Mode-Level Side-Channel Countermeasures

Olivier Pereira[*] Thomas Peters[*] François-Xavier Standaert[*]

March 6, 2023

1 Introduction

Let $\mathsf{BC}_k(.)$ be a n-bit block cipher parametrized by a n-bit secret key k. Differential Power Analysis (DPA) attacks are powerful side-channel attacks that continuously leak information about the secret key. For this purpose, the adversary monitors multiple executions of the block cipher $\mathsf{BC}_k(.)$ on different inputs x_i $(1 \leq i \leq q)$, each of these executions giving rise to a noisy leakage l_i. Assuming for simplicity that each execution of the cipher can be modeled as an independent communication channel and that its leakage provides the adversary with λ bits of information, a DPA can reveal the full key after the observation of approximately $\lceil \frac{n}{\lambda} \rceil$ block cipher executions.

The direct approach to prevent DPA is to implement the block cipher with hardware-level countermeasures (e.g., noise addition) or algorithmic countermeasures (e.g., masking). Yet, such a direct approach suffers from two drawbacks. On the one hand, these countermeasures can be expensive as the target security level increases. On the other hand, their secure implementation can be challenging and require strong expertise to deal with the various physical defaults that can break their underlying assumptions. As a result, an important line of research has investigated options to improve cryptographic designs in order to make their secure implementation cheaper and simpler. In this chapter, we survey the benefits of this approach. We denote it as mode-level countermeasures since (in the symmetric setting), it works by adapting cryptographic modes of operation to limit leakage rather than by adapting the implementation of the primitives used in these modes.

For this purpose, we start with an intuitive introduction of some basic building blocks that offer improved security against leakage. More precisely, we show in Section 2 that it is possible to design leakage-resilient Pseudo-Random Generators (PRGs) and Pseudo-Random Functions (PRFs) based on block ciphers that only require security against a weaker form of side-channel attacks, namely Simple Power Analysis (SPA). We then describe how these basic building blocks can be used to authenticate and encrypt with leakage. In Section 3, we adapt standard cryptographic definitions for this purpose. In Section 4, we discuss the models that are needed to analyze authentication and encryption in the presence of leakage, and their connection with concrete attacks like SPA and DPA. Eventually, we analyze how these security definitions can be matched by actual authentication, encryption and authenticated encryption schemes in Section 5. These results put forward that mode-level countermeasures provide an interesting path towards leveled implementations, where different parts of a cryptographic construction require different levels of security against leakage and therefore different (more or less expensive) hardware-level or algorithmic countermeasures.

[*] Crypto Group, ICTEAM Institute, UCLouvain, B-1348 Louvain-la-Neuve, Belgium.

2 Building blocks

A leakage-resilient PRG is depicted in Figure 1. From a random seed s_0, it works by iterating a length-doubling function that produces a fresh seed $s_{i+1} = \mathsf{BC}_{s_i}(\delta_0)$ and a pseudorandom output string $y_{i+1} = \mathsf{BC}_{s_i}(\delta_1)$, with δ_0 and δ_1 two distinct constant plaintexts. It is easy to see that in this case, a side-channel adversary can only exploit the execution of $q = 2$ plaintexts per key s_i.[1]

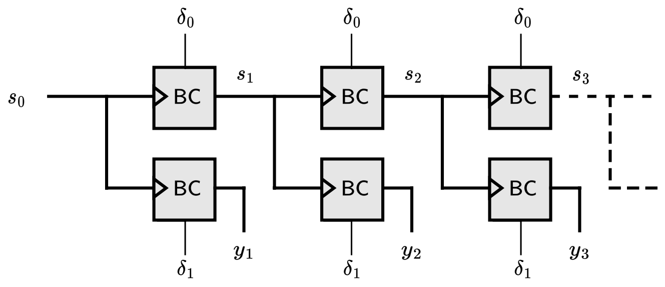

Figure 1: Leakage-resilient PRG.

From an operational viewpoint, leakage-resilient PRGs are limited by the need to agree on the initial seed s_0. For this purpose, a standard solution is to use a leakage-resilient PRF $\mathsf{F}_k(.)$, as depicted in Figure 2. From the master key k, it generates n intermediate keys k_i $(1 \leq i \leq n)$ such that $k_{i+1} = \mathsf{BC}_{k_i}(\delta_0)$ if $x(i) = 0$ and $k_{i+1} = \mathsf{BC}_{k_i}(\delta_1)$ if $x(i) = 1$, with $x(i)$ the ith bit of the input x, δ_0 and δ_1 two distinct constant plaintexts, an initialization $k_0 = k$ and the output set as $\mathsf{F}_k(x) = k_n$. This construction can be seen as a tree of depth n with 2^n leaves. Here as well, a side-channel adversary can only exploit the execution of $q = 2$ plaintexts per intermediate key. The output of this leakage-resilient PRF can then be used as a seed of the PRG of Figure 1.

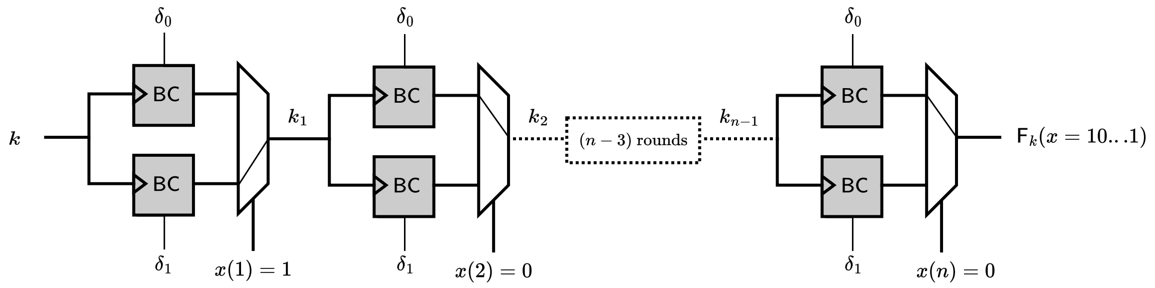

Figure 2: Leakage-resilient PRF.

The main physical security parameter of the previous leakage-resilient constructions is the bound q on the number of different block cipher executions per key that can be observed by the adversary (with different plaintexts). The leakage-resilient PRG of Figure 1 and the leakage-resilient PRF of Figure 2 are similar in this respect. In their basic form they bound the adversary to $q = 2$ block cipher executions per key. They both enable security vs. efficiency tradeoffs by increasing q. Since leakage security decreases exponentially in q, such a tradeoff can only be exploited to a limited extent (an unbounded q corresponds to DPA). For simplicity, we next assume $q = 2$.

However, these constructions also differ by one fundamental aspect. In the leakage-resilient PRG case (and assuming the initial key is fresh), the adversary cannot ask for multiple measurements

[1] Assuming no key collisions, which are expected with birthday probability only if a block cipher is used.

of the same (constant) plaintext δ_i. By contrast, in the leakage-resilient PRF case, she can repeat such measurements. The interest of these repeated queries is to get rid of the noise that possibly affects the leakage measurements. This difference leads us to the following definition.

Definition 1 (Bounded side-channel attacks)**.** *A side-channel attack against a block cipher implementation is (q, r)-bounded if the adversary is only able to observe the leakage of at most q different plaintexts per key and the leakage of each plaintext can be observed at most r times. By extension, a scheme is (q, r)-bounding if it limits the adversary to (q, r)-bounded attacks.*

Using this definition, we can see that a DPA is not (q, r)-bounded since both q and r are selected by the adversary and quantify the attack (data) complexity. The leakage-resilient PRG of Figure 1 limits the adversary to $(2, 1)$-bounded attacks, which are sometimes referred to as SPA. The leakage-resilient PRF of Figure 2 limits the adversary to $(2, .)$-bounded attacks (i.e., r is selected by the adversary), which we will refer to as an SPA with averaging (avgSPA for short). The SPA vs. avgSPA requirements of the block cipher calls in Figures 1 and 2 are reflected by their different shade of gray (and an even darker shade will later be used for DPA security).

The rationale behind mode-level countermeasures, which is widely supported by the literature, is that it is cheaper & simpler to implement countermeasures against SPA than against avgSPA, and it is cheaper & simpler to implement countermeasures against avgSPA than against DPA.

3 Security definitions

We now introduce the cryptographic primitive of authenticated encryption with the corresponding leaking algorithms (in Section 3.1), before defining integrity and confidentiality in the presence of leakage (in Sections 3.2 and 3.3, respectively). We consider a composite treatment of authenticated encryption with leakage, where integrity and confidentiality are defined and analyzed separately. While this is in contrast with the standard situation without leakage, where all-in-one definitions have become increasingly popular, this choice is motivated by the very different (physical) assumptions that can be used to ensure integrity and confidentiality with leakage. For example, integrity with leakage can be preserved with some intermediate computations leaked in full (which therefore removes the need of countermeasures), while such unbounded leakages cannot be used if confidentiality with leakage is required – we refer to Sections 4 and 5 for the details. Besides, we note that our focus is on the strongest possible security notions that allow leveling the implementations. In other words, it is of course always possible to define security with leakage just as security without leakage, by adding the appropriate leakage functions in the definitions. In the extreme case where a security definition can only be satisfied by requiring all the computations to be strongly protected against leakage thanks to hardware-level and algorithmic countermeasures, we fall back in the situation where mode-level protections are hardly effective, which we aim to avoid.

3.1 Authenticated encryption and leakage

The purpose of authenticated encryption is to provide both confidentiality and integrity in a single cryptographic scheme. First, we provide the general description of such a scheme.

Definition 2 (Nonce-Based Authenticated Encryption with Associated Data)**.** *A* nonce-based authenticated encryption scheme with associated data *is a tuple* $\mathsf{AE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *of algorithms such that, for any security parameter n, and keys in $\mathcal{K}$ generated from $\mathsf{Gen}(1^n)$:*

- $\mathsf{Enc} : \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{M} \to \mathcal{C}$ *deterministically maps a key from $\mathcal{K}$, a nonce from $\mathcal{N}$, some blocks of associated data from $\mathcal{AD}$, and a message from $\mathcal{M}$ to a ciphertext in $\mathcal{C}$.*

- $\mathsf{Dec} : \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{C} \to \mathcal{M} \cup \{\bot\}$ *deterministically maps a key from* $\mathcal{K}$*, a nonce from* $\mathcal{N}$*, some blocks of associated data from* $\mathcal{AD}$*, and a ciphertext from* $\mathcal{C}$ *to a message in* $\mathcal{M}$ *if integrity checking succeeds or to a special symbol* $\bot$ *if integrity checking fails.*

The sets $\mathcal{K}, \mathcal{N}, \mathcal{AD}, \mathcal{M}, \mathcal{C}$ *are completely specified by the security parameter* n*. Given a key* $k \leftarrow \mathsf{Gen}(1^n)$*,* $\mathsf{Enc}_k(N, A, M) := \mathsf{Enc}(k, N, A, M)$ *and* $\mathsf{Dec}_k(N, A, C) := \mathsf{Dec}(k, N, A, C)$ *are both deterministic functions of which the implementations may be probabilistic.*

Since we only focus on nonce-based authenticated encryption with associated data, we simply refer to it as *authenticated encryption*. Given a ciphertext $C \in \mathcal{C}$ of an encrypted message $M \in \mathcal{M}$, the decryption algorithm should always be able to recover M. This is captured by the *correctness* property: for any security parameter n, any key $k \in \mathcal{K}$, any nonce $N \in \mathcal{N}$, any associated data $A \in \mathcal{AD}$, and any message $M \in \mathcal{M}$, $\mathsf{Dec}_k(N, A, \mathsf{Enc}_k(N, A, M)) = M$. Whenever the algorithm Dec_k returns a message $M \neq \bot$ on input (N, A, C), we say that the ciphertext is *valid*.

Our definitions will additionally require the following algorithms:

Leaking Algorithm. A leaking version of an algorithm Algo is denoted LAlgo. It runs both Algo and a *leakage function* $\mathsf{L}_{\mathsf{Algo}}$ which captures the additional information given by an implementation of Algo during its execution. LAlgo simply returns the outputs of both Algo and $\mathsf{L}_{\mathsf{Algo}}$ which all take the same input. We thus have $\mathsf{LEnc} = (\mathsf{Enc}, \mathsf{L}_{\mathsf{Enc}})$ and $\mathsf{LDec} = (\mathsf{Dec}, \mathsf{L}_{\mathsf{Dec}})$.

Adversary. By a $(q_1, \ldots, q_\omega, t)$-bounded adversary $\mathcal{A}$ we mean a probabilistic algorithm that has access to ω oracles, $O_1, \ldots, O_\omega$, making at most q_i queries to its i-th oracle O_i, and performing computation bounded by running time t. We write $\mathcal{A}^{O_1,\ldots,O_\omega}$ to specify those oracles.

3.2 Integrity with leakage

Authenticity of authenticated encryption is defined as the infeasibility for any efficient adversary to produce a valid *fresh* ciphertext, known as a *forgery*. A ciphertext is fresh if it is not the output of any previous honest encryption that the adversary would have been able to eavesdrop.

More precisely, for a secret key $k \leftarrow \mathsf{Gen}(1^n)$, the goal of the adversary is to produce a forgery by providing (N, A, C) such that $\mathsf{Dec}_k(N, A, C) \neq \bot$. In order to capture the ability to observe ciphertexts and to strengthen the adversary, she is granted access to encryption and decryption oracles as detailed below. This security notion is called *ciphertext integrity* (CI).

Besides, in our definition the adversary is actually deemed successful as long as the triple (N, A, C) is fresh in the sense that C has not been returned by the encryption oracle given (N, A, M) for some message M. We next highlight that the adversary may repeat nonces in encryption and decryption oracles, as well as in the forgery, by using the term *nonce-misuse* (M).

We further extend this notion to capture the ability of the adversary to observe the *leakage* (L) of the encryption and decryption algorithms *during* their computation. We thus extend the oracles so as to return the result of LEnc_k and LDec_k instead of Enc_k and Dec_k in the black box case.

Definition 3 ($\mathsf{CIML2}$)**.** *An authenticated encryption* $\mathsf{AE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with leakage function pair* $\mathsf{L} = (\mathsf{LEnc}, \mathsf{LDec})$ *satisfies* (q, t, ϵ)-ciphertext integrity with nonce misuse and leakage both in encryption and decryption*, denoted as* $\mathsf{CIML2}$*, if for all* (q, t)*-bounded adversaries* $\mathcal{A}$*, we have:*

$$\Pr\left[\mathsf{CIML2}_{\mathsf{AE},\mathsf{L},\mathcal{A}} \Rightarrow 1\right] \leq \epsilon,$$

where the security game $\mathsf{CIML2}^b_{\mathsf{AE},\mathsf{L},\mathcal{A}}$ *is defined in Figure 1, and* $q = (q_e, q_d)$ *is an upper bound on the total number of queries made to the leaking encryption and decryption oracles, respectively.*

$\mathsf{CIML2}_{\mathsf{AE},\mathsf{L},\mathcal{A}}(1^\lambda)$ experiment	
Initialization:	Oracle $\mathsf{LEnc}(N, A, M)$:
$k \leftarrow \mathsf{Gen}(1^\lambda)$ s.t. $\lvert k\rvert = n := n(\lambda)$	$(C, \ell_e) = \mathsf{LEnc}_k(N, A, M)$
$\mathcal{S} \leftarrow \emptyset$	$\mathcal{S} \leftarrow \mathcal{S} \cup \{(N, A, C)\}$
Finalization:	return (C, ℓ_e)
$(N, A, C) \leftarrow \mathcal{A}^{\mathsf{LEnc}(\cdot,\cdot,\cdot),\mathsf{LDec}(\cdot,\cdot,\cdot)}$	
If $(N, A, C) \in \mathcal{S}$, return 0	Oracle $\mathsf{LDec}(N, A, C)$:
If $\perp = \mathsf{Dec}_k(N, A, C)$, return 0	$(M', \ell_d) = \mathsf{LDec}_k(N, A, C)$
Return 1	return (M', ℓ_d)

Table 1: CIML2 game.

We note that when the decryption device is not under the physical control of the adversary, it cannot measure the leakage when the decryption algorithm is running. For instance, this happens when a sender and a receiver use another secret key when their roles are permuted and the receiver wants to send encrypted plaintexts to the sender. We capture this scenario by only providing the adversary with the leakage during the encryption queries, and refer to CIML1 security in that case. In the corresponding CIML1 experiment, the oracle LDec is replaced by the black box oracle Dec. The number 1 or 2 in the CIML1 or CIML2 notations indicates the number of *leaking* oracles.

3.3 Confidentiality with leakage

Confidentiality of authenticated encryption follows the general blueprint of the indistinguishability of encrypted plaintexts against *Chosen Ciphertext Attacks* (CCA). In this scenario, the goal of the adversary is to compute two messages for which she will get the encryption of only one of them, resulting in the so-called *challenge ciphertext*, with the task of detecting which message has been encrypted. To make this adversary more powerful she can query the encryption of any message and the decryption of any ciphertext except the challenge ciphertext. This is captured by the encryption and decryption oracles that return the outputs of Enc_k and Dec_k, respectively.

We make the leakage available to the adversary by turning these oracles into their leaking versions LEnc_k and LDec_k. Even the challenge ciphertext is computed from LEnc_k, which captures the ability of the adversary to observe the device encrypting one message among the two she chooses. It means that no restriction is set on the algorithm execution with respect to leakage (L).

As far as nonce reuse is concerned, we allow the adversary to repeat nonces except the one used in the computation of the challenge ciphertext which may not appear in any other leaking encryption query. However, it may be involved in any decryption query. Such a notion is usually referred to as nonce misuse-resilience (m), as opposed to nonce misuse-resistance (M), where no restriction is made on the reuse of nonces. Misuse-resilience captures concrete situations where, for instance, a counter providing the nonce has been unintentionally reset or shifted. It ensures that the security of the challenges is then restored as soon as the counter recovers increments and provides fresh nonce values. We elaborate on this definitional choice in the Section 3.4.

Finally, we go one step further by allowing the adversary to observe the leakage corresponding to the decryption of the challenge ciphertext without returning the underlying plaintext. This addition is valuable in applications where users are allowed to use some code or content (e.g., firmware updates, or on-demand games or movies) but not to access this content due to IP restrictions.

$\mathsf{CCAmL2}^b_{\mathsf{AE},\mathsf{L},\mathcal{A}}$ is the output of the following experiment:

Initialization: generates a secret key $k \leftarrow \mathsf{Gen}(1^\lambda)$ s.t. $|k| = n := n(\lambda)$ and sets $\mathcal{E}_{ch}, \mathcal{E} \leftarrow \emptyset$

Leaking encryption queries: $\mathcal{A}^{\mathsf{L}}$ gets adaptive access to $\mathsf{LEnc}(\cdot,\cdot,\cdot)$,
$\mathsf{LEnc}(N, A, M)$ outputs $\perp$ if $(N, *, *) \in \mathcal{E}_{ch}$, else computes $C \leftarrow \mathsf{Enc}_k(N, A, M)$
and $\ell_e \leftarrow \mathsf{L}_{\mathsf{Enc}}(k, N, A, M)$, updates $\mathcal{E} \leftarrow \mathcal{E} \cup \{N\}$ and finally returns (C, ℓ_e)

Leaking decryption queries: $\mathcal{A}^{\mathsf{L}}$ gets adaptive access to $\mathsf{LDec}(\cdot,\cdot,\cdot)$,
$\mathsf{LDec}(N, A, C)$ outputs $\perp$ if $(N, A, C) \in \mathcal{E}_{ch}$, else computes $M \leftarrow \mathsf{Dec}_k(N, A, C)$
and $\ell_d \leftarrow \mathsf{L}_{\mathsf{Dec}}(k, N, A, C)$ and returns (M, ℓ_d)

Challenge queries: on a single occasion $\mathcal{A}^{\mathsf{L}}$ submits $(N_{ch}, A_{ch}, M^0, M^1)$,
If M^0 and M^1 have different length or $N_{ch} \in \mathcal{E}$, returns $\perp$, else computes
$(C^b, \ell_e^b) \leftarrow \mathsf{LEnc}_k(N_{ch}, A_{ch}, M^b)$, sets $\mathcal{E}_{ch} \leftarrow \{(N_{ch}, A_{ch}, C^b)\}$ and returns (C^b, ℓ_e^b)

Decryption challenge leakage queries: $\mathcal{A}^{\mathsf{L}}$ gets adaptive access to $\mathsf{L}_{\mathsf{decch}}(\cdot,\cdot,\cdot)$,
$\mathsf{L}_{\mathsf{Dec}}(N_{ch}, A_{ch}, C^b)$ computes and outputs $\ell_d^b \leftarrow \mathsf{L}_{\mathsf{Dec}}(k, N_{ch}, A_{ch}, C^b)$
if $(N_{ch}, A_{ch}, C^b) \in \mathcal{E}_{ch}$, else it outputs $\perp$;

Finalization: $\mathcal{A}^{\mathsf{L}}$ outputs a guess bit b' which is defined as the output of the game

Figure 3: The $\mathsf{CCAmL2}^b_{\mathsf{AE},\mathsf{L},\mathcal{A}}$ game.

Definition 4 (CCAmL2)**.** *An authenticated encryption* $\mathsf{AE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with leakage function pair* $\mathsf{L} = (\mathsf{L}_{\mathsf{Enc}}, \mathsf{L}_{\mathsf{Dec}})$*, is* (q, t, ϵ)-chosen-ciphertext secure with nonce misuse-resilience and leakage both in encryption and decryption*, denoted CCAmL2, if for all* (q, t)*-bounded adversary* $\mathcal{A}$*,*

$$\left|\Pr\left[\mathsf{CCAmL2}^0_{\mathsf{AE},\mathsf{L},\mathcal{A}} \Rightarrow 1\right] - \Pr\left[\mathsf{CCAmL2}^1_{\mathsf{AE},\mathsf{L},\mathcal{A}} \Rightarrow 1\right]\right| \leq \epsilon,$$

where the security game $\mathsf{CCAmL2}^b_{\mathsf{AE},\mathsf{L},\mathcal{A}}$ *is defined in Figure 3, and* $q = (q_e, q_d)$ *is an upper bound on the total number of queries made to the leaking encryption and decryption oracles, respectively.*

As for the relation between CIML2 and CIML1 for integrity, we capture the scenario where the decryption device is not under the physical control of the adversary by only providing her with the leakage during the encryption queries, and refer to CCAmL1 security in that case. In the corresponding CCAmL1 experiment, the oracle LDec is replaced by the black box oracle Dec.

3.4 Discussion

We next provide a more detailed discussion of the rationale behind our definitional choices.

On stronger security notions. As outlined by our notations, security definitions against leakage can consider leakage in encryption only (1) or in encryption and decryption (2), with nonce misuse-resilience (m) or misuse-resistance (M). As a result, it is explicit that the CIML2 guarantee is the strongest that can be achieved for integrity, while for confidentiality, we only focus on CCAmL2.

The reason why we do not focus on a combination of leakage-resistance and misuse-resistance (that would be be coined CCAML2) is that it would require all the blocks of the corresponding modes to be strongly protected against leakage, hence canceling the interest of mode-level protections. A bit more precisely, we first observe that standard side-channel attacks usually aim at extracting secrets: long-term ones in the case of DPA, ephemeral ones in the case of SPA. But CCAML2 security would further require that no adversary can detect nonce reuse for different messages. In this context, the following attack is possible: select two messages that are identical for some blocks and differ afterwards, and detect when the messages start to differ thanks to leakage. As a result, the two passes that misuse-resistant modes of operations leverage are circumvented thanks to leakage. And the attack does not require to extract secrets like standard DPAs and SPAs: it only requires to compare the leakage of two states. Concretely, such a *state comparison attack* is therefore much harder to prevent than a key recovery attack because all the operations of an implementation can directly be exploited in the comparison (which is in contrast with key recovery attacks that can only exploit the leakage of operations that can be guessed such as S-box computations in block ciphers). As a result, preventing state comparison attacks (which is necessary to reach $\mathsf{CCAML2}$ security) essentially requires that all the block cipher calls in a mode are sufficiently protected to make the leakage of two different messages vs. the leakage of two identical messages hard to identify, which eventually prevents any possibility of efficiently leveling the implementation. In other words, there is not theoretical impossibility to define $\mathsf{CCAML2}$ security and it could for example be achieved by requiring all the block cipher calls of a mode to be leak-free. Yet, as a result such a definition leads to quite uninteresting designs from the model-level leakage-resistance point-of-view.

Given that we cannot combine leakage-resistance, misuse-resistance and efficient leveled implementations, we next observe that one could then consider $\mathsf{CCAMl2}$ or $\mathsf{CCAmL2}$. The first one reflects a setting where the leakage of the challenge plaintext is excluded from the adversary's view and is denoted as leakage-resilience. The second one does not make restriction on the leakages obtained by the adversary, but considers that the nonce used to encrypt the challenge plaintext is fresh (i.e., a misuse-resilience setting). Following our rationale to focus on the strongest security notion that allows leveling the implementations thanks to mode-level design techniques, our discussions only consider $\mathsf{CCAmL2}$ security. The main reason of this choice is that by removing the leakage of the challenge ciphertext, $\mathsf{CCAMl2}$ deems designs that leak their plaintext in full as secure. By contrast, $\mathsf{CCAmL2}$ encourages the protection of the plaintext and, as discussed in Section 5, is therefore calling for modes leveraging internal re-keying processes that would otherwise not be needed. We insist that there is no theoretical impossibility nor practical difficulty to define $\mathsf{CCAMl2}$. There are even efficient constructions that match this security target. Overall, leakage-resilience (l) offers a different flavor of security than leakage-resistance (L), where confidentiality vanishes when leakage is available to the adversary, but is restored when this leakage is not available anymore.

We note that $\mathsf{CCAmL2}$ security implies that so-called *message comparison attacks*, where the adversary aims to distinguish two messages thanks to leakage, are a concern for the implementers. Whether strong protections against these attacks (e.g., leading to negligible advantage for the adversaries) can be formalized remains an open problem. Yet, contrary to the aforementioned state comparison attacks that can be mitigated by design (thanks to internal re-keying), message comparison attacks are an unavoidable issue, since the message at least has to be manipulated to be encrypted or decrypted. So either as a security target to ensure thanks to hardware-level and algorithmic countermeasures, or as an inherent weakness that must be considered when dealing with the confidentiality of cryptographic implementations, such a leakage cannot be ignored.

Differences with black box definitions. While the definitions of integrity and confidentiality with leakage can be viewed as natural extensions of the definitions without leakage with some additional impossibilities, they also differ in some fundamental aspects that we now detail.

Left-or-right vs. real-or-random. Our $\mathsf{CCAmL2}$ (and $\mathsf{CCAmL1}$) notions follow the so-called left-or-right paradigm to capture confidentiality. That is, the adversary has to distinguish the (say left) experiment $\mathsf{CCAmL2}^0_{\mathsf{AE},\mathsf{L},\mathcal{A}}$, where M_0 is encrypted in the challenge ciphertext, with the (say right) experiment $\mathsf{CCAmL2}^1_{\mathsf{AE},\mathsf{L},\mathcal{A}}$, where M_1 is encrypted in the challenge ciphertext. By contrast, in the real-or-random paradigm that can be used equivalently (and is usually considered more convenient) in the black box setting, the adversary has to distinguish a real encryption algorithm from an idealized function \$ returning random values in the ciphertext space. The reason why we use a left-or-right definition is that in order to adapt confidentiality with leakage in the real-or-random paradigm, we would have to define the leakage of an idealized function that (i) has by definition no implementation and (ii) creates a level of independence between the function's inputs and the outputs which is precisely what side-channel attacks threaten. As a result, left-or-right definitions are conceptually more appealing in a leakage setting, especially if aiming to obtain standard proofs without idealized assumptions, which we deem as an important long-term research goal.

Separation results. When considering AE without leakage, the combination of CPA security with ciphertext integrity implies CCA security. It can be proven that such an implication does not hold with leakage. Furthermore, combining CIML1 and CCAmL2 (resp., CIML2 and CCAmL1) is not sufficient to get CIML2 (resp., CCAmL2) either. This is shown by modifying a leaking AE that achieves the premise notions into another leaking AE that still satisfies them but fails to satisfy the consequent notion. Therefore the proposed combination of CIML2 and CCAmL2 is the strongest one (given the impossibility to ensure CCAML2 efficiently). Such separation results also lead to a strict hierarchy between the following model-level security grades:

Grade 1: An AE achieving either CCAL1 security (Grade 1a) or CIML2 security (Grade 1b).

Grade 2: An AE achieving both CIML2 security and CCAmL1 security.

Grade 3: An AE achieving both CIML2 security and CCAmL2 security. and two-pass decryption.

We insist that a higher mode-level security grade does not imply a higher concrete side-channel security. These grades should rather be seen as different tradeoffs between model-level protections and hardware-level/algorithmic countermeasures. In general, modes for which the security can be proven in leakage models that allow leveled implementations (i.e., for which the proof can leverage weaker security requirements for some computations) gain interest over modes that require a uniform protection of all their computations if the corresponding grade has to be reached. But it is always possible to reach all the grades thanks to strong physical assumptions.

Extensions. All our notions are described in the single-user setting and the confidentiality notions are also only given in the single-challenge setting. There are natural extensions of them modeling (realistic) treats where an adversary can see the execution of the scheme for many users and several challenge ciphertexts in a single experiment. These notions are actually asymptotically equivalent to those presented here, and they are mainly useful to prove better security bounds.

4 Leakage models

An implementation leaking all its secret in full cannot offer any security guarantee. Hence, satisfying the security definitions of Section 3 requires limiting the leakage with some hardware-level or algorithmic countermeasures. The goal of mode-level protections is to limit the need of such lower-level countermeasures and to prove the leakage security of authentication or encryption schemes with weak and falsifiable assumptions. We next discuss assumptions that can be used for this purpose, link them to our attack taxonomy of Section 2 and discuss the challenges they raise. We do this for integrity and confidentiality separately since, as already mentioned, it allows us to take advantage of the fundamentally different requirements to reach these security notions.

4.1 Models for integrity

Informally, it is possible to design a Message Authentication Codes (MAC) such that a majority of its internal computations can be leaked in full and only one block cipher execution is strongly protected against DPA. Formally, such MACs are proven in the unbounded leakage model.

Definition 5 (Unbounded leakage model). *An implementation of a scheme is said to offer a security property in the unbounded leakage model if that property is satisfied even if the leakage yields all the internal states produced during each execution of the scheme, including keys and random coins and excluding the internal state of strongly protected components if there are any.*

Informally, the strongly protected components capture the parts of the implementation that require security against DPA. Formally, there are different options to model such components. A first one is to assume them to be *leak-free*. That is, to assume that this component does not provide the adversary with any leakage. While conceptually simple, and frequently useful as a first step to identify schemes that have good properties against leakage, it suffers from the drawback of being a strong and idealized assumption. A weaker and more realistic (empirically falsifiable) alternative is to assume that the leaking block cipher implementation remains unpredictable with leakage.

Definition 6 (Unpredictability with leakage). *A block cipher* $\mathsf{BC} : \{0,1\}^n \times \{0,1\}^n \longmapsto \{0,1\}^n$ *with leakage function pair* $\mathsf{L} = (\mathsf{L}_{\mathsf{eval}}, \mathsf{L}_{\mathsf{inv}})$ *is* $(q_e, q_i, q_{\mathsf{L}}, t, \epsilon)$ *strongly unpredictable with leakage in evaluation and inversion, or* $(q_e, q_i, q_{\mathsf{L}}, t, \epsilon)$-SUL2*, if for any* $(q_e, q_i, q_{\mathsf{L}}, t)$*-adversary* A*, we have:*

$$\Pr[\mathsf{SUL2}^{\mathsf{BC},\mathsf{L}}_{\mathsf{exp}} \Rightarrow 1] \leq \epsilon,$$

with q_e *evaluation queries,* q_i *inversion queries,* q_{L} *(offline) queries to the leakage function (for profiling), time complexity* t *and the* $\mathsf{SUL2}^{\mathsf{BC},\mathsf{L}}_{\mathsf{exp},\mathsf{A}}$ *experiment defined in Table 2.*

```
$\mathsf{SUL2}^{\mathsf{BC},\mathsf{L}}_{\mathsf{exp},\mathsf{A}}$ experiment.

Initialization:
    $k \xleftarrow{\$} \{0,1\}^n$
    $\mathcal{L} \leftarrow \emptyset$

Finalization:
    $(x, y) \leftarrow \mathsf{A}^{\mathsf{L}_{\mathsf{eval}}(.,.),\mathsf{L}_{\mathsf{inv}}(.,.),\mathsf{L}}$
    If $(x, z) \in \mathcal{L}$
        Return 0
    If $y == \mathsf{BC}_k(x)$
        Return 1
    Return 0

Oracle LEval$(x)$:
    $y = \mathsf{BC}_k(x)$
    $\ell_{\mathsf{e}} = \mathsf{L}_{\mathsf{eval}}(k, x)$
    $\mathcal{L} \leftarrow \mathcal{L} \cup \{(x, y)\}$
    Return $(z, \ell_{\mathsf{e}})$

Oracle $\mathcal{L}_{\mathsf{inv}}(y)$:
    $x = \mathsf{BC}_k^{-1}(y)$
    $\ell_{\mathsf{i}} = \mathcal{L}_{\mathsf{inv}}(k, y)$
    $\mathcal{L} \leftarrow \mathcal{L} \cup \{(x, y)\}$
    Return $(x, \ell_{\mathsf{i}})$
```

Table 2: Strong unpredictability with leakage in evaluation and inversion experiment.

Concretely, the parts of an implementation that can leak in an unbounded manner do not require any lower-level countermeasure. Besides, unpredictability is among the weakest possible requirements for a block cipher. For example, it is widely believed that without leakage, block ciphers become unpredictable with less rounds than needed to become pseudorandom permutations. Combined with the fact that unpredictability with leakage is easy to test for evaluation laboratories, as breaking it essentially requires performing a successful key-recovery (which is the focus of most published side-channel attacks), it puts the analysis of integrity with leakage in a comfortable situation where theoretical assumptions easily translate into requirements for implementers.

4.2 Models for confidentiality

The situation of confidentiality with leakage is by far more delicate than the one of integrity with leakage. One reason is that, as explained in Section 3.4, the very definition of confidentiality with leakage is harder to capture. Another reason is that, contrary to integrity, confidentiality requires bounding the leakage of most of (if not all) its computations in some sense. For example, the unbounded leakage of a message trivially breaks its confidentiality (not its integrity). So on the one hand, confidentiality with leakage requires some strongly protected components that withstand DPA, and we can use the leak-free (or, ideally, unpredictability) assumption(s) of the previous section for this purpose. On the other hand, implementing all the components of an encryption scheme so that they offer these strong guarantees cancels the interest of mode-level countermeasures for confidentiality, since it implies that the implementation needs to be uniformly protected against DPA. As a result, and in order to enable leveled implementations, leakage-resistant encryption schemes aim to leverage the execution of a few strongly protected components, leaving the rest of their computations leaking in a more liberal, yet still bounded in some sense, manner.

Besides, one more difficulty is that formally, confidentiality with leakage not only requires bounding the leakage's informativeness but also its computational power. A notorious example is the one of so-called "future computation attacks" that are easily explained from Figure 1. Say the leakage function only leaks one bit per block cipher execution. If the leakage function is efficient (e.g., polytime) and adaptive, then it is possible to leak one bit of s_{128} when computing s_1 (letting the leakage function computing 128 block cipher executions), another bit of s_{128} when computing s_2, and so on and so forth, leading to a full key leakage after 128 iterations of the PRG round. The attack is admittedly unrealistic (i.e., it is unlikely that a leakage function leaks about future computations) but it shows that formalizing leakage with standard computational assumptions is challenging. In particular, it must deal with the paradox that solving Maxwell's equations for a complex circuit is a computationally intensive problem (that can take weeks with computer aided design software), but access to the physical circuit provides an instantaneous solution to it.

Overall, no physical assumption enables to analyze confidentiality with leakage in a way that is at the same time theoretically sound and practically relevant in the current state-of-the-art. Yet, existing assumptions share some necessary practical conditions. Namely, bounded leakage for some computation at least requires that it ensures security against SPA (as in the case of the PRG in Figure 1), or average SPA (as in the case of the PRF in Figure 2). In a bit more detail, the main solutions to bound the informativeness of the leakage for the analysis of confidentiality include:

1. *Hard-to-invert leakage*, which simply assumes that a value remains hard to guess even after the observation of its leakage. It is the weakest possible solution and, like the unpredictability with leakage, it is quantifiable/falsifiable by evaluation laboratories.

2. *Bounded range or information*, which rather assumes that the leakage function has a bounded range or that it preserves the (pseudo)entropy of the key. The first assumption is generally unrealistic for power or electromagnetic leakages since a single power or electromagnetic trace can contain hundreds or even thousands of samples, and therefore directly exhausts the key entropy. The second one is more realistic but is hard to quantify/falsify in practice.

As for the main solutions to bound the computational power of the leakage, they include:

a. *Combining the only computation leaks assumption with design tweaks.* By doubling the key size of the PRF of Figure 1 and tweaking it in such a way that block cipher executions use one or the other half of the key (using a so-called alternating structure), one prevents the future computation attack as long as the leakage does not leak about the unused key part.

b. *Oracle-free leakage functions*, which rather assumes that the block ciphers of Figure 1 are random oracles that can be queried by the adversary but not by the leakage function.

The most theoretically-satisfying solution would be to mix hard-to-invert leakages (1) with the only computation leaks assumption and design tweaks (a). The best state-of-the-art solutions either mix leakage with bounded range or information (2) with the only computation leaks assumption and design tweaks (a), or hard-to-invert leakages (1) with the idealized oracle-free assumption (b).

Eventually, another solution is to consider a simulatable leakage assumption. Its general idea is to avoid modeling the (computational power of the) leakage function and to directly assume that it is possible to simulate the leakage trace of a block cipher using the same hardware as used to generate it, but without knowledge of its secret key. Yet, in the current state-of-the-art, no proposal of leakage simulator can ensure simulatability that guarantees negligible adversarial advantage.

4.3 Practical guidelines

The assumptions used in proofs of security with leakage can be used in order to specify which parts of an implementation must be protected and how much. In particular, the computations that can leak in an unbounded manner do not require any protection. Block cipher implementations modeled as leak-free or unpredictable with leakage must ensure security against DPA. Block cipher implementations that require bounded leakage for confidentiality must ensure security against SPA in case of $(q, 1)$-bounding constructions, and security against avgSPA in case of $(q, .)$-bounding constructions. Concretely, protecting block cipher implementations against DPA is generally achieved thanks to the masking countermeasure. By contrast, security against SPA or avgSPA is best achieved thanks to parallel implementations (in hardware) of shuffling (in software). We finally note that the leakage-resilient PRF of Figure 2 can be used to instantiate a strongly protected component by reducing its DPA security requirement to an avgSPA security requirement.

5 Constructions

The most common generic approach to the design of an authenticated encryption scheme proceeds by encrypting messages with a CPA-secure encryption scheme, then authenticating the ciphertext with a MAC. This section explores the challenges associated to the design of these two key ingredients in the presence of leakages, starting with a MAC, then turning to encryption. We conclude by discussing the application of these results to the design of a leakage-resistant AE.

5.1 A leakage-resilient MAC

We consider the security of a MAC $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ in the presence of a side-channel adversary. By analogy with the $\mathsf{CIML2}$ game of Table 1, we assume an adversary who can repeatedly query the $\mathsf{LMac}(\cdot)$ and $\mathsf{LVrfy}(\cdot, \cdot)$ oracles that return the evaluation of Mac and Vrfy on a secret key, together with the corresponding leakages, when respectively queried on message and message-tag pairs. The adversary wins if she can produce a forgery, that is, a valid message-tag pair for a fresh message, based on the information that she collected thanks to her queries to leaking algorithms.

What can go wrong with a standard MAC? Let us consider, as a starting point, one of the most common block-cipher based MAC, namely $\mathsf{CBC\text{-}MAC}$, which is illustrated in Figure 4 and is part of the CCM authenticated encryption mode (standing for Counter with cipher block Chaining Message authentication code) that is included in the TLS 1.3 cipher suite for instance. We know that CBC-MAC is secure for fixed-length messages, so an adversary would need to rely on leakages in order to win the CIML2 security game. This could be done in at least two different ways:

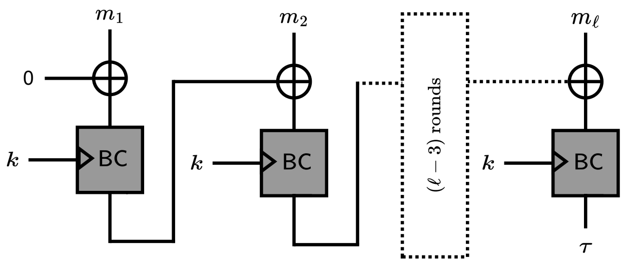

Figure 4: CBC-MAC used to authenticate a message $M = m_1, m_2, \dots, m_\ell$ with key k. Tag verification is performed by recomputing the correct tag and comparing it to the given one.

(1) The adversary can try to extract the secret key k from the leakages that she obtains from the LMac and LVrfy oracles. For an ℓ-block message, each query will give her ℓ leakages of the block cipher evaluation operating with key k. The first and last of these block-cipher evaluations would be particularly appealing as they proceed on a known plaintext or on a known ciphertext.

(2) The adversary can try to extract the tag associated to a message M for which she did not issue a $\mathsf{LMac}(M)$ query, by performing multiple $\mathsf{LVrfy}(M, \tau^*)$ queries for random values of τ^*, and trying to recover the correct tag τ when it is computed by Vrfy and compared to τ^*.

Designing a leakage-resilient MAC. Let us try to gain protection against these two attack avenues, starting with the first one. As k is a static parameter of Mac and Vrfy, and in the absence of a simple option to update the value of k between calls to LMac and LVrfy, we will rather require that BC is only evaluated with k as part of a strongly protected implementation.

Of course, one straightforward option would be to entirely evaluate Mac using the strongly protected BC implementation (which is reflected by the dark gray block cipher calls in Figure 4). But as already mentioned, this would be quite expensive in time and energy, and we therefore aim for better. Furthermore, it remains desirable to reduce the exposure of the strongly protected implementation (since strong protections may not be perfect), and to make use of it sparingly.

It is fortunately quite simple to make sure that the strongly protected block cipher is only used once per message, independently of the number blocks in the message. One option is to define $\mathsf{Mac}_k(M) = \mathsf{BC}_k(\mathsf{H}(M))$, where H is a collision-resistant hash function with an output of the length of a block (which can itself be implemented with an unprotected block cipher). If the strongly protected implementation of BC is modeled as leak-free, implying that it leaks nothing about the key k, then the LMac oracle won't provide any useful leakage information to the adversary, and the LMac is just as good as a Mac oracle that would not offer any leakage. The security of this construction can be proven (and gracefully degrades) with the weaker unpredictablity with leakage assumption. Of course, the black box security of this MAC is quite weak: if we use a block cipher of 128-bit block size, then the collision resistance of the hash function cannot go up the birthday bound, and an adversary who can evaluate around 2^{64} hashes may already be able to make a forgery. Improved solutions exist(e.g., from the use of tweakable block ciphers).

Turning to the second attack avenue, we can observe that we do not have much improvement if we follow the obvious way of defining $\mathsf{Vrfy}_k(M, \tau^*)$ as testing if $\mathsf{Mac}_k(M) \overset{?}{=} \tau^*$. Indeed, even if

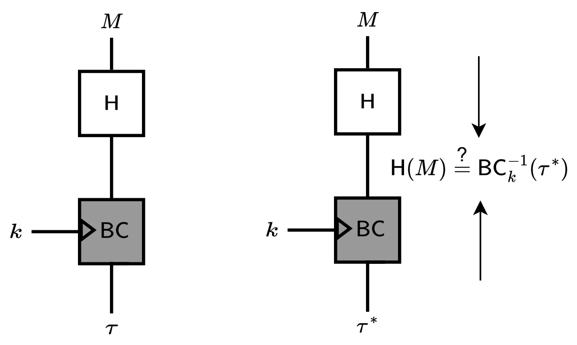

Figure 5: The HBC MAC together with its tag verification operation.

we assume that BC does not leak anything about its output either, the implementation would still require a strongly protected equality testing component, since otherwise the comparison may leak the value of $\mathsf{Mac}_k(M)$ thanks to a standard DPA leveraging the leakage of multiple τ^* values.

There is however another way of defining the Vrfy function, illustrated in the right part of Figure 5: we may also test whether $\mathsf{H}(M) \overset{?}{=} \mathsf{BC}^{-1}(\tau^*)$, taking advantage of the block cipher's invertibility. The advantage of this definition of Vrfy is that if the adversary provides an incorrect tag τ^*, and even if the comparison operation leaks its inputs in full, the adversary would only learn $\mathsf{BC}^{-1}(\tau^*) \neq \mathsf{H}(M)$ rather than learning the correct tag. The analysis of such a scheme is delicate though, as it requires considering the interaction between the hash function and the block cipher. Improved solutions can come from the use of tweakable block ciphers (e.g., by using $\mathsf{H}(M)$ as a tweak and encrypting a constant value with this tweak in order to produce the tag).

5.2 A leakage-resistant encryption scheme

We focus here on a simple CPAL security notion, which is considerably less demanding than the $\mathsf{CCAmL2}$ security notion discussed above (see Definition 4), but already brings important ingredients that are used in the design of a complete authenticated encryption scheme.

The CPAL security game is identical to the traditional CPA security game, except that all the encryption operations performed by the challenger return a leakage together with a ciphertext. More precisely, we consider an adversary who can repeatedly query a $\mathsf{LEnc}(\cdot)$ oracle that returns the evaluation of Enc on an adversarially chosen message with a secret key, together with the corresponding leakage. The adversary can additionally send a challenge query with two messages (M_0, M_1) of identical lengths. The challenger will flip a coin b and return an encryption of M_b together with the associated leakage. The adversary wins if he can guess the coin b.

What can go wrong with a standard CPA encryption mode? Let us consider the counter (CTR) mode illustrated in Figure 6 as our starting point, as it is used to encrypt messages in both the CCM and GCM modes included in the TLS 1.3 cipher suite for instance.

The CTR mode offers CPA security, so winning the CPAL game with non-negligible probability requires taking advantage of the leakages. Again, two main approaches can be considered:

(1) The adversary can try to extract the secret key k from the leakages that she obtains from the LEnc oracle. Each message block of each LEnc query and of the challenge query gives her a leakage of the block cipher evaluation operating with the key k. If k is recovered, the challenge ciphertext can be decrypted to determine whether M_0 or M_1 was encrypted.

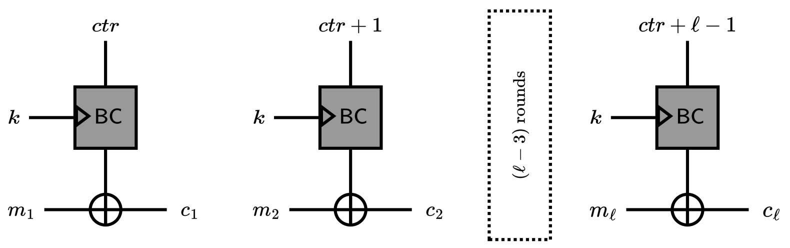

Figure 6: The CTR mode used to encrypt a message $M = m_1, m_2, \ldots, m_\ell$ with key k.

(2) The adversary can try to exploit the leakage obtained during the encryption of M_b during the challenge query, in order to identify whether it is M_0 or M_1 that is encrypted.

From a practical point-of-view, the two strategies come with very different requirements. The first one is a standard key recovery attack. It requires to recover a full secret value, with the help of a potentially large number of leakages. The second one is a message comparison attack. It requires to recognize a message within a set of two chosen messages which, as discussed in Section 3.4, is a considerably less demanding goal. But that goal must be accomplished with a single leakage.

Designing a CPAL-secure encryption scheme In order to mitigate attack strategies targeting the encryption key, we want, as we did for the MAC construction, to make sure that this key is used as little as possible, meaning in a few block cipher evaluations per message encryption, and have only these block cipher evaluations strongly protected. However, contrary to the MAC case, we cannot process the whole plaintext without any keying material: we need a key to hide it. One solution is to start the encryption process with a key derivation process: given a unique nonce N, we derive a per-message key $k^* := \mathsf{BC}_k(N)$, and use k^* in order to encrypt the message using the CTR mode (e.g., one could use the concatenation of the nonce and the message block number as a counter for the CTR mode). If the key derivation operation is sufficiently protected, then the only key that an adversary could recover using leakages would be the per-message key.

If we aimed for leakage-resilience (i.e., CPAl security) instead of leakage-resistance (i.e., CPAL security), then this would already be good enough: the challenge query would not offer any leakage, and the leakages obtained through the LEnc queries would, in the worst case, offer per-message keys that would be unrelated to the per-message key used in the challenge query. For CPAL security though, the per-message key derivation leaves us with the task of protecting the per-message key of the challenge query: that key is still used in every block in the CTR mode, which may therefore be enough to mount a successful DPA attack if long messages are encrypted by the adversary.

We can address this challenge by switching to an encryption mode that additionally derives a fresh key per message block. We also observe that this mode may only offer a weaker form of security: since we already have per-message keys from our key derivation operation, the mode only needs to offer eavesdropper security instead of CPA security. In effect, we only need a leakage-resilient PRG, as proposed in Figure 1, that would use the per-message key as its seed, and to XOR the output of this PRG with the message. The resulting mode is depicted in Figure 7.

More precisely, it is possible to demonstrate that this mode offers CPAL security when the initial key derivation block cipher is modeled as leak-free and when the leakages of all the other block cipher evaluations are simulatable (see discussion in Section 4.2). The analysis shows that, as long as two executions of a leaking block cipher with a single key do not leak too much information about that key, then winning the CPAL game will essentially be as hard as guessing which one of

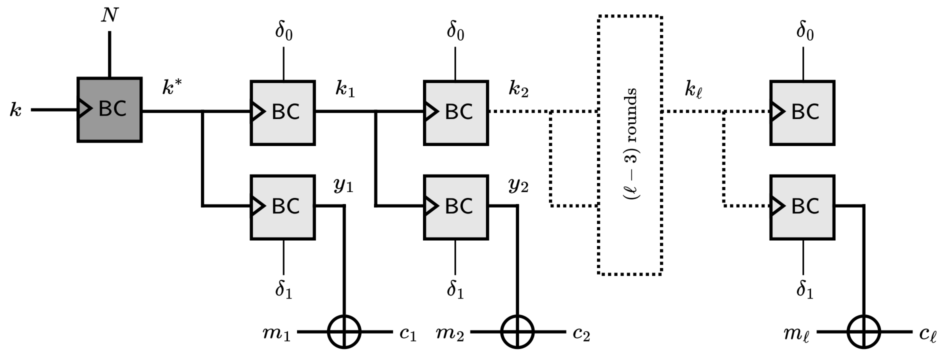

Figure 7: A CPAL secure encryption mode.

the two challenge messages is encrypted given the leakages of the XOR operation on the plaintext message blocks (i.e., a message comparison attack). That target seems to be the minimal demand that we can make from a mode of operation, since the plaintext has to be processed at some point, and we must leave the protection of that minimum processing to lower-level countermeasures.

We note that the encryption scheme of Figure 7 provides CPAmL security – we eluded a discussion of nonce misuse in this section to simplify the exposition of the scheme' main design ideas.

5.3 A leakage-resistant AE scheme

Eventually, the MAC of Section 5.1 and the encryption scheme of Section 5.2 can be combined into the leakage-resistant AE illustrated in Figure 8. Informally, this scheme offers CIML2 security under the sole assumption that the two dark gray block cipher calls are strongly protected against DPA (all the other block cipher calls and the hash function can then leak in full), and CCAmL2 security under the additional requirement that the light gray block cipher calls used in the encryption scheme are secure against SPA. It uses two different long-term keys, but it is possible to halve the key material with a separation bit for the block ciphers used in the encryption and the MAC parts. We note that the composite definitional framework presented in this chapter does not benefit from generic composition theorems. So the analysis of leakage-resistant AE like the one of Figure 8 for now requires an ad hoc investigation. The literature contains many other modes that offer better bounds or target different mode-level security grades. We list some of them in the final section.

6 Bibliographical notes and concrete instances

The leakage-resilient building blocks of Section 2 are based on the seminal work of Dziembowski and Pietrzak [DP08]. Early examples of PRG constructions and analyzes can be found in [Pie09, YSPY10]. Early examples of PRF constructions and analyzes can be found in [FPS12, YS13]. These formal results follow more heuristic investigations highlighting the interest of re-keying against leakage [Koc03, PSP$^+$08]. The definitional framework of Section 3 is from [GPPS19]. An alternative all-in-one definition can be found in [BMOS17]. The first one is composite and considers a mix of leakage-resistance with nonce misuse-resilience. The second one is all-in-one and considers a mix of leakage-resilience with nonce misuse-resistance. The difficulty to define confidentiality in the presence of leakage, hinting towards the need to distinguish leakage-resistance and leakage-resilience, dates back to the seminal work of Micali and Reyzin [MR04]. It is worth noticing that the state comparison attack that is argued to be very expensive to prevent in Section 3.4 is similar

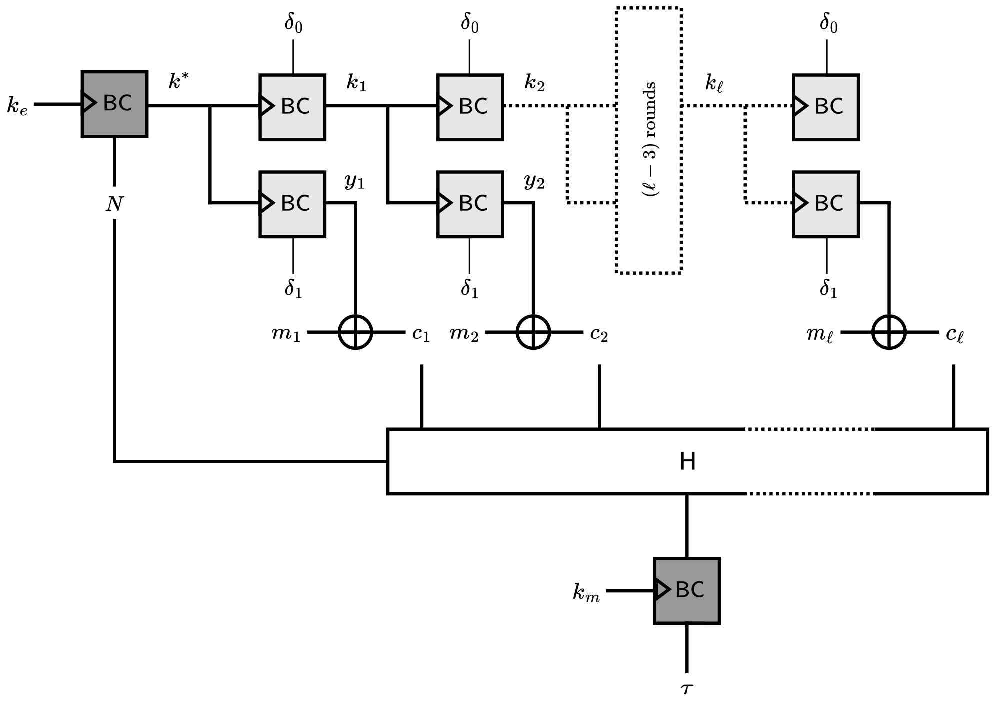

Figure 8: Exemplary leakage-resistant AE.

to the attacks against the FO transform that are known to raise fundamental challenges for the implementation security of post-quantum public-key encryption schemes [UXT+22, ABH+22]. The equivalence between real-or-random and left-or-righ definitions in the context of black box symmetric encryption is proven in [BDJR97]. The fact that CPA security combined with ciphertext integrity implies CCA security is proven in [KY00]. The unbounded leakage model described in Section 4 was introduced in [BKP+18]. Unpredictability with leakage was introduced in [BGP+19] and is used in [BGPS21] to prove the security of a leakage-resilient MAC without idealized assumptions. Hard-to-invert leakages were introduced in [DKL09] and used in [YSPY10] to analyze a leakage-resilient PRG under an idealized assumption. Bounded leakage or information are already used in the first work of Dziembowski and Pietrzak [DP08]. Simulatable leakages were introduced in [SPY13] and pitfalls regarding the instance of simulator proposed in this reference are discussed in [LMO+14]. Relations between these leakage assumptions are analyzed in [FH15]. The only computation leaks assumption dates back to Micali and Reyzin [MR04] and is combined with an alternating structure to prevent future computation leakage in [DP08, Pie09]. The assumption of oracle-free leakage function was introduced in [YSPY10] to analyze a more efficient PRG. A concrete application of the second attack path against CBC-MAC in Section 5.1 can be found in [BMPS21]. The following leakage-resilient MAC corresponds to the HBC construction analyzed in [BGP+19]. The leakage-resistant encryption scheme of Section 5.2 is from [PSV15]. We note that generating an ephemeral key as in this section may not only help in the context of side-channel attacks, but can also improve the bounds of black box security analyzes – see [GL17] for instance. An alternative treatment of leakage-resilient symmetric encryption based on re-keying can be found in [ABF13]. The leakage-resistant AE of Section 5.3 is a nonce-based variant of the EDT scheme in [BPPS17]. A longer discussion of definitions, models and designs for leakage-resistance was given in a Eurocrypt 2019 invited talk [Sta19]. A systematic investigation of some leakage-resistant AE according to their mode-level security grade (listed in Section 3.4) can be found in [BBC+20], which is also the basis of the practical guidelines in Section 4.3. It includes both constructions based on (tweakable) block ciphers, the formal analysis of which is based on the aforelisted references, and constructions based on permutations, the formal analysis of which is treated in [DM19, DJS19, GPPS20]. The additional challenges raised by the tag verification for permutation-based designs are discussed in [DM21]. Examples of constructions based on (tweakable) block ciphers include TEDT for Grade 3 [BGP+20], Triplex for Grade 2 [SPS+22] and AES-LR for Grade 1b [GKP20]. Examples of constructions based on permutations include ISAP for Grade 3 [DEM+20], Ascon for Grade 2 [DEMS21] and PHOTON-Beetle for Grade 1a [BCD+19]. For most of these schemes, the strongly protected primitive calls have to be implemented with state-of-the-art higher-order masking (e.g., [GR17] in software, [CGLS21] in hardware). Alternatively, it is possible to use a leakage-resilient PRF for this purpose: ISAP achieves this with a permutation-based design while LR-BC does it with block ciphers [BMPS21]. Yet another solution is to rely on a fresh re-keying scheme that generates a key with an easy-to-mask (e.g., key-homomorphic) primitive. Various models and proposals have been introduced for this purpose [MSGR10, DKM+15, DFH+16, Men20, DMMS21].

Acknowledgments. Thomas Peters and François-Xavier Standaert are respectively associate researcher and senior research associate of the Belgian Fund for Scientific Research (F.R.S.- FNRS). This work has been funded in parts by the ERC consolidator grant SWORD (724725) and by the Walloon region CyberExcellence project number 2110186 (acronym Cyberwal).

References

[ABF13] Michel Abdalla, Sonia Belaïd, and Pierre-Alain Fouque. Leakage-resilient symmetric encryption via re-keying. In *CHES*, volume 8086 of *Lecture Notes in Computer Science*, pages 471–488. Springer, 2013.

[ABH+22] Melissa Azouaoui, Olivier Bronchain, Clément Hoffmann, Yulia Kuzovkova, Tobias Schneider, and François-Xavier Standaert. Systematic study of decryption and re-encryption leakage: The case of Kyber. In *COSADE*, volume 13211 of *Lecture Notes in Computer Science*, pages 236–256. Springer, 2022.

[BBC+20] Davide Bellizia, Olivier Bronchain, Gaëtan Cassiers, Vincent Grosso, Chun Guo, Charles Momin, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Mode-level vs. implementation-level physical security in symmetric cryptography - A practical guide through the leakage-resistance jungle. In *CRYPTO (1)*, volume 12170 of *Lecture Notes in Computer Science*, pages 369–400. Springer, 2020.

[BCD+19] Zhenzhen Bao, Avik Chakraborti, Nilanjan Datta, Jian Guo, Mridul Nandi, Thomas Peyrin, and Kan Yasuda. PHOTON-Beetle authenticated encryption and hash family. *Submission to the NIST Lightweight Cryptography Standardization Effort*, 2019.

[BDJR97] Mihir Bellare, Anand Desai, E. Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. In *FOCS*, pages 394–403. IEEE Computer Society, 1997.

[BGP+19] Francesco Berti, Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Strong authenticity with leakage under weak and falsifiable physical assumptions. In *Inscrypt*, volume 12020 of *Lecture Notes in Computer Science*, pages 517–532. Springer, 2019.

[BGP$^+$20] Francesco Berti, Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. TEDT, a leakage-resist AEAD mode for high physical security applications. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(1):256–320, 2020.

[BGPS21] Francesco Berti, Chun Guo, Thomas Peters, and François-Xavier Standaert. Efficient leakage-resilient MACs without idealized assumptions. In *ASIACRYPT (2)*, volume 13091 of *Lecture Notes in Computer Science*, pages 95–123. Springer, 2021.

[BKP$^+$18] Francesco Berti, François Koeune, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Ciphertext integrity with misuse and leakage: Definition and efficient constructions with symmetric primitives. In *AsiaCCS*, pages 37–50. ACM, 2018.

[BMOS17] Guy Barwell, Daniel P. Martin, Elisabeth Oswald, and Martijn Stam. Authenticated encryption in the face of protocol and side channel leakage. In *ASIACRYPT (1)*, volume 10624 of *Lecture Notes in Computer Science*, pages 693–723. Springer, 2017.

[BMPS21] Olivier Bronchain, Charles Momin, Thomas Peters, and François-Xavier Standaert. Improved leakage-resistant authenticated encryption based on hardware AES coprocessors. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(3):641–676, 2021.

[BPPS17] Francesco Berti, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. On leakage-resilient authenticated encryption with decryption leakages. *IACR Trans. Symmetric Cryptol.*, 2017(3):271–293, 2017.

[CGLS21] Gaëtan Cassiers, Benjamin Grégoire, Itamar Levi, and François-Xavier Standaert. Hardware private circuits: From trivial composition to full verification. *IEEE Trans. Computers*, 70(10):1677–1690, 2021.

[DEM$^+$20] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, Bart Mennink, Robert Primas, and Thomas Unterluggauer. ISAP v2.0. *IACR Trans. Symmetric Cryptol.*, 2020(S1):390–416, 2020.

[DEMS21] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2: Lightweight authenticated encryption and hashing. *J. Cryptol.*, 34(3):33, 2021.

[DFH$^+$16] Stefan Dziembowski, Sebastian Faust, Gottfried Herold, Anthony Journault, Daniel Masny, and François-Xavier Standaert. Towards sound fresh re-keying with hard (physical) learning problems. In *CRYPTO (2)*, volume 9815 of *Lecture Notes in Computer Science*, pages 272–301. Springer, 2016.

[DJS19] Jean Paul Degabriele, Christian Janson, and Patrick Struck. Sponges resist leakage: The case of authenticated encryption. In *Krypto-Tag*. Gesellschaft für Informatik e.V. / FG KRYPTO, 2019.

[DKL09] Yevgeniy Dodis, Yael Tauman Kalai, and Shachar Lovett. On cryptography with auxiliary input. In *STOC*, pages 621–630. ACM, 2009.

[DKM$^+$15] Christoph Dobraunig, François Koeune, Stefan Mangard, Florian Mendel, and François-Xavier Standaert. Towards fresh and hybrid re-keying schemes with beyond birthday security. In *CARDIS*, volume 9514 of *Lecture Notes in Computer Science*, pages 225–241. Springer, 2015.

[DM19] Christoph Dobraunig and Bart Mennink. Leakage resilience of the Duplex construction. In *ASIACRYPT (3)*, volume 11923 of *Lecture Notes in Computer Science*, pages 225–255. Springer, 2019.

[DM21] Christoph Dobraunig and Bart Mennink. Leakage resilient value comparison with application to message authentication. In *EUROCRYPT (2)*, volume 12697 of *Lecture Notes in Computer Science*, pages 377–407. Springer, 2021.

[DMMS21] Sébastien Duval, Pierrick Méaux, Charles Momin, and François-Xavier Standaert. Exploring crypto-physical dark matter and learning with physical rounding towards secure and efficient fresh re-keying. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(1):373–401, 2021.

[DP08] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *FOCS*, pages 293–302. IEEE Computer Society, 2008.

[FH15] Benjamin Fuller and Ariel Hamlin. Unifying leakage classes: Simulatable leakage and pseudoentropy. In *ICITS*, volume 9063 of *Lecture Notes in Computer Science*, pages 69–86. Springer, 2015.

[FPS12] Sebastian Faust, Krzysztof Pietrzak, and Joachim Schipper. Practical leakage-resilient symmetric cryptography. In *CHES*, volume 7428 of *Lecture Notes in Computer Science*, pages 213–232. Springer, 2012.

[GKP20] Chun Guo, Mustafa Khairallah, and Thomas Peyrin. AET-LR: Rate-1 leakage-resilient AEAD based on the Romulus family. In *NIST LWC Workshop*, 2020.

[GL17] Shay Gueron and Yehuda Lindell. Better bounds for block cipher modes of operation via nonce-based key derivation. In *CCS*, pages 1019–1036. ACM, 2017.

[GPPS19] Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Authenticated encryption with nonce misuse and physical leakage: Definitions, separation results and first construction - (extended abstract). In *LATINCRYPT*, volume 11774 of *Lecture Notes in Computer Science*, pages 150–172. Springer, 2019.

[GPPS20] Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Towards low-energy leakage-resistant authenticated encryption from the Duplex sponge construction. *IACR Trans. Symmetric Cryptol.*, 2020(1):6–42, 2020.

[GR17] Dahmun Goudarzi and Matthieu Rivain. How fast can higher-order masking be in software? In *EUROCRYPT (1)*, volume 10210 of *Lecture Notes in Computer Science*, pages 567–597, 2017.

[Koc03] Paul C. Kocher. Leak-resistant cryptographic indexed key update, 2003. US Patent 6,539,092.

[KY00] Jonathan Katz and Moti Yung. Unforgeable encryption and chosen ciphertext secure modes of operation. In *FSE*, volume 1978 of *Lecture Notes in Computer Science*, pages 284–299. Springer, 2000.

[LMO$^+$14] Jake Longo, Daniel P. Martin, Elisabeth Oswald, Daniel Page, Martijn Stam, and Michael Tunstall. Simulatable leakage: Analysis, pitfalls, and new constructions. In

ASIACRYPT (1), volume 8873 of *Lecture Notes in Computer Science*, pages 223–242. Springer, 2014.

[Men20] Bart Mennink. Beyond birthday bound secure fresh rekeying: Application to authenticated encryption. In *ASIACRYPT (1)*, volume 12491 of *Lecture Notes in Computer Science*, pages 630–661. Springer, 2020.

[MR04] Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 278–296. Springer, 2004.

[MSGR10] Marcel Medwed, François-Xavier Standaert, Johann Großschädl, and Francesco Regazzoni. Fresh re-keying: Security against side-channel and fault attacks for low-cost devices. In *AFRICACRYPT*, volume 6055 of *Lecture Notes in Computer Science*, pages 279–296. Springer, 2010.

[Pie09] Krzysztof Pietrzak. A leakage-resilient mode of operation. In *EUROCRYPT*, volume 5479 of *Lecture Notes in Computer Science*, pages 462–482. Springer, 2009.

[PSP$^+$08] Christophe Petit, François-Xavier Standaert, Olivier Pereira, Tal Malkin, and Moti Yung. A block cipher based pseudo random number generator secure against side-channel key recovery. In *AsiaCCS*, pages 56–65. ACM, 2008.

[PSV15] Olivier Pereira, François-Xavier Standaert, and Srinivas Vivek. Leakage-resilient authentication and encryption from symmetric cryptographic primitives. In *CCS*, pages 96–108. ACM, 2015.

[SPS$^+$22] Yaobin Shen, Thomas Peters, François-Xavier Standaert, Gaëtan Cassiers, and Corentin Verhamme. Triplex: an efficient and one-pass leakage-resistant mode of operation. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(4):135–162, 2022.

[SPY13] François-Xavier Standaert, Olivier Pereira, and Yu Yu. Leakage-resilient symmetric cryptography under empirically verifiable assumptions. In *CRYPTO (1)*, volume 8042 of *Lecture Notes in Computer Science*, pages 335–352. Springer, 2013.

[Sta19] François-Xavier Standaert. Towards and open approach to secure cryptographic implementations (invited talk). In *EUROCRYPT I*, volume 11476 of *Lecture Notes in Computer Science*, pages xv, https://www.youtube.com/watch?v=KdhrsuJT1sE, 2019.

[UXT$^+$22] Rei Ueno, Keita Xagawa, Yutaro Tanaka, Akira Ito, Junko Takahashi, and Naofumi Homma. Curse of re-encryption: A generic power/EM analysis on post-quantum KEMs. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2022(1):296–322, 2022.

[YS13] Yu Yu and François-Xavier Standaert. Practical leakage-resilient pseudorandom objects with minimum public randomness. In *CT-RSA*, volume 7779 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 2013.

[YSPY10] Yu Yu, François-Xavier Standaert, Olivier Pereira, and Moti Yung. Practical leakage-resilient pseudorandom generators. In *CCS*, pages 141–151. ACM, 2010.