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 ideal cipher 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 [17]. Other existing constructions are for example XOR-MACs [3], HMAC [1] and UMAC [7]. 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 [11, 16]. 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

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

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 [11].

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 [13] and [8] 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 [14]. 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 [11] 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, 9]. Moreover, it was shown in [15] 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 ideal cipher model [6] and uses only block-ciphers with n-bit keys[1]. 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 [13] and [8], we know that encrypted CBC–MAC has a security level of $O(2^{n/2})$. In particular, in [13] 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 2.1.

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.

[1] A previous version of this paper used the random oracle model. It appears that the ideal cipher model, a slightly weaker model, suffices to address our needs.

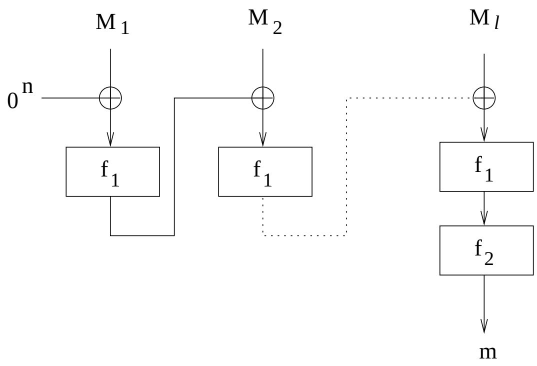

Fig. 1. The **DMAC** algorithm.

Another approach for dealing with messages of arbitrary length was proposed in [8]. 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 [8] is that the security proof it gives for **DMAC** is much simpler than the proof from [13]. However, the result stated in [8] is slightly weaker. Indeed, in [13] 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 [8], 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 [13] bounds the advantage of the adversary as $O(2^{-n/2})$ while the bound from [8] is $O(1)$. In truth, it seems that the authors of [8] 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 [8], 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 2.2.

When dealing with messages of arbitrary length, we can pad all messages as in [13]. Alternatively, we can also follow the approach from [8] (see section 2.1) to avoid padding messages whose length is already a multiple of n. This is simply 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 [8], i.e. using f_2 in one case and f_2' in the other.

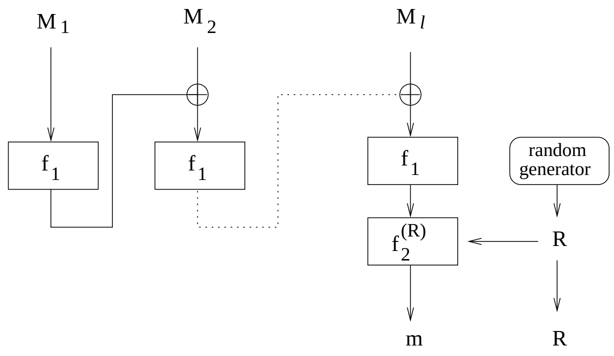

Fig. 2. The **RMAC** algorithm.

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 [13] or [8]). 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 $x \xleftarrow{R} A$ 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 note:

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

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

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

$\mathbf{Adv}_F^{\mathbf{mac}}(\mathcal{A})$ 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:

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

$\mathbf{Adv}_{\mathcal{F}}^{\mathbf{Rmac}}(\mathcal{A})$ 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 $\mathbf{RMAC}_{f_1,F_2}$ 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. Then:*

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

Proof of theorem 1. The inequality is obviously true for $L > N/4$ since $(4nL + 4L + 2)/N$ is then greater than one. We are now going to prove it for $L \leq N/4$. 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:

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

And then:

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

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. *[**RMAC** $\approx$ **Rand**] Fix $n \geq 1, r = n$ and let $N = 2^n$. Let $\mathcal{G}$ denotes the family of randomized MAC $\mathbf{RMAC}_{f_1,F_2}$ 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. Then:*

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

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. Then:*

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

The complete proof is given in appendix B. Here we only give the guidelines of the proof.

Sketch of proof of Lemma 1. The inequality is obviously true for $L > N/4$ since $(4nL + 4L + 2)/N$ is then greater than one. We are now going to prove it for $L \leq N/4$. 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 [8] 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. In our setting, any adaptive adversary can 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 (less than n messages with probability $1 - 1/2^n$, see in appendix B). 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:

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

Improving a lemma from [8] (see in the appendix A the lemma 3), we obtain that the probability of collision among q_R messages M_i of size m_i blocks is

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

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:

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

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 (see in the appendix A the lemma 4). We find that

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

when $\sum_{i=0}^{q} m_i \leq N/4$.

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.

Switching from random functions to random permutations is a well-known lemma from [4]. This lemma states that the advantage gained in distinguishing a random permutation of $\{0,1\}^n$ from a random function from $\{0,1\}^n$ to $\{0,1\}^n$ with p queries is at most $\dfrac{p(p-1)}{2^{n+1}}$.

This lemma can be used only when the adversary does not have access to an inverse oracle. This is the case of the generation queries phase. We use the lemma separating the calls made to the different permutations $f_2^{(R)}$. 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 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 $\dfrac{q}{2^n} + \mathbf{Adv}_B^{\mathbf{prp}}$, where $\mathbf{Adv}_B^{\mathbf{prp}}$ is the advantage for an adversary of distinguishing the block-cipher B from a "perfect" block-cipher.

Now we can express the advantage of an adversary in the standard model. We can bound q by L, the total number of queries done by the adversary.

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

Going further with the ideal cipher 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 ideal cipher 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* $\boldsymbol{RMAC}$ *with idealized block-cipher] Fix* $n \ge 2, r = n$ *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. Let* T *be the number of queries* $\mathcal{A}$ *asks to* C_f^{-1}*. Then:*

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

Sketch of proof of theorem 3. The full formal proof is given in appendix C.

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 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 to 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 plus the probability that $f_2^{(R)}$ and $f_3^{(R\oplus K)}$ have one common output plus the probability that $f_3^{(K\oplus R)^{-1}}$ has been evaluated on a value $V_{K\oplus R}$ that was the output of $f_2^{(R)}$ on some message input M_R.

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.

We will distinguish the incompatibility coming from the queries to C_f from the incompatibility coming from the queries to C_f^{-1}. We write:

$$\mathbf{Pr}\left[F_2 \text{ and } F_3 \text{ incompatible}\right] = \mathbf{Pr}_{C_f}\left[F_2 \text{ and } F_3 \text{ incompatible}\right] + \mathbf{Pr}_{C_f^{-1}}\left[F_2 \text{ and } F_3 \text{ incompatible}\right]$$

We first try to evaluate $\mathbf{Pr}_{C_f}\left[F_2 \text{ and } F_3 \text{ incompatible}\right]$. If we let Y_R represents a query Y done to $f_2^{(R)}$ and $X_{K\oplus R}$ a query X done to $f_3^{(K\oplus R)}$, the probability that one query Y_R to F_2 is incompatible with a query $X_{K\oplus R}$ in F_3 through C_f is:

$$\mathbf{Pr}_{C_f} \leq \sum_{X_{K\oplus R}} \left(\mathbf{Pr}\left[f_2^{(R)}(Y_R) = f_3^{(K\oplus R)}(X_{K\oplus R})\right] + \mathbf{Pr}\left[Y_R = X_{K\oplus R}\right]\right).$$

Let M_R be the message associated to Y_R, that is $Y_R = CBC_{f_1}(M_R)$, we have:

$$\mathbf{Pr}\left[Y_R = X_{K\oplus R}\right] = \mathbf{Pr}\left[CBC_{f_1}(M_R) = X_{K\oplus R}\right].$$

As seen in the classic proof of lemma 2, we have

$$\mathbf{Pr}\left[CBC_{f_1}(M_R) = X_{K\oplus R}\right] \leq \frac{|M_R|}{2^n}.$$

Similarly, since $f_2^{(R)}$ and $f_3^{(K\oplus R)}$ are permutations, we have:

$$\mathbf{Pr}\left[f_2^{(R)}(Y_R) = f_3^{(K\oplus R)}(X_{K\oplus R})\right] = \mathbf{Pr}\left[f_2^{(R)}(CBC_{f_1}(M_R)) = f_3^{(K\oplus R)}(X_{K\oplus R})\right] \leq \frac{|M_R|}{2^n}.$$

For a given key K, the probability that $\mathcal{A}$ wins is the sum over all queries Y_R of the probability that one query Y_R to F_2 is incompatible with all queries $X_{K\oplus R}$ in F_3. For a given key K, we obtain:

$$\mathbf{Pr}_{C_f}\left[F_2 \text{ and } F_3 \text{ incompatible}|K\right] \leq \sum_R \left(\sum_{M_R}\left(\sum_{X_{K\oplus R}} \frac{2|M_R|}{2^n}\right)\right).$$

Let $n_{K\oplus R}$ be the number of queries done to the permutation $f_3^{(K\oplus R)}$ of F_3 and let $L_{M_R} = \sum_{M_R} |M_R|$. We get:

$$\mathbf{Pr}_{C_f}\left[F_2 \text{ and } F_3 \text{ incompatible}|K\right] \leq \sum_R n_{K\oplus R} \frac{2L_{M_R}}{2^n}$$

We sum up for all K and divide by 2^n to get the probability for a randomly chosen K. Since the total length of queries L is less than 2^n, we have $\sum_R n_{K\oplus R} \leq 2^n$ and that gives us:

$$\begin{aligned}\mathbf{Pr}_{C_f}\left[F_2 \text{ and } F_3 \text{ incompatible}\right] &\leq \frac{1}{2^n}\sum_K \sum_R n_{K\oplus R} \frac{2L_{M_R}}{2^n} = \frac{1}{2^n}\sum_R \left(\frac{2L_{M_R}}{2^n}\sum_K n_{K\oplus R}\right)\\ &\leq \frac{1}{2^n}\sum_R \frac{2L_{M_R}}{2^n}2^n = \sum_R \frac{2L_{M_R}}{2^n}\\ &\leq \frac{2L}{2^n}.\end{aligned}$$

We now try to evaluate $\mathbf{Pr}_{C_f^{-1}}\left[F_2 \text{ and } F_3 \text{ incompatible}\right]$. It is the probability that we have asked the value of ${f_3^{(R')}}^{-1}$ on a value $V_{R'}$ that was the output of $f_2^{(R)}$ on some message M_R with R such that $R = R' \oplus K$.

For a given K and a given R' this probability is the probability that $R' = R \oplus K$ for all R such that a query $f_2^{(R)}(CBC_{f_1}(M_R))$ gave back the value $V_{R'}$.

The probability that $R' = R \oplus K$ is $1/2^n$ but we may have up to n collisions in the outputs of G_f, that is n different Rs such that $f_2^{(R)}(CBC_{f_1}(M_R)) = V_{R'}$.

So for a given key we have a probability

$$\mathbf{Pr}_{C_f^{-1}}\left[F_2 \text{ and } F_3 \text{ incompatible}|K\right] \leq \sum_{\text{all queries to } C_f^{-1}} \frac{n}{2^n}$$

Let T be the number of queries to C_f^{-1}, we sum up for all possible values of K

$$\mathbf{Pr}_{C_f^{-1}}\left[F_2 \text{ and } F_3 \text{ incompatible}\right] \leq \frac{1}{2^n}\sum_K \frac{nT}{2^n} = \frac{nT}{2^n}$$

Since $\mathbf{Adv}_{\mathcal{H}}^{\mathbf{Rmac}}(\mathcal{A}) \leq \mathbf{Adv}_{\mathcal{G}}^{\mathbf{Rmac}}(\mathcal{A}) + \mathbf{Pr}_{C_f} + \mathbf{Pr}_{C_f^{-1}}$, we have $\mathbf{Adv}_{\mathcal{H}}^{\mathbf{Rmac}}(\mathcal{A}) \leq \dfrac{4nL + 6L + 2 + nT}{2^n}$.

This concludes the sketch of proof of theorem 3.

5 Detailed Instantiations with the AES

Up to now, the only known attacks against the AES are generic attacks such as exhaustive search or Hellman's attacks [10]. 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 [8] 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 ideal cipher 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 $\mathbf{RMAC}_{f_1,F_2}$. See the algorithm of figure 5.

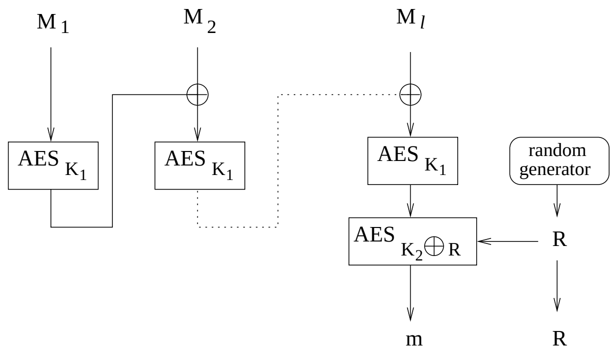

Fig. 3. The **RMAC** algorithm instantiated with the AES.

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:

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

using a key K_2 of 256 bits.

Since the proof for a key K_2 of 128 bits is in the ideal cipher model, expressing a concrete security in this case has no real meaning.

This security bound should be compared with the security of the traditional **DMAC**:

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

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.

MAC truncation. It is possible to reduce the size of the MAC by using some truncation on R and m. The idea is to reduce the size of the MAC while minimizing the security loss. In order to do this, we need to consider not only collision based attacks but also plain guessing attacks. Collision based attacks work by colliding both R and the output of the last AES evaluation, while guessing attacks need to correctly guess an m-bit value. In order to be as secure as possible, we balance the probability of success of the two kinds of attacks. The size of the blocks is n. Let r be the number of bits of the random R and let s be the number of bits of the output m that are kept to form the MAC. We want $(r+n)/2 = s$. For a security of $2^{-(n-d)}$ we take $r = n - 2d$ and $s = n - d$ and that gives us a MAC size of $t = r + s = 2n - 3d$.

For example, if $d = 43$, we get a security of 2^{-85} with a MAC tag of size $t = 127$. If we add one bit to avoid padding messages of length multiple of 128, as explained in section 2.2, we get a MAC tag of size 128 bits.

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.

The **RMAC** construction could also be described using the new model of tweakable block ciphers introduced by Liskov, Rivest and Wagner [12]. The descriptions and proofs may be simpler in this new setting.

References

1. M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In N. Koblitz, editor, *Advances in cryptology — CRYPTO'96*, volume 1109 of *Lecture Notes in Computer Science*. Springer, 1996.
2. 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.
3. M. Bellare, R. Guerin, and P. Rogaway. XOR MACs: New methods for message authentication using finite pseudo-random functions. In D. Coppersmith, editor, *Advances in Cryptology — CRYPTO'95*, volume 963 of *Lecture Notes in Computer Science*, 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 *Advances in Cryptology — CRYPTO'94*, volume 839 of *Lecture Notes in Computer Science*, 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 Kaisa Nyberg, editor, *Advances in Cryptology — EUROCRYPT'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 266–280. Springer, 1998.
6. M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In B. Preneel, editor, *Advances in Cryptology — EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 139–155. Springer, May 2000.
7. J. Black, S. Halevi, H. Krawczyk, T. Krovetz, and P. Rogaway. Umac: Fast and secure message authentication. In M. Wiener, editor, *Advances in Cryptology — CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 216–233. Springer-Verlag, 1999.
8. J. Black and P. Rogaway. CBC MACs for arbitrary-length messages: The three-key constructions. In Mihir Bellare, editor, *Advances in Cryptology — CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 197–215. Springer, 2000.
9. C. Hall, D. Wagner, J. Kelsey, and B. Schneier. Building PRFs from PRPs. In Hugo Keawczyk, editor, *Advances in Cryptology — CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 370–389. Springer, 1998.
10. M. E. Hellman. A cryptanalytic Time-Memory Trade-Off. In *IEEE Transactions on Information Theory*, volume IT-26, pages 401–406, July 1980.
11. 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.
12. M. Liskov, R. Rivest, and D.Wagner. Tweakable block ciphers. In Moti Yung, editor, *Advances in Cryptology — Proceedings of CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 31–46. Springer, 2002.
13. E. Petrank and C. Rackoff. CBC-MAC for real-time data sources. Technical Report 97-10, Dimacs, 1997.
14. B. Preneel and P. van Oorschot. MDx-MAC and building fast MACs from hash functions. In Don Coppersmith, editor, *Advances in cryptology — CRYPTO'95*, volume 963 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 1995.
15. M. Semanko. L-collision attacks against randomized MACs. In Mihir Bellare, editor, *Advances in Cryptology — CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 216–228. Springer, 2000.

16. 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.
17. 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.

A Collisions During CBC Computations

As seen in section 3, in order to prove the security of CBC–MACs, it is very important to bound the probability of collision between $\mathbf{CBC}_f(M)$ and $\mathbf{CBC}_f(M')$. The collision probability is defined as follows:

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

It is given by a lemma from [8]:

Lemma 2 (CBC collision bound). *Fix $n \geq 1$ and let $N = 2^n$. Then for $m, m' \leq N/4$, for any pair of messages M and M' of respective length m blocks and m' blocks:*

$$V_n(M,M') \leq \frac{(m+m')^2}{2^n}.$$

This lemma is in fact not strong enough for our purpose, so we need to improve it. We first state the two following lemmas.

Lemma 3 (CBC collision in a group). *Fix $n \geq 1$ and let $N = 2^n$. Fix $q \geq 1$ and let $M_1, \ldots M_q$ be q distinct messages having $m_1, \ldots, m_q$ blocks. Assume that $\sum_{i=1}^q m_i \leq N/4$. Then:*

$$\boldsymbol{Pr}[\pi \stackrel{R}{\leftarrow} \boldsymbol{Perm}(n) | \exists i \neq j \text{ such that } \boldsymbol{CBC}_\pi(M_i) = \boldsymbol{CBC}_\pi(M_j)] \leq \frac{3q\sum_{i=1}^q m_i}{2^n}.$$

Lemma 4 (CBC collision with a reference message). *Fix $n \geq 1$ and let $N = 2^n$. Fix $q \geq 1$ and let M_0, $M_1, \ldots M_q$ be $q+1$ distinct messages having $m_0, m_1, \ldots, m_q$ blocks. Assume that $\sum_{i=0}^q m_i \leq N/4$. Then:*

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

In order to prove these two lemmas, let us start with the simpler case of two different messages M and M'. Splitting each message in blocks, we write $M = (M_1, M_2, \cdots, M_m)$ and $M' = (M'_1, M'_2, \cdots, M'_{m'})$, we can assume that $m \geq m'$. Looking at the two messages, we have either $M_m = M'_{m'}$ or $M_m \neq M'_{m'}$. In the former case, we may drop the last block of each message without affecting the probability of collision. Iterating this last block removal, we end up with two truncated messages $M^{(T)}$ and ${M'}^{(T)}$ that satisfy one of the two following conditions:

- Either ${M'}^{(T)}$ is the empty message. This happens when M' is a suffix of M.
- Or the last blocks of $M^{(T)}$ and ${M'}^{(T)}$ differs.

Now we can forget about the truncation of messages and address the following issues:

- Given a message M of length $m \leq N/4$, bound the probability $T_n(M)$ that M collides with any of its suffixes. This happens if and only if some intermediate value C_i for $i \neq 0$ during the CBC–MAC computation of M collides with the initial value $C_0 = 0^n$.
- Given two messages M and M' of length $m \leq N/4$ and $m' \leq N/4$ satisfying $M_m \neq {M'}_{m'}$, bound $V_n(M,M')$ the probability that $\mathbf{CBC}_\pi(M) = \mathbf{CBC}_\pi(M')$.

Bounding T_n. Let us start by computing a bound on T_n. In order to do that, we compute for a given message M of length m a bound on the number of "bad" permutations π that lead to a collision between M and one of its suffixes. This can be done by writing an algorithm (see table 1) that can construct all permutations and that outputs a permutation and a status `good` or `bad`.

Looking at the algorithm, we see that it iteratively constructs a random permutation. Among all the different possible runs, it can build each of the $N!$ possible permutations exactly once. At each iteration, the algorithm defines π on exactly one new point. This point is either $I_i = C_{i-1} \oplus M_i$ or the smallest value where π is still undefined. Thus, in the first iteration there are N possible choices, in the second $N-1$ and so on.

Now, during each iteration, there is at most one choice that can turn the status to `bad`. Thus the probability $T_n(M)$ that M collides with any of its suffixes can be bounded by:

$$T_n(M) \leq \frac{1}{N!} \sum_{i=1}^{m} \frac{N!}{N+1-i} < \frac{m}{\frac{3}{4}N} < \frac{2m}{2^n}.$$

The second inequality comes from the fact that since $m \leq N/4$ we have $N+1-i > 3N/4$.

Bounding V_n. Given two messages M and M' of length $m \leq N/4$ and $m' \leq N/4$ satisfying $M_m \neq M'_{m'}$, we want to bound $V_n(M, M')$. As above, we are going to write an algorithm that constructs all permutations and outputs the permutation together with a status. Before writing the algorithm, let us notice that

$$\mathbf{CBC}_\pi(M) = \mathbf{CBC}_\pi(M') \quad \text{iff} \quad C'_{m'-1} = C_{m-1} \oplus M_m \oplus M'_{m'},$$

where C'_i denotes the intermediate CBC–MAC values while processing M'.

We also remark that the choice of π uniquely determines C_{m-1}. Moreover, for any n–bit number X, we can define Π_X^M, the set of all permutations π such that $C_{m-1} = X$. Clearly, the family of sets Π_X^M is a disjoint cover of the set of all permutations on n–bit numbers. We use this fact in the algorithm of table 2.

This algorithm iteratively constructs a random permutation. Among the different possible runs, each permutation is constructed once in its set Π_X^M, but it may additionally be constructed in other, wrong sets. In the latter case, the output status is `irrelevant`. When choosing X, the algorithm has N possible choices. Then it starts its determination of π on $m+m'-2$ points, with N choices for the first determination, $N-1$ for the next one, and so on up to determination number $m+m'-3$ with $N-(m+m'-4)$ possible choices. During the last determination, the algorithm tries to force C_{m-1} to be X; if this is not possible, it sets the status to `irrelevant`. Anyway, the last determination is deterministic and the algorithm has a single possible choice at that point. Then the rest of π is randomly chosen. We conclude that the total number of different runs of the algorithm is:

$$\frac{N \cdot N!}{N-(m+m'-3)}.$$

Among this runs $N!$ are relevant, the others are marked as `irrelevant`. Now, there are $m'-1$ iterations where the status can become bad. In each of them, there is at most one choice (among $N+1-i$ with $1 \leq i \leq m'-1$) that turns the status to `bad`. Thus the probability $V_n(M, M')$ can be bounded by:

$$\begin{aligned} V_n(M, M') &\leq \frac{1}{N!} \sum_{i=1}^{m'-1} N \cdot \frac{N!}{(N+1-i)\cdot(N-(m+m'-3))} \\ &< \frac{m'N}{\frac{3N}{4}\cdot\frac{N}{2}} = \frac{8}{3}\cdot\frac{m'}{N} < \frac{3m'}{2^n}. \end{aligned}$$

The second inequality comes from the fact that since $m \leq N/4$ and $m' \leq N/4$ we have $N+1-i > 3N/4$ and $N-(m+m'-3) > N/2$.

Set `Status=good`, Let $C_0 = 0^n$, Let $i = 1$
Set π as undefined at all points
Declare all n–bit numbers as currently unused
While $i \leq m$ do:
  Let $I_i = C_{i-1} \oplus M_i$
  If π is still undefined at I_i let $x = I_i$, else let x denote the smallest n-bit number where π is not yet defined.
  Randomly choose a value y for $\pi(x)$ among the unused n–bit numbers.
  Fix $\pi(x) = y$ and mark y as used
  If $y = 0^n$, let `Status=bad`
  Let $C_i = \pi(I_i)$
  Increment i
Randomly determines the rest of the permutation π.
Output π and `Status`.

Table 1. Algorithm for bounding $T_n(M)$

Set `Status=good`, Let $C_0 = 0^n$, $C'_0 = 0^n$, Let $i = 1$
Set π as undefined at all points
Declare all n–bit numbers as currently unused
Randomly choose a n–bit number X, thus deciding to construct a permutation from Π_X^M.
Let $X' = X \oplus M_m \oplus M'_{m'}$
While $i \leq m' - 1$ do:
  Let $I'_i = C'_{i-1} \oplus M'_i$
  If π is still undefined at I'_i let $x = I'_i$, else let x denote the smallest n-bit number where π is not yet defined.
  Randomly choose a value y for $\pi(x)$ among the unused n–bit numbers.
  Fix $\pi(x) = y$ and mark y as used
  If $y = X'$, let `Status=bad`
  Let $C'_i = \pi(I'_i)$
  Increment i
Let $i = 1$. While $i \leq m - 2$ do:
  Let $I_i = C_{i-1} \oplus M_i$
  If π is still undefined at I_i let $x = I_i$, else let x denote the smallest n-bit number where π is not yet defined.
  Randomly choose a value y for $\pi(x)$ among the unused n–bit numbers.
  Fix $\pi(x) = y$ and mark y as used
  Let $C_i = \pi(I_i)$
  Increment i
Let $I_{m-1} = C_{m-2} \oplus M_{m-1}$
If π is still undefined at I_{m-1} let $x = I_{m-1}$
If π is already defined at I_{m-1} let `Potential-Status=irrelevant` and let x denote the smallest n-bit number where π is not yet defined.
If X is still unused let $y = X$
If X is already used let `Potential-Status=irrelevant` and let y denote the smallest n-bit number marked as unused.
Fix $\pi(x) = y$ and mark y as used
If `Potential-Status=irrelevant` and $\pi(I_{m-1}) \neq X$ let `Status=irrelevant`
Let $C_{m-1} = \pi(I_{m-1})$
Randomly determines the rest of the permutation π.
Output π and `Status`.

Table 2. Algorithm for bounding $V_n(M, M')$

Proof of lemma 3 As in the statement of the lemma, we fix $n \geq 1$ and let $N = 2^n$. We fix $q \geq 1$ and let M_1, $\ldots M_q$ be q distinct messages having $m_1, \ldots, m_q$ blocks. We assume that $\sum_{i=1}^{q} m_i \leq N/4$. We let:

$$P = \mathbf{Pr}[\pi \stackrel{R}{\leftarrow} \mathbf{Perm}(n) | \exists i \neq j \text{ such that } \mathbf{CBC}_\pi(M_i) = \mathbf{CBC}_\pi(M_j)].$$

We can bound P as follows:

$$\begin{aligned}
P &\leq \sum_{i=1}^{q} \sum_{j \neq i} V_n(M_i, M_j) \\
&\leq \sum_{i=1}^{q} \left(\sum_{\substack{j \neq i \\ M_j \text{ suffix of } M_i}} V_n(M_i, M_j) + \sum_{\substack{j \neq i \\ M_j \text{ non suffix of } M_i}} V_n(M_i, M_j) \right) \\
&\leq \sum_{i=1}^{q} \left(T_n(M_i) + \sum_{\substack{j \neq i \\ M_j \text{ non suffix of } M_i}} V_n(M_i, M_j) \right) \\
&\leq \sum_{i=1}^{q} \left(\frac{2m_i}{2^n} + \sum_{j \neq i} \frac{3m_j}{2^n} \right) \leq \frac{3q \sum_{i=1}^{q} m_i}{2^n}.
\end{aligned}$$

This concludes the proof of lemma 3.

Proof of lemma 4 As in the statement of the lemma, we fix $n \geq 1$ and let $N = 2^n$. We fix $q \geq 1$ and let M_0, $M_1, \ldots M_q$ be q distinct messages having $m_0, m_1, \ldots, m_q$ blocks. We assume that $\sum_{i=0}^{q} m_i \leq N/4$. We let:

$$P = \mathbf{Pr}[\pi \stackrel{R}{\leftarrow} \mathbf{Perm}(n) | \exists i \text{ such that } \mathbf{CBC}_\pi(M_i) = \mathbf{CBC}_\pi(M_0)].$$

We can bound P as follows:

$$\begin{aligned}
P &\leq \sum_{i=1}^{q} V_n(M_0, M_j) \leq \sum_{\substack{i = 1 \\ M_i \text{ suffix of } M_0}}^{q} V_n(M_0, M_j) + \sum_{\substack{i = 1 \\ M_i \text{ non suffix of } M_0}}^{q} V_n(M_0, M_j) \\
&\leq T_n(M_0) + \sum_{\substack{i = 1 \\ M_i \text{ non suffix of } M_0}}^{q} V_n(M_0, M_j) \leq \frac{2m_0}{2^n} + \sum_{i=1}^{q} \frac{3m_i}{2^n} \leq \frac{3 \sum_{i=0}^{q} m_i}{2^n}.
\end{aligned}$$

This concludes the proof of lemma 4.

B Proofs of Theorem 2 and Lemma 1

Preliminaries. In order to prove the security of this construction, we closely mimic the proof from [8]. However, there are some differences between the two proofs. A crucial difference is that in the new construction, the adversary may use two different oracles (the generation and the verification oracle) while previously the generation oracle alone was sufficient. Indeed, to verify an ordinary MAC, it suffices to generate the MAC value and compare it to the submitted value, thus a verification oracle is not needed. Before giving the security proof of **RMAC**, let us state a few facts about adversaries.

- An adversary $\mathcal{A}$ against **RMAC** is a Turing machine with access to the generation and verification oracles. As before, we limit the total size of the queries that the adversary can make by L, with $L \leq N/4$. Clearly an adversary successfully forges when one of the two following events occurs:

- The result of a call to the generation oracle collides with the result of a previous call (on a different message). Indeed, with good probability this comes from a collision of the form

$$\mathbf{CBC}_{f_1}(M) = \mathbf{CBC}_{f_1}(M')$$

in the core CBC–MAC computation. In that case, any other valid MAC tag for M is also valid for M', thus forgery is easy in our security model.
- A call to the verification oracle outputs `valid MAC`, for a pair (M, m) where the candidate MAC value m is not a previous output of the generation oracle for the message M. Indeed, this by itself is a forgery. For the sake of simplicity, we also assume, in the use of a distinguishing adversary, that whenever this event happens the adversary wins.

– Any adversary $\mathcal{A}$ (that forges or distinguishes) can be rewritten as an adversary $\mathcal{A}'$ that first makes all its generation queries and then its verification queries. This is done as follows: $\mathcal{A}'$ simulates $\mathcal{A}$ except for verification queries. Whenever $\mathcal{A}$ makes a verification query, $\mathcal{A}'$ instead stores the query and assumes that the answer was `invalid MAC`. When $\mathcal{A}$ reaches its limit on the number of queries, it terminates and $\mathcal{A}'$ sends all the stored queries to the verification oracle. Clearly, $\mathcal{A}'$ uses the same resources as $\mathcal{A}$, however it is no longer adaptative on the results of the verification queries. Yet, this is not hurtful. Indeed, remember that if the verification oracle outputs `valid MAC` we assume that the adversary wins and stops. Thus delaying the verification queries just delays the success but never prevents it. This means that the adversary $\mathcal{A}'$ succeeds if and only if $\mathcal{A}$ does. In the sequel, we assume that all adversaries first make all the generation queries followed by the verification queries.

– The calls to the verification oracle are of two types:
  - The input of the verification can be a pair (M, m), where m was never outputted by the generation oracle. We call this kind of query a *guess.*
  - The input of the verification can be a pair (M, m), where m is the output of the generation oracle for a message M_0 with $M \neq M_0$. We call this kind of query a *collision check.*

– From the above point, we already know that the adversary $\mathcal{A}$ can be non-adaptative with respect to the verification queries. Now, we use the same argument as in [8] to show that in the proof of theorem 2, the adversary can be replaced by a non-adaptive adversary (with respect to all queries, generation and verification). Indeed, as long as $\mathcal{A}$ has not encountered a collision, he has only seen the images of random functions on distinct points. Consequently, any adaptative strategy can be replaced by a non-adaptive strategy where $\mathcal{A}$ precomputes its queries under the assumption that the previous queries produce random values. In other words, there exists a non-adaptative adversary $\mathcal{A}'$ that performs as well as $\mathcal{A}$. Since the adversary $\mathcal{A}'$ is non-adaptative, we can compute its probability of success by applying lemmas 3 and 4.

Proof of Lemma 1. Let $\mathcal{A}$ be a distinguishing adversary that, without loss of generality (see above), is non-adaptative and first makes all its generation queries followed by all its verification queries. Let us bound the probability for $\mathcal{A}$ of distinguishing a perfect randomized MAC from $\mathbf{RMAC}_{f_1,f_2}$ when $r = n$ and when the total length of the queries is bounded by $L \leq N/4$. Here f_1 is a random permutation and f_2 a family of random functions $f_2^{(R)}$. We proceed in two steps:

– We first bound the probability of success during the generation queries phase $\mathbf{Adv}_G(\mathcal{A})$. Let us assume that the adversary queries q messages (non necessarily different) $M_1, \ldots, M_q$. Since $\mathcal{A}$ is non-adaptative, the messages do not depend on f_1. This allows us to use lemmas 3 and 4 in order to compute $\mathbf{Adv}_G(\mathcal{A})$. The generation oracle allocates a random number of n bits to each message, $R_1, \ldots, R_q$. Whenever $M_i = M_j$, the oracle ensures that $R_i \neq R_j$ in order to avoid duplicates. Thus all pairs (M_i, R_i) are different. When the adversary runs against a perfect randomized MAC, he learns random independent numbers. When the adversary runs against $\mathbf{RMAC}_{f_1,f_2}$, we need to look at the pairs $(\mathbf{CBC}_{f_1}(M_i), R_i)$. If all these pairs are different, once again the adversary learns random independent numbers and thus gains nothing. Thus, we need to bound the probability of existence of a pair such that $(\mathbf{CBC}_{f_1}(M_i), R_i) = (\mathbf{CBC}_{f_1}(M_j), R_j)$.
Clearly, unless $R_i = R_j$ the two pairs cannot be equal. However, few messages share the same R. Indeed, if we let $No(R)$ denotes the number of messages such that $R_i = R$ we have:

$$\mathbf{Pr}\left[\exists R \text{ s.t. } No(R) \geq n\right] \leq \sum_R \left(\sum_{i_1 < i_2 \cdots < i_n} \mathbf{Pr}[R_{i_1} = R] \cdots \mathbf{Pr}[R_{i_n} = R] \right)$$

$$\leq \sum_R \left(\binom{q}{n}\left(\frac{1}{N}\right)^n\right) \leq 2^n \left(q^n \left(\frac{1}{N}\right)^n\right)$$
$$\leq 2^n \left(L^n \left(\frac{1}{N}\right)^n\right) \leq 2^n \left((N/4)^n \left(\frac{1}{N}\right)^n\right)$$
$$\leq 2^n (1/4)^n \leq \frac{1}{2^n}$$

With probability at most 2^{-n} some value of R is shared by more than n messages. In that case we suppose that the adversary wins. If this bad event does not occur, we now look for a collision on $\mathbf{CBC}_{f_1}(M_i)$ in groups of messages sharing R. According to lemma 3, the probability in a given group is smaller than $3q_R L_R 2^{-n}$, where q_R is the number of messages in the group and L_R is the sum of their length. Summing over all groups, we find an overall probability smaller than $3nL2^{-n}$. Adding together this probability with the probability of existence of a group larger than n, we get:

$$\mathbf{Adv}_G(\mathcal{A}) \leq \frac{3nL}{2^n} + \frac{1}{2^n}.$$

– Assuming that the adversary did not succeed during the generation query, we now look at the probability of success during the verification queries $\mathbf{Adv}_V(\mathcal{A})$. During this phase, the adversary can choose the value of R, the message M and a candidate value m. He learns whether these three values are compatible. However, he cannot choose a triple (R, M, m) when a MAC of the form (m', R) as been generated for M by the generation oracle. Indeed, in that case the answer is already known, it is `valid` if $m = m'$ and `invalid` otherwise. When the adversary runs against a perfect randomized MAC, he tests whether the value of a random function at a new point is m, the answer is thus a random event `valid/invalid`, with a probability 2^{-n} of outputting valid. When the adversary runs against $\mathbf{RMAC}_{f_1,f_2}$, we distinguish between *guesses* and *checks*. The adversary is said to make a *guess* query when the pair (m, R) was never outputted by the generation oracle. In that case, the fact that the MAC tags of messages are no longer independent can only decrease the probability of a correct *guess*. Since the number of queries is bounded from above by $N/4$ we cannot observe this phenomenom. Thus the adversary gains nothing when using *guesses*.
For *checks*, the oracle answers `valid` on query (M, R, m) where (m, R) was generated as a MAC for M_0, if and only if $\mathbf{CBC}_{f_1}(M) = \mathbf{CBC}_{f_1}(M_0)$. The adversary gains an advantage when such an event occurs. As in the generation case, we now make groups among the verification queries, one group for each pair (m, R) associated to a reference message M_0. Let $L(m, R)$ be the sum of the length of all messages in the group (including M_0), then according to lemma 4 the probability of existence of a message M in the group such that $\mathbf{CBC}_{f_1}(M) = \mathbf{CBC}_{f_1}(M_0)$ is bounded by $3L(m, R)2^{-n}$. Summing over all groups we bound $\mathbf{Adv}_V(\mathcal{A})$ by the following inequality:
$$\mathbf{Adv}_V(\mathcal{A}) \leq \frac{3L}{2^n}.$$
– Finally, adding $\mathbf{Adv}_G(\mathcal{A})$ and $\mathbf{Adv}_V(\mathcal{A})$ we conclude the proof of lemma 1.

Proof of theorem 2. In order to prove theorem 2, we start from lemma 1, we replace the family of random functions by a family of random permutations. In order to switch from functions to permutations, we are going to use the well-known lemma from [4]. We recall it here for the sake of completeness.

Lemma 5 (PRF/PRP Switching). *Fix $n \geq 1$. Let $\mathcal{A}$ be an adversary that ask at most p queries (to a computation oracle). Then*

$$\left| \boldsymbol{Pr}[\pi \stackrel{R}{\leftarrow} \boldsymbol{Perm}(n) | \mathcal{A}^\pi = 1] - \boldsymbol{Pr}[\rho \stackrel{R}{\leftarrow} \boldsymbol{Rand}(n,n) | \mathcal{A}^\rho = 1] \right| \leq \frac{p(p-1)}{2^{n+1}}.$$

Note that this lemma is affected by birthday paradox, since the bound on the advantage increases with the square of p. When using it, it is important to check that the switched function is called a small number of times only.

As in lemma 1, the adversary first makes its generation queries and then its verification queries. However, it is no longer non-adaptative in the verification phase. Indeed, when guessing the value of a random permutation on a new point, we clearly gain when excluding previously generated values.

We first prove that the generation queries do not help the adversary much and then that the verification queries do not either. According to the proof of lemma 1, we know that each permutation/function $f_2^{(R)}$ is used at most n times during the generation queries. Otherwise, we have already declared that the adversary has won. If q_R denotes the number of calls to $f_2^{(R)}$, the PRF/PRP switching lemma 5 states that the advantage for distinguishing the random permutation $f_2^{(R)}$ from a random function is bounded by $q_R^2/2^{n+1}$. Summing over all possible values of R, we find that during the generation phase, the adversary gains an advantage of:

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

During the verification phase, the adversary wins when he distinguishes the random permutations $f_2^{(R)}$ from random functions. This can occurs in two different ways. The first way is when on the random function side the adversary successfully *checks* a MAC generated for a reference message M_0 on some other message M while $\mathbf{CBC}_{f_1}(M) \neq \mathbf{CBC}_{f_1}(M_0)$. This happens with probability 2^{-n} for each new verification. When f_2 is a family of random permutations, the second way is with *guesses*. In that case, the probability of guessing the value of a permutation $f_2^{(R)}$ at a new point is bounded by $1/(2^n - n)$. Indeed, at most n values of $f_2^{(R)}$ are known from the generation phase and they should be excluded from the possible values. Note that the advantage gained for a *guess* or for a check are very similar. Summing over all verification (at most L), we see that the adversary gains an advantage of:

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

Adding $\mathbf{Adv}_G^{(2)}(\mathcal{A})$ and $\mathbf{Adv}_V^{(2)}(\mathcal{A})$, we find that the total advantage gained when going from theorem 2 to theorem 1 is:

$$\begin{aligned}\mathbf{Adv}^{(2)}(\mathcal{A}) &\leq \frac{nL}{2^{n+1}} + \frac{L}{2^n - n} \\ &\leq \frac{nL/2 + 2L}{2^n} \leq \frac{nL + L}{2^n},\end{aligned}$$

when $n \geq 2$. This concludes the proof of theorem 2, since

$$\mathbf{Adv}_G^{\mathbf{Rmac}}(\mathcal{A}) \leq \mathbf{Adv}_F^{\mathbf{Rprf}}(\mathcal{A}) + \mathbf{Adv}^{(2)}(\mathcal{A}).$$

C Full proof of theorem 3

Let $\mathcal{A}$ be an adversary trying to forge. We are going to compare the probability of success for $\mathcal{A}$ in two games. In game 0, $\mathcal{A}$ asks his queries to the four oracles G_f, V_f, C_f and C_f^{-1} that work as previously described. In game 1, these oracles are slightly modified. When G_f or V_f need to use a value of a permutation of the family F_2 this value is randomly chosen, except for the case when this value has already been asked, in this case the same value as before is chosen. The value is of course randomly chosen among the possible values left, in order to respect the fact that elements of F_2 are permutations. When C_f or C_f^{-1} need to use a value of a permutation of the family F_3 this value is also randomly chosen with the same process as with G_f or V_f, but the chosen values are independent from the values chosen with these oracles. At the end of game 1, a random value is chosen for the key K and the families F_2 and F_3 are completed, if possible, so that F_2 is a permuted copy of F_3. If it is not possible, they are completed randomly.

We define the following events. The event S_0 is the success of $\mathcal{A}$ in game 0. The event S_1 is the success of $\mathcal{A}$ in game 1. The event F corresponds to the case in game 1 where the family F_2 indexed by R is not the same as family F_3 indexed by $K \oplus R$.

It is clear that $\mathbf{Pr}\left[S_0 \wedge \neg F\right] = \mathbf{Pr}\left[S_1 \wedge \neg F\right]$, since if the answers for F_2 and F_3 are compatible, we are in the case where the small part of the family F_2 we have constructed during the execution is indeed a permuted copy of the small part of the family F_3 constructed independently. So there is no difference between the two games from the point of view of $\mathcal{A}$.

We can thus apply the following lemma:

Lemma 6. *Let S_0, S_1 and F be events verifying:*

$$\boldsymbol{Pr}[S_0 \wedge \neg F] = \boldsymbol{Pr}[S_1 \wedge \neg F]$$

then we have:

$$|\boldsymbol{Pr}[S_0] - \boldsymbol{Pr}[S_1]| \leq \boldsymbol{Pr}[F].$$

Proof of the lemma.

$$\begin{aligned}
|\mathbf{Pr}[S_0] - \mathbf{Pr}[S_1]| &= |\mathbf{Pr}\left[S_0 \wedge F\right] + \mathbf{Pr}\left[S_0 \wedge \neg F\right] - \mathbf{Pr}\left[S_1 \wedge F\right] - \mathbf{Pr}\left[S_1 \wedge \neg F\right]| \\
&= |\mathbf{Pr}\left[S_0 \wedge F\right] - \mathbf{Pr}\left[S_1 \wedge F\right] + \mathbf{Pr}\left[S_0 \wedge \neg F\right] - \mathbf{Pr}\left[S_1 \wedge \neg F\right]| \\
&= |\mathbf{Pr}\left[S_0 \wedge F\right] - \mathbf{Pr}\left[S_1 \wedge F\right]| \\
&= |\mathbf{Pr}\left[S_0|F\right].\mathbf{Pr}[F] - \mathbf{Pr}\left[S_1|F\right].\mathbf{Pr}[F]| \\
&= |\mathbf{Pr}\left[S_0|F\right] - \mathbf{Pr}\left[S_1|F\right]|.\mathbf{Pr}[F] \\
&\leq \mathbf{Pr}[F]
\end{aligned}$$

Now we want to evaluate $\mathbf{Pr}[F]$.

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. Let $f_2^{(R)}$ be the permutation in F_2 indexed by R and $f_3^{(R\oplus K)}$ be the permutation in F_3 indexed by $K \oplus R$. The answers are compatible if those two permutations are equal. That is, if $f_2^{(R)}$ and $f_3^{(R\oplus K)}$ were evaluated on the same point X, we must have $f_2^{(R)}(X) = f_3^{(R\oplus K)}(X)$; if $f_2^{(R)}(X) = f_3^{(R\oplus K)}(X')$, we must have $X = X'$; and if $f_2^{(R)}(X) = V_{K\oplus R}$ and $f_3^{(K\oplus R)^{-1}}(V_{K\oplus R}) = X'$, we must have $X = X'$. So for a given 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 plus the probability that $f_2^{(R)}$ and $f_3^{(R\oplus K)}$ have one common output plus the probability that $f_3^{(K\oplus R)^{-1}}$ has been evaluated on a value $V_{K\oplus R}$ that was the output of $f_2^{(R)}$ on some message input M_R. Since the simulator randomly 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. The queries X to F_3 are thus independent from the answers to the queries Y to F_2. The adversary cannot not adapt its strategy concerning F_2 with the answers from F_3.

We will distinguish the incompatibility coming from the queries to the oracle C_f from the incompatibility coming from the queries to C_f^{-1}. We write :

$$\mathbf{Pr}[F] = \mathbf{Pr}_{C_f}[F] + \mathbf{Pr}_{C_f^{-1}}[F].$$

We first evaluate $\mathbf{Pr}_{C_f}[F]$. If we let Y_R represents a query Y done to $f_2^{(R)}$ and $X_{K\oplus R}$ a query X done to $f_3^{(K\oplus R)}$, the probability that one query Y_R to F_2 is incompatible with all queries $X_{K\oplus R}$ in F_3 is:

$$\mathbf{Pr}_{C_f}[F] \leq \sum_{X_{K\oplus R}} \left(\mathbf{Pr}[f_2^{(R)}(Y_R) = f_3^{(K\oplus R)}(X_{K\oplus R})] + \mathbf{Pr}[Y_R = X_{K\oplus R}]\right).$$

Let M_R be the message associated to Y_R, that is $Y_R = CBC_{f_1}(M_R)$, we have:

$$\mathbf{Pr}[Y_R = X_{K\oplus R}] = \mathbf{Pr}[CBC_{f_1}(M_R) = X_{K\oplus R}].$$

As seen in the proof of lemma 2, we have

$$\mathbf{Pr}[CBC_{f_1}(M_R) = X_{K\oplus R}] \leq \frac{|M_R|}{2^n}.$$

Similarly, since $f_2^{(R)}$ and $f_3^{(K\oplus R)}$ are permutations, we have:

$$\mathbf{Pr}[f_2^{(R)}(Y_R) = f_3^{(K\oplus R)}(X_{K\oplus R})] = \mathbf{Pr}[f_2^{(R)}(CBC_{f_1}(M_R)) = f_3^{(K\oplus R)}(X_{K\oplus R})] \leq \frac{|M_R|}{2^n}.$$

For a given key K, the probability of event F is the sum over all queries Y_R of the probability that one query Y_R to F_2 is incompatible with all queries $X_{K\oplus R}$ in F_3. For a given key K, we obtain:

$$\mathbf{Pr}_{C_f}[F|K] \leq \sum_R \left(\sum_{M_R} \left(\sum_{X_{K\oplus R}} \frac{2|M_R|}{2^n} \right) \right).$$

Let $n_{K\oplus R}$ be the number of queries done to the permutation $f_3^{(K\oplus R)}$ of F_3 and let $L_{M_R} = \sum_{M_R} |M_R|$. We get:

$$\mathbf{Pr}_{C_f}[F|K] \leq \sum_R n_{K\oplus R} \frac{2L_{M_R}}{2^n}$$

We sum up for all K and divide by 2^n to get the probability for a randomly chosen K. Since in practice access to the oracles C_f and C_f^{-1} are cheaper than access to V_f and G_f (they represent offline computations), we authorize the adversary to do 2^n queries to those oracles. This is a natural limit since with 2^n computations, an adversary can already achieve exhaustive search on the MAC tag and can thus win. So we have $\sum_R n_{K\oplus R} = 2^n$ and that gives us:

$$\mathbf{Pr}_{C_f}[F] \leq \frac{1}{2^n} \sum_K \sum_R n_{K\oplus R} \frac{2L_{M_R}}{2^n} = \frac{1}{2^n} \sum_R \left(\frac{2L_{M_R}}{2^n} \sum_K n_{K\oplus R} \right) = \frac{1}{2^n} \sum_R \frac{2L_{M_R}}{2^n} 2^n = \sum_R \frac{2L_{M_R}}{2^n}.$$

Finally:

$$\mathbf{Pr}_{C_f}[F] \leq \frac{2L}{2^n}.$$

We now evaluate $\mathbf{Pr}_{C_f^{-1}}[F]$. It is the probability that we have asked the value of ${f_3^{(R')}}^{-1}$ on a value $V_{R'}$ that was the output of $f_2^{(R)}$ on some message M_R with R such that $R = R' \oplus K$.

For a given K and a given R' this probability is the probability that $R' = R \oplus K$ for all R such that a query $f_2^{(R)}(CBC_{f_1}(M_R))$ gave back the value $V_{R'}$.

The probability that $R' = R \oplus K$ is $1/2^n$ but we may have up to n collisions in the outputs of G_f, that is n different Rs such that $f_2^{(R)}(CBC_{f_1}(M_R)) = V_{R'}$.

So for a given key we have a probability

$$\mathbf{Pr}_{C_f^{-1}}[F|K] \leq \sum_{\text{all queries to } C_f^{-1}} \frac{n}{2^n}$$

Let T be the number of queries to C_f^{-1}, we sum up for all possible values of K:

$$\mathbf{Pr}_{C_f^{-1}}[F] \leq \frac{1}{2^n}\sum_K \frac{nT}{2^n} = \frac{nT}{2^n}.$$

Summing up the two contributions, we find that:

$$\mathbf{Pr}[F] \leq \frac{2L}{2^n} + \frac{nT}{2^n}.$$

We have thus proven that:

$$|\mathbf{Pr}[S_0] - \mathbf{Pr}[S_1]| \leq \mathbf{Pr}[F] \leq \frac{2L+nT}{2^n}.$$

Recall that $\mathbf{Adv}_{\mathcal{H}}^{\mathbf{Rmac}}(\mathcal{A}) = \mathbf{Pr}[f \stackrel{R}{\leftarrow} \mathcal{H}|\mathcal{A}^{G_f,V_f,C_f,C_f^{-1}}\text{forges}]$. By definition of game 0 and of event S_0 we have $\mathbf{Adv}_{\mathcal{H}}^{\mathbf{Rmac}}(\mathcal{A}) = \mathbf{Pr}[S_0]$. The probability of event S_1 correspond to the probability of $\mathcal{A}$ winning when the oracles C_f and $C_f^{(-1)}$ answer at random, it is thus equal to the probability of $\mathcal{A}$ winning when he has not access to those oracles. By definition, this corresponds to $\mathbf{Adv}_{\mathcal{G}}^{\mathbf{Rmac}}(\mathcal{A})$, so:

$$|\mathbf{Adv}_{\mathcal{H}}^{\mathbf{Rmac}}(\mathcal{A}) - \mathbf{Adv}_{\mathcal{G}}^{\mathbf{Rmac}}(\mathcal{A})| \leq \frac{2L+nT}{2^n}$$

$$\mathbf{Adv}_{\mathcal{H}}^{\mathbf{Rmac}}(\mathcal{A}) \leq \mathbf{Adv}_{\mathcal{G}}^{\mathbf{Rmac}}(\mathcal{A}) + \frac{2L+nT}{2^n}$$

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

This concludes the proof of theorem 3.