the well known CBC–MAC construction. It is provably secure against birthday paradox attacks. The size of the MAC tags in this construction is optimal, i.e. exactly twice the size of the block cipher. Moreover, this construction adds a negligible computational overhead compared to the cost of a plain, non-randomized CBC–MACs.

RMAC
A randomized MAC beyond the birthday paradox limit

Abstract. **RMAC** is a randomized message authentication code based on the well known CBC–MAC construction. It is provably secure against birthday paradox attacks. The size of the MAC tags in this construction is optimal, i.e. exactly twice the size of the block cipher. Moreover, this construction adds a negligible computational overhead compared to the cost of a plain, non-randomized CBC–MACs.

1 Mode specification

1.1 Description of RMAC

The mode of operation **RMAC** computes a message authentication code using a block cipher algorithm. It is a randomized message authentication code, meaning that the computation of the MAC requires the generation of a random value. The construction is based on the CBC–MAC construction. We differentiate in **RMAC** two sligthly different modes, based on different padding of the messages. The notations used in this paper are the following:

- n is the size of blocks,
- k is the size of the key of the block cipher algorithm,
- M represents the message to authenticate,
- $\mathbf{AES}_K(B)$ represents the **AES** enciphering of the block B with the key K.

The size of the blocks is $n = 128$ bits, the size of the key of the block cipher may be $k = 128$, 192 or 256 bits.

The figure 1 presents the diagram of the **RMAC**. The diagram is the same for both modes of operation.

The message M to be authenticated does not always has a size multiple of the block length. However in order to apply the CBC construction, we need to work on complete blocks. Thus an uncomplete message M must be padded. The two modes of **RMAC** differ on the padding method.

Mode 1. The first mode of **RMAC** deals with full padded messages. It means that, before processing, every message is padded, even messages whose length is a multiple of the block size. A simple and frequently encountered padding method is to append a '1' at the end of the message followed by enough '0's to fill the last block. Note that in cases where the size of the message is already a multiple of the block size, the padding implies appending a full block to the

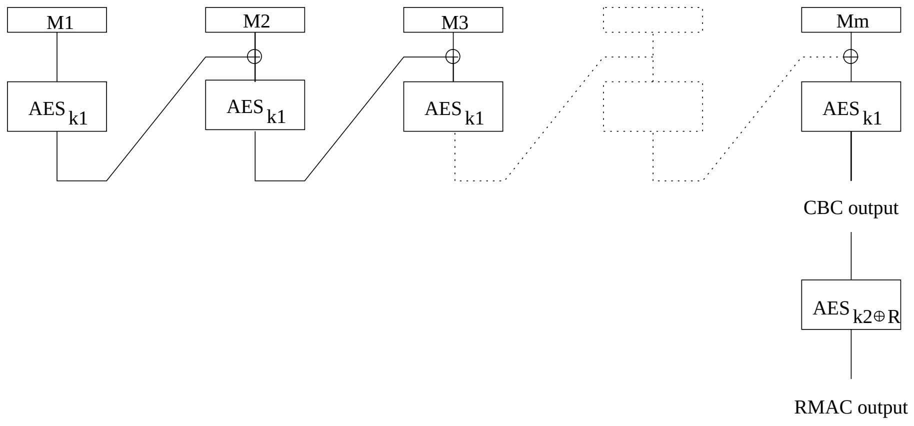

Fig. 1. Diagram of RMAC

message. The table 1 presents this first mode of operation. To obtain the MAC of a padded message, let us first compute the AES-CBC-MAC of the padded message M with a key K_1, then encipher the result with the **AES** algorithm using key $K_2 \oplus R$ where R is a n-bit random value. Note that the size of R depends on the size of blocks and not on the size of the key. If the key is longer than a block, R is padded with '0's before computing the xor with K_2. The MAC of the message is formed by the couple (m, R) where R is the chosen random value and m is the output of the last **AES**.

Input	Message $M = (M_1, M_2, \cdots, M_m)$
	Key $K = (K_1, K_2)$
Computation	
	$C_0 = 0^n,$
	$C_i = \mathbf{AES}_{K_1}(M_i \oplus C_{i-1})$ for i in $1 \cdots m$
	$B = \mathbf{AES}_{K_2 \oplus R}(C_m)$
Output	$\mathbf{RMAC}(M) = (B, R)$

Table 1. RMAC1 computation

To verify the MAC (B, R) of a given message M, let us first compute the AES-CBC-MAC of the padded message with the key K_1, then use R to compute $K_2 \oplus R$ and apply **AES** to the CBC output with this new key. The MAC is valid if the computed value is the same as the given B.

Mode 2. The second mode of **RMAC** avoids the padding of messages whose length is a multiple of the block size. In other cases, it fills the last block of the message with a '1' and as many '0's as necessary. To obtain the MAC of a padded message, let first compute the AES-CBC-MAC of the padded message M with a key K_1, then encipher the result with a last **AES** using key $K_2 \oplus R$ where R is a $(n+1)$-bit integer. The first n bits of R are randomly chosen and the last bit is equal to '0' if the message has been padded and to '1' otherwise. The MAC of the message is formed by the output of the last **AES** and by the first n bits of R. Note that in order to use this mode, one needs a key length of at least $n+1$ bits.

1.2 Security proof

In this section, we will briefly state the security of the RMAC construction. The security is the same for the two modes we have described. A detailed proof of the security analysis can be found in [2].

Let us consider an attacker asks for MAC computations and MAC verifications to an oracle. The total length of the queries transmitted to the oracle will be denoted by L and t is the total number of AES computations done by both the adversary and the oracle. The the advantage **Adv** of the attacker, i.e. the probability to forge a MAC, is bounded as follows:

$$\mathbf{Adv} \leq \frac{517L + t}{2^{128}}.$$

1.3 Important properties

The **RMAC** construction we propose gives an efficient solution to the problem of constructing a randomized CBC–MAC provably secure against birthday paradox attacks. The only previously known example of a birthday paradox resistant MAC was given in [1] and called MACRX. Compared to MACRX, **RMAC** has two main advantages. Firstly, its output has twice the length of the underlying block-cipher instead of three times for MACRX. Secondly, being a CBC–MAC variant, **RMAC** does not require any special function other than the block cipher.

Moreover, since **RMAC** provides a security in $O(2^n)$ for an algorithm using n-bit blocks, the security offered by a 128-bit block algorithm like AES is sufficient to make the need for 256-bit block cipher a very remote perspective.

2 Summary of properties

2.1 RMAC mode 1

Security function	Authentication
Error propagation	Infinite
Parallelizability	Sequential
Keying Material Requirements	2 keys
Counter/IV/Nonce Requirement	A random value of 1 block
Memory Requirements	None
Preprocessing capability	None
Message Length Requirements	Padding necessary
MAC tag size	2 blocks

2.2 RMAC mode 2

Security function	Authentication
Error propagation	Infinite
Parallelizability	Sequential
Keying Material Requirements	2 keys
Counter/IV/Nonce Requirement	A random value of 1 block
Memory Requirements	None
Preprocessing capability	None
Message Length Requirements	Arbitrary length
MAC tag size	2 blocks
Other requirements	The size of the key must be strictly greater than the size of the blocks

3 Test vectors

We have generated three test vectors for the three sizes of keys of the AES block cipher. Those test vectors are for RMAC used in mode 1. We are using the following notations: `MSG` represents the initial unpadded message. `PT` represents the plaintext fed to the CBC cipher, that is the padded message. `K1` is the key K_1 used during the CBC computation. `CBC` represents all the intermediary outputs C_i. `K2` is the key K_2. `R` is the random value. `K3` is $K_2 \oplus R$, that is the key used in the last encryption. Finally, `MAC` represents the output of RMAC.

```
KEYSIZE=128

MSG 000102030405060708090A0B0C0D0E0F 101112131415161718191A1B1C1D
PT  000102030405060708090A0B0C0D0E0F 101112131415161718191A1B1C18000
K1  000102030405060708090A0B0C0D0E0F
CBC 0A940BB5416EF045F1C39458C653EA5A 3C799ACECB066248FA06F6502D4EAF5A
K2  0F0E0D0C0B0A09080706050403020100
```

```
R   00020406080A0C0E10121416181A1C1E
K3  0F0C090A0300050617141112 1B181D1E
MAC E4CD62BD8824DDF33AB0C33DB3217BBB 00020406080A0C0E10121416181A1C1E

KEYSIZE=192

MSG 000102030405060708090A0B0C0D0E0F 101112131415161718191A1B1C1D
PT  000102030405060708090A0B0C0D0E0F 101112131415161718191A1B1C1D8000
K1  000102030405060708090A0B0C0D0E0F1011121314151617
CBC 0060BFFE46834BB8DA5CF9A61FF220AE 815CFD8CC0B2BA9FD9A195D742EE1388
K2  0F0E0D0C0B0A09080706050403020100FFFEFDFCFBFAF9F8
R   00020406080A0C0E10121416181A1C1E
K3  0F0C090A0300050617141112 1B181D1EFFFEFDFCFBFAF9F8
MAC 07B4CB1278AB823DC881ECE3488F3B28 00020406080A0C0E10121416181A1C1E

KEYSIZE=256

MSG 000102030405060708090A0B0C0D0E0F 101112131415161718191A1B1C1D
PT  000102030405060708090A0B0C0D0E0F 101112131415161718191A1B1C1D8000
K1  000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F
CBC 5A6E045708FB7196F02E553D02C3A692 80D19F4D978DCD5D0DFB41354BCAA493
K2  0F0E0D0C0B0A09080706050403020100FFFEFDFCFBFAF9F8F7F6F5F4F3F2F1F0
R   00020406080A0C0E10121416181A1C1E
K3  0F0C090A0300050617141112 1B181D1EFFFEFDFCFBFAF9F8F7F6F5F4F3F2F1F0
MAC 492AA4DAD27685658FB1539B25C1C71B 00020406080A0C0E10121416181A1C1E
```

References

1. M. Bellare, O. Goldreich, and H. Krawczyk. Stateless evaluation of pseudorandom functions: Security beyond the birthday barrier. In M. Wiener, editor, *Advances in Cryptology — CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 270–287. Springer, 1999.
2. E. Jaulmes, A. Joux, and F. Valette. On the security of randomized CBC–MAC beyond the birthday paradox limit. a new construction, 2001. Submitted. Available on request.