Leakage-Resilient Authentication and Encryption from Symmetric Cryptographic Primitives

Olivier Pereira
Université catholique de Louvain
ICTEAM – Crypto Group
B-1348, Belgium
olivier.pereira@uclouvain.be

François-Xavier Standaert
Université catholique de Louvain
ICTEAM – Crypto Group
B-1348, Belgium
fstandae@uclouvain.be

Srinivas Vivek
University of Luxembourg
University of Bristol
sv.venkatesh@bristol.ac.uk

ABSTRACT

Leakage-resilient cryptosystems aim to maintain security in situations where their implementation leaks physical information about their internal secrets. Because of their efficiency and usability on a wide range of platforms, solutions based on symmetric primitives (such as block ciphers) are particularly attractive in this context. So far, the literature has mostly focused on the design of leakage-resilient pseudorandom objects (e.g. PRGs, PRFs, PRPs). In this paper, we consider the complementary and practically important problem of designing secure authentication and encryption schemes. For this purpose, we follow a pragmatic approach based on the advantages and limitations of existing leakage-resilient pseudorandom objects, and rely on the (arguably necessary, yet minimal) use of a leak-free component. The latter can typically be instantiated with a block cipher implementation protected by traditional countermeasures, and we investigate how to combine it with the more intensive use of a much more efficient (less protected) block cipher implementation. Based on these premises, we propose and analyse new constructions of leakage-resilient MAC and encryption schemes, which allow fixing security and efficiency drawbacks of previous proposals in this direction. For encryption, we additionally provide a detailed discussion of why previously proposed (indistinguishability based) security definitions cannot capture actual side-channel attacks, and suggest a relaxed and more realistic way to quantify leakage-resilience in this case, by reducing the security of many iterations of the primitive to the security of a single iteration, independent of the security notion guaranteed by this single iteration (that remains hard to define).

1. INTRODUCTION

Motivation. Attacks based on the exploitation of side-channels [21] or faults [17], are an important issue for the security of cryptographic hardware. Motivated by their practical relevance for small embedded devices such as smart cards, a large body of research has investigated solutions to mitigate them, analyzed in more or less abstract models, and leading to more or less efficient implementations. In this context, symmetric cryptographic primitives such as block ciphers are of utmost importance. In general, they are considered as the workhorses of modern cryptography [19]. Because of their low cost and efficiency on a wide range of platforms, they are also a target of choice for physical attacks. Unfortunately, their lack of mathematical structure makes them particularly challenging to protect. Taking the example of side-channel attacks (that will be our main concern), probably the most investigated countermeasure is masking [5] (aka secret sharing [16]). But it implies overheads that are (at least) quadratic in the number of shares used (see e.g. [14] for a comparison of a couple of schemes), and its secure implementation is far from trivial (i.e., hardware engineers have to ensure that the leakage of each share is independent of each other, which is frequently contradicted in practice [22], or need to take this constraint into account by design, which implies additional costs [28]). Following, an alternative approach (*and, as we will see, complementary as well*), next denoted as leakage-resilience, has started to look for new primitives of which the design is inherently (more) secure against such physical attacks. So far, leakage-resilient symmetric cryptography has mostly focused on PRGs (aka stream ciphers) [7, 9, 30, 31, 36, 37, 38, 39], PRFs and PRPs [6, 9, 37, 38]. By contrast, much less work has been carried out on the exploitation of these primitives in the context of standard cryptographic tasks such as authentication and encryption. Our goal in this paper is therefore to clarify how and when to use leakage-resilience in these cases (and for which formal security guarantees).

Preliminaries. Our starting point for dealing with this problem is a recent work of Belaid et al. [3] which shows that concretely, the security improvements brought by leakage-resilience highly depend on whether the underlying primitive is stateful (like PRGs, typically) or stateless (like PRFs and PRPs, typically).[1] That is, despite proofs for both types of primitives being essentially based on the same assumptions (namely the leakage per iteration has to be limited in some sense), ensuring this condition in practice is sig-

CCS'15, October 12–16, 2015, Denver, Colorado, USA.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-3832-5/15/10 ...$15.00.
DOI: http://dx.doi.org/10.1145/2810103.2813626.

[1] By stateful, we mean that the implementation of the primitive has to maintain a state (typically a key) between its consecutive iterations, which implies that different parties involved in the use of this primitive have to be synchronized.

nificantly more difficult in the case of stateless primitives than in the case of stateful ones. In the case of PRGs and stream ciphers, leakage-resilient designs limit the number of measurements that an adversary can obtain per iteration. By contrast, for PRFs and PRPs, they only limit the number of plaintexts for which measurements can be obtained (which still allows the adversary to measure the same plaintext an arbitrary number of times, hence to reduce the noise). Therefore, implementations of leakage-resilient PRGs and stream ciphers (mostly) lead to concrete security against side-channel key recovery attacks at a lower cost than countermeasures like masking. By contrast, implementations of leakage-resilient PRFs and PRPs (mostly) lead to lower concrete security levels than standard PRFs and PRPs protected with such countermeasures. As a result, if we want to stick with constructions based on standard block ciphers for efficiency and usability reasons, there seems to be little hope to have a secure MAC or encryption scheme without further assumption. Indeed, stateless primitives are usually important ingredients of such schemes, and without properties such as a homomorphic structure, block cipher re-use will eventually leak the key in full, as just explained.

In this respect, a natural direction to investigate is to assume that we will need a well protected component (i.e., a block cipher in our case), that we will denote as "leak-free" for convenience. Admittedly, this leak-free component will be much (a dozen to hundred times) slower than an unprotected block cipher implementation, as it could be based on a combination of masking and other countermeasures – in fact, it could also be based on an asymmetric cryptographic primitive enjoying some exploitable homomorphic structure. Concretely, it will probably be imperfect to some extent as well (and we will detail how to capture these imperfections in our analysis). So our goal will be to make minimal use of this component (one call per message, independently of the message, typically), and to combine it with a faster implementation of block cipher in order to get a scheme that would still provide good protection against side-channel attacks (or at least, as good as we can hope), but would also be much more efficient than if we had to use the leak-free component only (or solutions that only exploit the mathematical structure of asymmetric cryptographic primitives such as [18] for encryption and [12, 23] for authentication).

Our contribution. First, we follow this goal of minimizing the need of leak-freeness for two important symmetric cryptographic functionalities, namely authentication and encryption. Second, we clarify and fix two important shortcomings in previously published approaches to these functionalities.

For leakage-resilient MACs, the only existing work based on symmetric primitives is the one by Schipper [35]. The basic idea is simple: take a leakage-resilient PRG and use it to generate keys for a one-time MAC. While this is indeed a stateful primitive, the main problem in this scheme is that the use of the key is limited per message, not per message block. This means that for long messages, and depending on the one-time MAC that is used (CBC-MAC would be problematic, for instance), the adversary can observe a large number of leaking operations exploiting the same key. One partial solution considered in Schipper's thesis is to use a MAC based on a leakage-resilient PRF. But this has a higher implementation cost (as noticed in [23]) and faces the previously discussed problem of stateless primitives. In order to improve this situation, we first propose a new (stateful) leakage-resilient MAC that limits the use of leak-free component to a single IV block (which can be pre-computed), and is efficient for large messages (i.e., requires a single block cipher execution per message block). We then propose a variant of this scheme based on a hash and MAC paradigm. Along these lines, we also put forward that certain standard MACs are better suited for leakage-resilience than others (e.g. HMAC is better than CBC-MAC in this respect).

For encryption, the literature based on symmetric primitives is also sparse. To the best of our knowledge, the work by Abdalla et al. [2] is the only one to address this question. Here, the problem is more general and definitional. That is, a central issue in all the leakage-resilient encryption notions proposed so far is that they exclude the leakage during the challenge queries, or focus on a restricted setting where an encryption is assumed to not leak any single bit of the plaintext that is encrypted (e.g. consider only key leakage). In fact, this is also true for public-key encryption schemes: see, e.g., [27] for an early proposal in this direction and [15] for a more recent one. On the one hand, this seems unavoidable: indeed a single bit of leakage on the plaintext trivially breaks the semantic security game. On the other hand, we argue that excluding challenge leakages is artificial and does not capture the actual adversarial scenario of leaking devices, at least in the context of side-channel attacks based on power consumption and electromagnetic radiation that we consider in this paper (but, we believe, in general as well). And, as a side effect, such definitions do not make a difference between an encryption implementation that would leak the full plaintexts from an implementation that would not leak any information about the plaintexts – a difference that seems to be of crucial importance in the context of encryption. Hence, we propose an alternative way to model the security in front of leakages where we do not try to enforce traditional security notions with a negligible advantage. We rather show that the security of multiple iterations reduces to the security of a single iteration. That is, we show that whatever the adversary is able to do against multiple iterations of our encryption scheme is also possible against a single iteration of this scheme. We believe this approach is more realistic since it does not give users the (illusory) feeling that semantic (or any indistinguishability-based) security can be obtained for encryption schemes with leakage. By contrast, we provide an efficient solution for which the designer is guaranteed that the security of the full construction reduces to the security of a single block (whatever security he is able to achieve).

Remarks. The combination of these results is in fact well in line with the early investigations of Micali and Reyzin, where it was shown that unpredictability-based security is easier to obtain than indistinguishability-based security in the presence of leakage [25]. Concretely, and based on present knowledge, it also means that if semantic (or equivalent) security is required for an application, the best option is to use leakage-resilient authentication to access a leak-free environment first, and to perform encryption only afterwards. In other words, the security guarantees of leakage-resilient encryption, despite practically meaningful (e.g. in order to prevent key recoveries), are indeed much harder to formalize in terms of message confidentiality. Note finally that our following constructions only consider the leakage-resilience of tag generation and encryption. This is a relevant first step, since it is a frequent scenario that only one (costconstrained) party in authentication and encryption has to be protected against side-channel analysis, while the other party (e.g., a reader) is much easier to shield through physical countermeasures [24]. Yet, we also admit that securing the tag verification and decryption parts will most likely be more challenging (since these algorithms are not randomized in most existing MAC and encryption schemes), and leave their investigation as an intersting research problem.

Leakage model. We consider the continuous leakage model since it is the only one capturing actual side-channel attacks. Indeed, if a system is used for a sufficiently long period of time, the amount of leakage observed by an attacker may exceed any apriori determined leakage bound. In this context, we capture the limited informativeness of actual leakages with the recently introduced "simulatable leakage" framework [36]. We are aware of the ongoing discussion about how to implement block ciphers ensuring this empirically verifiable assumption [11]. Yet, and as argued in this reference, it remains the most realistic assumption to reason about leakage we currently have (and in particular, the only one that can be challenged by hardware engineers). Besides, the recent discussion in [29] suggests that the main issue with the leakage simulators of [36] is due to the difficulty to capture the noise distribution in actual leakage traces (i.e., does not relate to exploitable key-dependent signal), and describes ways to design new instances of simulators to overcome this problem (of which the analysis is beyond the scope of this paper). Eventually, and more importantly, we believe our core contribution is the general investigation of leakage-resilient MAC and encryption, as well as the proposal of efficient constructions minimizing the need of leak-freeness. This contribution is quite independent of the quest for a perfectly realistic model of leakage-resilience, which indeed remains a great conceptual challenge. That is, we use a leakage model (here, the simulatability framework) to reason formally about our constructions and make sure they are theoretically founded. But in the first place, we hope that they will be helpful in practice, for cryptographic engineers.

2. LEAKAGE-RESILIENT MESSAGE AUTHENTICATION CODES

2.1 Security definition

Let us recollect the standard definition of a Message Authentication Code. A MAC is a tuple of three polynomial time algorithms $\mathsf{MAC} = (\mathsf{KeyGen}, \mathsf{Mac}, \mathsf{Vrfy})$ defined as follows:

- the **key generation** algorithm $\mathsf{KeyGen}(1^n)$ takes as input the security parameter n and generates a shared (master) secret key k to be used by both the tag generator and the verifier,

- the **MAC generation** algorithm $\mathsf{Mac}(m, k; r)$ takes as input the message m, the secret key k, and possibly some randomness r, and then outputs a tag τ. (In the following, we omit mentioning the randomness in contexts where it is not relevant.)

- the **tag verification** algorithm $\mathsf{Vrfy}(m, \tau, k)$ takes as input a message m, the corresponding tag τ and the secret key k. The algorithm outputs 1 (*accept*) if the tag is valid for the message, else it outputs 0 (*reject*).

We require the usual *correctness property* to be satisfied by the MAC, i.e., $\mathsf{Vrfy}(m, \mathsf{Mac}(m, k; r), k) \rightarrow 1$ for every message m and key k in the range of KeyGen. Informally, the MAC is said to be *existentially unforgeable in the presence of leakage during tag generation* (in short, LR-MAC) if the adversary is unsuccessful in the following security game. As usual, a key is selected as part of the experiment. We do not consider leakages at this step, as the actual way of loading the key into a device can vary quite a lot from one situation to another, and will usually happen at manufacturing time, out of reach of the adversary. Next, the adversary can ask for tags on messages of its choice, computed with that key. This time, a leakage corresponding to the each tag computation is provided to the adversary together with the tag, this leakage being computed through the leakage function L. We will of course place some restrictions on this leakage function later in the paper. Eventually the adversary will also have access to the tag verification oracle, but it will not get leakages during verification (which corresponds to the fact that we only secure the tag generation against side-channel attacks, not the tag verification).[2] The goal of the adversary is to output a valid forgery on a message for which it has not previously obtained a corresponding tag.

Formally, we consider the experiment $\mathsf{Forge}^{\mathsf{euf-cma}}_{\mathcal{A}^{\mathsf{L}},\mathsf{MAC}}(n)$:

```
$\mathsf{Forge}^{\mathsf{euf-cma}}_{\mathcal{A}^{\mathsf{L}},\mathsf{MAC}}(n)$:
    $k \leftarrow \mathsf{KeyGen}(1^n)$
    $\mathcal{F} \leftarrow \emptyset$
    $(m, \tau) \leftarrow \mathcal{A}^{\mathsf{L}, \mathcal{O}^{\mathsf{ML}}(\cdot), \mathcal{O}^{\mathsf{V}}(\cdot,\cdot)}(1^n)$
    If $m \in \mathcal{F}$, then Return $b := 0$
    $b \leftarrow \mathsf{Vrfy}(m, \tau, k)$
    Return $b$

Oracle $\mathcal{O}^{\mathsf{ML}}(m)$:
    $r \leftarrow \{0,1\}^n$
    $\mathcal{F} \leftarrow \mathcal{F} \cup m$
    Return $(\mathsf{Mac}(m, k; r),$
            $\mathsf{L}(m, k; r))$
Oracle $\mathcal{O}^{\mathsf{V}}(m, \tau)$:
    Return $\mathsf{Vrfy}(m, \tau, k)$
```

Conceptually, we do not want to place unnecessary restrictions on the leakage function L. In particular, it is an open problem to determine the complexity of such a function, and whether it can be computed efficiently. So, in order to abstract the complexity of the leakage function, we separate the process of obtaining a leakage, which is going through a physical measurement, from the internal computation of the adversary. The possibility for an adversary $\mathcal{A}$ to obtain leakages from a circuit is then expressed as an oracle access: $\mathcal{A}^{\mathsf{L}}$. Furthermore, we talk about (s, t)-bounded adversaries for adversaries who are able to perform s leakage queries and perform a computation bounded by running time t. Note that those s leakage queries are quite different from the $\mathcal{O}^{\mathsf{ML}}$ queries in the $\mathsf{Forge}^{\mathsf{euf-cma}}_{\mathcal{A}^{\mathsf{L}},\mathsf{MAC}}(n)$ game: when querying L, $\mathcal{A}$ must select the inputs (m, k) of the circuit by himself, while the $\mathcal{O}^{\mathsf{ML}}$ oracle provides $\mathcal{A}$ with leakages about a key k that $\mathcal{A}$ is not expected to know. Those s

[2] As mentioned in the introduction, preventing side-channel attacks during the tag verification will most likely be quite challenging. Indeed, if the adversary can observe the leakage during verification, it should be able to fully recover the key by re-using it many times in the verification phase. Besides, this problem is not specific to symmetric primitives. A similar (yet relaxed) restriction is also made in [23] for a construction of a leakage-resilient MAC based on pairings. In this case, the adversary gets the leakage during verification, but only once, for a given message and randomness pair. Otherwise, the adversary will be able to leak a correct tag bit-by-bit by repeatedly accessing the verification oracle with incorrect tags against the same message.

queries actually correspond to a training phase that $\mathcal{A}$ can perform as part of his attack of the circuit, a practice that is common in (profiled) side-channel analyses.

Definition 1. [LR-MAC] MAC is said to be a (q, s, t, ϵ) secure LR-MAC in the presence of leakage L, if ϵ is a bound on the the advantage of any (s, t)-bounded adversary $\mathcal{A}^{\mathsf{L}}$ playing the experiment $\mathsf{Forge}^{\mathsf{euf-cma}}_{\mathcal{A}^{\mathsf{L}},\mathsf{MAC}}(n)$ and making at most q queries to the $\mathcal{O}^{\mathsf{ML}}$ oracle, that is,

$$\Pr[\mathsf{Forge}^{\mathsf{euf-cma}}_{\mathcal{A}^{\mathsf{L}},\mathsf{MAC}}(n) = 1] \leq \epsilon.$$

Remark 1. The notion of *Strongly-Unforgeable LR-MAC* is a stronger security notion than that in Definition 1. This distinction is analogous to the difference between strong unforgeability and basic unforgeability notions for MAC in the traditional setting (without leakage). In the case of strongly-unforgeable LR-MAC, it suffices for the adversary to output a valid message-tag pair distinct from those pairs it received in its interaction. Hence, it is not necessarily required to output forgery on a distinct message. Otherwise, the security game remains the same as that for LR-MAC.

2.2 Why CBC-MAC is not leakage-resilient

To motivate our following investigations, we start with a brief explanation of why standard MAC constructions such as CBC-MAC, are not leakage-resilient by default. For this purpose, just look at the informal description of CBC-MAC in Figure 1. Here, the master key k is used in every iteration of the MAC (and kept constant among messages). So we are exactly in the scenario where a standard side-channel key recovery attack is the most devastating. As a result, a natural suggestion for improving the situation would be to combine CBC-MAC with a leakage-resilient stream cipher so that every message block would be processed with a different key. Yet, this would typically imply three block cipher executions per message block. In the following section, we show that a three times more efficient solution can be obtained.

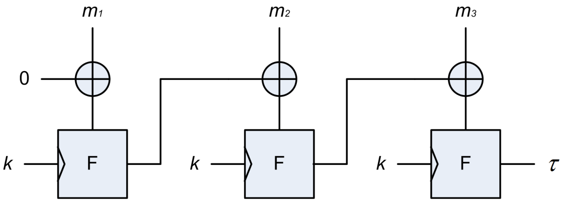

Figure 1: CBC-MAC.

2.3 Leakage-resilient tag generation

We next present a plausible construction of LR-MAC that is a variant of the standard CBC-MAC. The scheme $\mathsf{MAC}_1 = (\mathsf{KeyGen}_1, \mathsf{Mac}_1, \mathsf{Vrfy}_1)$, depicted in Figure 2 and described below, is a fixed length MAC that takes as input ℓ blocks of messages ($\ell \geq 1$), each block being n-bit long.[3] The construction requires two pseudorandom functions F and F^* that we will typically instantiate with a block cipher. The same block cipher could be used twice, but we distinguish between the two functions because their implementations will be quite different: we demand F^* for being essentially leak-free (or strongly protected) and expect its implementation to be fairly inefficient, while F is expected to be more efficient and require much less protection against leakages. How to realize the leak-free function F^* and to capture its possible imperfections will be discussed in Section 4.

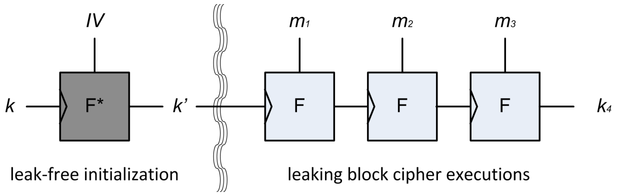

Figure 2: Re-keying MAC.

Description of MAC_1:

- $\mathsf{KeyGen}_\mathbf{1}(1^n)$: Choose a shared master secret key $k \xleftarrow{\$} \{0,1\}^n$.
- $\mathsf{Mac}_\mathbf{1}\,(m, k)$: Parse $m = \langle m_1, m_2, \ldots, m_\ell \rangle$. Choose $IV \xleftarrow{\$} \{0,1\}^n$. Compute the session key $k' := k_1 = \mathsf{F^*}_k(IV)$.
  - for $j = 2, \ldots \ell{+}1$: compute $k_j = \mathsf{F}_{k_{j-1}}\,(m_{j-1})$.

  Return $\tau = (IV, k_{\ell+1})$.
- $\mathsf{Vrfy}_\mathbf{1}\,(m, \tau, k)$: Parse $\tau = (IV, tg)$. Compute $\tau' \leftarrow \mathsf{Mac}_1\,(m, k, IV)$.
  - If $\tau' \overset{?}{=} \tau$, then **return** 1 (correct), else **return** 0 (incorrect).

Compared to CBC-MAC, one can directly see that this scheme brings improved leakage-resilience, since a new session key is used for every new message. Compared to the LR-MAC of Schipper in [35], we have the additional advantage that the key evolves for every message block, which allows us to state our requirements on the leakage for a single iteration of the scheme. We also exploit the block cipher quite efficiently since this new stateful construction essentially requires an execution of F per message block. Eventually, we require a very minimum use of the leak-free component F^* (depicted in dark grey on the figure): it is only needed to encrypt a single random IV under the master key k.

Remark 2. In the MAC_1 construction above, a random IV is chosen to compute every new tag on a message m. Nevertheless, the security of the construction will not be affected even if we choose the IVs arbitrarily, as long as they are distinct (cf. proof of Theorem 1). Hence, for instance, we could use a counter mode (i.e., start with $IV = 0$, and then successively increment it). This would only require the MAC implementation to maintain a public state.

We now prove the LR-MAC security of our MAC_1 construction based on the pseudorandomness and the simulatable leakage assumption of the block cipher F, assuming that the implementation of F^* used to compute the session key k' from IV is leak-free. We first recall these properties.

Definition 2. [Pseudorandom Function (PRF) [36, Definition 2]] A block cipher $\mathsf{F} : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ is a

[3] For clarity, we stick to a fixed-length MAC construction, just as CBC-MAC. The adaptations of CBC-MAC to a variable length MAC would apply here as well – see, e.g., [4].

(s, t, ϵ_{prf}) PRF in the presence of leakage function L if, for every (s, t)-bounded adversary $\mathcal{A}^{\mathsf{L}(\cdot,\cdot)}$, we have:

$$|\Pr[\mathcal{A}^{\mathsf{L}(\cdot,\cdot),\mathsf{F}_K(\cdot)} = 1] - \Pr[\mathcal{A}^{\mathsf{L}(\cdot,\cdot),\mathsf{R}(\cdot)} = 1]| \leq \epsilon_{prf},$$

where $K \xleftarrow{\$} \{0,1\}^n$ and R is a random function.

As discussed in [36], this definition would be exactly equivalent to the standard notion of PRF if the leakage function was polynomial time: indeed, in that case, $\mathcal{A}$ could emulate L internally. However, as discussed above, we do not want to make any such restriction, since it remains an open problem to determine the exact complexity of physical functions, and since leakages generally result from a physical process rather than a traditional computational process. This observation motivates the next q-simulatable leakage assumption that is defined via the following game and can be directly challenged by hardware engineers, as detailed in [36].

Game q-$\mathsf{sim}(\mathcal{A}, \mathsf{F}, \mathsf{L}, \mathcal{S}, b)$ [36, Section 2.1].

The challenger selects two random keys $k, k^* \xleftarrow{\$} \{0,1\}^n$. The output of the game is a bit b' computed by $\mathcal{A}^{\mathsf{L}}$ based on the challenger responses to a total of at most q adversarial queries of the following type:

Query	Response if $b = 0$	Response if $b = 1$
$\mathsf{Enc}(x)$	$\mathsf{F}_k(x)$, $\mathsf{L}(k, x)$	$\mathsf{F}_k(x)$, $\mathcal{S}^{\mathsf{L}}(k^*, x, \mathsf{F}_k(x))$

and one query of the following type:

Query	Response if $b = 0$	Response if $b = 1$
$\mathsf{Gen}(z, x)$	$\mathcal{S}^{\mathsf{L}}(z, x, k)$	$\mathcal{S}^{\mathsf{L}}(z, x, k^*)$

It directly leads to the following definition of a block cipher implementation with q-simulatable leakages.

Definition 3. [q-simulatable leakages [36, Defn. 1]] Let F be a block cipher having leakage function L. Then F is said to have $(s_{\mathcal{S}}, t_{\mathcal{S}}, s_{\mathcal{A}}, t_{\mathcal{A}}, \epsilon_{q\text{-}sim})$ q-simulatable leakages if there is an $(s_{\mathcal{S}}, t_{\mathcal{S}})$-bounded simulator $\mathcal{S}^{\mathsf{L}}$ such that, for every $(s_{\mathcal{S}}, t_{\mathcal{S}})$-bounded adversary $\mathcal{A}^{\mathsf{L}}$, we have:

$$|\Pr[q\text{-}\mathsf{sim}(\mathcal{A}, \mathsf{F}, \mathsf{L}, \mathcal{S}^{\mathsf{L}}, 1) = 1] - \Pr[q\text{-}\mathsf{sim}(\mathcal{A}, \mathsf{F}, \mathsf{L}, \mathcal{S}^{\mathsf{L}}, 0) = 1]| \leq \epsilon_{q\text{-}sim}.$$

Eventually, the following lemma is a consequence of Definition 2 and Definition 3 (for 2− simulatable leakages) [36].

Lemma 1. [2-simulatable ideal execution [36, Lemma 1]] *Let $\mathsf{F} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a (s, t, ϵ_{prf}) PRF in the presence of leakage function L having $(s_{\mathcal{S}}, t_{\mathcal{S}}, s, t, \epsilon_{2\text{-}sim})$ 2-simulatable leakages, and let $\mathcal{S}^{\mathsf{L}}$ be an appropriate $(s_{\mathcal{S}}, t_{\mathcal{S}})$-bounded leakage simulator. Then, for every $k^-, p_0, p_1, z \in \{0,1\}^n$ and every $(s - 3s_{\mathcal{S}}, t - \max(t_{prf}, t_{sim}))$-bounded distinguisher $\mathcal{D}^{\mathsf{L}}$, the following holds:*

$$\Big|\Pr\left[\mathcal{D}^{\mathsf{L}}\left(y^+, k^+, \mathsf{L}(k', p_0), \mathsf{L}(k', p_1), \mathcal{S}^{\mathsf{L}}(k^-, z, k')\right) = 1\right] - \Pr\left[\mathcal{D}^{\mathsf{L}}\left(y^*, k^*, \mathcal{S}^{\mathsf{L}}(k', p_0, y^*), \mathcal{S}^{\mathsf{L}}(k', p_1, k^*), \mathcal{S}^{\mathsf{L}}(k^-, z, k')\right) = 1\right]\Big| \leq \epsilon_{prf} + \epsilon_{2\text{-}sim},$$

where $k', y^*, k^* \xleftarrow{\$} \{0,1\}^n$, $y^+ = \mathsf{F}(k', p_0)$, $k^+ = \mathsf{F}(k', p_1)$. Furthermore, t_{prf} is equal to $3 \cdot t_{\mathcal{S}}$ augmented with the time needed to make 2 oracle queries to the PRF challenger and select a uniformly random key in $\{0,1\}^n$, and t_{sim} is the time needed to relay the content of two Enc *and one* Gen *queries from and to a q-sim challenger.*

Remark 3. We note that the output of the two Enc and the Gen queries in Lemma 1 can be obtained adaptively. More precisely, let $\langle d_1, d_2, d_3, d_4, d_5\rangle$ denote the input received by $\mathcal{D}^{\mathsf{L}}$. The above indistinguishability result holds even if $\mathcal{D}^{\mathsf{L}}$ adaptively obtains the input as $\langle d_1, d_3\rangle$, $\langle d_2, d_4\rangle$, and d_5, in any order of its choice. This observation will be useful in the following security analysis of MAC_1.

Remark 4. Note that besides the previously mentioned simulatability, we need to assume that the blocks leak independently of each other. This actually corresponds to the "only computation leaks" assumption (or "independent leakage" assumption) that is anyway required for any proof of leakage-resilience to hold. In the present case, we believe that it is reasonable to have it satisfied in practice, since we need it at the macroscopic level of fairly large blocks. That is, as for [7] and follow up works, it seems unlikely that our construction will be broken because of small deviations from this assumption (which can possibly be reduced at the hardware level, e.g. by shielding blocks with ground lines).

2.3.1 Security of MAC_1.

We now establish the LR-security of our MAC_1 construction.

Theorem 1. *Let $\mathsf{F} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be an (s, t, ϵ_{prf}) PRF having $(s_{\mathcal{S}}, t_{\mathcal{S}}, s, t, \epsilon_{2\text{-}sim})$ 2-simulatable leakages. Then, the instantiation of MAC_1 with F is an (q, s', t', ϵ')-strongly-unforgeable LR-MAC on messages of length ℓ with n-bit blocks, where:*

$$\epsilon' \leq \epsilon_{prf} + (q+1)\ell(\epsilon_{prf} + \epsilon_{2\text{-}sim}) + negl(n),$$

with $s' \approx s - q \cdot \ell \cdot s_{\mathcal{S}}$ and $t' \approx t - \tilde{t}$, where $\tilde{t}$ is the time required by the challenger to simulate the experiment $\mathsf{Forge}^{\mathsf{euf-cma}}_{\mathcal{A}^{\mathsf{L}},\mathsf{MAC}}(n)$ for the construction MAC_1, which essentially consists of $(q+1) \cdot (\ell+1)$ evaluations of F and $q \cdot \ell$ calls to the simulator $\mathcal{S}^{\mathsf{L}}$. Here, $negl(n)$ refers to a negligible function of n assuming that q and ℓ are polynomially bounded in n.

Proof. The proof of strongly-unforgeable LR-MAC security for our MAC_1 construction is available in Appendix A. It proceeds by a sequence of hybrid games, which essentially follows the strategy introduced in [36, Theorem 1]. □

Remark 5. A glance at Figure 2 might suggest that only the 1-simulatability leakage assumption would suffice for the security of the MAC_1 construction, but this does not seem to be the case. Indeed, for most parts of the security reduction, the 1-simulatability leakage assumption is enough. But because we allow the adversary to possibly output a forgery on a previously used IV, we need the second output pair of the leakage simulator to enable us to verify the attempted forgery by the adversary (on this particular IV). Note that this issue only relates to the reduction and has nothing to do with the MAC construction itself (for which we anyway exclude the leakage during verification).

Remark 6. From a performance point-of-view, the construction is essentially as fast as one can hope since it has an amortized cost of one (weakly protected) block cipher execution per message block – see the table in [23] for an overview. However, strict comparisons with these previous works is not possible due to their different leakage models. In particular, as recently discussed in [10], simulatable leakage and bounded leakage are not implied by each other.

2.4 Variation: the hash then MAC paradigm

To conclude this section, we note that in view of how the message in Figure 2 is processed, an alternative (and in fact even simpler) solution to build a leakage-resilient MAC is to rely on the hash then MAC paradigm. Such a proposal is intuitively depicted in Figure 3, where we can see that the leakage-resilience of the scheme now really boils down to the execution of the leak-free block cipher on a random IV, which comes at the cost of an additional building block (namely a collision-resistant hash function). This essentially results from the fact that the hash function is only executed on public inputs. Interestingly, this construction also suggests that a standard solution like HMAC could be slightly tweaked in order to become leakage-resilient (which is in contrast with the previously mentioned CBC-MAC).

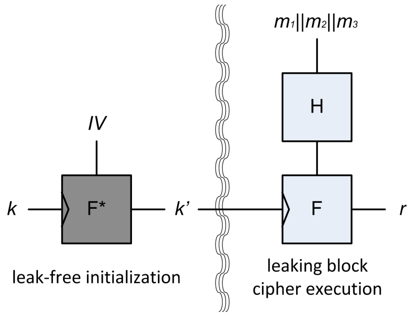

Figure 3: Hash then MAC.

More precisely, our construction $\mathsf{MAC}_2 = (\mathsf{KeyGen}_2, \mathsf{Mac}_2, \mathsf{Vrfy}_2)$, can be viewed as a special instantiation of MAC_1 where the messages of arbitrary length are first hashed to a single n-bit block using the hash function $\mathsf{H} : \{0,1\}^* \to \{0,1\}^n$. Then a tag is generated for this hashed block using MAC_1, which makes its analysis straightforward.

Description of MAC_2:

- $\mathsf{KeyGen}_\mathbf{2}(1^n)$: Choose a hash function $\mathsf{H} : \{0,1\}^* \to \{0,1\}^n$, and a shared symmetric-key $k \xleftarrow{\$} \{0,1\}^n$.
- $\mathsf{Mac}_\mathbf{2}\,(m, k)$: Choose $IV \xleftarrow{\$} \{0,1\}^n$. Compute $k' = \mathsf{F^*}_k(IV)$, $h = \mathsf{H}(m)$, and $r = \mathsf{F}_{k'}(h)$. **Return** $\tau = (IV, r)$.
- $\mathsf{Vrfy}_\mathbf{2}\,(m, \tau, k)$: Parse $\tau = (IV, tg)$. Compute $\tau' \leftarrow \mathsf{Mac}_2\,(m, k, IV)$.
  - If $\tau' \overset{?}{=} \tau$, then **return** 1 (correct), else **return** 0 (incorrect).

Our proof requires the definition of a collision resistant hash function (sampled from a family) which only operates on public values. Hence, we do not consider any leakage here.

Definition 4. [Collision Resistance]. A hash function $\mathsf{H} : \{0,1\}^* \to \{0,1\}^n$ sampled from a family of functions $\mathcal{H}$ is (t, ϵ_{cr}) collision-resistant if for any adversary $\mathcal{A}$ running for time at most t, its advantage in outputting $m, m' \in \{0,1\}^*$ such that $m \neq m'$ and $\mathsf{H}(m) = \mathsf{H}(m')$, is at most ϵ_{cr}.

Based on this definition, the following theorem establishes the leakage-resilience of our MAC_2 construction.

THEOREM 2. *Let $\mathsf{F} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be an (s, t, ϵ_{prf}) PRF having $(s_\mathcal{S}, t_\mathcal{S}, s, t, \epsilon_{2\text{-}sim})$ 2-simulatable leakages, and let $\mathsf{H} : \{0,1\}^* \to \{0,1\}^n$ be a $(t, \epsilon_{\mathsf{cr}})$ collision-resistant hash function. Then, the instantiation of MAC_2 with F and H is an (q, s', t', ϵ')-strongly-unforgeable LR-MAC on messages of* arbitrary length*, where:*

$$\epsilon' \leq \epsilon_{cr} + \epsilon_{prf} + (q+1)(\epsilon_{prf} + \epsilon_{2\text{-}sim}) + negl(n),$$

with $s' \approx s - q \cdot s_\mathcal{S}$ and $t' \approx t - \tilde{t}$, where $\tilde{t}$ is the time required by the challenger to simulate the experiment $\mathsf{Forge}^{\mathsf{euf-cma}}_{\mathcal{A}^\mathsf{L},\mathsf{MAC}}(n)$ for the construction MAC_2, which essentially consists of $2 \cdot (q+1)$ evaluations of F and q calls to the simulator $\mathcal{S}^\mathsf{L}$. Here, $negl(n)$ again refers to a negligible function of n assuming that q and ℓ are polynomially bounded in n.

Proof sketch. We just observe that if the adversary is unable to break the collision resistance of H, then it has to output a valid forgery on a new hash output (corresponding to some message) for a previously queried IV, or on an old hash output but for a new IV. By treating the n-bit hash outputs as the message space of MAC_1, we obtain the above bound on the advantage of $\mathcal{A}$ from Theorem 1 (with $\ell = 1$). Note that the adversary's advantage in breaking the collision-resistance of H is ϵ_{cr}. □

Remark 7. For a 128-bit block cipher such as the AES-128, we will have $\epsilon_{cr} = 2^{-64}$ (because the hash function outputs n bits) and $\epsilon_{prf} = 2^{-64}$ (because of the PRP to PRF conversion that we use for simplicity in our proof). However, beyond birthday security could potentially be obtained by hashing on $2n$-bit values and replacing the block cipher by a tweakable block cipher, e.g. as done in the context of authenticated encryption [20], which we leave as an open problem. Besides, note that this variant allows gaining a factor ℓ compared to the bound of Theorem 1, which comes at the cost of an additional primitive to implement.

3. LEAKAGE-RESILIENT ENCRYPTION

3.1 Security definition

We now turn to the construction of a leakage-resilient symmetric encryption scheme ENC with key generation algorithm Gen, encryption algorithm Enc and decryption algorithm Dec. The Enc algorithm proceeds on messages made of a variable number of blocks, i.e., messages from the set $(\{0,1\}^n)^*$ where n is the block size. For this scheme, we define a $\mathsf{PrivK}^{\mathsf{lmcpa,b}}_{\mathcal{A}^\mathsf{L},\mathsf{ENC}}$ game, analogue to the traditional IND-CPA security game, but in a physical setting where all encryption operations, including the test query, return to the adversary a leakage together with a ciphertext.

$\mathsf{PrivK}^{\mathsf{lmcpa,b}}_{\mathcal{A}^\mathsf{L},\mathsf{ENC}}$ is the output of the following experiment:

1. A key k is generated by running Gen.
2. $\mathcal{A}^\mathsf{L}$gets access to a leaking encryption oracle that, on messages of arbitrary block length, returns an encryption of these messages together with the leakage resulting from the encryption process.
3. $\mathcal{A}^\mathsf{L}$ submits two messages m_0 and m_1 of identical block length
4. A ciphertext $c \leftarrow \mathsf{Enc}_k(m_b)$ is computed, resulting in a leakage l. Both c and l are given to $\mathcal{A}^\mathsf{L}$.
5. $\mathcal{A}^\mathsf{L}$ can keep accessing the leaking encryption oracle.
6. $\mathcal{A}^\mathsf{L}$ outputs a bit b'.

Naturally, we will be interested in the difference $|\mathsf{PrivK}^{\mathsf{lmcpa},0}_{\mathcal{A}^{\mathsf{L}},\mathsf{ENC}} - \mathsf{PrivK}^{\mathsf{lmcpa},1}_{\mathcal{A}^{\mathsf{L}},\mathsf{ENC}}|$, which we would like to be minimal. However, and as discussed in the introduction, since we consider leakages even during the test query, we cannot expect this difference to be negligible. For that reason, we rather focus on establishing bounds that are derived from the security of a considerably simpler encryption scheme, that encrypts only one message made of a single block per key. That is, we want to show that any security guarantee that can be ensured for this simple (single-block, one-time) encryption scheme (next denoted by $\mathsf{ENC}s$) extends to our full construction ENC. This is eventually what we achieve in Theorems 3 and 4, which relate the CPA security of the multi-block ENC scheme to the eavesdropper security of the single-block $\mathsf{ENC}s$ scheme. We believe that such a result concretely helps the task of secure implementation and security evaluation in two ways:

1. The eavesdropper security game gives a unique leakage for a single-block message to the adversary, which is a most limited input to run a side-channel attack (in practice such attacks usually rely on a few hundred leakages). This means that a cheap and relatively weakly protected implementation could be used even in a setting where long messages are encrypted [3].
2. Security evaluations can also focus on the (comparatively) simpler task of assessing the security of a single encryption round, without needing to care about the combination of leakages from the encryption of multiple blocks of message. (For instance, leaking one bit of a key per encryption block is probably not a problem when encrypting a single block, but could become a problem if the encryption of each block leaks a new bit. Our proof guarantees that there is no such risk.)

The rest of this section is organized as follows. We start in Section 3.2 by defining our leakage-resilient encryption scheme ENC and its leakage model. Next, in Section 3.3, we define our one-time and one-block encryption scheme $\mathsf{ENC}s$, together with its leakage model. Based on these definitions, we build our security analysis as follows. In Section 3.4, we define an idealized version $\mathsf{ENC}s^{\mathsf{I}}$ of $\mathsf{ENC}s$ that has perfectly random outputs and simulated leakages for the PRFs. We also define a one-time (but multiple block) version of the ENC scheme, which we call $\mathsf{ENC}\ell$ (the ℓ referring to the ℓ blocks), as well as an ideal version $\mathsf{ENC}\ell^{\mathsf{I}}$ of it. We conclude this section by bounding the probability that an adversary distinguishes between real and ideal versions of the schemes. In Section 3.5, we push our analysis one step further, bounding the probability that an adversary breaks eavesdropper security for the $\mathsf{ENC}\ell^{\mathsf{I}}$ scheme as a function of the probability of breaking that same property on the $\mathsf{ENC}s^{\mathsf{I}}$ scheme. The result from Section 3.4 can then be used to move back to the real encryption schemes. Eventually, in Section 3.6, we conclude by relating the CPA security of the ENC scheme to the eavesdropper security of the $\mathsf{ENC}\ell$ scheme.

3.2 Leakage-resilient encryption scheme

The ENC *scheme.* Our starting point is the leakage-resilient stream cipher from [36], which we transform into an encryption scheme by XORing its output with the message to be encrypted. CPA security is obtained by adding an initialization round, which generates the stream-cipher seed by applying a leak-free PRF, keyed with the encryption key, to an initialization vector IV, as represented in Figure 4. As for the previous MAC constructions, we require the initialization step be leak-free in order to make sure that, despite the fact that it will be executed many times with the same key, that key will remain safe. And here as well, this use of leak-free component is minimal (a single execution per message) and independent of the message size (so that the fresh key k' can possibly be pre-computed.) The ENC encryption scheme is defined more formally in Table 1.

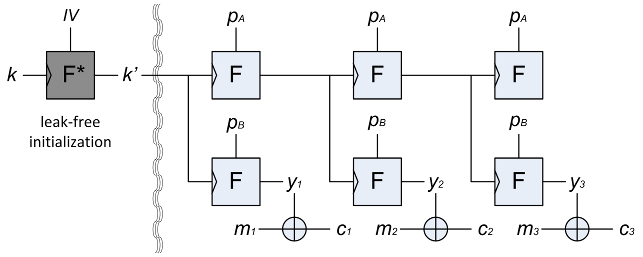

Figure 4: Leakage-resilient encryption scheme.

- Gen picks a random key $k \leftarrow \{0,1\}^n$.
- Enc picks a random $IV \leftarrow \{0,1\}^n$, then computes $k_0 = \mathsf{F^*}_k(IV)$ using the leak-free PRF. The encryption of the ℓ-block message $m = \langle m_1, \ldots, m_\ell \rangle$ is then computed as $IV, c_1, \ldots, c_\ell$, where $c_i = y_i \oplus m_i$, $y_i = \mathsf{F}_{k_{i-1}}(p_B)$ and $k_i = \mathsf{F}_{k_{i-1}}(p_A)$ (p_A and p_B are public constants, where $p_A \neq p_B$).
- Dec proceeds in the natural way.

Table 1: The ENC encryption scheme

Leakage model and assumptions. We capture the leakages of an implementation of this encryption scheme through two leakage functions: $\mathsf{L_F}(p,k)$ that defines the leakage of each PRF running on plaintext p with key k, and $\mathsf{L}_\oplus(m,y)$ that defines the leakage of computing the XOR of m and y. (When the adversary $\mathcal{A}^{\mathsf{L}}$ has the single L superscript, we mean that it can query both these leakage functions.) So, the encryption of each message block m_i causes the following leakages: $\mathsf{L_F}(p_A, k_{i-1})$, $\mathsf{L_F}(p_B, k_{i-1})$ and $\mathsf{L}_\oplus(m_i, y_i)$. Here and later, we precede an algorithm with the L letter (e.g. $\mathsf{LEnc}_k(m)$) to refer to both the output of an encryption and the resulting leakage. As in the previous section about MACs, we require the leakages of the PRF F to be 2-simulatable, but no more. As a consequence, leakage functions do not need to be efficient, and can possibly leak several bits of information on their inputs. In particular, they can provide several bits of information on y_i and m_i, which makes traditional security notions based on indistinguishability impossible to achieve. We believe that, without additional leak-free component, this is just unavoidable: at some point, messages need to be used during the encryption process, and this use must be expected to leak information that is sufficient to win any indistinguishability game.

3.3 Single-block one-time encryption scheme

We define, in the left part of Table 2, the $\mathsf{ENC}s$ single-block one-time encryption scheme, from the security of which we will infer the $\mathsf{PrivK}^{\mathsf{lmcpa},}_{\mathcal{A}^{\mathsf{L}},\mathsf{ENC}}$ security of our ENC multi-block

Description of $\mathsf{ENC}s$:	Description of $\mathsf{ENC}s^{\mathsf{I}}$:
- Gen picks $k_0 \leftarrow \{0,1\}^n$. - $\mathsf{Enc}s_{k_0}(m)$ returns (k_1, c_1), where $c_1 = y_1 \oplus m$, $y_1 = F_{k_0}(p_B)$ and $k_1 = F_{k_0}(p_A)$. - Dec proceeds in the natural way. The leakage resulting from $\mathsf{Enc}s_{k_0}(m)$ is defined as $\mathsf{L}_{\mathsf{Enc}s}(k_0, m) := (\mathsf{L}_{\mathsf{F}}(p_A, k_0), \mathsf{L}_{\mathsf{F}}(p_B, k_0), \mathsf{L}_{\oplus}(m, y_1), \mathcal{S}^{\mathsf{L}}(k^-, p_A, k_0), k^-)$ with $k^- \leftarrow \{0,1\}^n$.	- $\mathsf{Enc}s^{\mathsf{I}}{}_{k_0}(m)$ returns (k_1, c_1), where $c_1 = y_1 \oplus m$, $y_1 \leftarrow \{0,1\}^n$ and $k_1 \leftarrow \{0,1\}^n$. The leakage resulting from $\mathsf{Enc}s^{\mathsf{I}}{}_{k_0}(m)$ is defined as $\mathsf{L}_{\mathsf{ENC}s^{\mathsf{I}}}(k_0, k_1, y_1, m) := (\mathcal{S}^{\mathsf{L}}(k_0, p_A, k_1), \mathcal{S}^{\mathsf{L}}(k_0, p_B, y_1), \mathsf{L}_{\oplus}(m, y_1), \mathcal{S}^{\mathsf{L}}(k^-, p_A, k_0), k^-)$ with $k^- \leftarrow \{0,1\}^n$.

Table 2: The $\mathsf{ENC}s$ and $\mathsf{ENC}s^{\mathsf{I}}$ schemes.

scheme. The $\mathsf{ENC}s$ scheme, while being similar to the single block version of ENC, bears important differences with it:

1. It has no leak-free initialization process. This is not necessary for a one-time version of the scheme.

2. Its ciphertext contains k_1. While being harmless from a black-box point of view, including k_1 in the ciphertext will show to be useful in bounding the amount of information that leakages can transfer between rounds. We will provide constructive evidence that this addition is necessary after the proof of Lemma 3.

3. Its leakages contain $\mathcal{S}^{\mathsf{L}}(k^-, p_A, k_0), k^-$ with a random k^-. This leakage has a similar purpose as adding k_1 in the ciphertext. Namely, it will be used to bound the information that can leak, in the multi-block setting, from the encryption of previous blocks.

3.4 One-time ideal versions of our schemes

We now idealize the $\mathsf{Enc}s$ encryption process by replacing the use of F for computing k_1 and y_1 by the selection of random values. Furthermore, we adapt the corresponding leakages using $\mathcal{S}^{\mathsf{L}}$. The resulting algorithms are defined in the right part of Table 2. The following lemma expresses that leaking encryptions produced with these two schemes are hard to distinguish, by relying on the 2-simulatability assumption and the properties of the PRF.

Lemma 2. **Ideal single block encryption.** *Let $\mathsf{F} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a (s, t, ϵ_{prf})-PRF, whose implementation has a leakage function L_{F} having $(s_{\mathcal{S}}, t_{\mathcal{S}}, s, t, \epsilon_{2\text{-}sim})$ 2-simulatable leakages, and let $\mathcal{S}^{\mathsf{L}}$ be an appropriate $(s_{\mathcal{S}}, t_{\mathcal{S}})$-bounded leakage simulator. Then, for every $m, p_A, p_B, p \in \{0,1\}^n$, $p_A \neq p_B$, and every $(s - s_r, t - t_r)$-bounded distinguisher $\mathcal{D}^{\mathsf{L}}$, the following holds:*

$$\Big|\Pr\Big[\mathcal{D}^{\mathsf{L}}\left(m, \mathsf{LEnc}s(k_0, m)\right) = 1\Big] - \Pr\Big[\mathcal{D}^{\mathsf{L}}\Big(m, \mathsf{LEnc}s^{\mathsf{I}}(k_0, m)\Big) = 1\Big]\Big| \leq \epsilon_{prf} + \epsilon_{2\text{-}sim},$$

with $s_r := 3 \cdot s_{\mathcal{S}} + 1$, $t_r = \max(t_{prf}, t_{sim})$, t_{prf} being equal to $3 \cdot t_{\mathcal{S}}$ augmented with the time needed to make 2 oracle queries to the PRF challenger and select a uniformly random key in $\{0,1\}^n$, and t_{sim} the time needed to relay the content of two Enc and one Gen queries from and to a q-sim challenger.

Proof. We follow the same approach as used for proving Lemma 1: first replace the leakages of $\mathsf{LEnc}s$ with simulated leakages, relying on the simulatability assumption, then replace the outputs of the PRF of $\mathsf{LEnc}s$ with random values, relying on the assumption that F is a PRF. □

We now transpose the definitions of $\mathsf{ENC}s$ and $\mathsf{ENC}s^{\mathsf{I}}$ to the multi-block setting, but still focusing on the one-time encryption case. The resulting schemes, $\mathsf{ENC}\ell$ and $\mathsf{ENC}\ell^{\mathsf{I}}$ are described in Table 3. These ideal versions are closer to the ENC scheme definition: while we still ignore the leak-free initialization process, the ciphertexts do not contain the extra key block any more, and the leakages follow the natural definition. Just as before, we express that leaking encryptions produced with these two schemes are hard to distinguish.

Lemma 3. **Ideal multiple block encryption.** *Let F and $\mathcal{S}^{\mathsf{L}}$ be defined as in Lemma 2. Then, for every ℓ-block message m, every p_A, p_B and every $(s - s_r, t - t_r)$-bounded distinguisher $\mathcal{D}^{\mathsf{L}}$, the following holds:*

$$\Big|\Pr\Big[\mathcal{D}^{\mathsf{L}}\left(m, \mathsf{LEnc}\ell(k_0, m))\right) = 1\Big] - \Pr\Big[\mathcal{D}^{\mathsf{L}}\Big(m, \mathsf{LEnc}\ell^{\mathsf{I}}(k_0, m)\Big) = 1\Big]\Big| \leq \ell(\epsilon_{prf} + \epsilon_{2\text{-}sim}).$$

Here, $s_r = \ell(2s_{\mathcal{S}} + 3)$ and t_r is equal to $2\ell t_{\mathcal{S}}$ augmented with the time needed to pick 2ℓ random values in $\{0,1\}^n$, evaluate F 2ℓ times and compute ℓ $\oplus$ operations.

Proof. We define the hybrid distributions $H_0, \ldots, H_\ell$ in which $H_i(m)$ is defined as the concatenation of an ideal execution $\mathsf{LEnc}\ell^{\mathsf{I}}{}_{k_0}(m_{[1,i]})$ and a real execution $\mathsf{LEnc}\ell_{k_i}(m_{[i+1,\ell]})$ with k_0 chosen uniformly at random and k_i resulting from the evaluation of $\mathsf{Enc}\ell^{\mathsf{I}}$. It is clear that H_0 is distributed just as the inputs of $\mathcal{D}^{\mathsf{L}}$ in the first probability distribution from the lemma's statement, while H_ℓ is distributed as the inputs of $\mathcal{D}^{\mathsf{L}}$ in the second probability distribution. We now show that the probability with which $\mathcal{D}^{\mathsf{L}}$ can distinguish H_{i-1} from H_i is bounded by $\epsilon_{prf} + \epsilon_{2\text{-}sim}$, which will in turn imply the expected result. For this purpose, we build, from $\mathcal{D}^{\mathsf{L}}$, a (s, t)-bounded distinguisher $\mathcal{D}^{\mathsf{L}\prime}$ between the two distributions d_1 and d_2 that are the input of the distinguisher of Lemma 2. $\mathcal{D}^{\mathsf{L}\prime}$ receives its inputs m_i, c_i, k_i, and leakages l_A, l_B, $l_\oplus$, l_s, k_{i-2} sampled from d_1 or d_2. It then: (1) Samples the encryption of the $i-1$ first blocks of m from $\mathsf{LEnc}\ell^{\mathsf{I}}$, except that it uses k_{i-2} as key for the round $i-1$ and l_s as leakage for computing k_{i-1}; (2) Extends it with c_i and the leakages l_A, l_B and $l_\oplus$ for the i-th round; (3) Extends it with $\mathsf{LEnc}\ell_{k_i}(m_{[i+1,\ell]})$ for the last $\ell - i - 1$ rounds. It can be easily verified that, if the inputs of $\mathcal{D}^{\mathsf{L}\prime}$ are sampled from d_1 (resp d_2), then $\mathcal{D}^{\mathsf{L}\prime}$ produced something sampled according to H_{i-1} (resp. H_i). If fed to $\mathcal{D}^{\mathsf{L}}$, the result will enable $\mathcal{D}^{\mathsf{L}\prime}$ to distinguish H_{i-1} from H_i with the same probability $\mathcal{D}^{\mathsf{L}\prime}$ distinguishes d_1 from d_2. Furthermore, by inspection, we can verify that $\mathcal{D}^{\mathsf{L}\prime}$ is (s, t)-bounded. Applying Lemma 2, this probability is then bounded by $\epsilon_{prf} + \epsilon_{2\text{-}sim}$, as desired. □

Description of $\mathsf{ENC}\ell$:	Description of $\mathsf{ENC}\ell^{\mathsf{I}}$:
• Gen picks $k_0 \leftarrow \{0,1\}^n$. • $\mathsf{Enc}\ell_{k_0}(m_1,\ldots,m_\ell)$ returns $c_1,\ldots,c_\ell$, where $c_i = y_i \oplus m_i$, $y_i = \mathsf{F}_{k_{i-1}}(p_B)$ and $k_i = \mathsf{F}_{k_{i-1}}(p_A)$. • Dec proceeds in the natural way. The leakage $\mathsf{L}_{\mathsf{Enc}\ell}(k_0,m)$ resulting from computing $\mathsf{Enc}\ell_{k_0}(m)$ is defined by the sequence of $(\mathsf{L}_{\mathsf{F}}(p_A,k_{i-1}),\mathsf{L}_{\mathsf{F}}(p_B,k_{i-1}),\mathsf{L}_{\oplus}(m_i,y_i))$ for $i \in \{1,\ldots,\ell\}$.	• $\mathsf{Enc}\ell^{\mathsf{I}}{}_{k_0}(m_1,\ldots,m_\ell)$ returns $(c_1,\ldots,c_\ell)$, where $c_i = y_i \oplus m_i$, $y_1,\ldots,y_\ell \leftarrow \{0,1\}^n$ and $k_1,\ldots,k_\ell \leftarrow \{0,1\}^n$. The leakage $\mathsf{L}_{\mathsf{Enc}\ell^{\mathsf{I}}}(k,y,m)$ resulting from computing $\mathsf{Enc}\ell^{\mathsf{I}}{}_{k_0}(m)$ with the random vectors k and y is defined by the sequence of $(\mathcal{S}^{\mathsf{L}}(k_{i-1},p_A,k_i),\mathcal{S}^{\mathsf{L}}(k_{i-1},p_B,y_i),\ \mathsf{L}_{\oplus}(m_i,y_i))$ for $i \in \{1,\ldots,\ell\}$.

Table 3: The $\mathsf{ENC}\ell$ and $\mathsf{ENC}\ell^{\mathsf{I}}$ schemes.

Further remarks on the definition of $\mathsf{Enc}s$. The proof above heavily relies on the use of the extra leakages provided when running $\mathsf{Enc}s$, for the linking of the hybrids. This is however not just an artifact that we use to simplify our proof. Consider for instance a situation in which $\mathsf{Enc}s$ would not leak k_1 and a simple leakage function $\mathsf{L}_{\mathsf{F}}(p,k)$ would leak just the first bit of $k \oplus \mathsf{F}_k(p)$. In such a setting, if k_1 is not leaked, the leakage does not provide any useful information on the encrypted message (we just loose one bit of security for the key). So, if we encrypt the messages m_1 and m_2 with $\mathsf{Enc}s$ using two independent keys, the leakages do not provide us with any useful information. However, if we encrypt the message (m_1,m_2) using $\mathsf{Enc}\ell$, we will obtain c_1, c_2 and leakages containing the first bit of $k_0 \oplus y_1$, $k_0 \oplus k_1$ and $k_1 \oplus y_2$, from which we can derive the first bit of $y_1 \oplus y_2$, and eventually the first bit of $m_1 \oplus m_2$, which was not available before. This observation is a constructive evidence that encrypting two message blocks with $\mathsf{Enc}\ell$ can be far more damaging on the privacy than encrypting these blocks independently with a version of $\mathsf{Enc}s$ from which k_1 would not be leaked. The leakage of k_1 in $\mathsf{Enc}s$ prevents the shortcoming we just described, as k_1 would provoke the leakage of the first bit of each message block.

3.5 From 1-block to ℓ-block security

The above section demonstrated how one-time versions of our encryption scheme can be idealized with controlled security loss, in the case of single and multiple block encryption. We now use these idealized encryption processes to evaluate the (eavesdropper) security of an ℓ-block encryption with $\mathsf{Enc}\ell^{\mathsf{I}}$ by comparison with the security of ℓ encryptions with $\mathsf{Enc}s^{\mathsf{I}}$ performed with independent keys, block by block.

LEMMA 4. *For every pair of ℓ-block messages m^0 and m^1 and (s,t)-bounded adversary $\mathcal{A}^{\mathsf{L}}$, there is an $(s-s_r,t-t_r)$-bounded adversary $\mathcal{A}^{\mathsf{L}\prime}$ such that the following holds:*

$$\Big|\Pr\Big[\mathcal{A}^{\mathsf{L}}\Big(\mathsf{LEnc}\ell^{\mathsf{I}}{}_{k_0}(m^0)\Big)=1\Big] -\Pr\Big[\mathcal{A}^{\mathsf{L}}\Big(\mathsf{LEnc}\ell^{\mathsf{I}}{}_{k_0}(m^1)\Big)=1\Big]\Big| \leq \sum_{i=1}^{\ell}\Big|\Pr\Big[\mathcal{A}^{\mathsf{L}\prime}\Big(\mathsf{LEnc}s^{\mathsf{I}}{}_{k_i}(m_i^0)\Big)=1\Big] - \Pr\Big[\mathcal{A}^{\mathsf{L}\prime}\Big(\mathsf{LEnc}s^{\mathsf{I}}{}_{k_i}(m_i^1)\Big)=1\Big]\Big|,$$

with all k's chosen uniformly at random, $s_r = \ell(2s_{\mathcal{S}}+1)$ and t_r equal to $2\ell t_{\mathcal{S}}$ to which we add the time needed to sample 2ℓ random values and compute ℓ times the $\oplus$ operations.

PROOF. We proceed in two steps. We start by building a sequence of $\ell+1$ messages $m_{h,0},\ldots,m_{h,\ell}$ starting from m^0 and modifying its blocks one by one until obtaining m^1. That is, $m_{h,i} := m^1_{[1,i]}, m^0_{[i+1,\ell]}$. From the triangle inequality, it holds that:

$$\Big|\Pr\Big[\mathcal{A}^{\mathsf{L}}\Big(\mathsf{LEnc}\ell^{\mathsf{I}}{}_{k_0}(m^0)\Big)=1\Big] - \Pr\Big[\mathcal{A}^{\mathsf{L}}\Big(\mathsf{LEnc}\ell^{\mathsf{I}}{}_{k_0}(m^1)\Big)=1\Big]\Big| \leq \sum_{i=1}^{\ell}\Big|\Pr\Big[\mathcal{A}^{\mathsf{L}}\Big(\mathsf{LEnc}\ell^{\mathsf{I}}{}_{k_i}(m_{h,i-1})\Big)=1\Big] - \Pr\Big[\mathcal{A}^{\mathsf{L}}\Big(\mathsf{LEnc}\ell^{\mathsf{I}}{}_{k_i}(m_{h,i})\Big)=1\Big]\Big|.$$

The ℓ differences in the sum above can now be related to the probability of distinguishing the encryptions of single block messages: from an $\mathsf{Enc}s^{\mathsf{I}}$ encryption of $m_{0,i}$ or $m_{1,i}$ with the associated leakage $\mathsf{L}_{\mathsf{Enc}s^{\mathsf{I}}}$, it is immediate to sample an $\mathsf{Enc}\ell^{\mathsf{I}}$ encryption of $m_{h,i-1}$ or $m_{h,i}$ with the associated leakage $\mathsf{L}_{\mathsf{Enc}\ell^{\mathsf{I}}}$. The cost of this sampling is bounded by s_r leakage queries and running time t_r. □

Injecting Lemmas 2 and 3, which relate real and ideal encryptions, into Lemma 4, we obtain Theorem 3 which is our main result for eavesdropper security.

THEOREM 3. *Let F be a (s,t,ϵ_{prf})-PRF, with a leakage simulator $\mathcal{S}^{\mathsf{L}}$ as in Lemma 2, and let (s_r,t_r) be the bounds defined in Lemma 3. For every pair of ℓ-block messages m^0 and m^1 and $(s-s_r,t-t_r)$-bounded adversary $\mathcal{A}^{\mathsf{L}}$, there is an $(s-2s_r,t-2t_r)$-bounded adversary $\mathcal{A}^{\mathsf{L}\prime}$ such that the following holds:*

$$\Big|\Pr\Big[\mathcal{A}^{\mathsf{L}}\left(\mathsf{LEnc}\ell_{k_0}(m^0)\right)=1\Big] - \Pr\Big[\mathcal{A}^{\mathsf{L}}\left(\mathsf{LEnc}\ell_{k_0}(m^1)\right)=1\Big]\Big| \leq \sum_{i=1}^{\ell}\Big|\Pr\Big[\mathcal{A}^{\mathsf{L}\prime}\left(\mathsf{LEnc}s_{k_i}(m_i^0)\right)=1\Big] - \Pr\Big[\mathcal{A}^{\mathsf{L}\prime}\left(\mathsf{LEnc}s_{k_i}(m_i^1)\right)=1\Big]\Big| + 4\ell(\epsilon_{prf} + \epsilon_{2\text{-}sim}).$$

It indicates that, if we want to bound the probability that an attacker distinguishes the encryptions of two ℓ-block messages, we can focus on bounding the probabilities that an attacker distinguishes independent encryptions of the ℓ pairs of blocks, which is arguably easier, and derive the desired bound from it. Furthermore, the security degradation is moderate, being proportional to the number of blocks.

3.6 From eavesdropper to CPA security

The $\mathsf{ENC}\ell$ scheme is obviously insecure under chosen plaintext attack. However, the $\mathsf{PrivK}^{\mathsf{lmcpa,b}}_{\mathcal{A}^{\mathsf{L}},\mathsf{ENC}}$ security of the ENC scheme can now be derived from Theorem 3.

THEOREM 4. *Let $\mathcal{A}^{\mathsf{L}}$ be an $(s - s_r, t - t_r)$-bounded $\mathsf{PrivK}^{\mathsf{lmcpa,b}}_{\mathcal{A}^{\mathsf{L}},\mathsf{ENC}}$ adversary against the ENC scheme based on a (s, t, ϵ_{prf})-secure PRF. Then:*

$$\left|\Pr[\mathsf{PrivK}^{\mathsf{lmcpa,0}}_{\mathcal{A}^{\mathsf{L}},\mathsf{ENC}} = 1] - \Pr[\mathsf{PrivK}^{\mathsf{lmcpa,1}}_{\mathcal{A}^{\mathsf{L}},\mathsf{ENC}} = 1]\right| \leq$$
$$\left|\Pr[\mathcal{A}^{\mathsf{L}\prime}(\mathsf{LEnc}\ell_k(m^0)) = 1] - \Pr[\mathcal{A}^{\mathsf{L}\prime}(\mathsf{LEnc}\ell_k(m^1)) = 1]\right| +$$
$$2\epsilon_{prf} + q/2^n,$$

where $\mathcal{A}^{\mathsf{L}\prime}$ is $(s - 2s_r, t - 2t_r)$-bounded, m^0 and m^1 are the messages chosen by $\mathcal{A}^{\mathsf{L}}$ for the test query, $s_r := 3q$ where q is the number of encryption queries made by $\mathcal{A}^{\mathsf{L}}$, and t_r is the time needed to evaluate $\mathsf{LEnc}\ell$ on q messages of at most ℓ blocks and sample 2ℓ random values.

PROOF. We proceed in two steps. First, we modify the $\mathsf{PrivK}^{\mathsf{lmcpa,b}}_{\mathcal{A}^{\mathsf{L}},\mathsf{ENC}}$ game by replacing the leak-free PRF with a random function. Since the cost of the reduction of this change to the PRF security is less than the (s_r, t_r) bounds, the probability that $\mathcal{A}^{\mathsf{L}}$ detects the change is bounded by ϵ_{prf}. Next, we rely on the perfectly random distribution of the ephemeral keys (k') used by Enc to emulate the leak-free PRF and (consistently) answer all encryption queries with random IV's and random ephemeral key k_0's. For the test query, we generate a random IV, but use the $\mathsf{LEnc}\ell$ oracle to produce the ciphertext. This strategy will only fail when the random IV selected here is equal to one of the IV's generated during one of the at most q previous encryption queries, bounding the probability of this event by $q/2^n$. Again, the cost of answering these queries is bounded by (s_r, t_r). □

4. LEAK-FREE COMPONENT INSTANTIATION AND IMPERFECTIONS

Our constructions make use of a component modeled as leak-free used as part of the initialization of every MAC and encryption computation. Of course, it is unlikely that such a thing exists in reality. A simple work-around to this situation would be to require this component to be polynomially simulatable instead, i.e., require that it would satisfy the simulatability definition (Def. 3) for a number of observations q that would be high enough to not be reached within the life-time of the circuit. This would add a degradation factor corresponding to the one of the q-sim bound into all our reductions (which we did not do for clarity).

This requirement is of course much more demanding than the bound $q = 2$ that appears in the rest of our proofs, and would then require a highly protected implementation of this specific primitive, which might be considerably slower and demanding in energy. Concretely, we expect it to be a dozen to a few hundred times less efficient than the weakly protected one used in the leakage-resilient parts of our constructions (i.e. all the light grey blocks in Figures 2, 3 and 4). This is the typical gap between a standard implementation of the AES and a heavily protected implementation using masking and other countermeasures. For illustration, Table 4 reports some performance figures for unprotected and masked AES in software and hardware that are in these ranges.[4] We believe that this gap amply motivates our minimal use of the strongly protected component: it makes it possible to amortize its cost as soon as messages contain a few kilobits (the longer the better, of course), resulting in an efficiency gain of a factor comparable to the gap between the two implementations when comparing with a situation where a strongly-protected implementation would be used everywhere. Besides, and as previously mentioned, the random inputs of the highly protected component are independent of the messages that are manipulated, which makes it possible to run these (pre)computations in advance, therefore avoiding any delay when the messages are available.

As for the practical security of the proposed schemes, we anticipate that it will also follow the analyses in [3]. That is, the leakage-resilient parts of our constructions are expected to lower-bound the time complexity of the best side-channel attacks independently of the number of adversarial measurements (e.g. up to >80 bits depending of the implementations). As for the strongly protected (intialization) component, and as previously mentioned, the side-channel attacks' complexity cannot be bounded independently of the number of adversarial measurements. So the security level highly depends on the choice of q in this case. For example, Table 5 (for software implementations) and 18 (for hardware implementations) in [3] estimate the cost to maintain security levels of > 80 bits with up to q measurements. In general, masking increases security exponentially in the number of masks for a quadratic performance overheads. So we can theoretically have q arbitrarily large in the leak-free component, but for much larger overheads than in the leakage-resilient parts of our constructions – which again motivates our approach. Of course and as usual, further evaluations (on various devices) by third-parties remain welcome to improve the understanding of the concrete implementation security of our encryption and authentication schemes.

5. ACKNOWLEDGEMENTS

Olivier Pereira's work has been partly supported by the GreenTIC TrueDev Walloon region project 1317971. François-Xavier Standaert is a research associate of the Belgian Fund for Scientific Research (F.R.S.-FNRS). His work has been funded in part by the European Commission through the ERC project 280141 (acronym CRASH). Srinivas Vivek has been supported in part by the European Union's H2020 programme under grant agreement number ICT-644209.

6. REFERENCES

[1] *CHES 2013*, volume 8086 of *Lecture Notes in Computer Science*. Springer, 2013.

[2] M. Abdalla, S. Belaïd, and P. Fouque. Leakage-resilient symmetric encryption via re-keying. In *CHES 2013* [1], pages 471–488.

[3] S. Belaïd, V. Grosso, and F. Standaert. Masking and leakage-resilient primitives: One, the other(s) or both? *Cryptography and Communications*, 7(1):163–184, 2015.

[4] The cost functions are code size × cycle count for software implementations, and area / throughput for hardware ones. The physical assumptions are additional properties that justify the higher implementation cost of certain solutions (e.g. regarding the independent leakage assumption).

Software (8-bit) Implementations	*code size* (*bytes*)	*cycle count*	*cost function*	*physical assumptions*
Unprotected [8]	1659	4557	7.560	-
1-mask Boolean [34]	3153	$129 \cdot 10^3$	406.7	glitch-sensitive
1-mask polynomial [13, 32]	20 682	$1064 \cdot 10^3$	22 000	glitch-resistant
2-mask Boolean [34]	3845	$271 \cdot 10^3$	1042	glitch-sensitive
FPGA (Virtex-5) Implementations	*area* (*slices*)	*throughput* (*enc/sec*)	*cost function*	*physical assumptions*
Unprotected (128-bit) [33]	478	$\frac{245 \cdot 10^6}{11}$	21.46	-
1-mask Boolean (128-bit) [33]	1462	$\frac{100 \cdot 10^6}{11}$	160.8	glitch-sensitive
Threshold (8-bit) [26]	958	$\frac{170 \cdot 10^6}{266}$	1499	glitch-resistant

Table 4: Performance of some illustrative AES implementations (borrowed from [3]).

[4] M. Bellare, J. Kilian, and P. Rogaway. The security of the cipher block chaining message authentication code. *J. Comput. Syst. Sci.*, 61(3):362–399, 2000.

[5] S. Chari, C. S. Jutla, J. R. Rao, and P. Rohatgi. Towards sound approaches to counteract power-analysis attacks. In *CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999.

[6] Y. Dodis and K. Pietrzak. Leakage-resilient pseudorandom functions and side-channel attacks on feistel networks. In *CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 21–40. Springer, 2010.

[7] S. Dziembowski and K. Pietrzak. Leakage-resilient cryptography. In *FOCS 2008*, pages 293–302. IEEE Computer Society, 2008.

[8] T. Eisenbarth, Z. Gong, T. Güneysu, S. Heyse, S. Indesteege, S. Kerckhof, F. Koeune, T. Nad, T. Plos, F. Regazzoni, F. Standaert, and L. van Oldeneel tot Oldenzeel. Compact implementation and performance evaluation of block ciphers in attiny devices. In A. Mitrokotsa and S. Vaudenay, editors, *AFRICACRYPT 2012*, volume 7374 of *Lecture Notes in Computer Science*, pages 172–187. Springer, 2012.

[9] S. Faust, K. Pietrzak, and J. Schipper. Practical leakage-resilient symmetric cryptography. In *CHES 2012*, volume 7428 of *Lecture Notes in Computer Science*, pages 213–232. Springer, 2012.

[10] B. Fuller and A. Hamlin. Unifying leakage classes: Simulatable leakage and pseudoentropy. In *ICITS 2015*, volume 9063 of *Lecture Notes in Computer Science*, pages 69–86. Springer, 2015.

[11] J. L. Galea, D. P. Martin, E. Oswald, D. Page, M. Stam, and M. Tunstall. Simulatable leakage: Analysis, pitfalls, and new constructions. In *ASIACRYPT 2014*, volume 8873 of *Lecture Notes in Computer Science*, pages 223–242. Springer, 2014.

[12] D. Galindo and S. Vivek. A leakage-resilient pairing-based variant of the Schnorr signature scheme. In *IMA International Conference, IMACC 2013*, volume 8308 of *Lecture Notes in Computer Science*, pages 173–192. Springer, 2013.

[13] V. Grosso, F. Standaert, and S. Faust. Masking vs. multiparty computation: How large is the gap for aes? In *CHES 2013* [1], pages 400–416.

[14] V. Grosso, F. Standaert, and S. Faust. Masking vs. multiparty computation: how large is the gap for AES? *J. Cryptographic Engineering*, 4(1):47–57, 2014.

[15] C. Hazay, A. López-Alt, H. Wee, and D. Wichs. Leakage-resilient cryptography from minimal assumptions. In *EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 160–176. Springer, 2013.

[16] Y. Ishai, A. Sahai, and D. Wagner. Private circuits: Securing hardware against probing attacks. In *CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.

[17] M. Joye and M. Tunstall, editors. *Fault Analysis in Cryptography.* Information Security and Cryptography. Springer, 2012.

[18] E. Kiltz and K. Pietrzak. Leakage resilient ElGamal encryption. In *ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 595–612. Springer, 2010.

[19] L. R. Knudsen and M. Robshaw. *The Block Cipher Companion.* Information Security and Cryptography. Springer, 2011.

[20] M. Liskov, R. L. Rivest, and D. Wagner. Tweakable block ciphers. In *CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 31–46. Springer, 2002.

[21] S. Mangard, E. Oswald, and T. Popp. *Power analysis attacks - revealing the secrets of smart cards.* Springer, 2007.

[22] S. Mangard, T. Popp, and B. M. Gammel. Side-channel leakage of masked CMOS gates. In *CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 351–365. Springer, 2005.

[23] D. P. Martin, E. Oswald, and M. Stam. A leakage resilient MAC. *IACR Cryptology ePrint Archive*, 2013:292, 2013.

[24] M. Medwed, F. Standaert, J. Großschädl, and F. Regazzoni. Fresh re-keying: Security against side-channel and fault attacks for low-cost devices. In *AFRICACRYPT 2010*, volume 6055 of *Lecture Notes in Computer Science*, pages 279–296. Springer, 2010.

[25] S. Micali and L. Reyzin. Physically observable cryptography (extended abstract). In *TCC 2004*, volume 2951 of *Lecture Notes in Computer Science*, pages 278–296. Springer, 2004.

[26] A. Moradi, A. Poschmann, S. Ling, C. Paar, and H. Wang. Pushing the limits: A very compact and a threshold implementation of AES. In K. G. Paterson, editor, *EUROCRYPT*, volume 6632 of *Lecture Notes in Computer Science*, pages 69–88. Springer, 2011.

[27] M. Naor and G. Segev. Public-key cryptosystems resilient to key leakage. In *CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 18–35. Springer, 2009.

[28] S. Nikova, V. Rijmen, and M. Schläffer. Secure hardware implementation of nonlinear functions in the presence of glitches. *J. Cryptology*, 24(2):292–321, 2011.

[29] P. Pessl, F. Standaert, S. Mangard, and F. Durvaux. Towards leakage simulators that withstand the correlation distinguisher. *ASIACRYPT 2014 rump session talk*, 2014.

[30] C. Petit, F. Standaert, O. Pereira, T. Malkin, and M. Yung. A block cipher based pseudo random number generator secure against side-channel key recovery. In *ASIACCS 2008*, pages 56–65. ACM, 2008.

[31] K. Pietrzak. A leakage-resilient mode of operation. In *EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 462–482. Springer, 2009.

[32] E. Prouff and T. Roche. Higher-order glitches free implementation of the AES using secure multi-party computation protocols. In B. Preneel and T. Takagi, editors, *CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*, pages 63–78. Springer, 2011.

[33] F. Regazzoni, W. Yi, and F.-X. Standaert. FPGA implementations of the AES masked against power analysis attacks. In *COSADE* 2011, pp 56-66, Darmstadt, Germany, February 2011.

[34] M. Rivain and E. Prouff. Provably secure higher-order masking of AES. In S. Mangard and F. Standaert, editors, *CHES 2010*, volume 6225 of *Lecture Notes in Computer Science*, pages 413–427. Springer, 2010.

[35] J. Schipper. Leakage-resilient authentication. *Msc thesis, Centrum Wiskunde and Informatica, The Netherlands*, 2010.

[36] F. Standaert, O. Pereira, and Y. Yu. Leakage-resilient symmetric cryptography under empirically verifiable assumptions. In *CRYPTO 2013*, volume 8042 of *Lecture Notes in Computer Science*, pages 335–352. Springer, 2013.

[37] F. Standaert, O. Pereira, Y. Yu, J. Quisquater, M. Yung, and E. Oswald. Leakage resilient cryptography in practice. In *Towards Hardware-Intrinsic Security - Foundations and Practice*, Information Security and Cryptography, pages 99–134. Springer, 2010.

[38] Y. Yu and F. Standaert. Practical leakage-resilient pseudorandom objects with minimum public randomness. In *CT-RSA 2013*, volume 7779 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 2013.

[39] Y. Yu, F. Standaert, O. Pereira, and M. Yung. Practical leakage-resilient pseudorandom generators. In *CCS 2010*, pages 141–151. ACM, 2010.

APPENDIX

A. PROOF OF THEOREM 1

Consider the following hybrid games:

Hybrid H^{+}. This is the original security game executed as defined in the experiment $\mathsf{Forge}^{\mathsf{euf-cma}}_{\mathcal{A}^{\mathsf{L}},\mathsf{MAC}}(n)$ (Definition 1, Remark 1). In particular, the q session keys $k_1^{(i)}$ $(i = 1, 2, \ldots, q)$ are the output of F on random IVs. Let the queried messages (each containing ℓ blocks) be denoted as $m_i = \langle m_1^{(i)}, \ldots, m_\ell^{(i)} \rangle$ $(i = 1, 2, \ldots, q)$. The advantage of a q-query adversary $\mathcal{A}^{\mathsf{L}}$ in $\mathsf{Forge}^{\mathsf{euf-cma}}_{\mathcal{A}^{\mathsf{L}},\mathsf{MAC}}(n)$ is ϵ'.

Hybrid H^{++}. This is the same as hybrid H^{+} except that the q IVs randomly chosen are distinct. Let ϵ^{++} denote the advantage of $\mathcal{A}^{\mathsf{L}}$ in this hybrid. Using the birthday bound, we have that:

$$\left|\epsilon' - \epsilon^{++}\right| \leq \frac{q^2}{2^{n+1}}. \tag{1}$$

Hybrid H^{*}. This is the same as hybrid H^{++} except that the q session keys $k_1^{(i)} \xleftarrow{\$} \{0,1\}^n$ are chosen uniform randomly and independently. Let ϵ^{*} denote the advantage of $\mathcal{A}^{\mathsf{L}}$ in this hybrid. It is easy to see that:

$$\left|\epsilon^{++} - \epsilon^{*}\right| \leq \epsilon_{prf}. \tag{2}$$

This is because using the adversary $\mathcal{A}^{\mathsf{L}}$ of MAC_1, we can construct a (s, t', ϵ_{prf}) against F.

Hybrids $H^{*}_{i,j}$. Next, we successively transform the hybrid H^{*} into hybrids $H^{*}_{i,j}$ $(1 \leq i \leq q{+}1, 0 \leq j \leq \ell)$ by transforming the normal execution of the block cipher F into an ideal execution one message block at a time. More precisely, the hybrids H^{*} and $H^{*}_{1,0}$ are identical. In hybrid $H^{*}_{1,1}$, the output of $\mathsf{F}(k_1^{(1)}, m_1^{(1)})$ while processing the first message block during the first tag-generation query is a uniform random element in $\{0,1\}^n$ and leakages are simulated (cf. Lemma 1). The processing of the remaining message blocks in the first as well as the later queries is carried out "normally." By "normal" we mean that the actual values of F are used for all except the first block of the first message and all the session keys, unless for consistency we are forced to use the random sampled value in case the inputs $(k_1^{(1)}, m_1^{(1)})$ to F appear again elsewhere later.

More generally, in the hybrid $H^{*}_{i,j}$, all the evaluations of F are treated as ideal until (and including) the j^{th} message block of the i^{th} tag query. All the remaining evaluations of F are normal upto the consistency requirement mentioned above. The hybrids $H^{*}_{i,\ell}$ and $H^{*}_{i+1,0}$ are identical for $1 \leq i \leq q$, and the verification of the plausible forgery output by $\mathcal{A}^{\mathsf{L}}$ is considered as $q + 1$-st tag query except that the leakages and the tag are not output. This means that the hybrids $H^{*}_{q+1,j}$ correspond to idealizing the execution of F during the verification stage. Note that the goal of the adversary $\mathcal{A}^{\mathsf{L}}$ is to break the strong-unforgeability of MAC_1. This means that either it outputs a forgery on a distinct IV for a possibly previously queried message, or it outputs a forgery on a previously queried IV but on a distinct message. We assume without loss of generality that the forgery output by $\mathcal{A}^{\mathsf{L}}$ satisfies either of the above two conditions and it makes exactly q tag request queries. Hence the hybrids $H^{*}_{q+1,j}$ will be present. Note that this last sequence of hybrids may be avoided if we try to follow the standard approach of first showing that the construction is a PRF and hence it is a MAC. But it turns out this way we need more hybrid games than the current approach.

Next, we show that in the successive hybrids $H^*_{i,j}$ and $H^*_{i,j+1}$ ($1 \le i \le q+1, 0 \le j \le \ell - 1$), the views of $\mathcal{A}^{\mathsf{L}}$ are computationally identical upto the 2-simulatable leakage assumption. Let $\epsilon_{i,j}$ and $\epsilon_{i,j+1}$ denote the advantages of $\mathcal{A}^{\mathsf{L}}$ in the hybrids $H^*_{i,j}$ and $H^*_{i,j+1}$, respectively.

Lemma 5. $|\epsilon_{i,j} - \epsilon_{i,j+1}| \le \epsilon_{prf} + \epsilon_{2\text{-}sim} + \frac{(q+1)^2(\ell+1)^2}{2^n}$.

Proof. Using $\mathcal{A}^{\mathsf{L}}$ as a subroutine, we construct a (s', t')-bounded distinguisher $\mathcal{D}^{\mathsf{L}}$ for the distributions of Lemma 1 that has advantage as indicated on the R.H.S. of Lemma 5. $\mathcal{D}^{\mathsf{L}}$ simulates hybrid $H^*_{i,j}$ or hybrid $H^*_{i,j+1}$ depending on whether its input distribution is actual or ideal, respectively.

$\mathcal{D}^{\mathsf{L}}$ chooses a uniform random shared secret key k, random session keys $k_{i,1}$ $(1 \le i \le q)$, and possibly a random session key $k_{q+1,1}$ for verification if the target forgery is on a different IV. Whenever $\mathcal{D}^{\mathsf{L}}$ samples a random output value, say γ, on "key-message" input (α, β) it records the input/output pair $((\alpha, \beta), \gamma)$ in a table $\mathcal{T}$, in addition to the simulated leakage $\mathcal{S}^{\mathsf{L}}(\tilde{k}^*, \beta, \gamma)$ ($\tilde{k}^* \xleftarrow{\$} \{0,1\}^n$). This table is used to consistently return the same (random) output on the same input pair. Also, all the block cipher evaluations while the processing the tag requests return random outputs and simulated leakages until (including) the j^{th} message block of the i^{th} tag request. Recall the notation that the j'^{th} block of the i'^{th} message is denoted by $m_{i',j'}$, and the corresponding key and the output for the evaluation of F is denoted by $k_{i',j'}$ and $k_{i',j'+1}$, respectively. A '*' in the superscript of a parameter, for example, as in $k^*_{i',j'}$, explicitly denotes that the parameter was chosen uniform randomly and independently (and was not computed normally).

At this point, $\mathcal{D}^{\mathsf{L}}$ first receives its input d_5 upon querying its challenger with $\mathsf{Gen}(\tilde{k}^*_{i,j}, m_{i,j})$ with $\tilde{k}^*_{i,j} \xleftarrow{\$} \{0,1\}^n$ (cf. Remark 3). It then uses d_5 instead of the simulated leakage $\mathcal{S}^{\mathsf{L}}(\tilde{k}^*_{i,j}, m_{i,j}, k^*_{i,j})$. Note that the output of this round is implicitly set to k' (if $j = 0$, then the session key $k^*_{i,1}$ is implicitly set to k'). It then builds the view for the $(i, j+1)^{\text{th}}$ execution of F by querying its challenger for (d_1, d_3) by querying $\mathsf{Enc}(m_{i,j+1})$ $(p_0 := m_{i,j+1})$. $\mathcal{D}^{\mathsf{L}}$ then provides $\mathcal{A}^{\mathsf{L}}$ with (d_1, d_3). The remaining steps are executed normally by evaluating F with the (known) inputs. In case any inconsistency arises when F is evaluated with the inputs present in the table $\mathcal{T}$, then the corresponding output value recorded in the table is used. Such a situation does arise when the adversary outputs a possible forgery on a message m' w.r.t. the IV IV_i, such that $m' \neq m_i$ share a common prefix. Also, note that to evaluate F w.r.t. the (implicit) key k', $\mathcal{D}^{\mathsf{L}}$ has to use its input distribution to determine the output. This means that if there are more than two queries to F w.r.t. the (implicit) key k', then $\mathcal{D}^{\mathsf{L}}$ will abort the simulation. Denote this abort event by Abort.

In order for this simulation by $\mathcal{D}^{\mathsf{L}}$ to $\mathcal{A}^{\mathsf{L}}$ to be consistent with the working of MAC_1, two events must *not* occur.

- All the random values sampled (and listed in table $\mathcal{T}$) must be distinct.
- The abort event Abort mentioned above must not occur.

Let us denote both the events by $\mathsf{Collision}$. The reason for the first of the above conditions is that if the intermediate values repeat, then a possible pattern could arise in the successive intermediate values and there by implicitly setting the output of the $(i, j)^{\text{th}}$ step to k' may lead to inconsistency. Next, to ensure that the Abort event does not arise, and hence 2-simulatable assumption suffices, we need to make sure that k' is never used as key more than once later. This means that F is not queried with the input $(k^*_{i,j}, m_{i,j})$ more than once later. Note that if the first condition above does not occur, then the outputs of the later steps are function of parameters independent of the value $k^*_{i,j}$, except possibly once during the verification stage. It is easy to see that:

$$\Pr[\mathsf{Collision}] \le \frac{(q+1)^2(\ell+1)^2}{2^n}. \qquad (3)$$

Hence the lemma follows from Lemma 1 and (3). □

Note that there are at most $(q+1)\ell$ *distinct* hybrids $H^*_{i,j}$ $(1 \le i \le q+1, 0 \le j \le \ell)$ since the hybrids $H^*_{i,\ell}$ and $H^*_{i+1,0}$ are identical for $1 \le i \le q$. Also note that the hybrids H^* and $H^*_{1,0}$ are identical as well. Hence we obtain:

$$|\epsilon^* - \epsilon_{q+1,\ell}| \le (q+1)\ell(\epsilon_{prf} + \epsilon_{2-sim}) + \frac{(q+1)^3\ell(\ell+1)^2}{2^n}. \qquad (4)$$

Lemma 6. $\epsilon_{q+1,\ell} \le \frac{1}{2^n - ((q+1)(\ell+1))^2} + \frac{(q+1)^2(\ell+1)^2}{2^{n+1}}$

Proof. In this hybrid, all the evaluations of F are ideal, that means that distinct "key-message" input pairs produce random output values, and all the leakages are simulated. To break the strong-unforgeability property of MAC_1, $\mathcal{A}^{\mathsf{L}}$ must either output a forgery on a distinct IV for a possibly previously queried message, or it outputs a forgery on a previously queried IV but on a distinct message. This implies that during the verification step, conditioned on the event of no collision of randomly chosen output values, there will be distinct "key-message" pairs with which F will be queried and, as a consequence, a random output will be produced. Further, no collision would imply that the same would happen with the later message blocks during the verification step, including the final block. Note that the probability of collision is at most $\frac{(q+1)^2(\ell+1)^2}{2^{n+1}}$. Again, conditioned on the event of no collision, the probability that the tag τ output by $\mathcal{A}^{\mathsf{L}}$ is a valid forgery for the message m is at most $\frac{1}{2^n - ((q+1)(\ell+1))^2}$. Hence the lemma follows. □

Theorem 1 follows from (1), (2), (4) and Lemma 6. □