# Formal Security Proof of CMAC and its Variants

Cécile Baritel-Ruet, François Dupressoir, Pierre-Alain Fouque and Benjamin Grégoire

April 28, 2018

# Formal Security Proof of CMAC and its Variants

Cécile Baritel-Ruet[1,2], François Dupressoir[3], Pierre-Alain Fouque[4], and Benjamin Grégoire[2]

[1]UCA Université Côte d'Azur
[2]INRIA Sophia-Antipolis
[3]University of Surrey
[4]Université de Rennes 1 and Institut Universitaire de France

***Abstract*—The CMAC standard, when initially proposed by Iwata and Kurosawa as OMAC1, was equipped with a complex game-based security proof. Following recent advances in formal verification for game-based security proofs, we formalize a proof of unforgeability for CMAC in EasyCrypt. A side effect of this proof includes improvements and extensions to EasyCrypt's standard libraries. This formal proof obtains security bounds very similar to Iwata and Kurosawa's for CMAC, but also proves secure a certain number of intermediate constructions of independent interest, including ECBC, FCBC and XCBC. This work represents one more step in the direction of obtaining a reliable set of independently verifiable evidence for the security of international cryptographic standards.**

## I. Introduction

A *message authentication code* (MAC) is a short piece of information, a *tag*, appended to a message and used to confirm that the message came from the stated sender (*authenticity*) and has not been modified (*integrity*). When a MAC scheme–the algorithm used to produce the tag–is secure, it is unfeasibly difficult to produce a tag for a fresh message without knowing the secret key, even with the ability to obtain valid tags for chosen messages.

**The CMAC scheme** was first introduced by Iwata and Kurosawa as OMAC1, for One-key MAC, in [IK03a]. It is now a standardized MAC [Dwo16], widely used in practice. The scheme is based on the *cipher block chaining* MAC, shortened as CBC-MAC [EMST78], a very simple and textbook construction that builds a MAC for fixed-length messages out of a *block cipher*, a keyed permutation operation over *blocks*, or fixed-length bitstrings.

In the CBC-MAC scheme, the message to authenticate is split into a list of blocks. Each of these blocks is first masked using the result of the block cipher on the previous block before being encrypted itself, constructing a chain linked by the output of the block cipher. This dependence ensures that a change to any bit of a message will change the rest of the MAC in an unpredictable way. However, in its simplest form, CBC-MAC works only for messages that are non-empty lists of blocks, i.e whose length is a multiple of the size of a block.

Historically, to overcome this restriction and avoid unnecessary paddings, Black and Rogaway [BR05] propose and prove the security of some variants of CBC-MAC, named ECBC, FCBC and XCBC. However, each requires a key longer than the one of the underlying block cipher key. A variant of XCBC, CMAC was introduced as the first One-key CBC-MAC by Iwata and Kurosawa [IK03a]. This scheme uses only one block cipher key and was proven secure by Iwata and Kurosawa [IK03a] in a complex security proof that was later refined [IK03b].

*a) Our Contribution:* This paper is the first formalization verifying a concrete bound of the security of CMAC. We use the interactive computer-aided verification tool EasyCrypt[1] [BDG+14]. We reduce the security of CMAC as a MAC to the security of the underlying block cipher, without any aditional assumptions. The formalisation work makes use of EasyCrypt's rich libraries of game transformations and generic arguments, that we improve and extend with some generic lemmas. All results presented as Lemma or Theorem in this paper are formally verified, and potential unverified improvements will be explicitly noted as such.

Our proof is articulated as illustrated in Figure 1. The security of both the CMAC and XCBC schemes rely on the security proof of FCBC. We prove that the security of FCBC is equivalent to the security of ECBC. Finally, the security of ECBC relies on the fact that there is very little probability of collision in CBC-MAC, i.e. that two different messages chosen before the key is sampled share the same CBC-MAC value.

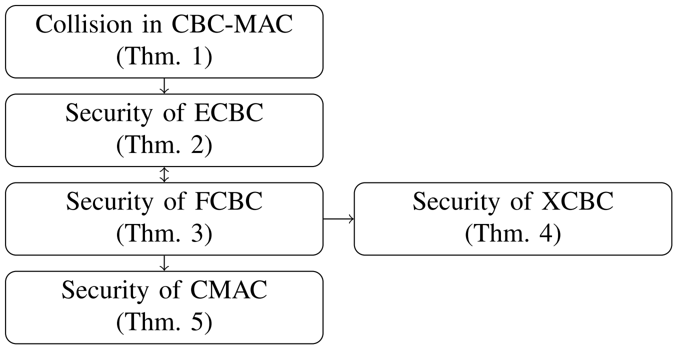


Fig. 1. Proof sketch

Our proof for CMAC is inspired from those by Iwata and Kurosawa [IK03a], [IK03b], but diverges from them and improves on them. In particular, our proof that CMAC is indistinguishable from FCBC is a slight generalisation of theirs:

[1]https://easycrypt.info

we show that any member of a larger class of MAC functions which includes CMAC is in fact indistinguishable from FCBC and therefore inherits its security. Our security proofs for FCBC, ECBC and XCBC, follow the proofs by Black and Rogaway [BR05]. Our formalisation efforts: i. reveal and correct a minor flaw in the proof (which has no effect on the bound), ii. produce a verified concrete bound—improving on some of their partial bounds, and iii. identify ways in which EasyCrypt falls short of supporting a full formalisation of Black and Rogaway's bounds.

Our formalization efforts also contribute to EasyCrypt and its libraries, producing: i. a tightening of the formalised bounds for the PRP/PRF switching lemma, ii. generic formalisations of some useful PRF-related game transformations, and iii. a new abstraction over reordering of random samplings in loops, allowing the EasyCrypt user to perform such transformations without making use of undocumented low-level tactics.

We start by informally introducing the constructions and their security (Section II), before formalizing definitions of the security assumptions and theorem statements we prove in EasyCrypt (Section III). We then give an overview of the proofs themselves (Section IV), detailing them, in particular, when our formalization diverges from existing pen-and-paper arguments [BR05], [IK03a], [IK03b]. We conclude (Section V) with a discussion of related work on the formalization of cryptography and of leads for future work on MAC functions in particular.

## II. Constructing CMAC: An Informal View

We first give an informal overview of syntax and security definitions for *message authentication codes* and *block ciphers*, the primitive underlying the CMAC construction. We then give an overview of intermediate constructions considered here – for which we obtain formal proofs of independent interest.

### A. Message Authentication Codes

We first give an informal overview of MAC schemes, their syntax, correctness and security. Formal definitions are given later. A MAC scheme is composed of three algorithms:

- a probabilistic *key generation* algorithm $\mathsf{Keygen}$ that, on input the security parameter outputs a key $\mathsf{k} \in \mathcal{K}$ (where $\mathcal{K}$ is the key space);
- a *tagging* algorithm $\mathsf{MAC}$ that, on input a key $\mathsf{k} \in \mathcal{K}$ and a message $\mathsf{m} \in \mathcal{M}$ outputs a tag $\mathsf{t} \in \mathcal{T}$ (where $\mathcal{M}$ is the message space and $\mathcal{T}$ is the tag space);
- a *verification* algorithm $\mathsf{Verify}$ that, on input a key $\mathsf{k} \in \mathcal{K}$, a message $\mathsf{m} \in \mathcal{M}$ and a tag $\mathsf{t} \in \mathcal{T}$, outputs a boolean identifying whether $\mathsf{t}$ is a valid tag for $\mathsf{m}$ with key $\mathsf{k}$.

*a) Correctness:* A MAC scheme is said to be *correct* if, for any key $\mathsf{k}$ output by the key generation algorithm and any message $\mathsf{m} \in \mathcal{M}$, a tag $\mathsf{t}$ computed for $\mathsf{m}$ using key $\mathsf{k}$ by the tagging algorithm always verifies successfully.

*b) Security:* A MAC scheme is said to be (existentially) *unforgeable* (against chosen message attacks) if, for any freshly generated key $\mathsf{k}$, it is difficult for an adversary that can make tag and verification queries for arbitrary messages using $\mathsf{k}$, to *forge* a valid tag $\mathsf{t}$ for a *fresh* message $\mathsf{m}$, for which a tag was not requested.

*c) Deterministic MACs:* In this paper, and for the purpose of reasoning about CMAC and its variants, we need consider only a restricted class of MAC schemes and assume:

1) a key generation algorithm that samples its output uniformly at random in $\mathcal{K}$;
2) a *deterministic* tagging algorithm; and
3) the verification algorithm defined by: $\mathsf{Verify}(\mathsf{m}^*, \mathsf{t}^*) := [\mathsf{t} \leftarrow \mathsf{MAC}(\mathsf{m}^*); \mathsf{t} = \mathsf{t}^*]$.

These standard restrictions mean that our MAC schemes can be fully defined by specifying only: i. a (finite) keyspace $\mathcal{K}$, ii. a message space $\mathcal{M}$, iii. a tag space $\mathcal{T}$, and iv. a tagging algorithm $\mathsf{MAC}$.

We also discuss, in Section III, other simplifications that follow from the consideration of stateless, deterministic MACs.

### B. Block Ciphers

CMAC and its variants construct secure MACs using a *block cipher* as primitive. A block cipher is a family of permutations over a finite set of *blocks*, say $\mathbb{B}$, indexed by a keyspace $\mathcal{K}$. As is standard, given a block cipher $E : \mathcal{K} \times \mathbb{B} \to \mathbb{B}$ and a key $\mathsf{k} \in \mathcal{K}$, we often denote with $E_\mathsf{k}$ the block cipher with fixed key $\mathsf{k}$. That is, for all $\mathsf{b} \in \mathbb{B}$, $E_\mathsf{k}(\mathsf{b}) = E(\mathsf{k}, \mathsf{b})$.

*a) Security:* A block cipher is said to be secure if, when its key is sampled uniformly at random in $\mathcal{K}$, it is computationally indistinguishable from a truly random permutation, sampled uniformly at random in the set of permutations over $\mathbb{B}$, denoted $\mathsf{Perm}(\mathbb{B})$.

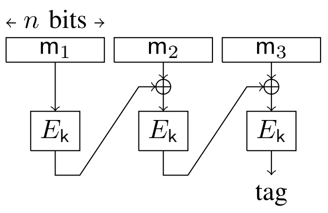
### C. Constructing CMAC

We now give intuitive descriptions of the various historical constructions leading up to the definition of CMAC. Apart from allowing us to illustrate the informal definitions above on increasingly complex schemes, many of these intermediate constructions also serve as intermediate steps in the security proof for CMAC.

*a) CBC-MAC:* Figure 2 gives both a graphical and programmatic view of the CBC-MAC tagging algorithm, which was initially introduced by Ehrsam et al. [EMST78].

CBC-MAC relies on a block cipher $E : \mathcal{K} \times \mathbb{B} \to \mathbb{B}$. Given some key space $\mathcal{K}$ and some block set $\mathbb{B}$, CBC-MAC instantiates the message space as $\mathbb{B}^m$ for some $m > 0$ and outputs tags in $\mathbb{B}$. Given an input $\mathsf{m}$, CBC-MAC splits it into blocks $\mathsf{m} = \mathsf{m}_1 || \ldots || \mathsf{m}_m$, and computes the tag as $\mathsf{CBC}_{E_\mathsf{k}}(\mathsf{m}) = \mathsf{c}_m$ where $\mathsf{c}_1 = E_\mathsf{k}(\mathsf{m}_1)$, $\mathsf{c}_{i+1} = E_\mathsf{k}(\mathsf{c}_i \oplus \mathsf{m}_{i+1})$, and $\oplus$ denotes the bitwise xor operation on blocks.

Bellare, Kilian and Rogaway [BKR94] prove the security of the CBC-MAC construction, by proving, as is the practice in cryptography, that any algorithm that finds CBC-MAC forgeries with non-negligible probability can be used to break the



```
CBC_{E_k}(m_1 || ... || m_m)
c ← E_k(m_1);
for i = [2 ... m] do
    c ← E_k(c ⊕ m_i);
return c
```

Fig. 2. Computing CBC-MAC with block cipher $E$ and key $k$ on an $m$ block message.

underlying block cipher with the same time complexity. However, their proof—and indeed the security of CBC-MAC—is limited to the case where the length $m$ of the message (in blocks) is fixed in advance. Indeed, it is easy to produce a chosen message forgery as follows when the tagging and verification algorithm accept arbitrary length messages:

1) Through chosen-message queries, obtain $t = \mathrm{CBC}(\mathsf{m})$ and $t' = \mathrm{CBC}(\mathsf{m}')$ for some messages $\mathsf{m}$ and $\mathsf{m}'$ of respective lengths $m > 0$ and $m' > 0$.
2) Then $t'$ is also a valid tag for the *fresh* message:

$$\mathsf{m}_1 \mid\mid \ldots \mid\mid \mathsf{m}_m \mid\mid [t \oplus \mathsf{m}'_1] \mid\mid \ldots \mid\mid \mathsf{m}'_{m'}$$

Thus CBC-MAC suffers from two major issues: i. the size of messages should be a multiple of $n$, and ii. all messages must contain the same, fixed, number of blocks. To overcome these restrictions, Black and Rogaway [BR05] propose three extensions of CBC-MAC:[2] i. ECBC, ii. FCBC and iii. XCBC. CMAC is a variant of XCBC that was first proposed by Iwata and Kurosawa [IK03a] as OMAC1.

*b) ECBC:* The first issue of CBC-MAC, that is being limited to computing tags for messages whose length is a multiple of the block length, is often dealt with using some injective padding scheme. However, such schemes require flexibility in the number of blocks that can be processed, as their injectivity may require them to add a full block of padding onto a message. ECBC combines two ideas to: i. allow padding messages to a multiple of the block length without overhead; and ii. securely support computing MAC tags for messages of different lengths.

First, in order to support messages with different numbers of blocks, it is possible to apply additional treatment to the final tag before releasing it. This is to ensure that it cannot predictably be used as a known intermediate value in the computation of a tag for a chosen message. In ECBC, this is done, using an idea by Vaudenay [Vau00], by computing a CBC-MAC tag using a block cipher key $\mathsf{k}_1$ and encrypting this tag with a different key $\mathsf{k}_2$ before releasing it.

[2]A simpler fix is to be prepend the length of the input message to the message itself before processing it: if this does indeed yield a secure MAC algorithm, it is not always possible to know the length of the payload when the processing begins. For example, the TLS protocol computes a MAC over the concatenation of all messages exchanged during its handshake protocol–the length of which is only known after it is over. These engineering considerations are often kept separate from security considerations when defining the syntax and security of cryptographic schemes. This provides abstraction but brings its own set of problems.

This first idea then brings an opportunity to do padding only when needed. Indeed, rather than ensuring the padding is injective, it is possible to apply padding to the message only when its length is not a multiple of the block length, but then compute the final encryption using a different key $\mathsf{k}_3$ when padding was applied.

Given some injective padding function pad : $\Sigma^* \to \mathbb{B}^+$ that pads arbitrary strings over some alphabet $\Sigma$ as non-empty sequence of blocks, and combining the two ideas above, the ECBC scheme (also shown in Figure 3) can be defined as the following three-key construction.

$$\mathrm{ECBC}_{E_{\mathsf{k}_1},E_{\mathsf{k}_2},E_{\mathsf{k}_3}}(\mathsf{m}) = \begin{cases} E_{\mathsf{k}_2}(CBC_{\mathsf{k}_1}(\mathsf{m})) & \text{if } \mathsf{m} \in \mathbb{B}^+ \\ E_{\mathsf{k}_3}(CBC_{\mathsf{k}_1}(\mathrm{pad}(\mathsf{m})) & \text{otherwise} \end{cases}$$

*c) FCBC:* In ECBC, as shown in Figure 3, processing the last block of message involves two consecutive calls to the block cipher with independent keys. This brings an opportunity for optimization. Indeed, since block ciphers are meant to be indistinguishable from random permutations, a single block cipher invocation is in fact sufficient for security. The FCBC construction, shown in Figure 4, implements this idea. It is defined as follows.

$$\mathrm{FCBC}_{E_{\mathsf{k}_1},E_{\mathsf{k}_2},E_{\mathsf{k}_3}}(\mathsf{m}) = \begin{cases} E_{\mathsf{k}_2}(CBC_{\mathsf{k}_1}(\mathsf{m}_1 \| \ldots \| \mathsf{m}_{m-1}) \oplus \mathsf{m}_m) & \text{if } \mathsf{m} \in \mathbb{B}^+ \\ E_{\mathsf{k}_3}(CBC_{\mathsf{k}_1}(\mathsf{m}'_1 \| \ldots \| \mathsf{m}'_{m-1}) \oplus \mathsf{m}'_m) & \text{otherwise} \end{cases}$$

with $\mathsf{m}'_1 \| \ldots \| \mathsf{m}'_m = \mathrm{pad}(\mathsf{m})$.

*d) XCBC:* Both of the above constructions do solve CBC-MAC's main issues, with FCBC giving a small performance advantage. However, their use of block ciphers keyed with independent keys makes them expensive to compute using block ciphers like the AES [DR13]. Indeed, AES involves a costly key expansion phase that can only be amortized if the same key is used many times. To better suit this engineering constraint, the XCBC construction (Figure 5) uses the same key k for all block cipher invocations, and uses the other two keys $(k_2, k_3)$ to mask the final block before processing it. This is done without loss of security. Formally, XCBC is defined as follows.

$$\mathrm{XCBC}_{E_\mathsf{k},k_2,k_3} = \mathrm{FCBC}_{E_\mathsf{k},E_\mathsf{k}(k_2 \oplus \cdot),E_\mathsf{k}(k_3 \oplus \cdot)}$$

*e) CMAC:* One final performance constraint needs dealt with: the key size for XCBC is indeed $2n$ plus the key size for CBC-MAC. CMAC (also known as OMAC1) proposes to derive $k_2$ and $k_3$ from k using the block cipher. We note that this makes CMAC easier to express as a refinement of FCBC—rather than XCBC. In particular, the security of XCBC cannot be easily used to prove the security of CMAC, as it requires that the three keys be independent.

At this stage, it is necessary to concretely instantiate the block space $\mathbb{B} = \mathsf{GF}(2^n)$, that is represented as the quotient of the polynomial ring $\mathsf{GF}(2)[\mathsf{x}]$ by a fixed irreductible polynomial of degree $n$. With this choice of block space, CMAC derives $k_2$ and $k_3$ as $k_2 := 2 \times E_\mathsf{k}(0^n)$ (when 2 is seen as

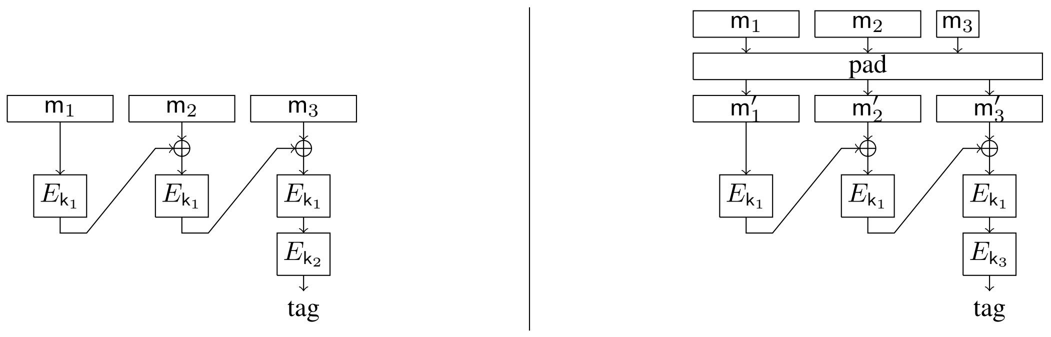


Fig. 3. Illustration of $\text{ECBC}_{E_{\mathsf{k}_1},E_{\mathsf{k}_2},E_{\mathsf{k}_3}}(\mathsf{m}_1||\mathsf{m}_2||\mathsf{m}_3)$

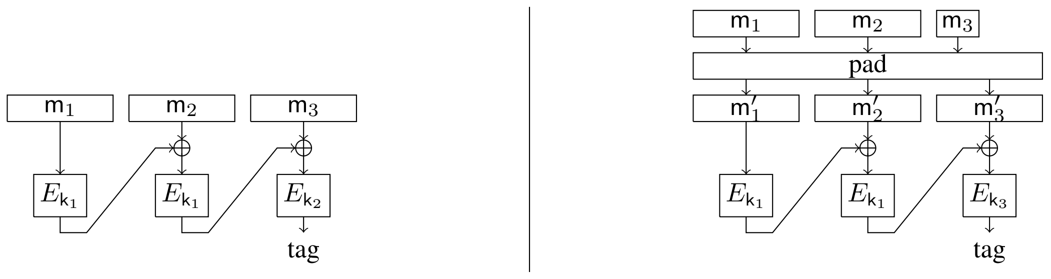


Fig. 4. Illustration of $\text{FCBC}_{E_{\mathsf{k}_1},E_{\mathsf{k}_2},E_{\mathsf{k}_3}}(\mathsf{m}_1||\mathsf{m}_2||\mathsf{m}_3)$

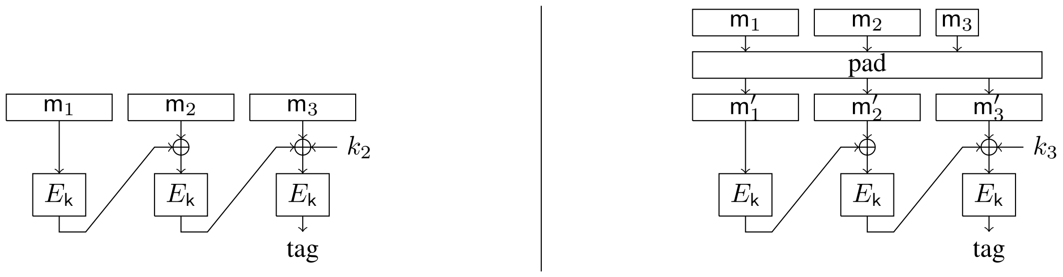


Fig. 5. Illustration of $\text{XCBC}_{E_{\mathsf{k}},k_2,k_3}(\mathsf{m}_1||\mathsf{m}_2||\mathsf{m}_3)$

`0x02`, or $0\dots010$, or the polynomial x) and $k_3 := 4 \times E_{\mathsf{k}}(0^n)$ (when 4 is seen as `0x04`, or $0\dots0100$, or the polynomial x$^2$), yielding the following definition.

$$\text{CMAC}_{E_{\mathsf{k}}} = \text{FCBC}_{E_{\mathsf{k}},E_{\mathsf{k}}([2\times E_{\mathsf{k}}(0^n)]\oplus\cdot),E_{\mathsf{k}}([4\times E_{\mathsf{k}}(0^n)]\oplus\cdot)}$$

## III. Formal Definitions

We now clarify the formal details of the various security notions needed to state our results. Our security definitions are expressed as standard cryptographic games and our theorems and their proofs are given in the code-based game-based style introduced by Bellare and Rogaway [BR06].

We express security using *games*, (probabilistic) algorithms that describe how an adversary can interact with a scheme to try and break its security and what such a break consists in. Security is then expressed as a property of the probability of such a break occurring for any adversary (within a certain admissible class of adversaries). Given a game $\mathcal{G}$ (seen as a sequence of commands, including calls to the adversary) and an event $E$, we denote with $\Pr[\mathcal{G} : E]$ the probability that event $E$ holds after executing $\mathcal{G}$.

In the following, we denote $X \leftarrow_{\$} \mathsf{Perm}(\mathbb{B})$ sampling a random variable $X$ in the uniform distribution over $\mathsf{Perm}(\mathbb{B})$, the set of all permutations in $\mathbb{B} \to \mathbb{B}$. We also denote $\mathsf{Rand}(\mathbb{B})$ and $\mathsf{Rand}(*, \mathbb{B})$, respectively, the uniform[3] distributions over $\mathbb{B} \to \mathbb{B}$ and $\{0,1\}^* \to \mathbb{B}$.

### A. Block cipher security

The security of ECBC, FCBC, XCBC and CMAC stand on a single cryptographic assumption: the block cipher $E$ is assumed to be a *secure pseudo-random permutation* (prp). This requirement is met when $E$ cannot be distinguished from a random permutation by any polynomial time adversary that can choose its inputs and view its outputs. The permutation oracle has the same interface as does the block cipher $E$ with a fixed key.

[3]We stress that $\{0,1\}^* \to \mathbb{B}$ is countably infinite. We abuse notation, and use the notation for sampling uniformly in that space to denote the instantiation of a lazy-sampling random function.

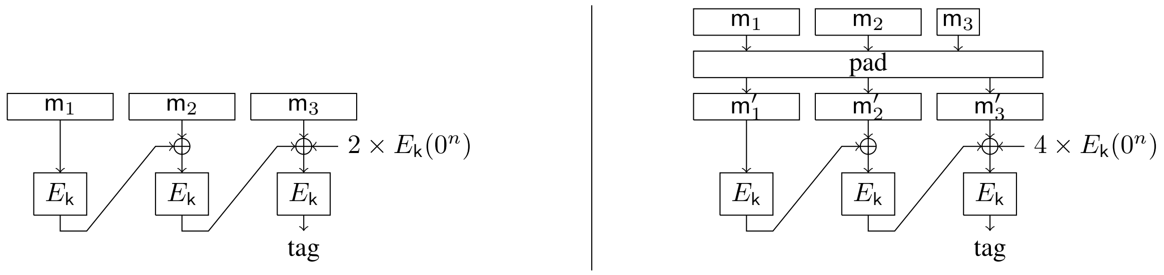


Fig. 6. Illustration of $\mathrm{CMAC}_{E_\mathsf{k}}(\mathsf{m}_1||\mathsf{m}_2||\mathsf{m}_3)$

For any adversary $\mathcal{A}$, we consider the game in which $\mathcal{A}$ is given oracle access to a permutation oracle $\mathsf{O}$ and is challenged to determine whether it is the block cipher $E_\mathsf{k}$ with a random selected key $\mathsf{k} \leftarrow_\$ \mathcal{K}$, or a random permutation $\pi \leftarrow_\$ \mathsf{Perm}(\mathbb{B})$. In the game, the adversary can adaptively make queries to $\mathsf{O}$ and receives its responses before simply returning a bit that indicates its guess as to which version of the oracle it was interacting with.

**Definition 1.** *The* PRP distinguishing advantage *of an adversary $\mathcal{A}$, denoted $\mathsf{Adv}^{\mathrm{prp}}_{E}(\mathcal{A})$, is defined as follows.*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{prp}}_{E}(\mathcal{A}) := \big|&\Pr\left[\mathsf{k} \leftarrow_\$ \mathcal{K}; b \leftarrow \mathcal{A}^{E_\mathsf{k}} : b = 1\right] \\ &- \Pr\left[\pi \leftarrow_\$ \mathsf{Perm}(\mathbb{B}); b \leftarrow \mathcal{A}^{\pi} : b = 1\right]\big|\end{aligned}$$

Formally, we say that a block cipher is secure if, for any efficient adversary $\mathcal{A}$, the advantage $\mathsf{Adv}^{\mathrm{prp}}_{E}(\mathcal{A})$ is small.

The first step in all the reductions we present later is to replace all block cipher with independent random keys by independent random permutations. This implies *an additional term* in all bounds shown below, which represents the adversary's ability to distinguish the blockcipher from a truly random permutation.

## *B. MAC security*

For any adversary $\mathcal{A}$, we consider the game in which $\mathcal{A}$ is given access to a tag generation oracle $\mathsf{MAC}_\mathsf{k}$ and a verification oracle $\mathsf{Verify}_\mathsf{k}$, for a randomly sampled $\mathsf{k}$ and is challenged to produce a valid tag for a fresh message. $\mathcal{A}$ is allowed to adaptively query its tagging and verification oracles, and is said to have produced a *forgery* if any of her queries to the $\mathsf{Verify}_\mathsf{k}$ oracle on a fresh message (that has not been queried to the $\mathsf{MAC}_\mathsf{k}$ oracle) succeeds.

**Definition 2.** *The* multi-challenge forgery advantage $\mathsf{Adv}^{\mathsf{m\text{-}mac}}_{\mathsf{MAC}}(\mathcal{A})$ *is the probability that the adversary succeeds in producing a forgery.*

$$\begin{aligned}&\mathsf{Adv}^{\mathsf{m\text{-}mac}}_{\mathsf{MAC}}(\mathcal{A}) := \\ &\quad \Pr\Big[\mathsf{k} \leftarrow_\$ \mathcal{K}; (\mathsf{m}^*, \mathsf{t}^*) \leftarrow \mathcal{A}^{\mathsf{MAC}_\mathsf{k}(\cdot), \mathsf{Verify}_\mathsf{k}(\cdot,\cdot)} : \\ &\quad \exists(\mathsf{m}^*, \mathsf{t}^*) \in \mathcal{L}_{\mathcal{A}}^{\mathsf{Verify}_\mathsf{k}(\cdot,\cdot)},\ \mathsf{m}^* \notin \mathcal{L}_{\mathcal{A}}^{\mathsf{MAC}_\mathsf{k}(\cdot)}{}_{|(\mathsf{m}^*,\mathsf{t}^*)} \\ &\qquad\qquad \wedge \mathsf{Verify}_\mathsf{k}(\mathsf{m}^*, \mathsf{t}^*) = 1\Big]\end{aligned}$$

*where $\mathcal{L}_{\mathcal{A}}^{\mathsf{MAC}_\mathsf{k}(\cdot)}{}_{|(\mathsf{m}^*,\mathsf{t}^*)}$ is the list of all queries made by $\mathcal{A}$ to its $\mathsf{MAC}_\mathsf{k}(\cdot)$ oracle before its first $(\mathsf{m}^*, \mathsf{t}^*)$ query to* $\mathsf{Verify}$, *and $\mathcal{L}_{\mathcal{A}}^{\mathsf{Verify}_\mathsf{k}(\cdot,\cdot)}$ is the list of all queries made by $\mathcal{A}$ to its $\mathsf{Verify}_\mathsf{k}(\cdot,\cdot)$ oracle.*

We note that, in the case of deterministic stateless MACs in which we place ourselves, the definition of unforgeability displayed above is equivalent to the following, which allows only a single query to the verification oracle. This was shown by Bellare et al. [BGM04]. Indeed, it is easy to see that, in this case, the Verify oracle can be perfectly implemented using the MAC oracle to which the adversary already has access. We use the definition below in the rest of the paper.

**Definition 3.** *The* forgery advantage $\mathsf{Adv}^{\mathsf{mac}}_{\mathsf{MAC}}(\mathcal{A})$ *is the probability that the adversary succeeds in producing a forgery.*

$$\begin{aligned}&\mathsf{Adv}^{\mathsf{mac}}_{\mathsf{MAC}}(\mathcal{A}) := \\ &\Pr\Big[\mathsf{k} \leftarrow_\$ \mathcal{K}; (\mathsf{m}^*, \mathsf{t}^*) \leftarrow \mathcal{A}^{\mathsf{MAC}_\mathsf{k}(\cdot)}; b \leftarrow \mathsf{Verify}_\mathsf{k}(\mathsf{m}^*, \mathsf{t}^*) : \\ &\qquad\qquad b = 1 \wedge \mathsf{m}^* \notin \mathcal{L}_{\mathcal{A}}^{\mathsf{MAC}_\mathsf{k}(\cdot)}\Big]\end{aligned}$$

*when $\mathcal{L}_{\mathcal{A}}^{\mathsf{MAC}_\mathsf{k}(\cdot)}$ is the list of all queries made by $\mathcal{A}$ to its oracle* $\mathsf{MAC}_\mathsf{k}(\cdot)$.

Formally, we say that a MAC scheme is secure when for any efficient adversary $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{mac}}_{\mathsf{MAC}}(\mathcal{A})$ remains small.

## *C. Pseudo-random Functions*

Another interesting property of keyed function families such as MAC schemes, is that of being pseudo-random. Intuitively, a function family is pseudo-random if a randomly sampled member of the family is computationally indistinguishable from a truly random function.

Formally, we consider the game in which the adversary $\mathcal{A}$ is given access to a tag generation oracle $\mathsf{O}$ and is challenged to determine whether it is the MAC scheme $\mathsf{MAC}_\mathsf{k}$ with a randomly selected key $\mathsf{k} \leftarrow_\$ \mathcal{K}$, or a random function $F \leftarrow_\$ \mathsf{Rand}(*, \mathbb{B})$ that has the same input/output spaces as the MAC scheme. In the game, the adversary makes adaptive queries to the oracle and receives its answers before returning a bit indicating its guess of the content of the oracle.

**Definition 4.** *The* PRF distinguishing advantage *of an adversary $\mathcal{A}$, denoted $\mathsf{Adv}^{\mathsf{prf}}_{\mathsf{MAC}}(\mathcal{A})$, is defined as follows.*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{prf}}_{\mathsf{MAC}}(\mathcal{A}) := \big|&\Pr\left[\mathsf{k} \leftarrow_\$ \mathcal{K}; b \leftarrow \mathcal{A}^{\mathsf{MAC}_\mathsf{k}} : b = 1\right] \\ &- \Pr\left[F \leftarrow_\$ \mathsf{Rand}(*, \mathbb{B}); b \leftarrow \mathcal{A}^{F} : b = 1\right]\big|\end{aligned}$$

Let us remark that PRF security is related to unforgeability.

**Lemma 1.** *For any forging adversary $\mathcal{A}$, there exists a distinguishing adversary $\mathcal{B}$ that uses $\mathcal{A}$ as a black box, and such that*

$$\mathsf{Adv}^{\mathsf{mac}}_{\mathsf{MAC}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathsf{MAC}}(\mathcal{B}(\mathcal{A})) + \frac{1}{|\mathbb{B}|}$$

*Proof.* Given an adversary $\mathcal{A}$ in the forgery game, we define $\mathcal{B}(\mathcal{A})^{\mathsf{O}}$ as the adversary that calls $\mathcal{A}^{\mathsf{O}}$ which outputs $(\mathsf{m}^*, \mathsf{t}^*)$, calls $\mathsf{O}(\mathsf{m}^*)$ and then return 1 if the output tag is the same as $\mathsf{t}^*$, and $\mathsf{m}^*$ has not been previously queried by $\mathcal{A}$, otherwise $\mathcal{B}$ returns 0. It is easy to see that $\mathsf{Adv}^{\mathsf{prf}}_{\mathsf{MAC}}(\mathcal{B}(\mathcal{A})) = \left|\mathsf{Adv}^{\mathsf{mac}}_{\mathsf{MAC}}(\mathcal{A}) - \frac{1}{|\mathbb{B}|}\right|$. In addition, it is easy to see that the PRF adversary $\mathcal{B}(\mathcal{A})$ makes exactly as many oracle queries as the initial adversary $\mathcal{A}$. □

Therefore it is sufficient to show that a MAC scheme is pseudo-random to prove that it is unforgeable.

*On concrete security:* We note here that our claims, such as the one illustrated in Lemma 1, are neither asymptotic, nor concrete in the standard sense of accounting precisely for oracle calls and execution time. Our formal proofs account precisely for oracle calls and constructively exhibit the reductions, which could be used to analyse the reduction's execution time. However, EasyCrypt itself does not formally support such an analysis, and we focus on presenting here the results as they are formalised. They are only semi-concrete, but can serve as support for a concrete analysis.

## IV. Security Proof

Our security proofs for ECBC, FCBC, XCBC and CMAC are in fact proofs that they cannot be distinguished from a random function assuming that *the underlying block cipher is secure*. In the following, superscripts are used to denote oracles to which the adversary has access.

We provide a verified security proof for the following statements. (The first three statements follow those by Black and Rogaway [BR05].)

a) ECBC is a secure MAC;
b) FCBC is perfectly indistinguishable from ECBC;
c) XCBC is computationally indistinguishable from FCBC; and
d) CMAC is computationally indistinguishable from FCBC.

*a) Security of ECBC:* To prove the security of ECBC, we formalize and rely on a much more general lemma, which states that, if $H \subseteq \mathbb{B}^+ \to \mathbb{B}$ is a family of hash functions, $\pi_2, \pi_3$ are random permutations and pad is an injective padding function, the forgery advantage of the following construction, when $h$ is a member of $H$ chosen uniformly at random.

$$\mathrm{EPAD}_h^{\pi_2,\pi_3}(M) = \begin{cases} \pi_2(h(M)) & \text{if } M \in \mathbb{B}^+ \\ \pi_3(h(\mathrm{pad}(M)) & \text{otherwise} \end{cases}$$

is only slightly higher than the *collision probability* of $h$.

This probability is defined as a game-based notion, through the following collision-finding game, in which the adversary $\mathcal{A}$ is asked to provide a list of messages of its choice, *before* the hash function $h$ is randomly selected from $H$. The adversary $\mathcal{A}$ succeeds if the list of messages initially chosen contains two distinct messages that have the same image by $h$. This yields the following definition of an adversary $\mathcal{A}$'s *collision-finding* advantage against $H$.

$$\Pr[l \leftarrow \mathcal{A}; h \leftarrow\!\!\$\, H : \exists\, m, m' \in l.\ m \neq m' \wedge h(m) = h(m')]$$

In this paper, we say that a hash function is *collision-resistant* if this probability is small.[4]

We also formalize the collision-resistance of CBC-MAC as a family of hash functions indexed by the block cipher key. This yields a concrete bound on the security of ECBC.

*b) Security of FCBC:* The security of FCBC is equivalent to the security of ECBC. This is due to the fact that the composition of two independent random permutations remains independent from one of the permutations.

*c) Security of XCBC:* The proof that XCBC is indistinguishable from FCBC is based on a more general lemma from [BR05], which states that an adversary with oracle access to two independent random permutations $\pi_1(\cdot)$ and $\pi_2(\cdot)$ cannot distinguish them from oracles $\pi(\cdot)$ and $\pi(k \oplus \cdot)$, when $\pi$ is a random permutation and $k$ is chosen uniformly at random and independently from $\pi$ (and remains hidden from the adversary).

*d) Security of CMAC:* The security of CMAC cannot be easily deduced from that of XCBC, since the core lemma of the security of XCBC imposes that the masks be independent from the permutation. Since this is not the case for CMAC, we use ideas from Iwata and Kurosawa [IK03a] to relate the security of CMAC to that of FCBC. Their key idea is to prevent the adversary from directly accessing the block cipher oracle by adding an independent random variable.

### A. Security of ECBC

We recall that CBC-MAC, as a mode of operation, can be seen as a function family in $\mathbb{B}^+ \to \mathbb{B}$ indexed by the key space $\mathcal{K}$. Sampling a key uniformly at random in $\mathcal{K}$ induces a distribution $H = \mathrm{CBC}_{\mathcal{K}}$ over $\mathbb{B}^+ \to \mathbb{B}$. We define $\mathrm{EPAD}^{\mathsf{p}}_H$ as the function family (using permutations and indexed by $H$) containing functions of the form

$$\mathrm{EPAD}_h^{\pi_2,\pi_3} = M \mapsto \begin{cases} \pi_2(h(M)) & \text{if } M \in \mathbb{B}^* \\ \pi_3(h(\mathrm{pad}(M))) & \text{otherwise} \end{cases}$$

where $\pi_2, \pi_3$ are independent random permutations and $h \leftarrow\!\!\$\, H$. We now prove that the MAC security of $\mathrm{EPAD}^{\mathsf{p}}_H$ follows from the collision resistance of $H$ as a function family.

First, we replace the independent random permutations $\pi_2, \pi_3$ by independent random functions $f_2, f_3$ using the PRP/PRF switching lemma (Lemma 2) twice.

**Lemma 2** (PRP/PRF switching)**.** *For any natural number c, any adversary $\mathcal{A}$ that can make at most c queries to its oracle can only distinguish a random permutation in* $\mathsf{Perm}(\mathbb{B})$ *from*

[4]We note that this is weaker than the usual notion of collision-resistance for hash functions, which requires resistance against adaptive queries.

*a random function in $\mathbb{B} \to \mathbb{B}$ with low probability. More formally, the following inequality holds.*

$$\begin{aligned}&|\Pr[\pi \leftarrow_{\$} \mathsf{Perm}(\mathbb{B}) : \mathcal{A}_c^{\pi} = 1]\\ &\quad - \Pr\big[f \leftarrow_{\$} \mathsf{Rand}(\mathbb{B}) : \mathcal{A}_c^{f} = 1\big]\big| \leq \frac{c \cdot (c-1)}{2 \cdot |\mathbb{B}|}\end{aligned}$$

We apply twice the PRP/PRF switching lemma on $\pi_2$ and then on $\pi_3$. Following our oracle query accounting, the adversary could cause either of these to be called $q$ times, yielding the following probability bound.

$$\begin{aligned}&\Big|\Pr\Big[h \leftarrow_{\$} H; \pi_2, \pi_3 \leftarrow_{\$} \mathsf{Perm}(\mathbb{B})^2; b \leftarrow \mathcal{A}_{\sigma,q}^{\mathrm{EPAD}_h^{\pi_2,\pi_3}} : b = 1\Big]\\ &- \Pr\Big[h \leftarrow_{\$} H; f_2, f_3 \leftarrow_{\$} \mathsf{Rand}(\mathbb{B})^2; b \leftarrow \mathcal{A}_{\sigma,q}^{\mathrm{EPAD}_h^{f_2,f_3}} : b = 1\Big]\Big|\\ &\leq \frac{q(q-1)}{|\mathbb{B}|}\end{aligned}$$

From now on, we refer using $\mathrm{EPAD}_H$ to the function family that mirrors $\mathrm{EPAD}_H^{\mathsf{p}}$ using functions instead of permutations, and thus contains functions of the form

$$\mathrm{EPAD}_h^{f_2,f_3} = M \mapsto \begin{cases} f_2(h(M)) & \text{if } M \in \mathbb{B}^* \\ f_3(h(\mathrm{pad}(M))) & \text{otherwise} \end{cases}$$

when $f_2, f_3$ are independent random functions and $h \leftarrow_{\$} H$. As $f_2, f_3$ are two independent random functions, the output of $\mathrm{EPAD}_h^{f_2,f_3}$ on fresh messages will be sampled at random unless $h$ maps it to an output value it has already produced for another query, which is a collision in $h$. Formalizing this argument, we prove that $\mathrm{EPAD}_h^{f_2,f_3}$ is indistinguishable from a random function unless a collision, as formally defined below, occurs in $h$.

$$\begin{aligned}\mathcal{C}oll_h(M, M') &:= M \neq M' \wedge h(M) = h(M')\\ \mathcal{C}oll_h(S) &:= \exists (M, M') \in S^2, \mathcal{C}oll_h(M, M')\end{aligned}$$

This proof is a standard cryptographic reduction, whereby, given a PRF adversary $\mathcal{A}$, we construct a collision-finding adversary $\mathcal{B}(\mathcal{A})$ against $h$ that operates with similar time complexity. $\mathcal{B}$ is constructed as follows. $\mathcal{B}$ samples a random function $f \leftarrow_{\$} \mathsf{Rand}(*, \mathbb{B})$ and runs $\mathcal{A}^f$. While doing so, $\mathcal{B}$ stores all the queries of $\mathcal{A}$ in a set,[5]pads all the queries whose length is not a multiple of $n$, then outputs the resulting set. It can then be shown that the following inequality holds, and it remains to bound the right-hand side, which is an instance of the collision-finding game.

$$\mathsf{Adv}_{\mathrm{EPAD}_H^{\mathsf{f}}}^{\mathsf{prf}}(\mathcal{A}) \leq \Pr[S \leftarrow \mathcal{B}(\mathcal{A})_{\sigma,q}; h \leftarrow_{\$} H : \mathcal{C}oll_h(S)]$$

[5]We note that the choice of data structure for formalizing collisions does not keep track of query order or multiplicity (in particular, when padding maps an unpadded message to a padded message that is separately queried to $h$). This relaxation is done without loss of precision since $\mathrm{CBC}_f$ is deterministic and two consecutive calls to $\mathrm{CBC}_f$ can be swapped without effect on the random function – even if it is sampled lazily.

From here on, our proof for ECBC differs from that by Black and Rogaway [BR05], due to the impossibility of precisely formalizing their arguments in EasyCrypt. Using their notations, in the proof of their Lemma 3, they compute probabilities of events of the form $\Pr[Y_{i-1} \oplus M_i = Y_{j-1} \oplus M_j]$, relying on their ability to compute the probability of sampling a particular value for $Y_{i-1}$. However, this sampling occurs in a previous iteration of the loop, and may in fact be overwritten, losing its "randomness" for the next iteration where the events are tested. One may argue that every value it may be overwritten with in fact follows the same distribution. However, EasyCrypt's logics–and indeed entire proof methodology relies on reasoning about values rather than distributions, and its logics cannot express the fact that some intermediate value follows a particular distribution. On the other hand, a standard way of dealing with similar issues would be to delay the random sampling until the value is used, allowing a precise probability computation. However, in this case, the value could in fact be overwritten between the point where it is initially sampled and the point where it is used. This introduces dependencies between random values and the adversary's view of the system that make it impossible to delay sampling operations as desired. However, if we cannot formalize precisely their argument, we can formalize a simpler–less precise bound that does not discount internal collisions when they are caused by a common prefix.

In particular, instead of computing the collision probability for the full set $S$ of messages, we guess which of the couples of messages may produce a collision and use standard arguments to bound the probability that any of them collide. We sample two of the messages from $S$ and only compute their collision probability. Let $\mathcal{C}(\mathcal{B})$ be the adversary that calls the adversary $\mathcal{B}$ which outputs a set $S \subset \mathbb{B}^+$, then samples two distinct messages $M, M' \leftarrow_{\$} S$ and outputs $(M, M')$. There are at most $\frac{|S|(|S|-1)}{2}$ different pairs.

**Lemma 3.** *For any function family $H \in \{\mathbb{B}^+ \to \mathbb{B}\}$, any adversary $\mathcal{B}$ that outputs a set of size at most c, with $2 \leq c$, has a comparable collision probability to the adversary that tries to guess where the collision may happen and only test this.*

$$\begin{aligned}&\Pr[S \leftarrow \mathcal{B}_c; h \leftarrow_{\$} H : \mathcal{C}oll_h(S)] \leq\\ &\frac{c(c-1)}{2} \cdot \Pr[(M, M') \leftarrow \mathcal{C}(\mathcal{B})_c; h \leftarrow_{\$} H : \mathcal{C}oll_h(m, m')]\end{aligned}$$

Taking into account an application of Lemma 2 that allows us to consider ECBC applied to a random function rather than a random permutation, we have the following bound on the prf distinguishing advantage of any adversary $\mathcal{A}$ against ECBC.

$$\begin{aligned}&\mathsf{Adv}_{\mathrm{ECBC}}^{\mathsf{prf}}(\mathcal{A})\\ &\leq \frac{q(q-1)}{2^n} + \Pr[S \leftarrow \mathcal{B}(\mathcal{A})_{\sigma}; \pi \leftarrow_{\$} \mathsf{Perm}(\mathbb{B}) : \mathcal{C}oll_{\mathrm{CBC}_\pi}(S)]\\ &\leq \frac{q(q-1)}{2^n} + \frac{0.5\sigma(\sigma-1)}{2^n} + \frac{q(q-1)}{2} \cdot\\ &\Pr\big[(M, M') \leftarrow \mathcal{C}(\mathcal{B}(\mathcal{A}))_q; f \leftarrow_{\$} \mathsf{Rand}(\mathbb{B}) : \mathcal{C}oll_{\mathrm{CBC}_f}(M, M')\big]\end{aligned}$$

We observe that the last term includes the probability-finding collision for $\mathrm{CBC}_f$. It therefore remains to bound the probability of finding collisions in CBC-MAC.

**Theorem 1** (CBC collision probability)**.** *For any natural number $c$, an adversary $\mathcal{A}_c$ that outputs two bit-strings $(M, M')$ whose lengths are at most $cn$, with $0 < c$, has a low probability of producing a collision with CBC when parameterized by a random function.*

$$\Pr\left[(M, M') \leftarrow \mathcal{A}_c; f \leftarrow_{\$} \mathsf{Rand}(\mathbb{B}) : \mathcal{C}oll_{\mathit{CBC}_f}(M, M')\right] \leq \frac{2c(2c-1)}{2^n} + \frac{1}{2^n}$$

*Proof.* Let $(M, M')$ be two distinct messages output by the adversary $\mathcal{A}_c$, and $f$ the random function sampled by the experiment. If there is no collision in any of the inputs of $f$ during the CBC chaining, $\mathrm{CBC}_f(M)$ will collide with $\mathrm{CBC}_f(M')$ only if the final call to $f$ yields a collision. This occurs with probability at most $\frac{1}{2^n}$.

It remains to bound the probability that a collision occurs somewhere along the chain of inputs to $f$. The chaining for the longest common prefix of $M$ and $M'$ is computed only once, and collisions are only considered afterwards. We then bound the probability of a collision occurring in one of the inputs to $f$ by the probability of a collision occurring when those $2c$ inputs are sampled. □

Combining all results above allows us to conclude with a security bound for ECBC.

**Theorem 2** (Security of ECBC)**.** *For any natural numbers $q, l, \sigma, n$, an adversary $\mathcal{A}$ making at most $q$ queries, each query of maximum size $ln$ and of total size in the number of blocks at most $\sigma$, has a low probability to distinguish ECBC from a random function, when parameterized by independent random permutations.*

$$\mathsf{Adv}^{\mathsf{prf}}_{\mathit{ECBC}}(\mathcal{A}) \leq \frac{0.5\sigma^2 + 2q^2l^2 + q^2}{2^n} \tag{1}$$

## B. FCBC security

We now relate the security of FCBC with that of ECBC. Indeed, we note that ECBC is an instance of FCBC with non-independent permutations. Given three permutations $\pi_1, \pi_2, \pi_3$, we have

$$\mathrm{ECBC}_{\pi_1,\pi_2,\pi_3} = \mathrm{FCBC}_{\pi_1,\pi_2\circ\pi_1,\pi_3\circ\pi_1}$$

The security proof for FCBC then simply relies on the fact that the composition of two independent random permutations remains independent from the first one. In other words, given two independent random permutations $\pi_1, \pi_2$, the distributions of $(\pi_1, \pi_2 \circ \pi_1)$ and $(\pi_1, \pi_2)$ are the same.

**Lemma 4.** *An adversary $\mathcal{A}$ making an unbounded number of oracle queries cannot distinguish the composition of two independent random permutations.*

$$\Pr\left[\pi_1, \pi_2 \leftarrow_{\$} \mathsf{Perm}(\mathbb{B})^2 : \mathcal{A}^{\pi_1,\pi_2} = 1\right] = \Pr\left[\pi_1, \pi_2 \leftarrow_{\$} \mathsf{Perm}(\mathbb{B})^2 : \mathcal{A}^{\pi_1,\pi_2\circ\pi_1} = 1\right]$$

*Proof of Lemma 4.* It is possible to pre-sample outputs for all inputs of both $\pi_1$ and $\pi_2$. This makes it clear that answers to $\pi_2 \circ \pi_1$ queries are independent from past queries to $\pi_1$ and that, with $\pi_1$ fixed, the distributions of $\pi_2 \circ \pi_1$ and $\pi_2$ are equal. □

**Theorem 3** (Security of FCBC)**.** *An adversary $\mathcal{A}$ cannot distinguish FCBC from ECBC, when parameterized by independent random permutations.*

$$\mathsf{Adv}^{\mathsf{prf}}_{\mathit{FCBC}}(\mathcal{A}) = \mathsf{Adv}^{\mathsf{prf}}_{\mathit{ECBC}}(\mathcal{A}) \tag{2}$$

*Proof.* This is a direct application of Lemma 4 (applied twice). □

## C. Security of XCBC

We view XCBC as a particular instance of FCBC. Indeed, given a permutation $\pi$ and two blocks $k_2, k_3$, we have

$$\mathrm{XCBC}_{\pi,k_2,k_3} = \mathrm{FCBC}_{\pi,\pi(k_2\oplus\cdot),\pi(k_3\oplus\cdot)}$$

We thus prove that the security of XCBC is implied by that of FCBC. The proof crucially relies on the following lemma, which states that one can always (computationally) simulate two independent random permutations using a unique random permutation and a random constant.

**Lemma 5.** *For any natural number $c$, an adversary $\mathcal{A}$ making at most $c$ oracle queries, with $0 < c \leq \frac{2^n}{2}$, has a low probability of distinguishing between two independent permutations $(\pi_1, \pi_2)$ and the pair of permutations $(\pi, \pi(k \oplus \cdot))$.*

$$\Big|\Pr\left[\pi_1, \pi_2 \leftarrow_{\$} \mathsf{Perm}(\mathbb{B})^2; b \leftarrow \mathcal{A}_c^{\pi_1,\pi_2} : b = 1\right] - \Pr\left[\pi, k \leftarrow_{\$} \mathsf{Perm}(\mathbb{B}) \times \mathbb{B}; b \leftarrow \mathcal{A}_c^{\pi,\pi(k\oplus\cdot)} : b = 1\right]\Big| \leq \frac{1.25c^2}{2^n}$$

*Proof of Lemma 5.* The proof closely follows that of Black and Rogaway [BR05]. Let $\mathcal{A}$ be an adversary that expects two permutations (as in the given game). The adversary's goal here is to find which distribution the oracles it was given come from. On one hand, there are two independent random permutations, and on the other hand there is a random permutation that simulates two permutations with a random constant.

We prove that the distributions are equal *unless* the adversary makes two queries $x$ and $y$ to $\mathsf{O}_1$ and $\mathsf{O}_2$ such that $\mathsf{O}_1(x) = \mathsf{O}_2(y)$. When two independent permutations are used, this may happen with low probability for any pair $(x, y)$. On the other hand, when the second permutation is a masked version of the first, this can only occur when $x \oplus y = k$.

Once such a pair of queries has been found, the adversary can decide with very high probability which oracles she is interacting with by simply checking, for some $z \neq x$ whether $\mathsf{O}_1(z) = \mathsf{O}_2(z \oplus x \oplus y)$, which is very unlikely if $\mathsf{O}_1$ and $\mathsf{O}_2$ are independent, but will hold with probability 1 otherwise.

We further prove that an adversary that makes such a pair of queries must have either: i. queried $x$ to $\mathsf{O}_1$ and $x \oplus k$ to $\mathsf{O}_2$ without knowing the value of $k$; or ii. queried $x$ to $\mathsf{O}_1$ and $y \neq x \oplus k$ to $\mathsf{O}_2$ and obtained $\mathsf{O}_1(x) = \mathsf{O}_2(y)$. We can then bound the probability of the first event by $\frac{0.25c^2}{2^n}$, and the probability of the second event by $\frac{c^2}{2^n}$. □

**Lemma 6** (Security of XCBC)**.** *For any natural numbers $q, \sigma, n$, an adversary $\mathcal{A}$ making at most $q$ queries, of total size in the number of blocks at most $\sigma$, has a low probability of distinguishing XCBC from FCBC, when XCBC is parameterized by a random permutation and two independent random constants, and FCBC is parameterized by three independent random permutations.*

$$\mathsf{Adv}^{\mathsf{prf}}_{XCBC}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{prf}}_{FCBC}(\mathcal{A}) + \frac{2.5\sigma^2}{2^n} \tag{3}$$

*Proof.* When an adversary succeeds in distinguishing XCBC from a random function, either it has distinguished XCBC from FCBC, or FCBC from a random function. To bound the distinguishing advantage of any adversary between FCBC and XCBC, let $\pi_1, \pi_2, \pi_3$ be three independent random permutations and $k_1, k_2$ two independent random constants. Lemma 5 is instantiated twice with the bound $\sigma$, making $\mathrm{FCBC}_{\pi_1,\pi_2,\pi_3}$ indistinguishable from $\mathrm{FCBC}_{\pi_1,\pi_1(k_1\oplus\cdot),\pi_3}$, and indistinguishable from $\mathrm{FCBC}_{\pi_1,\pi_1(k_1\oplus\cdot),\pi_1(k_2\oplus\cdot)}$, i.e. $\mathrm{XCBC}_{\pi_1,k_1,k_2}$. □

*a) A small flaw:* Note that our bound on the security of XCBC is different from Black and Rogaway's [BR05]. They instantiate their lemma called *Two Permutations From One* (Lemma 5 in the present paper) with the wrong bound on the total number of oracle queries. Indeed, the bound $c$ on the number of queries in Lemma 5 is a bound on the *total* number of oracle queries. On the other hand, in both instantiations of Lemma 5, Black and Rogaway only count the number of MAC queries that do not need padding (accounting only for queries to the first oracle in Lemma 5). The flaw is subtle, and has no effect on the security bound (which we improve below), but is present nonetheless, adding to the body of evidence that cryptographic proofs are difficult both to write and to evaluate.

*b) An improvement:* Furthermore, to tighten the bound back, we extend Lemma 5 into the following Lemma 7, which states that a random permutation can simulate three independent random permutations using two independent random constants. Proving this lemma directly, rather than relying on Lemma 5 twice allows us to improve the bound slightly, by allowing us to count queries to the first permutation once only.

**Lemma 7.** *For any natural number $c$, an adversary $\mathcal{A}$ making at most $c$ total oracle queries, with $0 < c \leq 2^{n-1}$, has a low probability to distinguish between three independent permutations $(\pi_1, \pi_2, \pi_3)$ and the tuple of permutations $(\pi, \pi(k_1 \oplus \cdot), \pi(k_2 \oplus \cdot))$.*

$$\begin{aligned}&\left|\Pr\left[\pi_1,\pi_2,\pi_3 \leftarrow_\$ \mathsf{Perm}(\mathbb{B})^3; b \leftarrow \mathcal{A}_c^{\pi_1,\pi_2,\pi_3} : b = 1\right] -\right.\\ &\Pr\left[\pi \leftarrow_\$ \mathsf{Perm}(\mathbb{B}); (k_1,k_2) \leftarrow_\$ \mathbb{B}^2; b \leftarrow \mathcal{A}_c^{\pi,\pi(k_1\oplus\cdot),\pi(k_2\oplus\cdot)} :\right.\\ &\left.\left. b = 1\right]\right| \leq \frac{1.75c^2}{2^n}\end{aligned}$$

*Proof.* This is very similar to Lemma 5 and Lemma 8. The proof needs to bound the events $\mathsf{coll}_{\mathsf{rng}}$ (defined for Lemma 8) and another one. The bound of $\mathsf{coll}_{\mathsf{rng}}$ has been proved as a generic result and then instanciated in Lemmas 7 and 8. The probability of the other one is bounded by $\frac{0.75\sigma^2}{2^n}$. □

**Theorem 4** (Security of XCBC)**.** *For any natural numbers $q, \sigma, n$, an adversary $\mathcal{A}$ making at most $q$ queries, of total size in the number of blocks at most $\sigma$, has a low probability of distinguishing XCBC from FCBC, when XCBC is parameterized by a random permutation and two independent random constants, and FCBC is parameterized by three independent random permutations.*

$$\mathsf{Adv}^{\mathsf{prf}}_{XCBC}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{prf}}_{FCBC}(\mathcal{A}) + \frac{1.75\sigma^2}{2^n} \tag{4}$$

*Proof.* As for Lemma 6, this is a direct application of Lemma 7. □

## D. *Security of CMAC*

Our security proof generalises CMAC slightly in that we use abstract public functions $f_1, f_2 : \mathbb{B} \to \mathbb{B}$ to capture the derivation of $k_2$ and $k_3$ from $\pi(0^n)$. The security of CMAC follows by instantiating $f_1 = x \mapsto 2 \times x$ and $f_2 = x \mapsto 4 \times x$.

The security of this generalised CMAC (or indeed of concrete CMAC) does not immediately follow from that of XCBC, since its constants $k_2 = f_1(\pi(0^n))$ and $k_3 = f_2(\pi(0^n))$ are not independent between themselves, and furthermore not independent from $\pi$. However, the security of CMAC is easily implied by that of FCBC, of which it is an instance. Indeed, given $\pi$,

$$\mathrm{CMAC}_\pi = \mathrm{FCBC}_{\pi,\pi(f_1(\pi(0^n))\oplus\cdot),\pi(f_2(\pi(0^n))\oplus\cdot)}$$

Here again, our security proof is close to that of Iwata and Kurosawa [IK03a], which relies on a more general construction, MOMAC, that generalises both CMAC and FCBC. We provide a game-based proof and go into more details.

*1) MOMAC:* The MOMAC construction [IK03a] is parameterised by 6 oracles $\mathsf{O}_{1\leq i\leq 6}$, which are used as follows.

$$\mathrm{MOMAC}_{\mathsf{O}_1,\mathsf{O}_2,\mathsf{O}_3,\mathsf{O}_4,\mathsf{O}_5,\mathsf{O}_6}(\mathsf{m}) :=$$
$$\begin{cases}\mathsf{O}_5(\mathsf{pad}(\mathsf{m})) & \text{if } 0 \leq |\mathsf{m}| < n\\ \mathsf{O}_3(\mathsf{m}) & \text{if } |\mathsf{m}| = n\\ \mathrm{FCBC}_{\mathsf{O}_1,\mathsf{O}_4,\mathsf{O}_6}(\mathsf{m}) & \text{if } n < |\mathsf{m}| \leq 2n\\ \mathrm{FCBC}_{\mathsf{O}_2,\mathsf{O}_4,\mathsf{O}_6}(\mathsf{m}_{\mathsf{O}_1}) & \text{if } 2n < |\mathsf{m}| \wedge \lceil |\mathsf{m}|/n \rceil = m\end{cases}$$

when $\mathsf{m}_{\mathsf{O}_1} = (\mathsf{O}_1(\mathsf{m}_1) \oplus \mathsf{m}_2) \mid\mid \ldots \mid\mid \mathsf{m}_m$.

The resulting construction is also illustrated in Figure 7.

*2) CMAC and FCBC as instances of MOMAC:* Given a random permutations $\pi$ and a random $n$-bit string $\mathsf{r}$, the following six oracles $Q_{1\le i\le 6}$, where $L^\pi = \pi(0^n)$, can be used with MOMAC to build CMAC.

$$\begin{array}{ll} Q_1(x) := \pi(x) \oplus \mathsf{r} & Q_2(x) := \pi(x \oplus \mathsf{r}) \oplus \mathsf{r} \\ Q_3(x) := \pi(x \oplus f_1(L^\pi)) & Q_4(x) := \pi(x \oplus \mathsf{r} \oplus f_1(L^\pi)) \\ Q_5(x) := \pi(x \oplus f_2(L^\pi)) & Q_6(x) := \pi(x \oplus \mathsf{r} \oplus f_2(L^\pi)) \end{array}$$

Given three independent random permutations $\pi_1, \pi_2, \pi_3$ and a random $n$-bit string $\mathsf{r}$, the following six oracles $R_{1\le i\le 6}$ can be used with MOMAC to build FCBC.

$$\begin{array}{ll} R_1(x) := \pi_1(x) \oplus \mathsf{r} & R_2(x) := \pi_1(x \oplus \mathsf{r}) \oplus \mathsf{r} \\ R_3(x) := \pi_2(x) & R_4(x) := \pi_2(x \oplus \mathsf{r}) \\ R_5(x) := \pi_3(x) & R_6(x) := \pi_3(x \oplus \mathsf{r}) \end{array}$$

*3) Security proof:* For any value of $\mathsf{r}$, the following functional equivalences follow from these definitions.

$$\begin{array}{rcl} \mathrm{CMAC}_\pi & \sim & \mathrm{MOMAC}_{Q_1,Q_2,Q_3,Q_4,Q_5,Q_6} \\ \mathrm{FCBC}_{\pi_1,\pi_2,\pi_3} & \sim & \mathrm{MOMAC}_{R_1,R_2,R_3,R_4,R_5,R_6} \end{array}$$

Therefore, distinguishing CMAC from FCBC can be reduced to distinguishing the $Q_i$ from the $R_i$. In the following, for any adversary $\mathcal{A}$, expecting six oracles and making at most $q$ oracle queries to $\mathsf{O}_3, \mathsf{O}_4, \mathsf{O}_5, \mathsf{O}_6$ and at most $\sigma$ total oracle queries before returning a boolean, we name $Q(\mathcal{A})$ (resp. $R(\mathcal{A})$) the game that initializes the oracles $Q_{1\le i\le 6}$ (resp. $R_{1\le i\le 6}$), then calls the adversary $\mathcal{A}^{Q_i}$ (resp. $\mathcal{A}^{R_i}$).

$$\begin{array}{rcl} Q(\mathcal{A}) & := & \left[\pi \leftarrow_{\$} \mathsf{Perm}(\mathbb{B}), \mathsf{r} \leftarrow_{\$} \mathbb{B}; b \leftarrow \mathcal{A}_{\sigma,q}^{Q_{1\le i\le 6}}\right] \\ R(\mathcal{A}) & := & \left[\pi_1,\pi_2,\pi_3 \leftarrow_{\$} \mathsf{Perm}(\mathbb{B}), \mathsf{r} \leftarrow_{\$} \mathbb{B}; b \leftarrow \mathcal{A}_{\sigma,q}^{R_{1\le i\le 6}}\right] \end{array}$$

**Lemma 8.** *For any bijective $f_1$ and $f_2$ such that $x \mapsto x \oplus f_1(x)$, $x \mapsto x \oplus f_2(x)$, $x \mapsto f_1(x) \oplus f_2(x)$, $x \mapsto x \oplus f_1(x) \oplus f_2(x)$ are also bijective, an adversary $\mathcal{A}$, making at most $q$ oracle queries to $\mathsf{O}_3, \mathsf{O}_4, \mathsf{O}_5, \mathsf{O}_6$ and at most $\sigma$ total oracles queries, has a low probability of distinguishing the game with the $R_i$ from the game with the $Q_i$.*

$$|\Pr[Q(\mathcal{A}) : b = 1] - \Pr[R(\mathcal{A}) : b = 1]| \le \frac{\sigma(\sigma+1)}{2^n} + \frac{\frac{1}{4}(\sigma+1)^2 + \frac{1}{2}q^2}{2^n} + \frac{\frac{3}{4}(\sigma+1)^2}{2^n}$$

*Proof sketch.* We first give an overview of the proof, using names that are defined later. The rest of this section details individual steps.

We formally prove that, for any adversary $\mathcal{A}$, the game $Q(\mathcal{A})$ is equivalent to $R(\mathcal{A})$ upto the event $\mathsf{find}_L \vee \mathsf{coll}_{\mathsf{rng}}$. It is easy to bound the probability of $\mathsf{coll}_{\mathsf{rng}}$ occurring, but the probability of $\mathsf{find}_L$ occurring in $R(\mathcal{A})$ requires additional work. We introduce two new games $S(\mathcal{A})$ and $T(\mathcal{A})$, represented in Figure 8. Game $S(\mathcal{A})$ is perfectly equivalent to $R(\mathcal{A})$, thus the probability of $\mathsf{find}_L$ occurring is equal in both games. We then prove that $T(\mathcal{A})$ is equivalent to $S(\mathcal{A})$ upto a third event $\mathsf{find}_{\mathsf{r}}$.

The following sequence of inequalities shows an outline of our detailed proof, with the justification for each non-trivial step given in the paragraph whose heading is listed beside the step.

$$\begin{array}{l} |\Pr[Q(\mathcal{A}) : b = 1] - \Pr[R(\mathcal{A}) : b = 1]| \\ \text{(using } Q(\mathcal{A}) \sim R(\mathcal{A}) \text{ upto } \mathsf{coll}_{\mathsf{rng}} \vee \mathsf{find}_L\text{)} \\ \quad \le \Pr[R(\mathcal{A}) : \mathsf{coll}_{\mathsf{rng}}] + \Pr[R(\mathcal{A}) : \mathsf{find}_L] \\ \text{(using } R(\mathcal{A}) \sim S(\mathcal{A})\text{)} \\ \quad = \Pr[R(\mathcal{A}) : \mathsf{coll}_{\mathsf{rng}}] + \Pr[S(\mathcal{A}) : \mathsf{find}_L] \\ \quad \le \Pr[R(\mathcal{A}) : \mathsf{coll}_{\mathsf{rng}}] + \Pr[T(\mathcal{A}) : \mathsf{find}_L] + \\ \qquad |\Pr[S(\mathcal{A}) : \mathsf{find}_L] - \Pr[T(\mathcal{A}) : \mathsf{find}_L]| \\ \text{(using } S(\mathcal{A}) \sim T(\mathcal{A}) \text{ upto } \mathsf{find}_{\mathsf{r}}\text{)} \\ \quad \le \Pr[R(\mathcal{A}) : \mathsf{coll}_{\mathsf{rng}}] + \Pr[T(\mathcal{A}) : \mathsf{find}_L] + \\ \qquad \Pr[T(\mathcal{A}) : \mathsf{find}_{\mathsf{r}}] \end{array}$$

We finally bound the probability of $\mathsf{coll}_{\mathsf{rng}}$ and $\mathsf{find}_{\mathsf{r}}$ in $T(\mathcal{A})$ and conclude. □

We now discuss individual steps in the proof, including definitions for the events. In Figure 8 and below, $L^\pi$ denotes the value $\pi(0^n)$ and $L_i^\pi$ denotes the value $f_i(L^\pi)$ for $i \in \{1,2\}$.

*a) $Q(\mathcal{A}) \sim R(\mathcal{A})$ upto $\mathsf{coll}_{\mathsf{rng}} \vee \mathsf{find}_L$:* In a proof reminiscent of that of Lemma 5, we note that games $Q(\mathcal{A})$ and $R(\mathcal{A})$ are equivalent unless $\mathcal{A}$ queries some $x$ to $\mathsf{O}_i$ and some $y$ to $\mathsf{O}_j$, with $(i,j) \in \{(1,3),(1,5),(3,5)\}$ and such that $\mathsf{O}_i(x) = \mathsf{O}_j(y)$. If this event (which we simply call $\mathsf{bad}$ below) occurs, then $\mathcal{A}$ can distinguish the two sets of oracles with high probability by testing, for some $z \neq x$ whether $\mathsf{O}_i(z) = \mathsf{O}_j(z \oplus x \oplus y)$. We now prove that event $\mathsf{bad}$ as defined above completely captures all cases in which the two games can be distinguished. To simplify bounding the probability of $\mathsf{bad}$ occurring, we consider instead a disjunction of two disjoint events $\mathsf{find}_L$ and $\mathsf{coll}_{\mathsf{rng}}$, depending on the value of $x \oplus y$. Event $\mathsf{find}_L$ occurs if $\mathcal{A}$ queries, for some $x$:

- $\mathsf{O}_1(x)$ and $\mathsf{O}_3(x \oplus f_1(L^\pi))$; or
- $\mathsf{O}_1(x)$ and $\mathsf{O}_5(x \oplus f_2(L^\pi))$; or
- $\mathsf{O}_3(x)$ and $\mathsf{O}_5(x \oplus f_1(L^\pi) \oplus f_2(L^\pi))$.

Event $\mathsf{coll}_{\mathsf{rng}}$ captures the remainder of the cases in $\mathsf{bad}$, and thus occurs when $\mathcal{A}$ queries, for some $x$ and $y$:

- $\mathsf{O}_1(x)$ and $\mathsf{O}_3(y)$ such that $\mathsf{O}_1(x) = \mathsf{O}_3(y)$ and $y \neq x \oplus f_1(L^\pi)$; or
- $\mathsf{O}_1(x)$ and $\mathsf{O}_5(y)$ such that $\mathsf{O}_1(x) = \mathsf{O}_5(y)$ and $y \neq x \oplus f_2(L^\pi)$; or
- $\mathsf{O}_3(x)$ and $\mathsf{O}_5(y)$ such that $\mathsf{O}_3(x) = \mathsf{O}_5(y)$ and $y \neq x \oplus f_1(L^\pi) \oplus f_2(L^\pi)$.

We consider two games $X_1(\mathcal{A})$ and $X_2(\mathcal{A})$, which are respectively equivalent to $Q(\mathcal{A})$ and $R(\mathcal{A})$. $X_1(\mathcal{A})$ and $X_2(\mathcal{A})$ are shown in Figure 9, where code inside the dotted boxes appears only in $X_1(\mathcal{A})$. Figure 9 only shows definitions for $\mathsf{O}_1$, $\mathsf{O}_3$ and $\mathsf{O}_5$, which are sufficient to define the entire games. Indeed, we note that, in both $Q(\mathcal{A})$ and $R(\mathcal{A})$, we

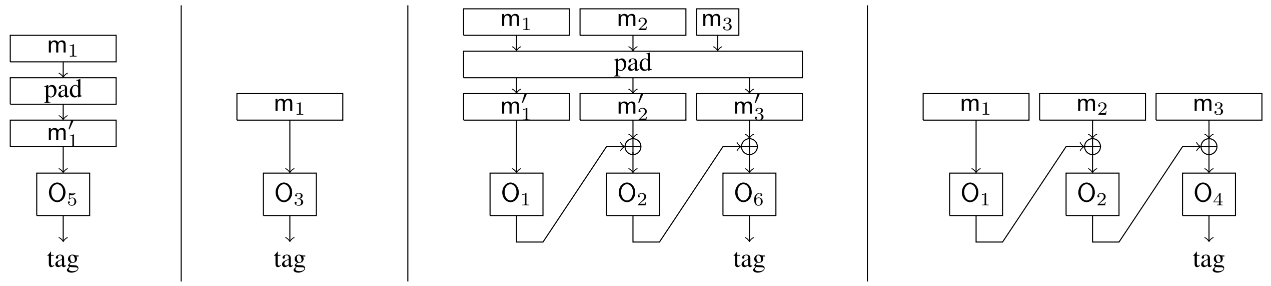


Fig. 7. Illustration of MOMAC, by increasing size order: $|\mathsf{m}| < n$, $|\mathsf{m}| = n$, $2n < |\mathsf{m}| < 3n$, $|\mathsf{m}| = 3n$,

| Game | $\mathsf{O}_1$ | $\mathsf{O}_2$ | $\mathsf{O}_3$ | $\mathsf{O}_4$ | $\mathsf{O}_5$ | $\mathsf{O}_6$ |
|---|---|---|---|---|---|---|
| $Q(\mathcal{A})$ | $\pi(\cdot)\oplus\mathsf{r}$ | $\pi(\cdot\oplus\mathsf{r})\oplus\mathsf{r}$ | $\pi(\cdot\oplus f_1(\pi(0^n)))$ | $\pi(\cdot\oplus\mathsf{r}\oplus f_1(\pi(0^n)))$ | $\pi(\cdot\oplus f_2(\pi(0^n)))$ | $\pi(\cdot\oplus\mathsf{r}\oplus f_2(\pi(0^n)))$ |
| $R(\mathcal{A})$ | $\pi_1(\cdot)\oplus\mathsf{r}$ | $\pi_1(\cdot\oplus\mathsf{r})\oplus\mathsf{r}$ | $\pi_2(\cdot)$ | $\pi_2(\cdot\oplus\mathsf{r})$ | $\pi_3(\cdot)$ | $\pi_3(\cdot\oplus\mathsf{r})$ |
| $S(\mathcal{A})$ | $\pi_1(\cdot)$ | $\pi_1(\cdot\oplus\mathsf{r})$ | $\pi_2(\cdot)$ | $\pi_2(\cdot\oplus\mathsf{r})$ | $\pi_3(\cdot)$ | $\pi_3(\cdot\oplus\mathsf{r})$ |
| $T(\mathcal{A})$ | $\pi_1(\cdot)$ | $\pi_1'(\cdot)$ | $\pi_2(\cdot)$ | $\pi_2'(\cdot)$ | $\pi_3(\cdot)$ | $\pi_3'(\cdot)$ |

Fig. 8. Sequence of games.

have $\mathsf{O}_2(x) = \mathsf{O}_1(x \oplus L_1^\pi)$ and that $\mathsf{O}_4$ and $\mathsf{O}_3$, and $\mathsf{O}_6$ and $\mathsf{O}_5$ obey the same relation.

Figure 9 shows very clearly that $Q(\mathcal{A})$ (or $X_1$) and $R(\mathcal{A})$ (or $X_2$) cease to be equivalent only when one of $\mathsf{find}_L$ or $\mathsf{coll}_{\mathsf{rng}}$ becomes true. Thus, we have:

$$\begin{aligned}&|\Pr[Q(\mathcal{A}) : b = 1] - \Pr[R(\mathcal{A}) : b = 1]|\\ =&|\Pr[X_1(\mathcal{A}) : b = 1] - \Pr[X_2(\mathcal{A}) : b = 1]|\\ \leq&\Pr[X_2(\mathcal{A}) : \mathsf{coll}_{\mathsf{rng}} \vee \mathsf{find}_L]\\ \leq&\Pr[X_2(\mathcal{A}) : \mathsf{coll}_{\mathsf{rng}}] + \Pr[X_2(\mathcal{A}) : \mathsf{find}_L]\end{aligned}$$

*b) Bounding* $\Pr[X_2(\mathcal{A}) : \mathsf{coll}_{\mathsf{rng}}]$*:* We note that $\mathsf{coll}_{\mathsf{rng}}$ is essentially the probability of a freshly sampled variable already appearing in some set. This can easily be bound, knowing that the total number of oracle queries is at most $\sigma \leq 2^{n-1}$.

For any adversary $\mathcal{A}$ that makes at most $\sigma$ oracle queries, we have,

$$\Pr[X_2(\mathcal{A}) : \mathsf{coll}_{\mathsf{rng}}] \leq \frac{\sigma(\sigma+1)}{2} \cdot \frac{1}{2^n - (\sigma+1)} \leq \frac{\sigma(\sigma+1)}{2^n}$$

*c)* $R(\mathcal{A}) \sim S(\mathcal{A})$*:* Event $\mathsf{find}_L$ in game $X_2(\mathcal{A})$ can also be expressed as follows.

$$\mathsf{find}_L \Leftrightarrow \bigvee \left\{\begin{array}{ll} L_1^{\pi_1} & \in \mathsf{dom}(\pi_1) \oplus \mathsf{dom}(\pi_2),\\ L_2^{\pi_1} & \in \mathsf{dom}(\pi_1) \oplus \mathsf{dom}(\pi_3),\\ L_1^{\pi_1} \oplus L_2^{\pi_1} & \in \mathsf{dom}(\pi_2) \oplus \mathsf{dom}(\pi_3)\end{array}\right\}$$

Its probability cannot be computed directly in $X_2(\mathcal{A})$, since the collision involves $L^\pi$, which is not independent from the adversary's view. We therefore need to modify $X_2(\mathcal{A})$ somewhat into a game $S(\mathcal{A})$ shown in Figure 10 to bound this probability. The objective of game $S(\mathcal{A})$ is to extend the $\mathsf{find}_L$ event to include fresh and independent randomness $\mathsf{r}$ in its definition. This will then enable us to make use of it to bound the probability of $\mathsf{find}_L$, since it is independent from the oracles' outputs. It is easy to see that for any adversary $\mathcal{A}$, $S(\mathcal{A})$ is perfectly equivalent to $X_2(\mathcal{A})$. We formally prove

$$\Pr[X_2(\mathcal{A}) : \mathsf{find}_L] = \Pr[S(\mathcal{A}) : \mathsf{find}_L]$$

*d) Event* $\mathsf{find}_L$ *in game* $S(\mathcal{A})$*:* In game $X_2(\mathcal{A})$, the value of $L^{\pi_1}$ as it appears in event $\mathsf{find}_L$ is equal to $\pi_1(0^n)$. In game $S(\mathcal{A})$, its value is related to $\mathsf{r}$ is equal to $\mathsf{r} \oplus \pi_1(0^n)$. Event $\mathsf{find}_L$ in game $S(\mathcal{A})$ can now be expressed as follows.

$$\mathsf{find}_L \Leftrightarrow$$
$$\bigvee \left\{\begin{array}{ll} f_1(L^{\pi_1} \oplus \mathsf{r}) & \in \mathsf{dom}(\pi_1) \oplus \mathsf{dom}(\pi_2),\\ f_2(L^{\pi_1} \oplus \mathsf{r}) & \in \mathsf{dom}(\pi_1) \oplus \mathsf{dom}(\pi_3),\\ f_1(L^{\pi_1} \oplus \mathsf{r}) \oplus f_2(L^{\pi_1} \oplus \mathsf{r}) & \in \mathsf{dom}(\pi_2) \oplus \mathsf{dom}(\pi_3)\end{array}\right\}$$

*e)* $S(\mathcal{A}) \sim T(\mathcal{A})$ *upto* $\mathsf{find}_{\mathsf{r}}$*:* As it stands in game $S(\mathcal{A})$, the use of $\mathsf{r}$ in $\mathsf{find}_L$ is not sufficient to allow us to bound its probability of occurring. To do so, we wish to show that it is possible to delay sampling $\mathsf{r}$ until the end of the game. However, $\mathsf{r}$ is correlated to the output of oracles $S_2$, $S_4$ and $S_6$. Again, we observe that $\mathsf{r}$ is a random value, unknown to the adversary, and independent from the permutations. Rather than using this fact as in Lemma 5 to replace the three permutations and their re-randomised version with six truly independent permutations, we simply use it to replace the three permutations with six permutations that are simply independent from the value of $\mathsf{r}$.

In our formal proof, we use three pairs of incomplete functions $(\pi_i, \pi_i')$ that are guaranteed to never output the same result for any fresh input. We implement this by sampling each new fresh output from the uniform distribution over $\mathbb{B} \setminus (\mathsf{rng}(\pi_i) \cup \mathsf{rng}(\pi_i'))$. The games we use to formally prove that $(\pi_1, \pi_1')$ are equivalent to $(\pi_1, \pi_1(\cdot \oplus \mathsf{r}))$ upto the event $\mathsf{find}_{\mathsf{r}}$ are illustrated in Figure 10. We extend this definition to $(\pi_2, \pi_2')$ and $(\pi_3, \pi_3')$ and name $T(\mathcal{A})$ the game that uses the corresponding definitions for $\mathsf{O}_{1\leq i\leq 6}$ (which can be seen in Figure 8).

```
$X_1(\mathcal{A})$ or $X_2(\mathcal{A})$
$\pi, \pi_1, \pi_2, \pi_3 \leftarrow$ Undefined
$L \leftarrow\$ \mathbb{B}$
$\pi[0^n] \leftarrow L$
$\pi_1[0^n] \leftarrow L$
$\mathsf{r} \leftarrow\$ \mathbb{B}$
$\mathsf{coll_{rng}} \leftarrow$ false
$\mathsf{find}_L \leftarrow$ false
$b \leftarrow \mathcal{A}_{\sigma,q}^{\mathsf{O}_1,\mathsf{O}_3,\mathsf{O}_5}$
```

```
$\mathsf{O}_1(\mathsf{m})$
if $(\mathsf{m} \in \mathsf{dom}(\pi_1))$
    return $\pi_1[\mathsf{m}]$
else if $(\mathsf{m} \in \mathsf{dom}(\pi))$
    $\mathsf{find}_L \leftarrow$ true
    $\pi_1[\mathsf{m}] \leftarrow \pi[\mathsf{m}]$
    return $\pi_1[\mathsf{m}]$
$y \leftarrow\$ \mathbb{B} \setminus \mathsf{rng}(\pi_1)$
if $(y \in \mathsf{rng}(\pi))$
    $\mathsf{coll_{rng}} \leftarrow$ true
    $y \leftarrow\$ \mathbb{B} \setminus \mathsf{rng}(\pi)$
$\pi_1[\mathsf{m}] \leftarrow y$
$\pi[\mathsf{m}] \leftarrow y$
return $y$
```

```
$\mathsf{O}_3(\mathsf{m})$
if $(\mathsf{m} \in \mathsf{dom}(\pi_2))$
    return $\pi_2[\mathsf{m}]$
else if $(\mathsf{m} \oplus f_1(L) \in \mathsf{dom}(\pi))$
    $\mathsf{find}_L \leftarrow$ true
    $\pi_2[\mathsf{m}] \leftarrow \pi[\mathsf{m} \oplus f_1(L)]$
    return $\pi_2[\mathsf{m}]$
$y \leftarrow\$ \mathbb{B} \setminus \mathsf{rng}(\pi_2)$
if $(y \in \mathsf{rng}(\pi))$
    $\mathsf{coll_{rng}} \leftarrow$ true
    $y \leftarrow\$ \mathbb{B} \setminus \mathsf{rng}(\pi)$
$\pi_2[\mathsf{m}] \leftarrow y$
$\pi[\mathsf{m} \oplus f_1(L)] \leftarrow y$
return $y$
```

```
$\mathsf{O}_5(\mathsf{m})$
if $(\mathsf{m} \in \mathsf{dom}(\pi_3))$
    return $\pi_3[\mathsf{m}]$
else if $(\mathsf{m} \oplus f_2(L) \in \mathsf{dom}(\pi))$
    $\mathsf{find}_L \leftarrow$ true
    $\pi_3[\mathsf{m}] \leftarrow \pi[\mathsf{m} \oplus f_2(L)]$
    return $\pi_3[\mathsf{m}]$
$y \leftarrow\$ \mathbb{B} \setminus \mathsf{rng}(\pi_3)$
if $(y \in \mathsf{rng}(\pi))$
    $\mathsf{coll_{rng}} \leftarrow$ true
    $y \leftarrow\$ \mathbb{B} \setminus \mathsf{rng}(\pi)$
$\pi_3[\mathsf{m}] \leftarrow y$
$\pi[\mathsf{m} \oplus f_2(L)] \leftarrow y$
return $y$
```

Fig. 9. Games $X_1(\mathcal{A})$ (including dotted boxes) and $X_2(\mathcal{A})$ (excluding dotted boxes).

```
$\mathsf{O}_1(\mathsf{m})$
if $(\mathsf{m} \notin \mathsf{dom}(\pi_1))$
    $\mathsf{y} \leftarrow\$ \mathbb{B} \setminus$
            $(\mathsf{rng}(\pi_1) \cup \mathsf{rng}(\pi_1'))$
    if $(\mathsf{m} \oplus \mathsf{r} \in \mathsf{dom}(\pi_1'))$
        $\mathsf{find_r} \leftarrow$ true
        $\mathsf{y} \leftarrow \pi_1'[\mathsf{m} \oplus \mathsf{r}]$
    $\pi_1[\mathsf{m}] \leftarrow \mathsf{y}$
$\mathsf{y} \leftarrow \pi_1[\mathsf{m}]$
return $\mathsf{y}$
```

```
$\mathsf{O}_2(\mathsf{m})$
if $(\mathsf{m} \notin \mathsf{dom}(\pi_1'))$
    $\mathsf{y} \leftarrow\$ \mathbb{B} \setminus$
            $(\mathsf{rng}(\pi_1) \cup \mathsf{rng}(\pi_1'))$
    if $(\mathsf{m} \oplus \mathsf{r} \in \mathsf{dom}(\pi_1))$
        $\mathsf{find_r} \leftarrow$ true
        $\mathsf{y} \leftarrow \pi_1[\mathsf{m} \oplus \mathsf{r}]$
    $\pi_1'[\mathsf{m}] \leftarrow \mathsf{y}$
$\mathsf{y} \leftarrow \pi_1'[\mathsf{m}]$
return $\mathsf{y}$
```

Fig. 10. Games $S(\mathcal{A})$ (including dotted boxes) and $T(\mathcal{A})$ (excluding dotted boxes).

Thanks to this more flexible goal we need only consider a smaller event, $\mathsf{find_r}$, whose occurrence allows the adversary to distinguish between the $S_i$ and the $T_i$. In essence, $\mathsf{find_r}$ corresponds in this setting to the $\mathsf{coll_{rng}}$ event from the first step. With this in mind, it is easy to prove that the behaviour of $S(\mathcal{A})$ and $T(\mathcal{A})$ can only diverge if $\mathsf{find_r}$ occurs. This allows us to prove the following inequality.

$$|\Pr[S(\mathcal{A}) : \mathsf{find}_L] - \Pr[T(\mathcal{A}) : \mathsf{find}_L]| \leq \Pr[T(\mathcal{A}) : \mathsf{find_r}]$$

*f) Bounding* $\Pr[T(\mathcal{A}) : \mathsf{find_r}]$*:* As when bounding $\mathsf{find}_L$ earlier, we note that $\mathsf{find_r}$ is the probability of a freshly sampled value already appearing in a set. For all adversary $\mathcal{A}$ that makes at most $q$ to $\mathsf{O}_3$, $\mathsf{O}_4$, $\mathsf{O}_5$ and $\mathsf{O}_6$, and at most $\sigma$ total oracle queries, we have

$$\Pr[T(\mathcal{A}) : \mathsf{find_r}] \leq \frac{1}{4}\frac{(\sigma+1)^2}{2^n} + \frac{1}{2}\frac{q^2}{2^n}$$

The proof relies on the fact that, $\forall x, y, z$, if $0 \leq x + y \leq z$ then $xy \leq \frac{z^2}{4}$, which allows us to clean complex bounds.

*g) Bounding* $\Pr[T(\mathcal{A}) : \mathsf{find}_L]$*:* It now remains to bound the probability that $\mathsf{find}_L$ occurs in $T(\mathcal{A})$. As shown in Figure 8, game $T(\mathcal{A})$ no longer uses the value of $\mathsf{r}$, and that variable can therefore be leveraged to bound the probability of $\mathsf{find}_L$.

Therefore, we can use the randomness of $\mathsf{r}$ to bound the probability of $\mathsf{find}_L$ in $T(\mathcal{A})$. The transformation from $S(\mathcal{A})$ to $T(\mathcal{A})$ have modified $\mathsf{find}_L$, in the sense that every occurence of $\mathsf{dom}(\pi_i)$ is now replaced by $\mathsf{dom}(\pi_i) \cup \mathsf{dom}(\pi_i')$, for $i \in \{1, 2, 3\}$. To simplify all this, $\mathsf{find}_L$ is equivalent to a disjunction of six sub-events. Denoting $D_i := \mathsf{dom}(\pi_i)$ and $D_i' := \mathsf{dom}(\pi_i')$, $\mathsf{find}_L$ can be expressed, as it appears in $T(\mathcal{A})$ as follows.

$$\mathsf{find}_L \Leftrightarrow \bigvee \left\{ \begin{array}{ll} f_1(L^{\pi_1} \oplus \mathsf{r}) & \in (D_1 \oplus D_2) \cup (D_1' \oplus D_2'), \\ \mathsf{r} \oplus f_1(L^{\pi_1} \oplus \mathsf{r}) & \in (D_1 \oplus D_2') \cup (D_1' \oplus D_2), \\ f_2(L^{\pi_1} \oplus \mathsf{r}) & \in (D_1 \oplus D_3) \cup (D_1' \oplus D_3'), \\ \mathsf{r} \oplus f_2(L^{\pi_1} \oplus \mathsf{r}) & \in (D_1 \oplus D_3') \cup (D_1' \oplus D_3), \\ f_1(L^{\pi_1} \oplus \mathsf{r}) \oplus f_2(L^{\pi_1} \oplus \mathsf{r}) & \in (D_2 \oplus D_3) \cup (D_2' \oplus D_3'), \\ \mathsf{r} \oplus f_1(L^{\pi_1} \oplus \mathsf{r}) \oplus f_2(L^{\pi_1} \oplus \mathsf{r}) & \in (D_2 \oplus D_3') \cup (D_2' \oplus D_3) \end{array} \right\}$$

Recall that $f_1$, $f_2$, $x \mapsto x \oplus f_1(x)$, $x \mapsto x \oplus f_2(x)$, $x \mapsto f_1(x) \oplus f_2(x)$ and $x \mapsto x \oplus f_1(x) \oplus f_2(x)$ are bijective. Thus, for any adversary $\mathcal{A}$ that makes at most $\sigma$ total queries to its oracles, we can prove that

$$\Pr[T(\mathcal{A}) : \mathsf{find}_L] \leq \frac{3}{4}\frac{(\sigma+1)^2}{2^n}$$

This concludes the proof of Lemma 8, and we can now seek to apply it to the security of CMAC.

**Lemma 9** (Indistinguishability of CMAC and FCBC)**.** *For any natural numbers $q, \sigma, n$, an adversary $\mathcal{A}$ making at most $q$ queries, of total size in the number of blocks of at most $\sigma$, has a low probability of distinguishing CMAC from FCBC, when*

*CMAC is parameterized by a random permutation and FCBC is parameterized by three independent random permutations.*

$$\mathsf{Adv}^{\mathsf{prf}}_{\mathit{CMAC}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathsf{prf}}_{\mathit{FCBC}}(\mathcal{A}) + \frac{2(\sigma+1)^2}{2^n} + \frac{0.5q^2}{2^n} \quad (5)$$

*Proof.* This is a direct application of the definition of MOMAC and Lemma 8. In CMAC, the assumptions involving bijections are reduced in GF($2^n$) to the fact that, for every $p \in \{\mathsf{x}, \mathsf{x} \oplus 1, \mathsf{x}^2, \mathsf{x}^2 \oplus 1, \mathsf{x}^2 \oplus \mathsf{x}, \mathsf{x}^2 \oplus \mathsf{x} \oplus 1\}$, the function $q \mapsto p \times q$ is bijective. □

**Theorem 5** (Security of CMAC)**.** *For any natural numbers $q, l, \sigma, n$, an adversary $\mathcal{A}$ making at most $q$ queries, each query of maximum size $ln$ and of total size in the number of blocks of at most $\sigma$, has a low probability of distinguishing CMAC from a random function, when parameterized by a random permutation.*

$$\mathsf{Adv}^{\mathsf{prf}}_{\mathit{CMAC}}(\mathcal{A}) \leq \frac{2.5(\sigma+1)^2 + 1.5q^2 + 2q^2l^2}{2^n} \quad (6)$$

*Proof.* This is a direct application of Theorem 2, Theorem 3 and Lemma 9. □

## V. Discussion and Related Work

We have formalized the security of CMAC in EasyCrypt, assuming only that the underlying block cipher is secure. Our proof relies on the computational indistinguishability of CMAC from FCBC. With this formalization, we also verify that all intermediate lemmas are correct and correctly instantiated, identifying and preventing minor flaws in previously published proofs [BR05]. We have already contributed some of these intermediate lemmas, in their general forms, to EasyCrypt's standard library.

Formalizing Black and Rogaway's bound on the probability of collisions in CBC-MAC [BR05] would tighten our verified bound, but requires changes to the EasyCrypt tool. In addition, Black and Rogaway's proof does not yield the best known bounds. Bellare, Pietrzak and Rogaway [BPR05] provide improved bounds for CBC-MAC and ECBC using less generic arguments in their reduction. Our formalization effort can be used as a basis for a formalization of these specific arguments, and perhaps a further generalization, left as future work. Nandi [Nan09] describes an alternative proof strategy for XCBC and CMAC, which yields better bounds for these two constructions by relying on a general theorem–the *Strong Interpolation Theorem*. Using this theorem, Nandi obtains a much tighter asymptotic bound on the security of CMAC: $O(\frac{l \cdot q^2}{2^n})$. However, our goal was not only to formalize a security proof for CMAC, but also to obtain generic results applicable to intermediate constructions and variants. Nandi's direct approach would not have supported this dual goal. The formalization of this theorem, making it suitable for concrete security analysis, and formally applying it to the MAC schemes discussed here would certainly be an interesting lead for future work.

Beringer et al. [BPYA15], [App15] verify the security and functional correctness of a restricted case of OpenSSL's HMAC-SHA256 implementation of the HMAC scheme. Their verification is carried out entirely within the Coq proof assistant through the VST [App11] and FCF [PM15] libraries. Ye et al. [YGS+17] further extend their proof to cover an HMAC-based random number generator. We do not push our proof to implementation level, but could leverage their techniques, and those of others [ABBD16] to get similar implementation-level results, including considerations of side-channel security. In a similar direction of verifying implementations, Zinzindohoué et al. [ZBPB17] present a verified low-level cryptographic library whose functional correctness and some aspects of side-channel security are verified. Their verification does not include cryptographic security considerations, but combining their approach with formal cryptographic proofs as performed in EasyCrypt or FCF is an interesting direction for future work.

Malozemoff et al. [MKG14] propose automated techniques for the analysis and synthesis of secure modes of operation for block cipher modes of operation, including CBC. Hoang, Katz and Malozemoff [HKM15] present an extended approach for the analysis and synthesis of block-cipher modes of operation for *authenticated encryption*, which combine confidentiality and authenticity guarantees. Their technique needs the mode of operation to operate on pairs of blocks, making it inapplicable to CBC-MAC. Automated analysis has also been implemented using EasyCrypt [BCG+13], and has been used to synthesise and verify many *public-key encryption schemes* built from hash functions and trapdoor permutations. However, the approach is very focused and has only limited applicability to other cryptographic constructions. It would be interesting, based on the insights gained from this formal effort, to see whether either technique could be used to analyze and synthesize block cipher-based MAC schemes, including CMAC. Formalizing a primitive's security proof is very different from proving security properties for larger systems, such as protocols. Indeed, if proofs for primitives often involve relatively complex probabilistic arguments, proofs for larger protocols are typically made complex by the presence of state and by their sheer scale. Others have succeeded in formalising security proofs for much larger constructions such as electronic voting [CDD+17] multi-party computation [ABB+17] or authenticated key exchange [BCLS15], including TLS [BFK+13]. Again, our work is complementary to theirs in that these large proofs often stop short of formalising the security of their primitives.

CryptoVerif [Bla08] is another tool for finding sequences of games that constitute proofs in the computational model. This highly automated tool attempts to synthesize intermediate games to prove the security of primitives and protocols [BJST08] and has been used to produce a provably secure implementation of SSH [CB13]. Its automated nature makes some proofs very easy, but proofs for very low-level primitives such as CMAC are for now out of its reach. Studying interactions between automated techniques, such as those used in CryptoVerif, and finer-grained but more effort-intensive interactive techniques such as those used in EasyCrypt is also an interesting direction for future work.

*Acknowledgments:* C. Baritel-Ruet, P.-A. Fouque and B. Grégoire are grateful for the support of the ANR through grant 16 CE39 0012 (SafeTLS) and 14 CE28 0015 (BRUTUS).

## References

[ABB$^{+}$17] José Bacelar Almeida, Manuel Barbosa, Gilles Barthe, Francois Dupressoir, Benjamin Grégoire, Vincent Laporte, and Vitor Pereira. A fast and verified software stack for secure function evaluation. In *Proceedings of The 2017 ACM Conference on Computer and Communications Security (CCS 2017)*. Association for Computing Machinery (ACM), 2017.

[ABBD16] José Bacelar Almeida, Manuel Barbosa, Gilles Barthe, and François Dupressoir. Verifiable side-channel security of cryptographic implementations: constant-time MEE-CBC. In *International Conference on Fast Software Encryption*, pages 163–184. Springer, 2016.

[App11] Andrew W. Appel. Verified software toolchain - (invited talk). In *Programming Languages and Systems - 20th European Symposium on Programming, ESOP 2011, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2011, Saarbrücken, Germany, March 26-April 3, 2011. Proceedings*, pages 1–17, 2011.

[App15] Andrew W Appel. Verification of a cryptographic primitive: SHA-256. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 37(2):7, 2015.

[BCG$^{+}$13] Gilles Barthe, Juan Manuel Crespo, Benjamin Grégoire, César Kunz, Yassine Lakhnech, Benedikt Schmidt, and Santiago Zanella-Béguelin. Fully automated analysis of padding-based encryption in the computational model. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 1247–1260. ACM, 2013.

[BCLS15] Gilles Barthe, Juan Manuel Crespo, Yassine Lakhnech, and Benedikt Schmidt. Mind the gap: Modular machine-checked proofs of one-round key exchange protocols. *IACR Cryptology ePrint Archive*, 2015:74, 2015.

[BDG$^{+}$14] Gilles Barthe, François Dupressoir, Benjamin Grégoire, César Kunz, Benedikt Schmidt, and Pierre-Yves Strub. *EasyCrypt: A Tutorial*, pages 146–166. Springer International Publishing, Cham, 2014.

[BFK$^{+}$13] Karthikeyan Bhargavan, Cédric Fournet, Markulf Kohlweiss, Alfredo Pironti, and Pierre-Yves Strub. Implementing TLS with verified cryptographic security. In *Security and Privacy (SP), 2013 IEEE Symposium on*, pages 445–459. IEEE, 2013.

[BGM04] Mihir Bellare, Oded Goldreich, and Anton Mityagin. The power of verification queries in message authentication and authenticated encryption. *IACR Cryptology ePrint Archive*, 2004:309, 2004.

[BJST08] Bruno Blanchet, Aaron D Jaggard, Andre Scedrov, and J-K Tsay. Computationally sound mechanized proofs for basic and public-key kerberos. In *Proceedings of the 2008 ACM symposium on Information, computer and communications security*, pages 87–99. ACM, 2008.

[BKR94] Mihir Bellare, Joe Kilian, and Phillip Rogaway. The security of cipher block chaining. In *Advances in Cryptology—CRYPTO'94*, pages 341–358. Springer, 1994.

[Bla08] Bruno Blanchet. A computationally sound mechanized prover for security protocols. *IEEE Transactions on Dependable and Secure Computing*, 5(4):193–207, 2008.

[BPR05] Mihir Bellare, Krzysztof Pietrzak, and Phillip Rogaway. Improved security analyses for CBC MACs. In *Crypto*, volume 3621, pages 527–545. Springer, 2005.

[BPYA15] Lennart Beringer, Adam Petcher, Katherine Q. Ye, and Andrew W Appel. Verified correctness and security of openssl HMAC. In *USENIX Security Symposium*, pages 207–221, 2015.

[BR05] John Black and Phillip Rogaway. CBC MACs for arbitrary-length messages: The three-key constructions. *Journal of Cryptology*, 18(2):111–131, 2005.

[BR06] Mihir Bellare and Phillip Rogaway. Code-based game-playing proofs and the security of triple encryption. In *Advances in Cryptology–EUROCRYPT*, volume 4004, page 10, 2006.

[CB13] David Cadé and Bruno Blanchet. From computationally-proved protocol specifications to implementations and application to SSH. *JoWUA*, 4(1):4–31, 2013.

[CDD$^{+}$17] Véronique Cortier, Constantin Cătălin Drăgan, François Dupressoir, Benedikt Schmidt, Pierre-Yves Strub, and Bogdan Warinschi. Machine-checked proofs of privacy for electronic voting protocols. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pages 993–1008. IEEE, 2017.

[DR13] Joan Daemen and Vincent Rijmen. *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media, 2013.

[Dwo16] Morris J Dworkin. Recommendation for block cipher modes of operation: The CMAC mode for authentication. *Special Publication (NIST SP)-800-38B*, 2016.

[EMST78] William F Ehrsam, Carl HW Meyer, John L Smith, and Walter L Tuchman. Message verification and transmission error detection by block chaining, February 14 1978. US Patent 4,074,066.

[HKM15] Viet Tung Hoang, Jonathan Katz, and Alex J Malozemoff. Automated analysis and synthesis of authenticated encryption schemes. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 84–95. ACM, 2015.

[IK03a] Tetsu Iwata and Kaoru Kurosawa. OMAC: One-key CBC MAC. In *FSE*, volume 2887, pages 129–153. Springer, 2003.

[IK03b] Tetsu Iwata and Kaoru Kurosawa. Stronger security bounds for OMAC, TMAC and XCBC. Cryptology ePrint Archive, Report 2003/082, 2003. https://eprint.iacr.org/2003/082.

[MKG14] Alex J Malozemoff, Jonathan Katz, and Matthew D Green. Automated analysis and synthesis of block-cipher modes of operation. In *Computer Security Foundations Symposium (CSF), 2014 IEEE 27th*, pages 140–152. IEEE, 2014.

[Nan09] Mridul Nandi. Improved security analysis for OMAC as a pseudorandom function. *Journal of Mathematical Cryptology*, 3(2):133–148, 2009.

[PM15] Adam Petcher and Greg Morrisett. The foundational cryptography framework. In Riccardo Focardi and Andrew Myers, editors, *Principles of Security and Trust*, pages 53–72, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[Vau00] Serge Vaudenay. Decorrelation over infinite domains: the encrypted CBC-MAC case. In *Selected Areas in Cryptography*, volume 2012, pages 189–201. Springer, 2000.

[YGS$^{+}$17] Katherine Q. Ye, Matthew Green, Naphat Sanguansin, Lennart Beringer, Adam Petcher, and Andrew W Appel. Verified correctness and security of mbedtls HMAC-DRBG. 2017.

[ZBPB17] Jean-Karim Zinzindohoué, Karthikeyan Bhargavan, Jonathan Protzenko, and Benjamin Beurdouche. Hacl*: A verified modern cryptographic library. In *ACM Conference on Computer and Communications Security (CCS)*, 2017.