Authenticated Encryption with Nonce Misuse and Physical Leakage: Definitions, Separation Results & First Construction
(Extended Abstract)

Chun Guo*[⋆], Olivier Pereira*, Thomas Peters*, François-Xavier Standaert*

* ICTEAM/ELEN/Crypto Group, UCLouvain, Louvain-la-Neuve, Belgium
chun.guo.sc@gmail.com,olivier.pereira@uclouvain.be,thomas.peters@uclouvain.be,francois-xavier.standaert@uclouvain.be

Abstract. We propose definitions of authenticated encryption (AE) schemes that offer security guarantees even in the presence of nonce misuse and side-channel information leakage. This is part of an important ongoing effort to make AE more robust, while preserving appealing efficiency properties. Our definitions consider an adversary enhanced with the leakage of all the computations of an AE scheme, together with the possibility to misuse nonces, be it during all queries (in the spirit of misuse-resistance), or only during training queries (in the spirit of misuse-resilience recently introduced by Ashur et al.). These new definitions offer various insights on the effect of leakage in the security landscape. In particular, we show that, in contrast with the black-box setting, leaking variants of INT-CTXT and IND-CPA security do not imply a leaking variant IND-CCA security, and that leaking variants of INT-PTXT and IND-CCA do not imply a leaking variant of INT-CTXT. They also bring a useful scale to reason about and analyze the implementation properties of emerging modes of operation with different levels of leakage-resistance, such as proposed in the ongoing NIST lightweight cryptography competition. We finally propose the first instance of mode of operation that satisfies our most demanding definitions.

1 Introduction

Authenticated encryption (AE) has become the de facto standard primitive for the protection of secure communications, by offering a robust and efficient alternative to the combination of encryption and MACs, a combination that is challenging enough to have been the source of security issues in numerous high-profile systems [2,18,29]. This effort towards robustness has been intensely pursued and, as a result, a number of strengthened requirements for AE schemes have been proposed in the literature.

A first focus has been on reducing functional requirements, in order to protect users from their failure to provide appropriate inputs to the system. The

[⋆] Corresponding author.

typical requirement of using random IVs has been lowered to the requirement of providing unique nonces. Further efforts have then been made to reduce the impact of a repeated nonce, by requiring that such a repetition only makes it possible to recognize the repetition of a message, which is the strict minimal consequence. These considerations led Rogaway and Shrimpton to define the central notion of misuse-resistant nonce-based AE [32], which goes even one step further, by requiring ciphertexts to be indistinguishable from random strings. Satisfying this notion of misuse-resistance is extremely appealing, as it goes as far as possible in protecting users from their own mistakes or from devices offering poor sources of randomness. However, it comes with a significant memory penalty (two successive passes are needed to perform encryption) and, as we will argue, may also be infeasible to achieve in the presence of many natural types of leakage (e.g., based on the power consumption or electromagnetic radiation of an implementation). More recently, Ashur et al. [4] proposed the relaxed security notion of misuse-resilience, which requires that nonce misuse does not have an impact on the security of messages encrypted with a fresh nonce. This notion can be satisfied by much more efficient schemes, and we will show that it is also compatible with the side-channel attack scenarios mentioned above.

A second line of efforts aims at protecting from weaknesses that implementers could introduce in an AE scheme, by creating observable behaviors that are not part of its specifications. One type of implementation weakness comes from the decryption of invalid ciphertexts [13,3,24,27,4]. While security models usually assume that the decryption of an invalid ciphertext returns an error signal, the reality is often different, and some implementations return different messages depending on the step at which decryption fails, or would even go as far as releasing the partially decrypted message to the adversary, either explicitly, or by treating it as public garbage. Another source of weakness coming from implementation is the possibility of side-channel attacks [10,5,11]. Here, the attacker does not (only) exploit explicit software messages, but extracts information from side-effects such as the computation time, the power consumption, or the electromagnetic radiation of the device performing cryptographic operations. In this context, the previous focus on decryption failures must be broadened, as side-channel leakage happens at encryption and decryption times, and happen at decryption time whether a ciphertext is valid or not.

What can be achieved in the presence of leakage of course depends on the type, the permanence and the amount of leakage granted to the adversary. As far as the type is concerned, we separate scalar leakage (like timing) that allows so-called univariate attacks and vector leakage (like the aforementioned power consumption or electromagnetic radiation) that allows so-called multivariate attacks. As far as the permanence is concerned, we separate between full leakage (that allows leakage during all interactions with the device) and partial leakage that excludes the leakage of some interactions. This leads us to define a first taxonomy of leakage as Vector & Full (VF), Scalar & Full (SF), Vector and Partial (VP), Scalar & Partial (SP). In the rest of this paper, we are concerned with the strongest category of VF leakage.

As for the amount of information leaked, it is in general hard to quantify and highly depends on the implementation and measurement devices that are at hand. In this respect, our starting observation is that the leakage of all secrets makes confidentiality-preserving cryptography impossible, but the full protection against leakage at the implementation level brings us back to a situation in which we put a lot of pressure on implementers, who we must completely trust to limit the leakage. Furthermore, even in that case, this will come at high cost in terms of extra computation time, energy, or circuit area, since strong protections against side-channel attacks (especially in the case of VF leakage) typically increase the "code size $\times$ cycle count" metric by 2 or 3 orders of magnitude compared to a non-protected implementation [7,19].

This state-of-the-art motivated the design of authentication, encryption and AE schemes allowing "leveled implementation" (or implementation in the leveled leakage setting). By leveled implementation, we mean that different levels of security are required for different parts of the computations: some computations must be well protected, while a weaker protection would be sufficient for others. As put forward in [30,10,11], this setting usually allows lower cost or more efficient implementation with symmetric building blocks.

The design of such schemes being guided by the security definitions to target, it can be viewed as a tradeoff between the pressure on implementers to limit the leakage and the pressure on the modes to deal with the remaining leakage. Our contributions therefore aim at defining security targets and modes of operation with an effective balance between leakage reduction at the implementation level and leakage-resistance at the cryptographic mode level, in order to reach AE with high physical security and a minimum implementation cost.

Contributions. Our main contributions target deterministic (nonce-based) authenticated encryption with associated data (AEAD) [31]. We:

(i) Define confidentiality and integrity notions in the presence of nonce misuse and leakage. Our security notions capture leakage in encryption and decryption and allow the computation of the "challenge ciphertexts" to leak. Several variants are explored and compared.
(ii) Identify the strongest form of security that an AEAD can offer to protect messages in the presence of nonce misuse and VF leakage, which we call $\mathsf{AEML\text{-}VF}$ security, as a combination of black-box misuse-resistance, ciphertext integrity with VF leakage and misuse-resistance and CCA security with VF leakage, and misuse-resilience. We also argue why CCA security with VF leakage and misuse-resistance cannot be achieved.

Inspired by the misuse-resistance vs. misuse-resilience terminology, we denote as leakage-resistant the modes that aim to cope with full leakage, and as leakage-resilient the modes that aim to cope with partial leakage. The resulting set of definitions is relatively large, but we believe that it offers a valuable resource.

– An increased number of AE schemes claim to offer some form of leakage-resistance: among the candidates to the NIST lightweight cryptographic algorithm competition, we may mention Ascon, ISAP, Spook and others.[1] Our structured set of definitions should help comparing the claimed guarantees, which we would view as an interesting future work.

- Applications may raise different risks in terms of side-channel attacks, resulting in the adoption of different security requirements. For instance, we may have an encrypting device exposed to side-channel attacks, while decryption would only happen in a very well protected environment, which may suggest the adoption of a security definition that excludes decryption leakages. And aiming at weaker requirement could very well lead to the adoption of more efficient schemes.
- Exploring relations between definitions may sometimes offer implications that would support simpler security proofs. We compare all our definition variants, and demonstrate that none of our security notions are equivalent.

Security definition. Our definition of AEML-VF security (written more simply as AEML security when VF is understood) is a combination of three requirements: (i) The AE scheme must be misuse-resistant (MR) in the black-box setting (without leakage), in the usual sense of Rogaway and Shrimpton [32]. (ii) The AE scheme must offer CIML2 security, which is a natural extension of ciphertext integrity and nonce misuse resistance in the presence of (full) leakages, as introduced by Berti et al. [10,11]. (iii) The AE scheme must offer CCAmL2 security, which is an extension of CCA security with misuse-resilience in the presence of (full) leakage that we propose here. Misuse-resilience and full leakage are reflected by the small m and large L in these notations.

The first requirement is there to ensure that, for someone who does not have access to leakage, an AEML scheme is also a traditional MR AE scheme.

In the presence of leakage, we unfortunately cannot just easily extend the Rogaway-Shrimpton definition of MR AE in any natural way that would uniformly combine confidentiality and integrity. Indeed, their definition requires that ciphertexts look random as soon as they are produced from a fresh (nonce, message) pair. But defining the leakage function corresponding to the generation of such random-looking ciphertexts is difficult, since the very definition of this function is implementation-dependent. In order to avoid this caveat, we therefore turn back to the original definitional approach for AE security, as a combination of confidentiality and ciphertext integrity, which we gradually extend to the leakage world in the presence of nonce misuse. Such a combination turns out to be especially relevant in the leakage setting where ensuring confidentiality and integrity may benefit from different types of physical assumptions.

The extension of INT-CTXT (the hardness to forge a fresh ciphertext that would pass decryption) to the setting of misuse and leakage has been recently proposed as the CIML2 notion [11]. (The same notion excluding decryption leakage is denoted as CIML1 [10]). This extension can be viewed as natural as it provides the adversary with full nonce misuse capabilities (as in the black box

[1] https://csrc.nist.gov/projects/lightweight-cryptography/

setting) and leakage from all the computations performed in encryption and decryption. Furthermore and as will be detailed next, it can be obtained under quite mild leakage assumptions, making it a particularly desirable property to reach in practice.

It would be tempting to complete this picture with an extension of CPA security to the misuse and leakage setting, e.g., based on the notions defined/used in [33,10]. However, this leads to guarantees that are weaker than what we can hope to achieve. While INT-CTXT + IND-CPA implies (the desired) IND-CCA security in the black box setting [8], we show that this is not true anymore when leakage enters the picture: the implication towards an extension of CCA security with leakage does not hold, mainly because it does not capture the risks associated to decryption leakage.

With this difficulty in mind, we introduce the notion of CCAmL2 security as an extension of CCA security that also offers nonce misuse-resilience [4] in the presence of full leakage: as long as the nonce used in the test query is fresh, confidentiality must hold. Besides the aforementioned separation, we also show that leaking variants of INT-PTXT (the hardness to forge a ciphertext that would decrypt to a fresh message) and CCAmL2 do not imply CIML2 security: the alternate definition of AE as INT-PTXT and IND-CCA security [25] does not suffice in the leaking setting either. By the above, we propose to use CCAmL2 in combination with CIML2 (and MR) to define AEML-VF security.

Finally, the reason of our focus on nonce misuse-resilience for CCAmL2 security, rather than on the stronger requirement of nonce misuse-resistance, is due to the nature of VF leakage. Concretely, if an implementation of an AE scheme processes a message block-by-block, as it is standard, the leakage happening during the processing of the first blocks will only depend on these blocks, and not on all blocks. So, if an adversary asks for an encryption of two messages that have identical first blocks (but differ otherwise), using a single nonce, the leakage of the computation associated to these first blocks will be highly similar, something that can be easily observed and trivially contradicts misuse-resistance. Nonce misuse-resilience is then the natural form of protection against misuse that can be aimed for in the VF setting. Note that this argument may not hold for scalar leakage (i.e., in the SF or SP settings): if there is a single scalar leaked for a full message encryption process, then that scalar may depend on the full message, not just on its first blocks. It also does not hold in the VP setting since, in that case, the security game does not provide the adversary with the challenge leakage that can help her to distinguish.

New mode of operation. Based on the above definitions, we propose a new mode of operation for which the driving motivation, and our choice of leakage models, is to push towards the most effective balance between the pressure on implementers and the pressure on designers of modes (i.e., trading more complicated leakage-resistant modes for simpler implementations).

Traditional (non leakage-resistant) encryption modes, when intended to be used in a VF leakage setting, require implementers to offer an implementation that can at least withstand so-called Differential Power Analysis attacks (DPAs).

Informally, DPAs are the most commonly exploited side-channel attacks and take advantage of the leakage about a secret (e.g., key) obtained from computations based on different inputs (e.g., [26,15,14] and follow ups). A DPA reduces the computational secrecy of the state manipulated by a device at a rate that is exponential in the number of leakage traces, by combining the information of these different inputs (e.g., plaintexts).

Leakage-resistant modes have the potential to considerably lower the costs of physical protection (in terms of time/energy/code size needed to perform an encryption) by removing, to a large extent, DPAs from the attack surface (mostly via consistently refreshing internal secrets, so that it's impossible to collect multiple traces), leaving the adversary with the more challenging task of exploiting leakage with Simple Power Analysis attacks (SPAs). SPAs are side-channel attacks taking advantage of the leakage of a single input, possibly measured multiple times to reduce the measurement noise, and therefore correspond to a minimum threat that targets the unavoidable manipulation of the messages to encrypt/decrypt. SPA protection is considerably cheaper than DPA protection (see [16]), and it is expected that the overhead coming from the use of a more demanding encryption mode (e.g., requiring more block cipher calls per message block) can be compensated by the cheaper physical protections against SPA.

With the above spirit, we define a mode of operation FEMALE (for Feedback-based Encryption with Misuse, Authentication and LEakage) that is based on a block cipher and a hash function and offers AEML-VF security. As encouraged by our composite definitions, the security proofs of FEMALE are obtained under physical assumptions that differ depending on whether we target confidentiality or integrity guarantees. In this respect, it is important to note that our definition of AEML security allows expressing gradual security degradations in the sense (present in our modes) that a weakly protected component that would leak too much to satisfy the assumptions that we need for CCAmL2 security may still offer CIML2 security. In other words, CIML2 and CCAmL2 should be seen as gradual improvements that modes of operation can bring to better cope with leakage, with a risk of non-negligible adversarial advantages (especially for CCAmL2) that are at the same time inherent to physical security issues, but also significantly reduced compared to modes of operations ignoring physical leakage in their design.

In the extended version of this work, we additionally propose a more efficient (one-pass) mode of operation, that ensures CIML2 and CCAmL2 without black box MR, a combination that we denoted as AEmL [20].

Related works. Recently, Barwell et al. [5] introduced notions of misuse-resistant and leakage-resilient AE, and proposed modes of operation satisfying their definitions. We will refer to this work as BMOS, by the initial of its authors. The BMOS definition captures leakage-resilient AE security as follows (we just focus on encryption queries for simplicity): they first follow the Rogaway-Shrimpton strategy by challenging the adversary to distinguish between non-leaking real or random encryption oracles, then augment the power of the adversary by giving him access to a leaking encryption oracle that cannot be queried with inputs identical to those of the non-leaking oracles. As a result, it is impossible to win their game by exploiting a leakage that would reveal information about the messages that the AE scheme is supposed to hide: leakage is excluded from the "challenge" queries. In our terminology, BMOS focuses on the VP (Vector & Partial) leakage model, which we reflect with a small l in our notations.

As a result of the weaker VP leakage model, BMOS can realistically require misuse-resistance to hold for confidentiality, and not only misuse-resilience (i.e., $\mathsf{CCAMl1}$, $\mathsf{CCAMl2}$), which is in line with our previous discussions. As such, the BMOS work can be viewed as dual to ours: we consider full leakage (i.e., leakage-resistance), and as a consequence have to exclude full misuse-resistance; they consider partial leakage (i.e., leakage-resilience), which makes misuse-resistance possible. On the positive side, the BMOS authors show that their definition is compatible with strong composition results. Yet, the VP model may be insufficient in many practical cases: an implementation that leaks plaintexts in full during encryption may still satisfy the BMOS security definition. By contrast, such an implementation would be considered insecure in the VF model.

Our $\mathsf{CCAmL2}$ definition builds on the notion of misuse-resilience introduced by Ashur, Dunkelman and Luykx [4]. The actual definitions and their motivations are quite different, though. Ashur et al. introduce misuse-resilience to offer a finer grained evaluation of several standard AE schemes that are not misuse-resistant. They do not consider side-channel leakage. In contrast, this type of leakage is the central concern of our definitions, and is our motivation for departing from traditional misuse-resistance in the VF leakage model.

Eventually, we mention the line of works about "after-the-fact" leakage which is complementary to ours [23] and allows the adversary to obtain leakage information after the challenge ciphertext. While the latter is meaningful in certain scenarios (e.g., in the context of a cold boot attack [22], the adversary could first see the encrypted disk – hence getting access to the ciphertext – and then try to design a method of measuring the memory for the purpose of decrypting this ciphertext), it still excludes information leakage during the challenge phase, as will be available in the context of a side-channel attack based on power consumption leakage, which is our main concern here.

2 Preliminaries

Throughout the paper n denotes the security parameter.

2.1 Notations

Adversary. We denote by a $(q_1, \ldots, q_\omega, t)$-bounded adversary a probabilistic algorithm that has access to ω oracles, can make at most q_i queries to its i-th oracle, and can perform computations bounded by running time t. For algorithms that have no oracle to access, we simply call them t-bounded. In this paper, we use subscripts to make a clear distinction between the number of queries to different oracles: the number of queries to the (authenticated) encryption oracle, decryption oracle, and leakage oracle L are denoted by q_e, q_d, and q_l respectively. For example, a (q_e, q_d, q_l, t)-bounded adversary runs in time t, makes q_e and q_d queries to the encryption and decryption oracles of the Authenticated Encryption with Associated Data (AEAD) scheme respectively, and makes q_l additional queries to the leakage oracle L. In all cases, the access to leakage happens through oracle queries. This captures the fact that the leakage function is typically unknown, and that it is only queried by running a physical computation process (rather than being emulated).

Leaking algorithm. For an algorithm Algo, a leaking version 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 returns the outputs of both Algo and $\mathsf{L}_{\mathsf{algo}}$ which all take the same input.

2.2 Definitions of primitives

We will focus on authenticated encryption with the following formalism.

Definition 1 (Nonce-Based AEAD [31]). *A* nonce-based authenticated encryption scheme with associated data *is a tuple* $\mathsf{AEAD} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *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 selected from $\mathcal{K}$, a nonce value from $\mathcal{N}$, some associated data selected from $\mathcal{AD}$, and a message from $\mathcal{M}$ to a ciphertext in $\mathcal{C}$.*

– $\mathsf{Dec} : \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{C} \to \mathcal{M} \cup \{\perp\}$ *deterministically maps a key from $\mathcal{K}$, a nonce from $\mathcal{N}$, associated data from $\mathcal{AD}$, and a ciphertext from $\mathcal{C}$, to a message in $\mathcal{M}$ or to a special symbol $\perp$ if integrity verification fails.*

The sets $\mathcal{K}, \mathcal{N}, \mathcal{AD}, \mathcal{M}, \mathcal{C}$ are completely specified by n. Given a key $k \leftarrow \mathsf{Gen}(1^n)$, $\mathsf{Enc}_k(N, A, M) := \mathsf{Enc}(k, N, A, M)$ and $\mathsf{Dec}_k(N, A, M) := \mathsf{Dec}(k, N, A, M)$ are deterministic functions whose implementations may be probabilistic. Furthermore, the input length of Enc publicly determines its output length.

The correctness is defined in the natural way (see [31]). Since we only focus on (correct) nonce-based authenticated encryption with associated data in this paper, we will simply refer to it as *authenticated encryption*.

We recall the definition of Misuse-Resistance (MR) formalized in [32].

Definition 2 (MR). *A nonce-based authenticated encryption scheme with associated data $\mathsf{AEAD} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is $(q_e, q_d, t, \varepsilon)$ misuse resistant for a security parameter n if, for all (q_e, q_d, t)-bounded adversaries $\mathcal{A}$,*

$$\left|\Pr\left[k \xleftarrow{\$} \mathsf{Gen}(1^n) : \mathcal{A}^{\mathsf{Enc}_k, \mathsf{Dec}_k}(1^n) \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{\$, \perp}(1^n) \Rightarrow 1\right]\right| \leq \varepsilon,$$

where $\$(N, A, M)$ outputs and associates a fresh random ciphertext $C \leftarrow \mathcal{C}$ of appropriate length to fresh inputs, and the associated C otherwise, and $\perp(N, A, C)$ outputs $\perp$ except if C was associated to (N, A, M) for some message M, in which case it returns M.

3 AE with full vectorial leakage

In order to define AEML security, our proposed security notion for authenticated encryption in the full vectorial leakage setting in the presence of nonce misuse, we start by extending the existing black-box security notions to the leakage setting. Surprisingly, the combination of our strongest extensions of confidentiality and integrity is separated from *any other* combinations unlike the situation without misuse and leakages. This motivates our definition to be at least as secure as this strongest combination.

3.1 Variants of black-box notions with misuse and leakages

We first adapt the $\mathsf{IND\text{-}CPA}$ and the $\mathsf{IND\text{-}CCA}$ confidentiality notions of nonce-based authenticated encryption in the setting of nonce misuse and leakages. We focus then on the extension of the $\mathsf{INT\text{-}PTXT}$ and $\mathsf{INT\text{-}CTXT}$ integrity notions.

Confidentiality Contrary to existing confidentiality notions in a leaking setting (e.g., [5]), our definition includes leakages during encryption and decryption even on the challenge ciphertext(s). We first focus on security against chosen-ciphertext attacks with misuse-resilience and leakages, denoted $\mathsf{CCAmL2}$. Then, we derive the weaker notion of security against chosen-plaintext attacks with misuse-resilience and leakages, denoted $\mathsf{CPAmL2}$.

Chosen-ciphertext security with misuse and leakages. To capture $\mathsf{CCAmL2}$ security, as motivated in the introduction, we define the game $\mathsf{PrivK}^{\mathsf{CCAmL2},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$ detailed in Figure 1. This game takes as parameters an adversary $\mathcal{A}$, a nonce-based authenticated encryption AEAD and a (possibly probabilistic) leakage function pair $\mathsf{L} = (\mathsf{L}_{\mathsf{enc}}, \mathsf{L}_{\mathsf{dec}})$ resulting from the implementation of the scheme. During $\mathsf{PrivK}^{\mathsf{CCAmL2},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$, the adversary $\mathcal{A}$ selects and submits a tuple $(N_{ch}, A_{ch}, M^0, M^1)$, where the nonce N_{ch} must be fresh and the messages M^0, M^1 must have identical block length. It then receives an encryption of (N_{ch}, A_{ch}, M^b) and the associated leakage, and must guess the bit b. All along this game $\mathcal{A}$ is also granted unbounded and adaptive access to three types of oracles: LEnc, a leaking encryption oracle; LDec, a leaking decryption oracle; and $\mathsf{L}_{\mathsf{decch}}$, a challenge decryption leakage oracle that provides the leakage of the decryption process, but not the resulting plaintext.

Overall, this definition follows the general pattern of CCA security. In terms of misuse-resilience, it only forbids the adversary to reuse a nonce N_{ch} in its challenge query. This captures real situations where, for instance, a counter providing the nonce has been unintentionally reset or shifted. As long as the counter recovers increments and provides fresh nonce values, the security of the challenges should remain unaltered even if the previous encryptions leaked. In terms of leakage, both the encryption and the decryption oracles leak (hence the "2" of $\mathsf{CCAmL2}$ for the two leaking oracles), including during the challenge query. We go one step further with the $\mathsf{L}_{\mathsf{decch}}$ oracle, which offers the leakages corresponding to the decryption of the challenge ciphertext (but not the corresponding plaintext, as it would offer a trivial win). This addition captures the fact that the adversary may be allowed to observe the decryption of this challenge ciphertext

$\mathsf{PrivK}^{\mathsf{CCAmL2},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}(1^n)$ is the output of the following experiment:

Initialization: generates a secret key $k \leftarrow \mathsf{Gen}(1^n)$ and sets $\mathcal{E} \leftarrow \emptyset$

Pre-challenge queries: $\mathcal{A}^{\mathsf{L}}$ gets adaptive access to $\mathsf{LEnc}(\cdot,\cdot,\cdot)$ and $\mathsf{LDec}(\cdot,\cdot,\cdot)$

(1) $\mathsf{LEnc}(N, A, M)$ computes $C \leftarrow \mathsf{Enc}_k(N, A, M)$ and $\mathsf{leak_e} \leftarrow \mathsf{L_{enc}}(k, N, A, M)$ updates $\mathcal{E} \leftarrow \mathcal{E} \cup \{N\}$ and finally returns $(C, \mathsf{leak_e})$

(2) $\mathsf{LDec}(N, A, C)$ computes $M \leftarrow \mathsf{Dec}_k(N, A, C)$ and $\mathsf{leak_d} \leftarrow \mathsf{L_{dec}}(k, N, A, C)$ and returns $(M, \mathsf{leak_d})$ — we stress that $M = \bot$ may occur

Challenge query: on a single occasion $\mathcal{A}^{\mathsf{L}}$ submits a tuple $(N_{\mathsf{ch}}, A_{\mathsf{ch}}, M^0, M^1)$

If M^0 and M^1 have different (block) length or $N_{\mathsf{ch}} \in \mathcal{E}$ return $\bot$

Else compute $C^b \leftarrow \mathsf{Enc}_k(N_{\mathsf{ch}}, A_{\mathsf{ch}}, M^b)$ and $\mathsf{leak}^b_{\mathsf{e}} \leftarrow \mathsf{L_{enc}}(k, N_{\mathsf{ch}}, A_{\mathsf{ch}}, M^b)$ and return $(C^b, \mathsf{leak}^b_{\mathsf{e}})$

Post-challenge queries: $\mathcal{A}^{\mathsf{L}}$ can keep accessing LEnc and LDec with some restrictions but it can also get an unlimited access to $\mathsf{L_{decch}}$

(3) $\mathsf{LEnc}(N, A, M)$ returns $\bot$ if $N = N_{\mathsf{ch}}$ otherwise computes $C \leftarrow \mathsf{Enc}_k(N, A, M)$ and $\mathsf{leak_e} \leftarrow \mathsf{L_{enc}}(k, N, A, M)$ and finally returns $(C, \mathsf{leak_e})$

(4) $\mathsf{LDec}(N, A, C)$ returns $\bot$ if $(N, A, C) = (N_{\mathsf{ch}}, A_{\mathsf{ch}}, C^b)$ otherwise computes $M \leftarrow \mathsf{Dec}_k(N, A, C)$ and $\mathsf{leak_d} \leftarrow \mathsf{L_{dec}}(k, N, A, C)$ and returns $(M, \mathsf{leak_d})$

(5) $\mathsf{L_{decch}}$ outputs the leakage trace $\mathsf{leak}^b_{\mathsf{d}} \leftarrow \mathsf{L_{dec}}(k, N_{\mathsf{ch}}, A_{\mathsf{ch}}, C^b)$ of the challenge

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

Fig. 1: The $\mathsf{PrivK}^{\mathsf{CCAmL2},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}(1^n)$ game.

through side-channels, which might be valuable in applications such as secure bootloading or firmware update with a device controlled by $\mathcal{A}$ [28] (see [11] for more discussion). We let the adversary query the $\mathsf{L_{decch}}$ oracle multiple times, as leakages can be non-deterministic (e.g., contain noise), and $\mathcal{A}$ may benefit from observing the leakages from multiple decryptions of the same plaintext. However, a single leakage is provided for the encryption of the challenge message, as we require the encrypting party to be nonce-respecting for that query: as a result, the challenge encryption process can happen only once.

We focus on a single challenge definition but the multi-challenge setting is treated in the extended version of our work, where we establish their equivalence.

Definition 3 (CCAmL2). *A nonce-based authenticated encryption with associated data* $\mathsf{AEAD} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with leakage function pair* $\mathsf{L} = (\mathsf{L_{enc}}, \mathsf{L_{dec}})$ *is* $(q_e, q_d, q_c, q_l, t, \varepsilon)$*-CCAmL2 secure for a security parameter* n *if, for every* (q_e, q_d, q_c, q_l, t)*-bounded adversary* $\mathcal{A}^{\mathsf{L}}$*,*[2] *we have:*

$$\left| \Pr\left[\mathsf{PrivK}^{\mathsf{CCAmL2},0}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}(1^n) \Rightarrow 1\right] - \Pr\left[\mathsf{PrivK}^{\mathsf{CCAmL2},1}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}(1^n) \Rightarrow 1\right] \right| \leq \varepsilon,$$

where the adversary $\mathcal{A}^{\mathsf{L}}$ *makes at most* q_e *leaking encryption queries,* q_d *leaking decryption queries,* q_c *challenge decryption leakage queries and* q_l *leakage evaluation queries on arbitrarily chosen keys.*

[2] The notation of $\mathcal{A}^{\mathsf{L}}$ indicates that the adversary may query L on chosen inputs including *chosen keys* selected and known by $\mathcal{A}$.

We will sometimes refer to $\mathsf{CCAmL2}^*$ as a weakened version of $\mathsf{CCAmL2}$ where we drop the challenge decryption leakage oracle $\mathsf{L}_{\mathsf{decch}}$ from the game of Figure 1.

Chosen-plaintext security with misuse and leakages. Derived from $\mathsf{CCAmL2}$, $\mathsf{CPAmL2}$ is defined by a game $\mathsf{PrivK}^{\mathsf{CPAmL2},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$ which is exactly as $\mathsf{PrivK}^{\mathsf{CCAmL2},b}_{\mathcal{A}^{\mathsf{L}},\mathsf{AEAD}}$ except that we remove $\mathcal{A}$'s access to the leaking decryption oracle LDec in Figure 1 (Items 2,4). Yet, $\mathcal{A}$ is still able to get challenge decryption leakages $\mathsf{leak}^b_{\mathsf{d}}$—this corresponds to settings in which a passive adversary tries to break confidentiality while being able to observe leakages of encryption and decryption operations. In section 4.1, we will also introduce the $\mathsf{CPAmL1}$ security notion, which can be seen as the $\mathsf{CPAmL2}$ variant without $\mathsf{L}_{\mathsf{decch}}$, corresponding to situations in which an adversary cannot observe any decryption operation, which could happen in settings where keys are dependent of the communication direction, and the adversary only has physical access to one end of the communication.

Definition 4 (CPAmL2)**.** *A nonce-based authenticated encryption with associated data AEAD* $= (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with leakage function pair* $\mathsf{L} = (\mathsf{L}_{\mathsf{enc}}, \mathsf{L}_{\mathsf{dec}})$ *is* $(q_e, q_c, q_l, t, \varepsilon)$-$\mathsf{CPAmL2}$ *secure for a security parameter n if, for every (q_e, q_c, q_l, t)-bounded adversary $\mathcal{A}$, we have:*

$$\left| \Pr\left[\mathsf{PrivK}^{\mathsf{CPAmL2},0}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}(1^n) \Rightarrow 1\right] - \Pr\left[\mathsf{PrivK}^{\mathsf{CPAmL2},1}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}(1^n) \Rightarrow 1\right] \right| \leq \varepsilon,$$

where the adversary $\mathcal{A}^{\mathsf{L}}$ makes at most q_e leaking encryption queries, q_c challenge decryption leakage queries and q_l leakage evaluation queries on chosen keys.

Integrity We next adopt the natural and strong extensions of $\mathsf{INT\text{-}CTXT}$ and $\mathsf{INT\text{-}PTXT}$ to nonce-misuse resistance and (full) leakages in encryption and decryption. The $\mathsf{INT\text{-}CTXT}$ extension, called Ciphertext Integrity with Misuse and Leakages, noted $\mathsf{CIML2}$, comes from [11] and is an earlier proposal $\mathsf{CIML1}$ [10] extended with with decryption leakage. Based on this definition, we propose here the corresponding extension of $\mathsf{INT\text{-}PTXT}$ security, which we call $\mathsf{PIML2}$.

Definition 5 (CIML2, PIML2)**.** *An authenticated encryption* $\mathsf{AEAD} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with leakage function pair* $\mathsf{L} = (\mathsf{L}_{\mathsf{enc}}, \mathsf{L}_{\mathsf{dec}})$ *provides* $(q_e, q_d, q_l, t, \varepsilon)$*-ciphertext (resp. plaintext) integrity with nonce misuse and leakages for security parameter n if, for all (q_e, q_d, q_l, t)-bounded adversaries $\mathcal{A}^{\mathsf{L}}$, we have:*

$$\Pr\left[\mathsf{PrivK}^{\mathsf{CIML2}}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}(1^n) \Rightarrow 1\right] \leq \varepsilon,$$
$$\textit{(resp.} \Pr\left[\mathsf{PrivK}^{\mathsf{PIML2}}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}(1^n) \Rightarrow 1\right] \leq \varepsilon\textit{)},$$

where the security game $\mathsf{PrivK}^{\mathsf{CIML2}}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$ *(resp.* $\mathsf{PrivK}^{\mathsf{PIML2}}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$*) is defined in the left part of Table 1 (resp. right part) when $\mathcal{A}^{\mathsf{L}}$ makes at most q_e leaking encryption queries, q_d leaking decryption queries and q_l leakage evaluation queries.*

$\mathsf{PrivK}^{\mathsf{CIML2}}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}(1^n)$ experiment	$\mathsf{PrivK}^{\mathsf{PIML2}}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}(1^n)$ experiment
Initialization:	*Initialization*:
1. $k \leftarrow \mathsf{Gen}(1^n)$, $\mathcal{S} \leftarrow \emptyset$	1. $k \xleftarrow{\$} \mathsf{Gen}(1^n)$, $\mathcal{S} \leftarrow \emptyset$
Finalization:	*Finalization*:
1. $(N, A, C) \leftarrow \mathcal{A}^{\mathsf{LEnc}_k,\mathsf{LDec}_k,\mathsf{L}}(1^n)$	1. $(N, A, C) \leftarrow \mathcal{A}^{\mathsf{LEnc}_k,\mathsf{LDec}_k,\mathsf{L}}(1^n)$
2. If $(N, A, C) \in \mathcal{S}$, return 0	2. $M \leftarrow \mathsf{Dec}_k(N, A, C)$
3. If $\mathsf{Dec}_k(N, A, C) = \perp$, return 0	3. If $M = \perp$ or $(A, M) \in \mathcal{S}$, return 0
4. Return 1	4. Return 1
Leaking encryption: $\mathsf{LEnc}_k(N, A, M)$	*Leaking encryption:* $\mathsf{LEnc}_k(N, A, M)$
1. $C \leftarrow \mathsf{Enc}_k(N, A, M)$	1. $C \leftarrow \mathsf{Enc}_k(N, A, M)$
2. $\mathcal{S} \leftarrow \mathcal{S} \cup \{(N, A, C)\}$	2. $\mathcal{S} \leftarrow \mathcal{S} \cup \{(A, M)\}$
3. Return $(C, \mathsf{L}_{\mathsf{enc}}(k, N, A, M))$	3. Return $(C, \mathsf{L}_{\mathsf{enc}}(k, N, A, M))$
Leaking decryption: $\mathsf{LDec}_k(N, A, C)$	*Leaking decryption:* $\mathsf{LDec}_k(N, A, C)$
1. Return $(\mathsf{Dec}_k(N, A, C), \mathsf{L}_{\mathsf{dec}}(k, N, A, C))$	1. Return $(\mathsf{Dec}_k(N, A, C), \mathsf{L}_{\mathsf{dec}}(k, N, A, C))$

Table 1. The CIML2 and PIML2 security games. Both games ask the adversary to forge a fresh (N, A, C) triple, based on inputs received from leaking encryption and decryption oracles. The CIML2 is won as soon as (N, A, C) is valid, while PIML2 also requires that its decryption leads to a fresh pair of message and associated data.

3.2 Overall requirement on AE

We eventually require that a secure AE intended to support nonce misuse and full leakage satisfies the strongest achievable guarantee presented in the paper: an AEML scheme is expected to offer CCAmL2 and CIML2 security, together with being a MR AEAD scheme without leakages. This definition departs from the traditional ones in the black-box setting, which is based on the combination of CPA and INT-CTXT security [8] or CCA and INT-PTXT security [25]. We will actually show that there are important separations between these notions: CCAmL2 + PIML2 + MR $\not\Rightarrow$ CIML2, and CPAmL2 + CIML2 + MR $\not\Rightarrow$ CCAmL2. Furthermore, even if CCAmL2 (resp., CIML2) implies both IND-CCA and CPAmL2 (resp., INT-CTXT and PIML2), the combination of CCAmL2 and CIML2 security does not imply MR, which is therefore a separate requirement.

Definition 6 (AEML-VF). *An AE scheme with security against nonce misuse and full vectorial leakages (denoted as* AEML-VF*) is an AE scheme* AEAD = (Gen, Enc, Dec) *with a leakage function pair* $\mathsf{L} = (\mathsf{L}_{\mathsf{enc}}, \mathsf{L}_{\mathsf{dec}})$ *satisfying the following assertions: (i)* AEAD *is misuse resistant; (ii)* AEAD *is* CIML2 *secure with leakage function* L*; (iii)* AEAD *is* CCAmL2 *secure with leakage function* L.

As indicated above, AEML-VF security will typically be abbreviated as AEML in our paper, since the VF attack setting is understood.

3.3 Separation results

We now explain why the strong security notions of MR, CCAmL2 and CIML2 are needed to define AEML, while one could be tempted to make a definition as a combination of weaker notions, which may be easier to prove. Unfortunately, there is no such equivalence and we show that AEML is strictly stronger than any other combinations, assuming that AEML-secure AEAD exists.

$$\begin{tikzcd}
\boxed{\mathsf{CCAmL2}} \arrow[d, dotted, shift right, "|" marking] & \boxed{\mathsf{CIML2} \wedge \mathsf{CPAmL2}} \arrow[l, "\wedge\mathsf{MR}"', "|" marking] \arrow[d, dotted, shift right] \arrow[dl, dotted, shift right, "|" marking] \\
\boxed{\mathsf{PIML2} \wedge \mathsf{CCAmL2}} \arrow[u, dotted, shift right] \arrow[r, "\wedge\mathsf{MR}"', "|" marking] \arrow[ur, dotted, shift right, "|" marking] & \boxed{\mathsf{CIML2}} \arrow[u, dotted, shift right, "|" marking]
\end{tikzcd}$$

Fig. 2: Relations among notions. Arrows (resp., barred arrows) denote implications (resp., separations). Dotted arrows are trivially implied by other relations.

We summarize even more relations in Figure 2. In contrast to the black-box setting, these relations show that one cannot choose between different ways to achieve AEML since, for instance, $\mathsf{CCAmL2} \wedge \mathsf{PIML2} \not\Leftarrow \mathsf{CPAmL2} \wedge \mathsf{CIML2}$.

$\mathsf{MR} \wedge \mathsf{CPAmL2} \wedge \mathsf{CIML2} \not\Rightarrow \mathsf{CCAmL2}^*$. It is not surprising that MR does not imply $\mathsf{CCAmL2}$ since the leakage function L is absent from the black-box notion. Contrarily, L appears both in $\mathsf{CPAmL2}$ and $\mathsf{CIML2}$. This claim thus says that leakages in decryption may not alter integrity but may alter confidentiality. To reflect this intrinsic separation of the leakage setting, we show that the implication does not even hold for $\mathsf{CCAmL2}^*$ where the challenge decryption leakage oracle $\mathsf{L}_{\mathsf{decch}}$ is unavailable. While the latter leakages are motivated in the context of side-channel attacks [11], is is also quite specific to such attacks. So ignoring it in the separation makes our result stronger and more general.

Theorem 1. *Assuming that there exists an AE scheme which satisfies* MR*,* $\mathsf{CPAmL2}$*, and* $\mathsf{CIML2}$ *in the unbounded leakage setting, then there exists an AE with the same security properties but which fails to achieve* $\mathsf{CCAmL2}$*, even without challenge decryption leakages (i.e.,* $\mathsf{CCAmL2}^*$*).*

The proof utilizes information leaked by invalid decryption queries, which was also exploited in the "protocol leakage" setting [6]. The implication does not hold either when starting from the multiple challenge variant of $\mathsf{CPAmL2}$.

Proof. Let $\mathsf{AEAD} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with leakage function $\mathsf{L} = (\mathsf{L}_{\mathsf{enc}}, \mathsf{L}_{\mathsf{dec}})$ be MR, $\mathsf{CPAmL2}$ with respect to L and $\mathsf{CIML2}$ with respect to L^* as such authenticated encryption exists by assumption. Then we build $\mathsf{AEAD}' = (\mathsf{Gen}', \mathsf{Enc}, \mathsf{Dec})$ with leakage $\mathsf{L}' = (\mathsf{L}'_{\mathsf{enc}}, \mathsf{L}'_{\mathsf{dec}})$ such that, for a fixed message $M^\dagger \in \mathcal{M}$:

$\mathsf{Gen}'(1^n)$: returns $k \leftarrow \mathsf{Gen}(1^n)$ and $k' \leftarrow \mathsf{Gen}(1^n)$;

$\mathsf{L}'_{\mathsf{enc}}((k, k'), N, A, M)$: outputs $(\mathsf{leak}_{\mathsf{e}}, C', \mathsf{leak}_{\mathsf{e}'})$ where $\mathsf{leak}_{\mathsf{e}} = \mathsf{L}_{\mathsf{enc}}(k, N, A, M)$ (comes from the computation of $C \leftarrow \mathsf{Enc}_k(N, A, M)$), the ciphertext $C' = \mathsf{Enc}_{k'}(N, A, M)$ and consequently $\mathsf{leak}_{\mathsf{e}'} = \mathsf{L}_{\mathsf{enc}}(k', N, A, M)$;

$\mathsf{L}'_{\mathsf{dec}}((k, k'), N, A, C)$: outputs $\mathsf{leak}_{\mathsf{d}} = \mathsf{L}_{\mathsf{dec}}(k, N, A, C)$ if $M \neq \perp$ (which comes from the computation of $M \leftarrow \mathsf{Dec}_k(N, A, C)$) and outputs $(\mathsf{leak}_{\mathsf{d}}, C^\dagger, \mathsf{leak}^\dagger_{\mathsf{e}'})$ otherwise, where $\mathsf{leak}_{\mathsf{d}} = \mathsf{L}_{\mathsf{dec}}(k, N, A, C)$, $C^\dagger \leftarrow \mathsf{Enc}_{k'}(N, A, M^\dagger)$ and consequently also $\mathsf{leak}^\dagger_{\mathsf{e}'} = \mathsf{L}_{\mathsf{enc}}(k', N, A, M^\dagger)$.

From a black-box standpoint, k' does not even exist so AEAD' is still MR. Therefore, let us focus on the security notions involving leakages.

$\mathsf{CPAmL2}$. In the $\mathsf{PrivK}^{\mathsf{CPAmL2},b}_{\mathcal{A}',\mathsf{AEAD}',\mathsf{L}'}(1^n)$ game, the adversary $\mathcal{A}'$ does not have access to $\mathsf{L}'_{\mathsf{dec}}$ except from the challenge decryption leakage through $\mathsf{L}'_{\mathsf{decch}}$. But since the challenge ciphertext is valid, $\mathsf{L}'_{\mathsf{decch}} = \mathsf{L}_{\mathsf{decch}}$ which returns $\mathsf{L}_{\mathsf{dec}}(k, N_{\mathsf{ch}}, A_{\mathsf{ch}}, C^b)$. Consequently, an adversary $\mathcal{A}$ in $\mathsf{PrivK}^{\mathsf{CPAmL2},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$ can easily simulate the view of $\mathcal{A}'$, simply by picking $k' \leftarrow \mathsf{Gen}(1^n)$, transmitting all the queries to its own oracles, and adding the encryption leakage $(C', \mathsf{leak}_{\mathsf{e}'})$ if necessary.

$\mathsf{CIML2}$. In the $\mathsf{PrivK}^{\mathsf{CIML2}}_{\mathcal{A}',\mathsf{AEAD}',(\mathsf{L}')^*}(1^n)$ game, the adversary $\mathcal{A}'$ still needs to forge a fresh ciphertext of AEAD with key k while the additional unbounded leakage given by $(\mathsf{L}')^*$ only depends k'. Then, building a reduction to $\mathsf{PrivK}^{\mathsf{CIML2}}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}^*}(1^n)$ is straightforward.

$\neg\mathsf{CCAmL2}^*$. We build a distinguisher $\mathcal{A}'$ against AEAD'. In the security game $\mathsf{PrivK}^{\mathsf{CCAmL2},b}_{\mathcal{A}',\mathsf{AEAD}',\mathsf{L}'}(1^n)$, the adversary queries leaking decryption of $(N_{\mathsf{ch}}, A_{\mathsf{ch}}, C)$ for any chosen $N_{\mathsf{ch}}, A_{\mathsf{ch}}$ and C. If the ciphertext is valid, it receives some $M \neq \perp$ and it sets $(M^0, C^0) = (M, C)$. If not, it receives $(\perp, (\mathsf{leak}_{\mathsf{e}}, C^\dagger, \mathsf{leak}^\dagger_{\mathsf{e}'}))$ from $\mathsf{LDec}_{k,k'}(N_{\mathsf{ch}}, A_{\mathsf{ch}}, C)$ and sets $(M^0, C^0) = (M^\dagger, C^\dagger)$. In the challenge phase, $\mathcal{A}'$ sends $(N_{\mathsf{ch}}, A_{\mathsf{ch}}, M^0, M^1)$ for any distinct M^1 than M^0. Since the pair $(N_{\mathsf{ch}}, A_{\mathsf{ch}})$ has never been queried for (leaking) encryption, $\mathcal{A}'$ does not receive $\perp$. In the answer $\mathsf{LEnc}_k(N_{\mathsf{ch}}, A_{\mathsf{ch}}, M^b)$, $\mathcal{A}'$ gets C^b. If C^b equals the known C^0, $\mathcal{A}'$ outputs 0, otherwise it outputs 1. Obviously the distinction holds with probability 1.

Now, it is easy to see that AEAD' with leakage L' fulfills all the desired requirements of the theorem based on the existence of AEAD. □

$\mathsf{MR} \wedge \mathsf{CCAmL2} \wedge \mathsf{PIML2} \not\Rightarrow \mathsf{CIML2}$. As for the previous assertion, being MR does not say anything about leakages, so not being $\mathsf{CIML2}$ is obviously compatible. The most interesting part comes from $\mathsf{CCAmL2}$ and $\mathsf{PIML2}$ which include leakages. This claim exploits the fact that leakages on repeated queries may degrade ciphertext integrity but neither confidentiality nor plaintext integrity.

Theorem 2. *Assuming that there exists an AE scheme which satisfies* MR*,* $\mathsf{CCAmL2}$*, and* $\mathsf{PIML2}$ *in the unbounded leakage model, then there exists an AE which satisfies the same security properties but which fails to achieve* $\mathsf{CIML2}$ *(even if not in the unbounded leakage model).*

The proof is available in the full version. It proceeds by building a $\neg\mathsf{CIML2}$ scheme AEAD' from an $\mathsf{MR} \wedge \mathsf{CCAmL2} \wedge \mathsf{PIML2}$ scheme AEAD. An interesting feature is that this counterexample AEAD' preserves the tidiness of AEAD, that is, the property that $\mathsf{Enc}_k(N, A, \mathsf{Dec}_k(N, A, C)) = C$ when $\mathsf{Dec}_k(N, A, C) \neq \perp$. This deviates from Bellare and Namprempre's well-known approach for establishing $\mathsf{INT\text{-}PTXT} \not\Rightarrow \mathsf{INT\text{-}CTXT}$, which did utilize non-tidy counterexamples [8]. It is possible in our case due to the presence of leakages.

4 Completing the definitions' zoo

To give a complete picture of the different security flavors of AE with misuse-resistance or resilience and full vectorial leakages, we list all the security definitions that can be derived from our confidentiality and integrity notions. We then study their relations which may be useful in order to guide future designs with relaxed requirements (e.g., in order to reach better performances). It shows that, apart from the obvious implications between the different flavors of confidentiality (resp., integrity), all the notions are separated from each other. In this section we concentrate on the single challenge notions. The extension to the multi-challenge setting is discussed in the extended version of our work.

4.1 Security definition list (single challenge setting)

The $\mathsf{CCAmL2}$ security game $\mathsf{PrivK}^{\mathsf{CCAmL2},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$ is defined in Section 3.1, Figure 1. By dropping some accesses to the distinct oracles of this game, we naturally derive other confidentiality notions. For instance $\mathsf{CPAmL2}$ is defined by removing items (2) and (4) from the security game. By doing similar modifications, we can define different integrity notions from the $\mathsf{CIML2}$ security game $\mathsf{PrivK}^{\mathsf{CIML2}}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$ defined in Section 2.2, Table 1. We next formalize these variants.

Prefix-suffix definitions. In all the notions derived from $\mathsf{CCAmL2}$ and $\mathsf{CIML2}$ we only focus on those capturing full leakages (partial leakage is covered by BMOS). Therefore all the definitions below keep the large L in their notation. This leads us to consider 16 different notions denoted as "$\mathsf{pre\text{-}suf}$" with prefix $\mathsf{pre} \in \{\mathsf{CCA}, \mathsf{CPA}, \mathsf{CI}, \mathsf{PI}\}$ and suffix $\mathsf{suf} \in \{\mathsf{ML2}, \mathsf{ML1}, \mathsf{mL2}, \mathsf{mL1}, \mathsf{L2}, \mathsf{L1}\}$: a large "$\mathsf{M}$" corresponds to misuse resistance, a small "m" corresponds to misuse-resilience and no "M/m" means that the security game is nonce-respecting (which only restricts leaking encryption queries).

Zoo of confidentiality notions. For $\mathsf{pre} \in \{\mathsf{CCA}, \mathsf{CPA}\}$ we obtain the following 8 notions, by starting from $\mathsf{CCAmL2}$ and by removing one security layer at a time:

$$\mathsf{CCAmL2} \to \mathsf{CCAmL1}, \mathsf{CCAL2}, \mathsf{CPAmL2} \to \mathsf{CPAmL1}, \mathsf{CPAL2}, \mathsf{CCAL1} \to \mathsf{CPAL1}.$$

Definition 7. *A nonce-based authenticated encryption with associated data* $\mathsf{AEAD} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with leakages* $\mathsf{L} = (\mathsf{L}_{\mathsf{enc}}, \mathsf{L}_{\mathsf{dec}})$ *is* $(q_{\mathsf{pre\text{-}suf}}, q_l, t, \varepsilon)$*-*$\mathsf{pre\text{-}suf}$ *secure for a security parameter n if, for every* $(q_{\mathsf{pre\text{-}suf}}, q_l, t)$*-bounded adversary* $\mathcal{A}^{\mathsf{L}}$*, we have* $\mathsf{pre} \in \{\mathsf{CCA}, \mathsf{CPA}\}$ *and:*

$$\left| \Pr\left[\mathsf{PrivK}^{\mathsf{pre\text{-}suf},0}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}(1^n) \Rightarrow 1\right] - \Pr\left[\mathsf{PrivK}^{\mathsf{pre\text{-}suf},1}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}(1^n) \Rightarrow 1\right] \right| \leq \varepsilon,$$

where the adversary $\mathcal{A}^{\mathsf{L}}$ *makes at most* $q_{\mathsf{pre\text{-}suf}}$ *queries defined in* $\mathsf{PrivK}^{\mathsf{pre\text{-}suf},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$ *below, and* q_l *leakage evaluation queries on arbitrarily chosen keys.*

(i) $\mathsf{PrivK}^{\mathsf{CCAmL2},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$: $q_{\mathsf{CCAmL2}} = (q_e, q_d, q_c)$ with the $\mathsf{CCAmL2}$ game in Figure 1.

(ii) $\mathsf{PrivK}^{\mathsf{CCAmL1},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$: $q_{\mathsf{CCAmL1}} = (q_e, q_d)$ and the $\mathsf{CCAmL1}$ security game "removes 2" from the $\mathsf{CCAmL2}$ game, meaning that $\mathsf{L}_{\mathsf{dec}}$ is removed from all the oracles. In other words items (2),(4) become black-box and (5) disappears.

(iii) $\mathsf{PrivK}^{\mathsf{CCAL2},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$: $q_{\mathsf{CCAL2}} = (q_e, q_d, q_c)$ and the $\mathsf{CCAL2}$ security game "removes M" from the $\mathsf{CCAmL2}$ game which becomes nonce-respecting.

(iv) $\mathsf{PrivK}^{\mathsf{CPAmL2},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$: $q_{\mathsf{CPAmL2}} = (q_e, q_c)$ and no decryption oracle access is given in Figure 1: items (2) and (4) are removed but not item (5), hence the 2.

(v) $\mathsf{PrivK}^{\mathsf{CPAmL1},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$: $q_{\mathsf{CPAmL1}} = (q_e)$ and the $\mathsf{CPAmL1}$ game only keeps items (1) and (3) from the $\mathsf{CCAmL2}$ game, (like the $\mathsf{CPAmL2}$ game without $\mathsf{L}_{\mathsf{decch}}$).

(vi) $\mathsf{PrivK}^{\mathsf{CPAL2},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$: $q_{\mathsf{CPAL2}} = (q_e, q_c)$, the $\mathsf{CPAL2}$ game is a nonce-respecting version of the $\mathsf{CPAmL2}$ and $\mathsf{L}_{\mathsf{decch}}$ is still available in item (5).

(vii) $\mathsf{PrivK}^{\mathsf{CCAL1},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$: $q_{\mathsf{CCAL1}} = (q_e, q_d)$ and the $\mathsf{CCAL1}$ is a nonce-respecting version of $\mathsf{CCAmL1}$ (with black-box dec. and nonce-respecting leaking enc.).

(viii) $\mathsf{PrivK}^{\mathsf{CPAL1},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$: $q_{\mathsf{CPAL1}} = (q_e)$ with only nonce-respecting leaking encryption.

Zoo of integrity notions. For $\mathsf{pre} \in \{\mathsf{CI}, \mathsf{PI}\}$ we obtain the following 8 notions, by starting from $\mathsf{CIML2}$ and by removing one security layer at a time:

$$\mathsf{CIML2} \rightarrow \mathsf{CIML1}, \mathsf{CIL2}, \mathsf{PIML2} \rightarrow \mathsf{PIML1}, \mathsf{PIL2}, \mathsf{CIL1} \rightarrow \mathsf{PIL1}.$$

Definition 8. *A nonce-based authenticated encryption with associated data* $\mathsf{AEAD} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with leakages* $\mathsf{L} = (\mathsf{L}_{\mathsf{enc}}, \mathsf{L}_{\mathsf{dec}})$ *is* $(q_e, d_d, q_l, t, \varepsilon)$*-pre-suf secure for a security parameter* n *if, for every* (q_e, q_d, q_l, t)*-bounded adversary* $\mathcal{A}^{\mathsf{L}}$*, we have* $\mathsf{pre} \in \{\mathsf{CI}, \mathsf{PI}\}$ *and:*

$$\Pr\left[\mathsf{PrivK}^{\mathsf{pre\text{-}suf}}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}(1^n) \Rightarrow 1\right] \leq \varepsilon,$$

where the adversary $\mathcal{A}^{\mathsf{L}}$ *makes at most* q_e *encryption queries and* q_d *decryption queries defined in* $\mathsf{PrivK}^{\mathsf{pre\text{-}suf}}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$ *below, and* q_l *leakage evaluation queries on arbitrarily chosen keys.*

(i) $\mathsf{PrivK}^{\mathsf{CIML2}}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$: the $\mathsf{CIML2}$ game, see Table 1.

(ii) $\mathsf{PrivK}^{\mathsf{CIML1}}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$: the $\mathsf{CIML1}$ game removes $\mathsf{L}_{\mathsf{dec}}$ (i.e., decryption is black-box).

(iii) $\mathsf{PrivK}^{\mathsf{CIL2}}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$: the $\mathsf{CIL2}$ game is a nonce-respecting version of $\mathsf{CIML2}$.

(iv) $\mathsf{PrivK}^{\mathsf{PIML2}}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$: in the $\mathsf{PIML2}$ game the winning condition changed (Table 1).

(v) $\mathsf{PrivK}^{\mathsf{PIML1}}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$: the $\mathsf{PIML1}$ game removes $\mathsf{L}_{\mathsf{dec}}$ from $\mathsf{PIML2}$.

(vi) $\mathsf{PrivK}^{\mathsf{PIL2}}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$: the $\mathsf{PIL2}$ game is a nonce respecting version of $\mathsf{PIML2}$.

(vii) $\mathsf{PrivK}^{\mathsf{CIL1}}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$: the $\mathsf{CIL1}$ game is a nonce-respecting version of $\mathsf{CIML2}$ free of $\mathsf{L}_{\mathsf{dec}}$.

(viii) $\mathsf{PrivK}^{\mathsf{PIL1}}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$: the $\mathsf{PIL1}$ game is a nonce-respecting version of $\mathsf{PIML2}$ free of $\mathsf{L}_{\mathsf{dec}}$.

Connection with previous works. Among the above sixteen notions, three of them are equivalent to already defined ones: $\mathsf{CPAL1}$ appeared in [30] under the name of LMCPA, $\mathsf{CIML1}$ was introduced in [10] (under the name CIML) and $\mathsf{CIML2}$ was introduced in [11].

4.2 Relations within the zoo (single challenge setting)

We picture all the 16 notions with their natural implications in Figure 3.

Theorem 3 (Long diagonals). *There exist authenticated encryptions schemes showing that:*

$$\begin{array}{ccc} \mathsf{CCAmL1} \not\Rightarrow \mathsf{CPAL2} & \mathsf{CCAL2} \not\Rightarrow \mathsf{CPAmL1} & \mathsf{CPAmL2} \not\Rightarrow \mathsf{CCAL1} \\ \mathsf{CIML1} \not\Rightarrow \mathsf{PIL2} & \mathsf{CIL2} \not\Rightarrow \mathsf{PIML1} & \mathsf{PIML2} \not\Rightarrow \mathsf{CIL1} \end{array}$$

$$\begin{tikzcd}
 & \mathsf{CCAmL2} \arrow[ld] \arrow[rr] \arrow[dd, dashed] & & \mathsf{CPAmL2} \arrow[ld] \arrow[dd] \\
\mathsf{CCAmL1} \arrow[rr] \arrow[dd] & & \mathsf{CPAmL1} \arrow[dd] & \\
 & \mathsf{CCAL2} \arrow[ld, dashed] \arrow[rr, dashed] & & \mathsf{CPAL2} \arrow[ld] \\
\mathsf{CCAL1} \arrow[rr] & & \mathsf{CPAL1} &
\end{tikzcd}
\qquad
\begin{tikzcd}
 & \mathsf{CIML2} \arrow[ld] \arrow[rr] \arrow[dd, dashed] & & \mathsf{PIML2} \arrow[ld] \arrow[dd] \\
\mathsf{CIML1} \arrow[rr] \arrow[dd] & & \mathsf{PIML1} \arrow[dd] & \\
 & \mathsf{CIL2} \arrow[ld, dashed] \arrow[rr, dashed] & & \mathsf{PIL2} \arrow[ld] \\
\mathsf{CIL1} \arrow[rr] & & \mathsf{PIL1} &
\end{tikzcd}$$

Fig. 3: Single-challenge security notions with various combinations of C/P (Ciphertext/Plaintext), m/M (misuse), 1/2 (# of leaking oracles). Left: confidentiality notions, right: integrity notions. Arrows indicate implications.

As a corollary, all the arrows of Figure 3 are strict. The proof only requires to show 4 of the 6 assertions.

Proof. We are to prove 12 non-implications:

(i)	$\mathsf{CCAmL1} \not\Rightarrow \mathsf{CPAL2}$,	(vii)	$\mathsf{CIML1} \not\Rightarrow \mathsf{PIL2}$,
(ii)	$\mathsf{CPAL2} \not\Rightarrow \mathsf{CCAmL1}$,	(viii)	$\mathsf{PIL2} \not\Rightarrow \mathsf{CIML1}$,
(iii)	$\mathsf{CCAL2} \not\Rightarrow \mathsf{CPAmL1}$,	(ix)	$\mathsf{CIL2} \not\Rightarrow \mathsf{PIML1}$,
(iv)	$\mathsf{CPAmL1} \not\Rightarrow \mathsf{CCAL2}$,	(x)	$\mathsf{PIML1} \not\Rightarrow \mathsf{CIL2}$,
(v)	$\mathsf{CPAmL2} \not\Rightarrow \mathsf{CCAL1}$,	(xi)	$\mathsf{PIML2} \not\Rightarrow \mathsf{CIL1}$,
(vi)	$\mathsf{CCAL1} \not\Rightarrow \mathsf{CPAmL2}$,	(xii)	$\mathsf{CIL1} \not\Rightarrow \mathsf{PIML2}$.

We first show that a security notion $\mathsf{X1}$ *without* decryption leakages cannot imply the corresponding notion $\mathsf{X2}$ *with* decryption leakages. This would establish six separations (i), (iv), (vi), (vii), (x), and (xii). For this, assume that AEAD is a $\mathsf{X1}$ secure scheme with master-key K. We define a new scheme AEAD^*, which is the same as AEAD except that its leakages for decryption queries explicitly include the master-key K. In this way, AEAD^* is clearly not $\mathsf{X2}$ secure (as the key is leaked). But it remains $\mathsf{X1}$ secure, since this enhancement of decryption leakage *cannot* be observed in the $\mathsf{X1}$ security game.

We then show that a security notion X *without* supporting nonce-misuse resistance/resilience cannot imply the corresponding notion XM *with* misuse-resilience. This would establish four separations (ii), (iii), (viii), and (ix). For this, assume that $\mathsf{AEAD} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with leakage $\mathsf{L} = (\mathsf{L}_{\mathsf{enc}}, \mathsf{L}_{\mathsf{dec}})$ is a X secure scheme. We define a new scheme $\mathsf{AEAD}^* = (\mathsf{Gen}', \mathsf{Enc}, \mathsf{Dec})$ with leakage $\mathsf{L} = (\mathsf{L}'_{\mathsf{enc}}, \mathsf{L}'_{\mathsf{dec}})$ as follows:

$\mathsf{Gen}'(1^n)$: generates two keys $k \leftarrow \mathsf{Gen}(1^n)$ and $k' \leftarrow \mathsf{Gen}(1^n)$, and selects a public pair $(N^\dagger, A^\dagger)$.

$\mathsf{L}'_{\mathsf{enc}}((k, k'), N, A, M)$: outputs $\mathsf{leak_e} = \mathsf{L}_{\mathsf{enc}}(k, N, A, M)$ as well as the additional value B but in only two cases:
- if $N = N^\dagger$ and $A = A^\dagger$, $B = k \oplus k'$;
- if $N = N^\dagger$ and $A \neq A^\dagger$, $B = k'$;

Clearly, when multiple encryption queries with the same nonce $N^\dagger$ is made, then both $k \oplus k'$ and k' could be leaked, and the key of the underlying scheme AEAD could be recovered. Therefore, AEAD^* is not misuse-resistant in *any* security setting. This is not the case in the nonce-respecting setting, and it thus remains X secure.

It remains to prove $\mathsf{CPAmL2} \not\Rightarrow \mathsf{CCAL1}$ and $\mathsf{PIML2} \not\Rightarrow \mathsf{CIL1}$. For this we follow the standard idea of showing CPA$\not\Rightarrow$CCA and $\mathsf{INT\text{-}PTXT} \not\Rightarrow \mathsf{INT\text{-}CTXT}$. In detail, consider $\mathsf{CPAmL2} \not\Rightarrow \mathsf{CCAL1}$ first, and assume that $\mathsf{AEAD} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is $\mathsf{CPAmL2}$ secure. We define a new scheme $\mathsf{AEAD}^* = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}')$ as follows:

$\mathsf{Dec}'_k(N, A, C)$: outputs $\mathsf{Dec}_k(N, A, C)\|k$, i.e. the main key k is appended to the decrypted plaintext.

This very artificial scheme "gives up" by appending its key to the decrypted message upon any decryption query. Therefore, it cannot be CCA secure under any reasonable definition. Thus $\mathsf{CPAmL2} \not\Rightarrow \mathsf{CCAL1}$.

For $\mathsf{PIML2} \not\Rightarrow \mathsf{CIL1}$, assume that $\mathsf{AEAD} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is $\mathsf{PIML2}$ secure. We define a new scheme $\mathsf{AEAD}^* = (\mathsf{Gen}, \mathsf{Enc}', \mathsf{Dec}')$ as follows:

$\mathsf{Enc}'_k(N, A, M)$: outputs $\mathsf{Enc}_k(N, A, M)\|0\|0$, i.e., two bits are appended to the ciphertext.

$\mathsf{Dec}'_k(N, A, C)$: parses $C = C'\|b\|b'$, and outputs $\mathsf{Dec}_k(N, A, C')$ if and only if $b = b'$.

Then it's clear that AEAD^* is not $\mathsf{CIL1}$ since from any valid ciphertext $(N, A, C\|0\|0)$ obtained before the adversary could use $(N, A, C\|1\|1)$ as a forgery. Yet, it remains $\mathsf{PIML2}$ secure.

Remark. By revisiting the proof for $\mathsf{MR} \wedge \mathsf{CCAmL2} \wedge \mathsf{PIML2} \not\Rightarrow \mathsf{CIML2}$ in subsection 3.3, it can be seen that the exhibited $\mathsf{CIML2}$ adversary *only* relies on the leaking encryption. This means that it also breaks the $\mathsf{CIML1}$ security. Therefore, we already know that $\mathsf{MR} \wedge \mathsf{CCAmL2} \wedge \mathsf{PIML2} \not\Rightarrow \mathsf{CIML1}$.

5 First AEML instantiation: FEMALE

We finally present the AEML mode FEMALE that makes only two calls to a strongly protected block cipher per message to be encrypted and enables leveled implementations. FEMALE is named after *Feedback-based Encryption with Misuse, Authentication and LEakage* as it starts processing the message blocks using a (re-keying) ciphertext feedback mode (see the top of Figure 4). The encryption processes the key only twice and the message blocks only once.

Given a hash function $\mathsf{H} : \{0,1\}^* \mapsto \mathcal{B}$ and a block cipher E on $\mathcal{M} = \{0,1\} \times \mathcal{B}$ as well as two distinct public constants p_A and p_B of $\mathcal{M}$, the FEMALE encryption algorithm has 3 stages:

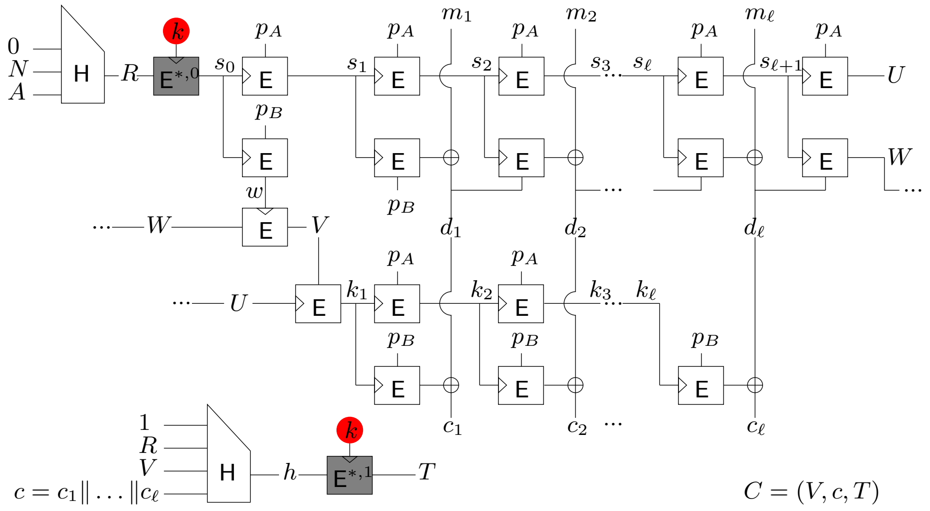

Fig. 4: FEMALE encryption algorithm. "Leak free" block ciphers are in gray with "*". $\mathsf{E}^{*,b} : \mathcal{B} \mapsto \mathcal{M}$ is such that $\mathsf{E}^{*,b}(B) := \mathsf{E}^*(b, B)$. Triangles in block ciphers indicate key inputs. If $M \in \mathcal{M}^*$ is such that $M = (m_1, \ldots, m_\ell)$, the output ciphertext is $C = (V, c, T) \in \mathcal{M}^{\ell+2}$. (Top) Generates (U, V) and $d = (d_1, \ldots, d_\ell)$, a pre-encryption of $M = (m_1, \ldots, m_\ell)$. (Middle) One-time encryption of d into $c = (c_1, \ldots, c_\ell)$ with one-time key U and pseudorandom IV V. (Bottom) Authentication from tag T.

(i) *Ephemeral key-IV generation:* on input (N, A, M) with $M \in \mathcal{M}^*$, derives a pseudorandom ephemeral key U depending only on (N, A) as well as a pseudorandom IV V depending on the whole triple. During this process all the blocks m_i of $M = (m_1, \ldots, m_\ell)$ are "pre-encrypted" as d_i, resulting in $d = (d_1, \ldots, d_\ell)$, where d_i depends on $(N, A, m_1, \ldots, m_i)$;

(ii) *One-time encryption:* on input (V, d) and the ephemeral key U, produces a one-time encryption c of d with initialized vector V;

(iii) *Authentication:* on input (R, V, c), where $R = \mathsf{H}(0\|N\|A)$, computes a pseudorandom tag T.

The ciphertext is given by $C = (V, c, T)$ and it does not include d. To decrypt the ciphertext (N, A, C), FEMALE first checks (iii) before deriving the one-time key U from (N, A) as in step (i) in order to decrypt c into d as the reverse process of step (ii). Eventually, (N, A, d) allows retrieving M at step (i). The full specification of FEMALE is available in Figure 5.

For the sake of space, the security analysis of FEMALE is deferred to the full version of this paper [20]. Informally, built upon secure cryptographic functions and assuming that the circuits of E^* is "leak-free", FEMALE offers a (somewhat standard) birthday security, i.e., it can preserve MR, CIML2, and CCAmL2 up to $2^{n/2}$ computations and processing $2^{n/2}$ message and associated data blocks.

Description of FEMALE:

Gen(1^n) picks a random key $k \xleftarrow{\$} \{0,1\}^n = \mathcal{K}$.

Enc$_k$(N, A, M) parses $M \in \mathcal{M}^*$ into as many blocks as needed as $M = (m_1, \ldots, m_\ell)$ for some ℓ. Computes $R \leftarrow \mathsf{H}(0\|N\|A)$. Then:

1. *Ephemeral key-IV generation:* (skip step (b) if $\ell = 0$)
   (a) Computes $s_0 \leftarrow \mathsf{E}^*_k(0\|R)$, $w \leftarrow \mathsf{E}_{s_0}(p_B)$, $s_1 \leftarrow \mathsf{E}_{s_0}(p_A)$, and sets $d_0 \leftarrow p_B$;
   (b) Computes $s_{i+1} \leftarrow \mathsf{E}_{s_i}(p_A)$, $y_i \leftarrow \mathsf{E}_{s_i}(d_{i-1})$, $d_i \leftarrow y_i \oplus m_i$, for $i = 1$ to ℓ;
   (c) Computes $U \leftarrow \mathsf{E}_{s_{\ell+1}}(p_A)$, $W \leftarrow \mathsf{E}_{s_{\ell+1}}(d_\ell)$, $V \leftarrow \mathsf{E}_w(W)$.
2. *One-time encryption:* first computes $k_1 \leftarrow \mathsf{E}_U(V)$ and then, for $i = 1$ to $\ell - 1$, computes $k_{i+1} \leftarrow \mathsf{E}_{k_i}(p_A)$, $z_i \leftarrow \mathsf{E}_{k_i}(p_B)$, and $c_i \leftarrow z_i \oplus d_i$.
3. *Authentication:* sets $c = c_1\|\ldots\|c_\ell$, computes $h \leftarrow \mathsf{H}(1\|R\|V\|c)$, and computes $T \leftarrow \mathsf{E}^*_k(1\|h)$.

Eventually, returns the ciphertext $C = (V, c, T)$.

Dec$_k$(N, A, C) parses $C = (V, c, T)$, $c = c_1\|\ldots\|c_\ell$, then proceeds in four phases:

1. *Integrity Checking:* computes $R \leftarrow \mathsf{H}(0\|N\|A)$, $h \leftarrow \mathsf{H}(1\|R\|V\|c)$, and $h^* \leftarrow \mathsf{trunc}((\mathsf{E}^*_k)^{-1}(T))$, where trunc drops the first bit of its input. Then, if $h^* = h$, it enters the next phase, and returns $\perp$ otherwise.
2. *Ephemeral key extraction:* first computes $s_0 \leftarrow \mathsf{E}^*_k(0\|R)$ and $s_{i+1} \leftarrow \mathsf{E}_{s_i}(p_A)$, for $i = 0$ to ℓ, and finally $U \leftarrow \mathsf{E}_{s_{\ell+1}}(p_A)$.
3. *One-time decryption:* first computes $k_1 \leftarrow \mathsf{E}_U(V)$ and then, for $i = 1$ to $\ell - 1$, computes $k_{i+1} \leftarrow \mathsf{E}_{k_i}(p_A)$, $z_i \leftarrow \mathsf{E}_{k_i}(p_B)$, and $d_i \leftarrow z_i \oplus c_i$; Set $d_0 \leftarrow p_B$.
4. *Message recovery:* for $i = 1$ to ℓ, computes $y_i \leftarrow \mathsf{E}_{s_i}(d_{i-1})$ and $m_i \leftarrow y_i \oplus d_i$.

Eventually, returns the message $M = (m_1, \ldots, m_\ell)$.

Fig. 5: The FEMALE AEAD scheme.

5.1 Other possible instances

To demonstrate the usefulness of the various definitions, we also list some other possible instances of modes that satisfy some form of resistance against leakage. First, it has been known that the state of a duplex construction [12] can be easily recovered when the initial state can be fixed [1], which may enable universal forgery. A standard 1-pass duplex AE starts processing the inputs by deriving a secret duplex state from the nonce N and the AE key, and thus runs a duplex construction on A and M. For such an AE, the initial state indeed can be fixed if N can be reused for encryption, or if decryption leakages are available. This means such a standard 1-pass duplex AE cannot be CIML1 not CIL2. The inability to offer CIL2 enables a simple forging-and-testing attack against the CCAL2 security. However, given a side-channel secure state-derivation function (like our leak-free block cipher), such a 1-pass duplex AE seems to offer CIL1 and CCAL1, since in these settings N cannot be reused and the above attacks turn impossible. We leave the proof of this conjecture as an open problem.

On the other hand, if a (side-channel secure) keyed finalization function is added to the 1-pass duplex AE, which is adopted by a CAESAR final winner Ascon [17], then the above universal forgery disappears. Indeed, Ascon is CIML2 and CCAmL1 under certain assumptions: see our recent work [21]. Furthermore, if we can use 2 passes, then we can achieve CIML2 and CCAmL2: this is achieved by the AEDT design [20], or the recent TEDT [9]. Sponge-based 2-pass AEs like ISAP [16] or S1P [21] are also expected to offer similar guarantees.

Acknowledgments. Thomas Peters and François-Xavier Standaert are post-doctoral researcher and senior associate researcher of the Belgian Fund for Scientific Research (FNRS-F.R.S). This work has been funded in part by the ERC consolidator grant SWORD (724725), and also by the EU and Walloon Region through the FEDER project USERMedia (501907-379156).

References

1. Adomnicai, A., Fournier, J.J., Masson, L.: Masking the lightweight authenticated ciphers acorn and ascon in software. Cryptology ePrint Archive, Report 2018/708 (2019), appeared at BalkanCryptSec 2018.
2. Albrecht, M.R., Paterson, K.G., Watson, G.J.: Plaintext recovery attacks against SSH. In: IEEE Symposium on Security and Privacy. pp. 16–26. IEEE Computer Society (2009)
3. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., Yasuda, K.: How to securely release unverified plaintext in authenticated encryption. In: ASIACRYPT (1). LNCS, vol. 8873, pp. 105–125. Springer (2014)
4. Ashur, T., Dunkelman, O., Luykx, A.: Boosting authenticated encryption robustness with minimal modifications. In: CRYPTO (3). LNCS, vol. 10403, pp. 3–33. Springer (2017)
5. Barwell, G., Martin, D.P., Oswald, E., Stam, M.: Authenticated encryption in the face of protocol and side channel leakage. In: ASIACRYPT (1). LNCS, vol. 10624, pp. 693–723. Springer (2017)
6. Barwell, G., Page, D., Stam, M.: Rogue decryption failures: Reconciling AE robustness notions. In: IMA Int. Conf. LNCS, vol. 9496, pp. 94–111. Springer (2015)
7. Belaïd, S., Grosso, V., Standaert, F.: Masking and leakage-resilient primitives: One, the other(s) or both? Cryptography and Communications 7(1), 163–184 (2015)
8. Bellare, M., Namprempre, C.: Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. J. Cryptology 21(4), 469–491 (2008)
9. Berti, F., Guo, C., Pereira, O., Peters, T., Standaert, F.: Tedt, a leakage-resilient AEAD mode for high (physical) security applications. IACR Cryptology ePrint Archive 2019, 137 (2019), https://eprint.iacr.org/2019/137
10. Berti, F., Koeune, F., Pereira, O., Peters, T., Standaert, F.: Ciphertext integrity with misuse and leakage: Definition and efficient constructions with symmetric primitives. In: AsiaCCS. pp. 37–50. ACM (2018)
11. Berti, F., Pereira, O., Peters, T., Standaert, F.: On leakage-resilient authenticated encryption with decryption leakages. IACR Trans. Symmetric Cryptol. 2017(3), 271–293 (2017)
12. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Duplexing the sponge: Single-pass authenticated encryption and other applications. In: SAC. pp. 320–337. sacpub (2012)
13. Boldyreva, A., Degabriele, J.P., Paterson, K.G., Stam, M.: On symmetric encryption with distinguishable decryption failures. In: FSE. LNCS, vol. 8424, pp. 367–390. Springer (2013)
14. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: CHES. LNCS, vol. 3156, pp. 16–29. Springer (2004)

15. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: CHES. LNCS, vol. 2523, pp. 13–28. Springer (2002)
16. Dobraunig, C., Eichlseder, M., Mangard, S., Mendel, F., Unterluggauer, T.: ISAP - towards side-channel secure authenticated encryption. IACR Trans. Symmetric Cryptol. 2017(1), 80–105 (2017)
17. Dobraunig, C., Eichlseder, M., Mendel, F., Schlaffer, M.: Ascon v1.2. Submission to the CAESAR Competition (2016), https://competitions.cr.yp.to/round3/asconv12.pdf.
18. Duong, T., Rizzo, J.: Cryptography in the web: The case of cryptographic design flaws in ASP.NET. In: IEEE Symposium on Security and Privacy. pp. 481–489. IEEE Computer Society (2011)
19. Goudarzi, D., Rivain, M.: How fast can higher-order masking be in software? In: EUROCRYPT (1). LNCS, vol. 10210, pp. 567–597 (2017)
20. Guo, C., Pereira, O., Peters, T., Standaert, F.: Authenticated encryption with nonce misuse and physical leakages: Definitions, separation results and leveled constructions. IACR Cryptology ePrint Archive 2018, 484 (2018), https://eprint.iacr.org/2018/484
21. Guo, C., Pereira, O., Peters, T., Standaert, F.: Towards Lighter Leakage-Resilient Authenticated Encryption from the Duplex Construction. IACR Cryptology ePrint Archive 2019, 193 (2019), https://eprint.iacr.org/2019/193
22. Halderman, J.A., Schoen, S.D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J.A., Feldman, A.J., Appelbaum, J., Felten, E.W.: Lest we remember: cold-boot attacks on encryption keys. Commun. ACM 52(5), 91–98 (2009)
23. Halevi, S., Lin, H.: After-the-fact leakage in public-key encryption. In: TCC. LNCS, vol. 6597, pp. 107–124. Springer (2011)
24. Hoang, V.T., Krovetz, T., Rogaway, P.: Robust authenticated-encryption AEZ and the problem that it solves. In: EUROCRYPT (1). LNCS, vol. 9056, pp. 15–44. Springer (2015)
25. Katz, J., Yung, M.: Unforgeable encryption and chosen ciphertext secure modes of operation. In: FSE. LNCS, vol. 1978, pp. 284–299. Springer (2000)
26. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Proceedings of Crypto'99. pp. 388–397. Springer-Verlag - LNCS Vol. 1666 (1999)
27. Martin, D.P., Oswald, E., Stam, M., Wójcik, M.: A leakage resilient MAC. In: IMA Int. Conf. LNCS, vol. 9496, pp. 295–310. Springer (2015)
28. O'Flynn, C., Chen, Z.D.: Side channel power analysis of an AES-256 bootloader. In: CCECE. pp. 750–755. IEEE (2015)
29. Paterson, K.G., AlFardan, N.J.: Plaintext-recovery attacks against datagram TLS. In: NDSS. The Internet Society (2012)
30. Pereira, O., Standaert, F., Vivek, S.: Leakage-resilient authentication and encryption from symmetric cryptographic primitives. In: ACM Conference on Computer and Communications Security. pp. 96–108. ACM (2015)
31. Rogaway, P.: Authenticated-encryption with associated-data. In: ACM Conference on Computer and Communications Security. pp. 98–107. ACM (2002)
32. Rogaway, P., Shrimpton, T.: A provable-security treatment of the key-wrap problem. In: EUROCRYPT. LNCS, vol. 4004, pp. 373–390. Springer (2006)
33. Standaert, F., Pereira, O., Yu, Y.: Leakage-resilient symmetric cryptography under empirically verifiable assumptions. In: CRYPTO (1). LNCS, vol. 8042, pp. 335–352. Springer (2013)