On the Security of Randomized CBC–MAC Beyond the Birthday Paradox Limit A New Construction

Éliane Jaulmes, Antoine Joux and Frédéric Valette

DCSSI Crypto Lab
18, rue du Dr. Zamenhof
F-92131 Issy-Les-Moulineaux.
email: eliane.jaulmes@wanadoo.fr
antoine.joux@m4x.org
fred.valette@wanadoo.fr

Abstract. In this paper, we study the security of randomized CBC–MACs and propose a new construction that resists birthday paradox attacks and provably reaches full security. The size of the MAC tags in this construction is optimal, i.e., exactly twice the size of the block cipher. Up to a constant, the security of the proposed randomized CBC–MAC using an n–bit block cipher is the same as the security of the usual encrypted CBC–MAC using a $2n$–bit block cipher. Moreover, this construction adds a negligible computational overhead compared to the cost of a plain, non-randomized CBC–MAC. We give a full standard proof of our construction using one pass of a block-cipher with $2n$-bit keys but there also is a proof for n-bit keys block-ciphers in the random oracle model.

1 Introduction

The message authentication code (MAC) is a well-known and widely used cryptographic primitive whose goal is to authenticate messages and to check their integrity in a secret key setting. For historical and efficiency reasons, MACs are often based on block ciphers. Of course, other constructions are possible. A well-known method to build MACs is for example to start from a hash function and transform it into a secure MAC. The idea first appeared in the work of Wegman and Carter [15]. Other existing constructions are for example XOR-MACs [3], HMAC [1] and UMAC [6]. However, in low end cryptographic devices, the ability to reuse an existing primitive is an extremely nice property. In practice, a simple construction called CBC–MAC is frequently encountered. Several variants of the CBC–MAC are described in normative documents [9, 14]. The simplest of those works as follows: let E be a block cipher using a key K to encrypt n–bit blocks. To compute the CBC–MAC of the message M with the key K, we split M into a sequence of n–bit blocks $M_1, \ldots, M_l$ and compute

$$\begin{aligned} &C_0 = 0^n, \\ &C_i = E_K(M_i \oplus C_{i-1}) \text{ for } i \text{ in } 1 \cdots l. \end{aligned}$$

After this computation, the value of the CBC–MAC is $\mathbf{CBC}_{E_K}(M) = C_l$. Note that the length of the message M has to be a multiple of the block size n, however several padding techniques have been proposed to remove this constraint [9].

This simple CBC–MAC has been proved secure in [4] for messages of fixed (non zero) length. However, when the length is no longer fixed, forgery attacks exist. The simplest of those uses two messages of one block each M and M', and queries their MACs C and C'. Then it can forge the MAC of $M \| (M' \oplus C)$, namely C'.

In order to remove this limitation, it is shown in [11] and [7] that it suffices to encrypt the plain CBC–MAC of a message with another key. However, the security level offered by these CBC–MACs is not optimal, since they all suffer from a common weakness: birthday paradox based attacks. In fact, all iterated MACs suffer from this kind of attacks, as has been shown in [12]. The basic idea beyond the birthday paradox attacks is to find two different messages with the same MAC value. Due to the birthday paradox, this search can be done in $2^{n/2}$ MAC computations, where n is the size of the MAC tag. Then one just need to append any fixed string to these messages and the MAC values of the extended messages are again the same. Thus forgery is easy since it suffices to query the MAC of one of the extended messages and use it as a forged MAC of the other extended message.

In order to protect MACs from birthday paradox attacks, it is suggested in [9] to add to each message a unique identifier, leading to a stateful MAC, or some kind of randomization, leading to a randomized MAC. These ideas have been studied in deeper details by some recent papers. In [3], a stateful construction based on XOR-MAC is given. It turns out that this leads to a reasonably simple and efficient construction. However, this approach has a major drawback, since it forces the MAC generation device to maintain an internal state from one generation to the next, which is extremely inconvenient when several MAC generation devices share the same key. On the other hand, randomization is much easier to deal with in practice. However, building a randomized MAC provably secure against birthday paradox attacks is not a simple matter. Indeed, the best currently known solution, called MACRX [2], is not CBC–MAC based and it expands the size of the MAC values by a factor of 3 instead of the expected 2. Indeed since with a MAC of size kn, an adversary can always obtain collisions in $2^{\frac{kn}{2}}$, in order to have a security against birthday paradox, the size of the MAC must be at least $2n$ bits. So MACRX is not optimal. Moreover it proposes to use hash functions and requires the use of a pseudorandom function family which itself is secure beyond the birthday paradox limit. Up to now, the only known solutions for designing pseudorandom functions with security beyond the birthday limit using block-ciphers (pseudorandom permutations) are counter-based [5, 8]. Moreover, it was shown in [13] that the simple and arguably reasonable approach of adding a random value at the beginning of a message before computing its CBC–MAC does not give full security. Indeed, this construction suffers from the so-called L-collision attack and forgery is possible after $2^{\alpha n}$ queries, where $\alpha = 2/3$.

Our paper is organized as follows. In section 2 we recall the standard deterministic CBC–MAC algorithm, **DMAC**, and explain how we construct our randomized CBC-MAC **RMAC** from **DMAC**. We also present our security model and recall a few notations. The section 3 contains the theorems stating the security of our construction as well as some sketches of proof. In section 4 we show how to instantiate our construction with a block-cipher. Two proofs are given. The first one is in the standard model but make use of block-ciphers with $2n$-bit keys, the second one is in the random oracle model and uses only block-ciphers with n-bit keys. Then section 5 proposes a detailed instantiation using the AES block-cipher and we conclude in section 6.

2 Preliminaries

2.1 Standard Deterministic CBC–MAC

According to [11] and [7], we know that encrypted CBC–MAC has a security level of $O(2^{n/2})$. In particular, in [11] the security of a CBC–MAC named **DMAC** is analyzed. We briefly recall the definition of **DMAC**. Given two random permutations f_1 and f_2 on n bits, $\mathbf{DMAC}_{f_1,f_2}$ is defined on messages whose length is a multiple of n. Given $M = (M_1, M_2, \cdots, M_l)$, we compute:

$$\begin{aligned} C_0 &= 0^n, \\ C_i &= f_1(M_i \oplus C_{i-1}) \ \text{ for } i \text{ in } 1 \cdots l, \\ \mathbf{CBC}_{f_1}(M) &= C_l \\ \mathbf{DMAC}_{f_1,f_2}(M) &= f_2(\mathbf{CBC}_{f_1}(M)). \end{aligned}$$

The first block appearing in the computation C_0 is called the initial value, it can safely be chosen as the all-zero block 0^n. The resulting algorithm may be seen on figure 1.

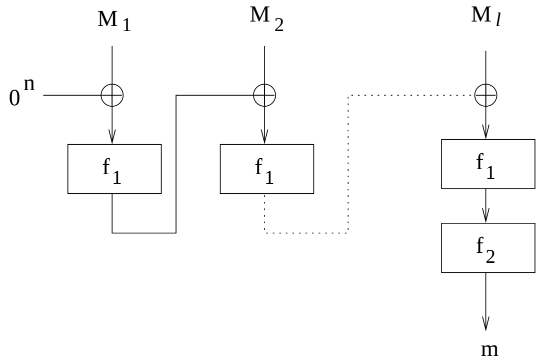

Fig. 1. The **DMAC** algorithm.

In order to deal with messages of arbitrary size, it suffices to define a padding process **Pad** such that for any pair of distinct messages M and M', we have $\mathbf{Pad}(M) \neq \mathbf{Pad}(M')$. Such a padding can be obtained by simply adding a '1' bit at the end of the message followed by enough '0' bits to turn the length of the padded message into a multiple of n. Note that in order to ensure that $\mathbf{Pad}(M) \neq \mathbf{Pad}(\mathbf{Pad}(M))$, we need to pad messages whose length is already a multiple of n. In that case one full block is added.

Another approach for dealing with messages of arbitrary length was proposed in [7]. This approach nicely avoids the padding of messages which already contain an integral number of blocks. This is achieved by taking one permutation f_2 for messages that need to be padded and a different permutation f_2' for others. In fact, this is a first step towards randomizing the function f_2 and it neatly fits into the construction we propose in this paper. However, to avoid cumbersome details, we ignore this variation in the proofs.

An advantage of [7] is that the security proof it gives for **DMAC** is much simpler than the proof from [11]. However, the result stated in [7] is slightly weaker. Indeed, in [11] the probability for an adversary of attacking **DMAC** is bounded by a function of the form $O(L^2/2^n)$, where L is the sum of the length (in blocks) of the messages whose MAC are computed during the attack. In [7], the result is expressed in terms of the number of messages q and of the length l of the longer message as a function of the form $O(l^2q^2/2^n)$. When all the messages are roughly of the same length, the two are equivalent. However, if the adversary queries $2^{n/4}-1$ messages of one block and a single message of $2^{n/4}$ blocks, then $L = 2^{n/4+1}-1$, $q = 2^{n/4}$ and $l = 2^{n/4}$, we see that the result from [11] bounds the advantage of the adversary as $O(2^{-n/2})$ while the bound from [7] is $O(1)$. In truth, it seems that the authors of [7] chose to present a weaker result for the sake of clarity. In the security proof we present in this paper, we closely mimic the proof from [7], however we bound the advantage of the adversary as a function of L instead of using q and l.

2.2 Randomizing CBC-MACs

The above definition can easily be turned into a randomized CBC–MAC. Let f_1 be a random permutation on n bits and F_2 be a set of random permutations or functions $f_2^{(R)}$ on n bits, indexed by R a r–bit number. A randomized CBC–MAC is built on the following function:

$$\mathbf{RMAC}_{f_1,F_2}(M,R) = \left(\mathbf{DMAC}_{f_1,f_2^{(R)}}(M), R\right).$$

To compute the MAC of a message, we proceed as follows: we choose a random r–bit value R and returns $\mathbf{RMAC}_{f_1,F_2}(M,R)$. To verify a given MAC (m,R) of a message M, we check whether $\mathbf{RMAC}_{f_1,F_2}(M,R) = (m,R)$. The algorithm may be seen on figure 2.

When dealing with messages of arbitrary length, we can pad all messages as in [11]. Alternatively, we can also follow the approach from [7] (see section 2.1) to avoid padding messages whose length is already a multiple of n. This is simply

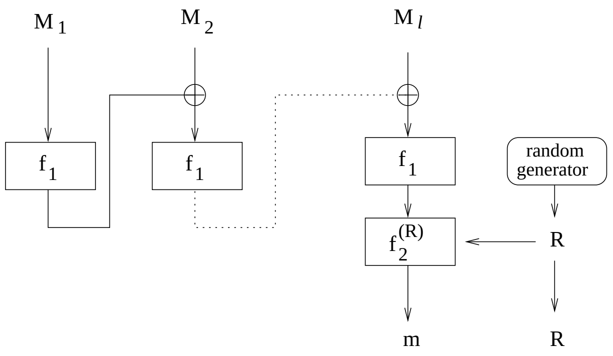

Fig. 2. The **RMAC** algorithm.

done by adding one bit to R, thus turning it into a $(r+1)$–bit number. The added bit is set to '0' when computing or verifying the MAC tag of a padded message and is set to '1' for an unpadded message. This ensures that a padded and a non-padded message never share the same R. In the boundary (non-randomized) case $r = 0$, we are clearly back to the proposal from [7], i.e. using f_2 in one case and f_2' in the other.

2.3 Security Model

The main goal of the paper is to prove that **RMAC** achieves full security. In order to make this statement precise, we need to define a new security model.

Perfect MACs. A perfect (ordinary) MAC is usually seen as a random function f from the set of messages $\{0,1\}^*$ to the set of possible MAC tags $\{0,1\}^n$. Thus to each message the function associates a random MAC tag. Similarly, a **perfect randomized MAC** is a family of independent random functions $f^{(R)}$ indexed by R, a r–bit number. Each function in the family goes from $\{0,1\}^*$ to $\{0,1\}^n$ and is randomly and independently chosen for each $R \in \{0,1\}^r$ among all possible such functions. This family of functions can be accessed through two oracles, a MAC generation oracle G_f and a MAC verification oracle V_f. The generation oracle takes a message M, chooses a random r–bit value R and returns $(f^{(R)}(M), R)$. The verification oracle takes a message M and a MAC tag (m, R), checks whether $f^{(R)}(M) = m$ and accordingly returns `valid MAC` or `invalid MAC`.

When there is no randomness, i.e. when $r = 0$, we get a perfect MAC as special case. In that case, the verification oracle becomes redundant since verification can be achieved by generating a MAC for M and testing equality with m.

Information theoretic model. The classical approach in proving the security of **DMAC** is to show the security of an information theoretic version of the construction and then come to the computational result (see [11] or [7]). Recall that **DMAC** uses two functions f_1 and f_2. For a padded message $M = (M_1, M_2, \cdots, M_l)$, we compute:

$$\mathbf{DMAC}_{f_1,f_2}(M) = f_2(\mathbf{CBC}_{f_1}(M)).$$

In the information theoretic version of the construction, it is first assumed that the functions f_1 and f_2 are randomly chosen among all possible functions and the security of the resulting construction is shown. Then f_1 and f_2 are replaced by block ciphers and it is proved that such an instantiation still offers a good security.

Now if we look at **RMAC**, we see that

$$\mathbf{RMAC}_{f_1,F_2}(M,R) = \left(\mathbf{DMAC}_{f_1,f_2^{(R)}}(M), R\right).$$

Here we assume that f_1 is a random permutation and that F_2 is a family of independent random permutations indexed by R and we are going to prove the security of **RMAC** under these assumptions. But before proceeding further, we need to define a few notations.

Notations. Let $\mathbf{Rand}(A,B)$ be the set of all functions from A to B. When A or B is replaced by a positive number n, then the corresponding set is $\{0,1\}^n$. Let $\mathbf{Perm}(n)$ be the set of all permutations on $\{0,1\}^n$. By $x \overset{R}{\leftarrow} A$ we denote the choice of an element x uniformly at random in A.

A function family F is a set of functions from A to B where A and B are subsets of $\{0,1\}^*$. Each element in F is indexed by a key K. A block cipher is a function family from A to A that contains permutations only.

Adversaries against ordinary MACs. When dealing with ordinary MACs, an adversary is an algorithm given access to an oracle that computes some function. Adversaries are assumed to never ask queries outside the domain of the oracle and to never repeat the same query.

Let F be a function family from A to B, f be a function randomly chosen in F and $\mathcal{A}$ be an adversary. We say that $\mathcal{A}^f$ forges, if $\mathcal{A}$ outputs $(x, f(x))$ and $\mathcal{A}$ never queried its oracle f at x. We denote:

$$\mathbf{Adv}_F^{\mathbf{mac}}(\mathcal{A}) = \mathbf{Pr}[f \overset{R}{\leftarrow} F | \mathcal{A}^f \text{ forges}],$$

$$\mathbf{Adv}_F^{\mathbf{prf}}(\mathcal{A}) = \left|\mathbf{Pr}[f \overset{R}{\leftarrow} F | \mathcal{A}^f = 1] - \mathbf{Pr}[f \overset{R}{\leftarrow} \mathbf{Rand}(A,B) | \mathcal{A}^f = 1]\right|,$$

and when $A = B = \{0,1\}^n$:

$$\mathbf{Adv}_F^{\mathbf{prp}}(\mathcal{A}) = \left|\mathbf{Pr}[f \overset{R}{\leftarrow} F | \mathcal{A}^f = 1] - \mathbf{Pr}[f \overset{R}{\leftarrow} \mathbf{Perm}(n) | \mathcal{A}^f = 1]\right|.$$

$\mathbf{Adv}_F^{\mathbf{mac}}(\mathcal{A})$ represents the probability for the adversary $\mathcal{A}$ of forging a valid MAC knowing that the MAC function f is not a true random function but is randomly chosen among the family F. $\mathbf{Adv}_F^{\mathbf{prf}}(\mathcal{A})$ represents the advantage for adversary $\mathcal{A}$ of distinguishing a function f randomly chosen from one chosen in the family F. $\mathbf{Adv}_F^{\mathbf{prp}}(\mathcal{A})$ is the same as above but with permutations instead of functions.

We also write $\mathbf{Adv}^{\mathbf{mac}}(t,\mu)$ for the maximal value of $\mathbf{Adv}^{\mathbf{mac}}(\mathcal{A})$ among adversaries that are bounded as follows: the running time should be less than t, and the sum of the bit length of all the oracle queries should be less than μ. We likewise define $\mathbf{Adv}^{\mathbf{prf}}(t,\mu)$ and $\mathbf{Adv}^{\mathbf{prp}}(t,\mu)$. In the case of $\mathbf{Adv}^{\mathbf{mac}}(t,\mu)$, μ also counts the length of an additional query to verify if the adversary's output is a valid forgery.

Adversaries against randomized MACs. When dealing with randomized MACs, an adversary is an algorithm given access to the generation and to the verification oracles for some randomized MAC. Adversaries are assumed to never ask queries outside the domain of the oracle, however, they may repeat the same query. Indeed, it might be useful to obtain several different MAC tags for the same message. Without loss of generality, since the adversary can always get rid of duplicates, we may assume that when MAC generation is queried several times with the same message, the generation oracle always chooses a different random value (among a total of 2^r possibilities). In that case, the adversary should not be allowed to query a given message more than 2^r times from the generation oracle. Moreover, we may assume that the adversary never repeats verifications, and never verifies previously generated MAC tags or obviously false tags. This means that when a tag (m,R) was generated for a message M, the adversary never verifies $(M,(m',R))$. Indeed, the answer is obviously `valid` when $m=m'$ and `invalid` otherwise.

Let $\mathcal{P}$ be the family of all perfect randomized MAC from a set A to a set B, let $\mathcal{F}$ be a given family of randomized MAC from A to B and let f be a randomized MAC randomly chosen in $\mathcal{F}$. We say that $\mathcal{A}^{G_f,V_f}$ forges, if $\mathcal{A}$ obtains the answer `valid MAC` from the oracle V_f for a tag $(x,(f^{(R)}(x),R))$ where $\mathcal{A}$ never got this MAC tag from its generation oracle G_f. We let:

$$\begin{aligned}\mathbf{Adv}_{\mathcal{F}}^{\mathbf{Rmac}}(\mathcal{A}) &= \mathbf{Pr}[f \stackrel{R}{\leftarrow} \mathcal{F} | \mathcal{A}^{G_f,V_f} \text{forges}],\\ \mathbf{Adv}_{\mathcal{F}}^{\mathbf{Rprf}}(\mathcal{A}) &= \left|\mathbf{Pr}[f \stackrel{R}{\leftarrow} \mathcal{F} | \mathcal{A}^{G_f,V_f}=1] - \mathbf{Pr}[f \stackrel{R}{\leftarrow} \mathcal{P} | \mathcal{A}^{G_f,V_f}=1]\right|.\end{aligned}$$

$\mathbf{Adv}_{\mathcal{F}}^{\mathbf{Rmac}}(\mathcal{A})$ represents the probability for an adversary $\mathcal{A}$ of forging a valid MAC knowing that the family f is not a perfect randomized MAC but is randomly chosen among the set $\mathcal{F}$. $\mathbf{Adv}_{\mathcal{F}}^{\mathbf{Rprf}}(\mathcal{A})$ represents the advantage for an adversary $\mathcal{A}$ of distinguishing a family f randomly chosen among all possible perfect randomized MACs from one chosen in the set $\mathcal{F}$.

As before, we write $\mathbf{Adv}^{\mathbf{Rmac}}(t,\mu)$ for the maximal value of $\mathbf{Adv}^{\mathbf{Rmac}}(\mathcal{A})$ among adversaries that are bounded as follows: the running time should be less than t, and the sum of the bit length of all the oracle queries should be less than μ. We likewise define $\mathbf{Adv}^{\mathbf{Rprf}}(t,\mu)$. In the case of $\mathbf{Adv}^{\mathbf{Rmac}}(t,\mu)$, no additional queries are necessary, since the adversary has access to a verification oracle and can test its forgery by itself. This differs from $\mathbf{Adv}^{\mathbf{mac}}(t,\mu)$ in the case of non randomized MAC.

3 Security of RMAC

We are now going to state the security reached by **RMAC** in the information-theoretic model. We evaluate this security in terms of $\mathbf{Adv}_{\mathcal{G}}^{\mathbf{Rmac}}(\mathcal{A})$ when $\mathcal{G}$ is the family described in 2.3, i.e. $\mathcal{G}$ is the family of all couples (f_1, F_2) where f_1 is a random permutation and F_2 is a family of independent random **permutations** indexed by R.

Theorem 1 states that the advantage of a forging adversary against $\mathbf{RMAC}_{f_1,F_2}$ with f_1 and F_2 as above increases as a linear function of L, the total length of messages.

Theorem 1. *[Forging* ***RMAC*** *is hard] Fix $n \geq 2, r = n$ and let $N = 2^n$. Let $\mathcal{G}$ denotes the family of randomized MAC $\boldsymbol{RMAC}_{f_1,F_2}$ built from the couple (f_1, F_2) where f_1 is a random permutation and F_2 a family of random permutations $f_2^{(R)}$. Let $\mathcal{A}$ be an adversary which asks queries of total length at most L n–bit blocks. Assume $L \leq N/4$, then:*

$$\boldsymbol{Adv}_{\mathcal{G}}^{\boldsymbol{Rmac}}(\mathcal{A}) \leq \frac{4nL + 4L + 2}{N}.$$

Proof of theorem 1. If an adversary A is able to forge, then he is able to distinguish between **RMAC** and a **Rprf**. Indeed recall that $\mathcal{A}$ forges when he has verified his forgery through the verification oracle, thus forgery leads directly to distinction. We have:

$$\mathbf{Adv}_{\mathcal{G}}^{\mathbf{Rmac}}(\mathcal{A}) \leq \mathbf{Adv}_{\mathcal{G}}^{\mathbf{Rprf}}(\mathcal{A}) + \frac{1}{N}.$$

We just need to prove an indistinguishability theorem in the information-theoretic model. Theorem 2 states that the advantage for distinguishing $\mathbf{RMAC}_{f_1,F_2}$ from a perfect randomized MAC increases as a linear function of L.

Theorem 2. *[$\boldsymbol{RMAC} \approx \boldsymbol{Rand}$] Fix $n \geq 1, r = n$ and let $N = 2^n$. Let $\mathcal{G}$ denotes the family of randomized MAC $\boldsymbol{RMAC}_{f_1,F_2}$ built from the couple (f_1, F_2) where f_1 is a random permutation and F_2 a family of random permutations. Let $\mathcal{A}$ be an adversary which asks queries of total length at most L n–bit blocks. Assume $L \leq N/4$, then:*

$$\boldsymbol{Adv}_{\mathcal{G}}^{\boldsymbol{Rprf}}(\mathcal{A}) \leq \frac{4nL + 4L + 1}{N}.$$

In order to prove this theorem, we are first going to prove a lemma where the family F_2 of random permutations has been replaced by a family of random functions.

Lemma 1. *Fix $n \geq 1, r = n$ and let $N = 2^n$. Let $\mathcal{F}$ denotes the family of randomized MAC $\boldsymbol{RMAC}_{f_1,F_2}$ built from the couple (f_1, F_2) where f_1 is a random permutation and F_2 a family of random functions. Let $\mathcal{A}$ be an adversary which asks queries of total length at most L n–bit blocks. Assume $L \leq N/4$, then:*

$$\boldsymbol{Adv}_{\mathcal{F}}^{\boldsymbol{Rprf}}(\mathcal{A}) \leq \frac{3nL + 3L + 1}{N}.$$

Sketch of proof of Lemma 1. [1] Here f_1 is a random permutation and F_2 a family of random functions. The proof of the theorem is close to the proof given in [7] but there are some fundamental differences. The adversary $\mathcal{A}$ has access to the two oracles described in section 2.3, the generation and the verification oracles. The total length of the queries it may ask is bounded by L. An adaptive adversary can always be replaced by a non-adaptative adversary that performs as well, so we are going to separately bound the advantage $\mathbf{Adv}_G(\mathcal{A})$ gained by $\mathcal{A}$ by means of the generation queries and the advantage $\mathbf{Adv}_V(\mathcal{A})$ gained by means of the verification queries, those two advantages being independent from each other.

In order to bound $\mathbf{Adv}_G(\mathcal{A})$, we observe that only a small number of messages will be processed with the same R[2]. Moreover since all the functions $f_2^{(R)}$ for different Rs are independent, the adversary only learns information from MACs generated with the same R (else he only sees outputs of independent functions). Within such a group, the adversary only learns information when the CBC output of two messages is the same (else he only sees the outputs of a random function on different inputs). So we need to evaluate the probability of collision within a group of messages at the end of the CBC computation. The collision probability is defined as follows:

$$V_n(M, M') = \mathbf{Pr}[\pi \xleftarrow{R} \mathbf{Perm}(n) | \mathbf{CBC}_\pi(M) = \mathbf{CBC}_\pi(M')].$$

We improve a lemma from [7][3] and obtain that the probability of collision among q_R messages M_i of size m_i blocks is

$$\mathbf{Pr}[\pi \xleftarrow{R} \mathbf{Perm}(n) | \exists i \neq j \text{ such that } \mathbf{CBC}_\pi(M_i) = \mathbf{CBC}_\pi(M_j)] \leq \frac{3q_R \sum_{i=1}^{q_R} m_i}{2^n},$$

with $\sum_{i=1}^{q_R} m_i \leq N/4$.

So the advantage $\mathbf{Adv}_G(\mathcal{A})$ is bounded by the sum of the probability of collision within the different groups plus the probability of existence of a group larger than n:

$$\mathbf{Adv}_G(\mathcal{A}) \leq \sum_R \frac{3q_R \sum_{i=1}^{q_R} m_i}{2^n} + \frac{1}{2^n} \leq \frac{3n}{2^n} \sum_R \sum_{i=1}^{q_R} m_i + \frac{1}{2^n} \leq \frac{3nL}{2^n} + \frac{1}{2^n}.$$

[1] The full proof is given in [10].

[2] Less than n messages with probability $1 - 1/2^n$, see in the full paper [10].

[3] See in the full paper [10].

In order to bound $\mathbf{Adv}_V(\mathcal{A})$, we observe that the adversary learns information only when he checks a previously received MAC with a new message (else he just guesses at random). The adversary succeeds if the new message collides with the reference message at the end of the CBC computation. We thus need to evaluate the probability of collision of messages with a reference message[4]. We find that

$$\mathbf{Pr}[\pi \stackrel{R}{\leftarrow} \mathbf{Perm}(n) | \exists i \in [1, q] \text{ such that } \mathbf{CBC}_\pi(M_i) = \mathbf{CBC}_\pi(M_0)] \leq \frac{3\sum_{i=0}^{q} m_i}{2^n}.$$

Summing over all reference messages of total length L we get:

$$\mathbf{Adv}_V(\mathcal{A}) \leq \frac{3L}{2^n}.$$

Finally, adding $\mathbf{Adv}_G(\mathcal{A})$ and $\mathbf{Adv}_V(\mathcal{A})$, we conclude the proof of lemma 1.

$$\mathbf{Adv}_{\mathcal{F}}^{\mathbf{Rprf}}(\mathcal{A}) \leq \frac{3nL + 3L + 1}{N}.$$

Sketch of proof of Theorem 2. In theorem 2 we replace the family F_2 of random functions by a family of random permutations. We evaluate the advantages $\mathbf{Adv}_G^{(2)}(\mathcal{A})$ and $\mathbf{Adv}_V^{(2)}(\mathcal{A})$ obtained by $\mathcal{A}$ respectively with generation and verification queries when we do this modification.

We use the well-known PRF/PRP switching lemma [4] on each permutation $f_2^{(R)}$. Indeed the adversary tries to separately distinguish the different permutations from functions. If q_R denotes the number of calls made to $f_2^{(R)}$, we recall from the proof of theorem 2 that with probability $1/2^n$ we have $q_R \leq n$. So we obtain

$$\mathbf{Adv}_G^{(2)}(\mathcal{A}) \leq \sum_R \frac{q_R^2}{2^{n+1}} \leq n \sum_R \frac{q_R}{2^{n+1}} \leq \frac{nL}{2^{n+1}}.$$

During the verification phase, the adversary wins when he distinguishes the random permutations $f_2^{(R)}$ from random functions. This happens when the verification oracle answers `valid MAC` for either a guessed MAC or a MAC obtained for another message. We find that:

$$\mathbf{Adv}_V^{(2)}(\mathcal{A}) \leq \frac{L}{2^n - n}.$$

Finally, adding $\mathbf{Adv}_G^{(2)}(\mathcal{A})$ and $\mathbf{Adv}_V^{(2)}(\mathcal{A})$ with $\mathbf{Adv}_{\mathcal{F}}^{\mathbf{Rprf}}(\mathcal{A})$, we conclude the proof of theorem 2.

[4] See in the full paper [10].

4 Instantiation of the RMAC Construction with a Block Cipher

4.1 The Computational Model

Proof in the standard model. When proving the security of a MAC construction, it is customary to first show that their information-theoretic versions approximate random functions. Then, we need to transport the result from the information-theoretic model to the computational complexity model. This improves the advantage of the adversary since he can now try to distinguish the pseudo-random functions or permutations from truly random ones. It is a general principle that the advantage in the computational-complexity model is the sum of the advantage in the information-theoretic model and of the advantages to distinguish each component of the construction from its idealized version with the number of calls made in the construction. An example of this principle appears in section 4 of [4].

To go from the information theoretic model to the computational model, we replace the random permutation f_1 with a block-cipher B. The adversary gains that way an advantage $\mathbf{Adv}_B^{\mathbf{prf}}$ of distinguishing the block-cipher B from a random function.

The random family of permutations F_2 indexed by a n-bit key can be viewed as a function $f_2(R, X) = f_2^{(R)}(X)$ of $2n$-bit to n-bit where $f_2^{(R)}$ is a permutation. We want to replace the family F_2 by a construction based on a block-cipher B with keys of $2n$ bits. We propose to choose a random $2n$-bit key K and to let $f_2^{(R)}(X) = B_{K \oplus R}(X)$, where R has been padded with n zeroes for the XOR. When such a construction is used, the adversary gains some new advantage. This advantage comes either from a weakness in the block cipher or from a weakness in the construction itself. In order to separate the two kinds of weaknesses, we would like to assume that the block cipher is "perfect". In order to do this, we use the following model. Assume that the block cipher is replaced by a family F_3 containing 2^{2n} random permutations together with a numbering. Given access to F_3, can an adversary distinguish the case where F_2 is built from F_3 as above from the case where F_2 itself is a family of random permutations ? Clearly, the adversary gains no advantage unless in the former case he manages to query the same function once through F_2 and once through F_3. In order to bound the probability that this event occurs and since the adversary is computationally bounded, we assume that he makes less than 2^n calls to F_3. Thus he queries at mots 2^n permutations among a total of 2^{2n}. On the other hand, with q queries, at most q permutations can be seen on the F_2 side. Unless the two sets collide, the adversary sees nothing. Thus, the probability for an adversary to detect that F_2 is a subfamily of F_3 is at most $\dfrac{2^n q}{2^{2n}} = \dfrac{q}{2^n}$.

Now, when using a real block-cipher, some new weaknesses may arise. In that case, this leads to a correlated key attack against the block-cipher. Indeed, the MAC construction allows us to distinguish B from F_3, when given access to $B_{K \oplus R}$ (and the corresponding decryption oracle).

The advantage gained by the adversary from the information-theoretic model to the computational complexity model is thus equal to $\frac{q}{2^n} + \mathbf{Adv}_B^{F_3}$, where $\mathbf{Adv}_B^{F_3}$ is the advantage for an adversary of distinguishing the block-cipher B from a "perfect" block-cipher.

Now that no attack other than exhaustive search is possible against B, we can express the advantage of an adversary is the standard model. We can bound q by L, the total number of queries done by the adversary.

$$\mathbf{Adv}_{\mathbf{RMAC}_B} \leq \mathbf{Adv}_{\mathcal{G}}^{\mathbf{RMAC}} + \mathbf{Adv}_B^{\mathbf{prf}} \leq \frac{4nL + 5L + 2}{2^n} + \frac{t}{2^n}.$$

Going further with the random oracle model. Instead of a $2n$-bit key block cipher, it would be more satisfying to use a standard n-bit key. We see in this section that this can be done if we accept a weaker security proof. Indeed with n-bit keys, we only prove security in the random oracle model. As above, we define $f_2(R, X) = f_3^{(R \oplus K)}(X)$. However, F_3 is now a smaller family made of 2^n permutations "only". The adversary trying to forge the MAC in this model has still access to the two oracles G_f and V_f and to F_3 through two other oracles C_f and C_f^{-1}. These computation oracles work as follows. In C_f, the adversary queries a chosen function $f_3^{(S)}$ of the family F_3, indexed by some n-bit integer S, with some input X and the oracle returns $f_3^{(S)}(X)$. In C_f^{-1}, the adversary also queries a chosen instance of the block-cipher F_3, indexed by S and asks for the value of X corresponding to the output U; the oracle returns $\left(f_3^{(S)}\right)^{-1}(U)$.

Let $\mathcal{H}$ be the family of all triplets (f_1, F_2, F_3) as described above. We want to bound the probability of forging for the adversary $\mathcal{A}$:

$$\mathbf{Adv}_{\mathcal{H}}^{\mathbf{Rmac}}(\mathcal{A}) = \mathbf{Pr}\left[f \stackrel{R}{\leftarrow} \mathcal{H} | \mathcal{A}^{G_f, V_f, C_f, C_f^{-1}} \text{forges}\right].$$

Theorem 3. *[Forging **RMAC** with idealized block-cipher] Fix $n \geq 2, r = n$ and let $N = 2^n$. Let $\mathcal{H}$ denotes the family of randomized MAC $\boldsymbol{RMAC}_{f_1, F_2, F_3}$ built from the triplet (f_1, F_2, F_3) where F_3 is a random family of 2^n permutations, f_1 is a random permutation and F_2 is a permuted copy of the family F_3 determined by a key K. Let $\mathcal{A}$ be an adversary which asks queries of total length at most L n–bit blocks. Assume $L \leq N/4$, then:*

$$\boldsymbol{Adv}_{\mathcal{H}}^{\boldsymbol{Rmac}}(\mathcal{A}) \leq \frac{4nL + 6L + 2}{N}.$$

Sketch of proof of theorem 3. [5] Let $\mathcal{A}$ be an adversary trying to forge. $\mathcal{A}$ has access to the four oracles G_f, V_f, C_f and C_f^{-1}. Against $\mathcal{A}$, we play a simulator $\mathcal{S}$ that works as follows. When $\mathcal{A}$ queries G_f or V_f on a value of a permutation of the family F_2, $\mathcal{S}$ chooses this value randomly under the condition that the

[5] The full formal proof is given in the full paper [10].

underlying function $f_2^{(R)}$ is a permutation. Of course this implies that when asked twice the value of some permutation it answers twice the same result. When $\mathcal{A}$ queries C_f or C_f^{-1} on a value of a permutation of the family F_3, $\mathcal{S}$ also chooses this value randomly under the same conditions as above. The important fact here is that the simulator answers questions about F_3 independently from questions about F_2. When the attacker $\mathcal{A}$ terminates, the simulator chooses an n-bit key K uniformly at random. Then he tries to redefine F_2 using the formula $f_2^{(R)} = f_3^{(K \oplus R)}$. Unless two incompatible answers were given while the attacker asks questions, this can be done easily. Indeed, all the answers define some $f_3^{(S)}$ and thus $f_2^{(S \oplus K)}$ or some $f_2^{(R)}$ and thus $f_3^{(R \oplus K)}$. The rest of F_3 (and F_2) can be chosen randomly (under the condition that all $f_3^{(S)}$ are permutations). When two incompatible answers were given, we assume that the simulator has lost, i.e. that the adversary wins.

We want to evaluate the probability $\mathbf{Pr}[F_2 \text{ and } F_3 \text{ incompatible}]$. The answers to F_2 and F_3 match if the tables of the answers for F_2 and those for F_3 are compatible for the chosen key K. The probability that F_2 and F_3 are not compatible is less than the probability that $f_2^{(R)}$ and $f_3^{(R \oplus K)}$ have been evaluated on one common point or that $f_2^{(R)}$ and $f_3^{(R \oplus K)}$ have one common output. Since the simulator independently answers questions on permutations of F_3 and F_2, the adversary cannot adapt its queries to one family from the answers to the queries of the other family. Moreover, when K is chosen, $\mathcal{A}$ has already terminated and it is too late for him to be adaptative. Since $\mathcal{A}$ cannot be adaptative on K, we can compute an upper bound on the probability that F_2 and F_3 mismatch. We find:

$$\mathbf{Pr}\left[F_2 \text{ and } F_3 \text{ incompatible}\right] \leq \frac{2L}{2^n}.$$

Since $\mathbf{Adv}_{\mathcal{H}}^{\mathbf{Rmac}}(\mathcal{A}) \leq \mathbf{Adv}_{\mathcal{G}}^{\mathbf{Rmac}}(\mathcal{A}) + \frac{2L}{2^n}$, we have $\mathbf{Adv}_{\mathcal{H}}^{\mathbf{Rmac}}(\mathcal{A}) \leq \frac{4nL + 6L + 2}{2^n}$.

This concludes the sketch of proof of theorem 3.

5 Detailed Instantiations with the AES

Up to now, the only known attack against the AES is exhaustive search. We propose two different instantiations of **RMAC** with the AES. The first one assumes that all messages are padded. The second instantiation takes advantage of the technique from [7] that allows not to pad messages which are formed from an integral number of blocks (see section 2.2). We describe here the instantiations with the AES using n-bit keys and $2n$-bit keys. In these instantiations, the longest key size for K_2 gives security in the standard model, while the shortest key size restricts us to security in the random oracle model.

First instantiation. Let K_1 be a 128-bit key and K_2 a 128 or 256-bit key. Let $f_1 = \mathbf{AES}_{K_1}$ and $f_2^{(R)} = \mathbf{AES}_{K_2 \oplus R}$. Here R is a 128-bit integer padded with zeros for the XOR if necessary. The proposed instantiation is simply $\mathbf{RMAC}_{f_1,F_2}$.

Second instantiation. Let K_1 be a 128-bit key and K_2 be a 192 or 256-bit key. Let $f_1 = \mathbf{AES}_{K_1}$ and $f_2^{(R)} = \mathbf{AES}_{K_2 \oplus R}$. Here R is a 129-bit number padded with zeros for the XOR. The 128 low order bits of R are randomly chosen by the generation oracle. The 129-th bit is a '0' when the message needs to be padded and a '1' otherwise. This additional bit is never included as part of the MAC tags, it should be set by the verification oracle according to the properties of the message being verified.

Security of the instantiations. Glueing together theorem 1 and theorem 3 with the known attacks against the AES, we claim that the advantage of an adversary making queries of total length at most L and with runtime t – including the run time of the generation and verification queries themselves – is at most:

$$\mathbf{Adv}_{\mathbf{RMAC}_{\mathbf{AES}}} \leq \frac{4 \cdot 128L + 5L + 2}{2^{128}} + \frac{t}{2^{128}} \leq \frac{518L + t}{2^{128}},$$

using a key K_2 of 256 bits and

$$\mathbf{Adv}_{\mathbf{RMAC}_{\mathbf{AES}}} \leq \frac{4 \cdot 128L + 6L + 2}{2^{128}} + \frac{t}{2^{128}} \leq \frac{519L + t}{2^{128}},$$

using a key K_2 of 128 bits.
This should be compared with the security of the traditional **DMAC**:

$$\mathbf{Adv}_{\mathbf{DMAC}_{\mathbf{AES}}} \leq \frac{2L^2 + t}{2^{128}}.$$

In other words, $\mathbf{RMAC}_{\mathbf{AES}}$ is secure as long as the total length of the queries is smaller than 2^{118}, while $\mathbf{DMAC}_{\mathbf{AES}}$ is secure as long as the total length of the queries is smaller than 2^{63}. In fact, the security of $\mathbf{RMAC}_{\mathbf{AES}}$ is almost as good as the security of **DMAC** with a good 256–bit block cipher.

6 Conclusion

The **RMAC** construction proposed in this paper 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 [2] 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 functions other than the block cipher.

Moreover, **RMAC** unleashes the full power of the AES in MAC computation, thus making the need for 256-bit block ciphers a very remote perspective. Quite interestingly, the proof is stronger when using 256-bit keys in AES.

References

1. M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In *CRYPTO'96*, volume 1109 of *LNCS*. Springer, 1996.
2. M. Bellare, O. Goldreich, and H. Krawczyk. Stateless evaluation of pseudorandom functions: Security beyond the birthday barrier. In *CRYPTO'99*, volume 1666 of *LNCS*, pages 270–287. Springer, 1999.
3. M. Bellare, R. Guerin, and P. Rogaway. XOR MACs: New methods for message authentication using finite pseudorandom functions. In *CRYPTO'95*, volume 963 of *LNCS*, pages 15–28. Springer-Verlag, 1995.
4. M. Bellare, J. Killian, and P. Rogaway. The security of the cipher block chaining message authentication code. In *CRYPTO'94*, volume 839 of *LNCS*, pages 341–358. Springer, 1994. See new version at `http://www.cs.ucdavis.edu/~rogaway/`.
5. M. Bellare, T. Krovetz, and P. Rogaway. Luby-rackoff backwards: increasing security by making block-ciphers non-invertible. In *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 266–280. Springer, 1998.
6. J. Black, S. Halevi, H. Krawczyk, T. Krovetz, and P. Rogaway. UMAC: Fast and secure message authentication. In *CRYPTO'99*, volume 1666 of *LNCS*, pages 216–233. Springer-Verlag, 1999.
7. J. Black and P. Rogaway. CBC MACs for arbitrary-length messages: The three-key constructions. In *CRYPTO 2000*, volume 1880 of *LNCS*, pages 197–215. Springer, 2000.
8. C. Hall, D. Wagner, J. Kelsey, and B. Schneier. Building PRFs from PRPs. In *CRYPTO'98*, volume 1462 of *LNCS*, pages 370–389. Springer, 1998.
9. International Organization for Standards, Geneva, Switzerland. *ISO/IEC 9797-1. Information Technology - Security Techniques - Data integrity mechanism using a cryptographic check function employing a block cipher algorithm*, second edition edition, 1999.
10. É. Jaulmes, A. Joux, and F. Valette. On the security of randomized cbc–mac beyond the birthday paradox limit: A new construction. Available at `http://eprint.iacr.org`, 2002. Full version of this paper.
11. E. Petrank and C. Rackoff. CBC-MAC for real-time data sources. Technical Report 97-10, Dimacs, 1997.
12. B. Preneel and P. van Oorschot. MDx-MAC and building fast MACs from hash functions. In *CRYPTO'95*, volume 963 of *LNCS*, pages 1–14. Springer, 1995.
13. M. Semanko. L-collision attacks against randomized MACs. In *CRYPTO 2000*, volume 1880 of *LNCS*, pages 216–228. Springer, 2000.
14. U.S. Department of Commerce/National Bureau of Standards, National Technical Information Service, Springfield, Virginia. *FIPS 113. Computer Data Authentication. Federal Information Processing Standards Publication 113*, 1994.
15. M. Wegman and J. Carter. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, 22(3):265–279, 1981.