Authenticated Encryption with Nonce Misuse and Physical Leakages: Definitions, Separation Results, and Leveled Constructions

Chun Guo\textsuperscript{*}, Olivier Pereira\textsuperscript{*}, Thomas Peters\textsuperscript{*}, François-Xavier Standaert\textsuperscript{*}
\textsuperscript{*} ICTEAM/ELEN/Crypto Group, UCL, Louvain-la-Neuve, Belgium

Abstract. We propose definitions and constructions of authenticated encryption (AE) schemes that offer security guarantees even in the presence of nonce misuse and side-channel leakages. 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 leakages 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 leakages 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. Eventually, we propose first instances of modes of operations that satisfy our definitions. In order to optimize their efficiency, we aim at modes that support “leveled implementations” such that the encryption and decryption operations require the use of a small constant number of evaluations of an expensive and heavily protected component, while the bulk of the computations can be performed by cheap and weakly protected block cipher implementations.

1 Introduction

1.1 Authenticated encryption

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 [1,13,31]. 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 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 [35], 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 leakages (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 [1,2,21,28,4]. 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 implementations is the possibility of side-channel attacks [9,5,10]. 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 leakages happen 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 leakages of course depends on the type, the permanence and the amount of leakages granted to the adversary. As far as the type is concerned, we separate scalar leakages (like timing) that allow so-called univariate attacks and vector leakages (like the aforementioned power consumption or electromagnetic radiation) that allow so-called multivariate attacks. As far as the permanence is concerned, we separate between full leakages (that allow to leak during all interactions with the device) and partial leakages that exclude the leakages of some interactions. This leads us to define a first taxonomy of leakages 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 leakages.

As for the amount of leakage, it is in general hard to quantify and highly depends on the implementations and measurement devices that are at hand. In this respect, our starting observation is that the leakage of all secrets makes privacy-preserving cryptography impossible, but the full protection against leakages 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 leakages. 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 leakages) typically increase the “code size × cycle count” metric by 2 or 3 orders of magnitude compared to a non-protected implementation [7].

This state-of-the-art motivated the design of authentication, encryption and AE schemes allowing “leveled implementations” (or implementations in the leveled leakage setting). By leveled implementations, 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 [32,9,10], this setting usually allows lower cost or more efficient implementations 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 leakages and the pressure on the modes to deal with the remaining leakages. Our following contributions therefore aim at defining security targets and modes of operation with an effective balance between leakage reduction at the implementation level and leakage-resilience at the cryptographic mode level, in order to reach AE with high physical security and a minimum implementation cost.

1.2 Contributions

Our main contributions target deterministic (nonce-based) authenticated encryption with associated data (AEAD) [34]. We:

(i) Define confidentiality and integrity notions in the presence of nonce misuse and leakages. Our security notions capture leakages in encryption and decryption and allow the computation of the “challenge ciphertexts” to leak. Several variants are explored and put in relation.

(ii) Identify the strongest form of security that an AEAD can offer in the presence of nonce misuse and vector & full (VF) leakages, which we call AEML-VF security, as a combination of black-box misuse-resistance, ciphertext integrity with VF leakages and misuse-resistance and CCA security with VF leakages, and misuse-resilience. We also argue why CCA security with VF leakages and misuse-resistance cannot be achieved.

(iii) Define modes of operation for AEML-VF that satisfy our definitions and allow leveled implementations, using a set of weaker assumptions for the integrity part of the definition and stronger ones for its confidentiality part, leading to “gradual security degradations”: even if some of the security requirements for confidentiality are not satisfied, integrity may be preserved.

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 leakages), in the usual sense of Rogaway and Shrimpton [35]. (ii) The AE scheme must offer CICML2 security, which is a natural extension of ciphertext integrity considering the presence of (full) nonce misuse and (full) leakages introduced by Berti et al. [9,10]. (iii) The AE scheme must offer CCAML2 security, which is an extension of CCA security with misuse-resilience in the presence of (full) leakages that we propose here. Misuse-resilience and full leakages 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 leakages, an AEML scheme is also a traditional MR AE scheme.

In the presence of VF leakages, we unfortunately cannot just extend 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, a requirement that is known to be unrealistic in a leakage setting since the early works on leakage-resilience by Micali and Reyzin on pseudorandom generators [29]. For example and concretely, if an implementation of an AE scheme processes a message block-by-block, as it is standard, the leakages 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 leakages of the computation associated to these first blocks will be highly similar, something that can be easily observed. But this directly contradicts the requirement that ciphertexts must look independent as soon as two different messages are encrypted, even if they only differ by one bit in their last block. Note that this argument may not hold for scalar leakages (i.e., in the SF of 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 leakages that can help her to distinguish.

In order to avoid this caveat of the VF case, we 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, as detailed next.

The extension of INT-CTX to the setting of misuse and leakages has been recently proposed as the CICML2 notion [10]. (The same notion excluding decryption leakages has been proposed beforehand in [9] - we denote it as CICML1). This extension can be viewed as natural as it provides the adversary with full nonce misuse capabilities (as in the black box setting) and leakages 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 [36,9]. However, this leads to guarantees that are weaker than the theoretical limits. While INT-CTX + IND-CPA implies (the desired) IND-CCA security in the black box setting [8], we show that this is not true anymore when leakages enter the picture: the implication towards an extension of CCA security with leakages does not hold, mainly because it does not capture the risks associated to decryption leakages.

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 leakages: 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 and CCAML2 do not imply CICML2 security: the alternate definition of AE as INT-PTXT and IND-CCA security [24] does not suffice in the leaking setting either. By the above, we propose to use CCAML2 in combination with CICML2 (and MR) to define AEML-VF security.

The reason of our focus on nonce misuse-resilience for CCAML2 security, rather than on the stronger requirement of nonce misuse-resistance, is identical to the reason why pseudorandom ciphertexts for each (nonce, message) pair is an excessive requirement in the VF leakage setting: querying an encryption oracle with identical nonces and messages starting with identical blocks is something that would cause identically distributed leakages during the encryption of these first blocks, something that can be expected to be noticed, making a misuse-resistant CCA game trivial to win. Nonce misuse-resilience is then the natural form of protection against misuse that can be aimed for in the VF setting.

**New modes of operation.** Based on the above definitions, the driving motivation for the design of our modes of operation, and our choice of leakage models, is to push towards the most effective balance between the pressure on implementers and the pressure on mode of operation designers.
Traditional (non leakage-resilient) 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 leakages about a secret (e.g., key) obtained from computations based on different inputs [27]. A DPA reduces the computational secrecy of the state manipulated by a device at a rate that is exponential in the number of leakages, by combining the information of these different inputs (e.g., plaintexts).

Leakage-resilient 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, leaving the adversary with the more challenging task of exploiting leakages with Simple Power Analysis attacks (SPAs). SPAs are side-channel attacks taking advantage of the leakages 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, 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. This concern is reflected in the assumptions that we make about leakages. They differ in strength depending on the security property that we analyze but, based on these assumptions, any mode offering the possibility to mount a traditional DPA, message block by message block, would be considered insecure.

We define two modes of operation. Our first mode, FEMALE (for Feedback-based Encryption with Misuse, Authentication and LEakages), offers AEML-VF security. The second one, AEDT, is CIML2 and CCAmL2 but not MR, a combination which we denote as AEmL-VF (where the lower-case m indicates that black-box misuse-resistance is abandoned – still, misuse-resilience is offered). It is proposed for situations in which the processing of the message in two passes (inherent to misuse-resistance) leads to excessive memory requirements.

FEMALE is a two-pass encryption scheme offering traditional AE security, which is compatible with leveled implementations: independently of the length of the message and associated data, a strongly protected block cipher (BC) must be called only twice – which we model as a “leak-free” block cipher for simplicity. Apart from that, a weakly protected BC must be called $4\ell + 5$ times for a message of $\ell$ blocks. The MR security of FEMALE holds in the standard model. The CIML2 security holds in the unbounded leakage model [10], which lets weakly protected components leak their state completely, hence only relying on the two heavily protected blocks. The CCAmL2 security holds based on the assumption that leakages are simulatable [36]. Our reduction shows that any attacker that breaks the CCAmL2 security of FEMALE is either violating the simulatable leakage assumption, or able to break the eavesdropper security of a very simple encryption scheme that can only encrypt one single block and has no leak-free component (so in some sense we extend the domain of the single block encryption in a security-preserving manner and enhance functionality). The single-block security formally reflects the aforementioned “minimum attack surface” that our modes aim to provide. It cannot be derived from the simulatable leakage assumption: the leakage of one single bit of a secret message may not contradict simulatability but break confidentiality [32]. So this approach is as far as one can go based on the current understanding of leakage-resilience and the fact that accurately defining the confidentiality guarantees offered by a single message block remains an open problem. Still, it makes the application of DPA attacks impossible by design (excepted for the calls to a strongly protected block, which we make two times per message to be encrypted, independently of the length of this message). Also, testing the eavesdropper security of a single-block encryption scheme is a much simpler target for a side-channel evaluation laboratory than evaluating the CCAmL2 security of a fully fledged scheme.

AEDT is a simple variant of the EDT scheme of Berti et al. [10] which was designed to support leveled implementations. It is already known to be CIML2 secure, and we show that AEDT also offers CCAmL2 security (the combination of which gives AEmL-VF). Giving up on black-box MR security leads to efficiency improvements: AEDT requires two calls of leak-free BC, but only $2\ell - 1$ calls of a weakly protected BC, and

---

1 One interesting open question would be to investigate the extent to which proposals of leakage-resilient (S)PRP, e.g. [12,15], could be used here. The security definitions used there do not seem usable for our purpose, as we need guarantees of limited leakage about the output of the PRP, something that is out of the picture in their definition, in which leakages are always given together with the full PRP output.
the evaluation of a hash function on public values. We note that, if the hash function proceeds by blocks
(as any function based on the Merkle-Damgård transformation or sponge constructions), then AEDT does
not require any latency for producing ciphertexts, and reduces the memory complexity for encryption to
constant instead of requiring the storage of the message in full as for traditional MR. Still, and as usual, the
full ciphertext is needed before decryption can start. Combining leakage-resilience with modes like in the
works of Fleischmann et al. [16] and Hoang et al. [22] on on-line misuse-resistance, which explore weakened
notions of misuse-resistance that are compatible with single-pass encryption (but do not consider the presence
of leakages) is an interesting scope for further research.

The properties achieved by these constructions are summarized in Table 1. We do not mention standard
modes that are not designed for leakage-resilience since they do not provide any of these properties (e.g., the
CIML1 security of the SIV construction [35] is broken with a DPA in [9]).

Table 1. Existing AE designs that support leveled leakage-resilient implementations. LR hash indicates whether
the scheme relies a hash functions that’s resilient to some leakages. AD indicates whether the scheme supports the
processing associated data.

<table>
<thead>
<tr>
<th></th>
<th>LR hash</th>
<th>LF calls</th>
<th>AD</th>
<th>CIML</th>
<th>CCAmL2</th>
<th>MR</th>
<th>Ref.</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTE</td>
<td>✓</td>
<td>2</td>
<td>×</td>
<td>1</td>
<td>×</td>
<td>✓</td>
<td>[9]</td>
</tr>
<tr>
<td>DTE2</td>
<td>✓</td>
<td>2</td>
<td>×</td>
<td>2</td>
<td>✓</td>
<td>×</td>
<td>[10]</td>
</tr>
<tr>
<td>EDT</td>
<td>×</td>
<td>2</td>
<td>×</td>
<td>2</td>
<td>✓</td>
<td>×</td>
<td>[10]</td>
</tr>
<tr>
<td>FEMALE</td>
<td>×</td>
<td>2</td>
<td>✓</td>
<td>2</td>
<td>✓</td>
<td>✓</td>
<td>new</td>
</tr>
<tr>
<td>AEDT</td>
<td>×</td>
<td>2</td>
<td>✓</td>
<td>2</td>
<td>✓</td>
<td>×</td>
<td>extends [10]</td>
</tr>
</tbody>
</table>

Remark on gradual security degradations. The security proofs of FEMALE and AEDT are obtained under
physical assumptions that differ depending on whether we target confidentiality or integrity guarantees, as
outlined above, and as will be carefully discussed in Section 5.

In this respect, it is important to note that our definitions of AEML and AEML security allow expressing
gradual security degradations in the sense (present in our modes) that imperfectly simulatable leakages or
strongly leaking single-block encryptions (which would harm CCAmL2 security) do not affect the CIML2
guarantees. In other words, CIML2 and CCAmL2 should be seen as gradual improvements that modes of
operation can bring to better cope with leakages, with a risk of non-negligible adversarial advantages (espe-
cially for CCAmL2) that are in the same time inherent to physical security issues, and significantly reduced
compared to modes of operations ignoring physical leakages in their design.

1.3 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 leakages that would
reveal information about the messages that the AE scheme is supposed to hide: leakages are excluded from
the “challenge” queries. In the terminology introduced above, BMOS focuses on the VP (Vector & Partial)
leakage model.

As a result of the weaker VP leakage model, BMOS can realistically require misuse-resistance to hold for
confidentiality, and not only misuse-resilience, which is in line with our previous discussions. As such, the
BMOS work can be viewed as dual to ours: we consider full leakages, and as a consequence have to exclude
full misuse-resistance; they consider partial leakages, 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: according to the BMOS definition, an implementation
that leaks plaintexts in full during encryption may be considered as perfectly secure. By contrast, such an
implementation would be considered completely insecure in the VF model, which is our focus here.
A secondary difference is on the modes of operation of BMOS, which require “uniform implementations” (or an implementations in the uniform leakage setting). In contrast with the leveled implementations we propose, all the components in their constructions are required to offer the same strong level of protection against side-channel attacks. This is expected to lead to considerably more expensive implementations. For example, the proposed implementation strategy of the BMOS modes of operation processes each message block with a pairing-based leakage-resilient PRF (while we leverage symmetric building blocks).

For the rest, the proofs of BMOS and ours rely on very similar requirements. Namely, and as will be discussed carefully in Section 5, both works require a kind of ideal (i.e., leak-free) component for the authentication and integrity guarantees, and an additional assumption (e.g., simulatability or bounded leakages [14]) for the confidentiality guarantees.

Note that in general, our approach is motivated by the observation that restricting the misuse capabilities may be easier to control by implementers than excluding some leakages (which usually depends on when the target device is monitored – a hard to control parameter). Our following results therefore focus on this important (and so far unexplored) part of the problem, and analyze how to deal with adversaries having full leakage capabilities. Yet, we insist that they do not dismiss the interest of the BMOS definition which rather corresponds to a situation where leakages can be first accumulated in a profiling phase and are then exploited in a leak-free challenge query with full misuse. In this respect, investigating whether an AE scheme allowing leveled implementations (e.g., a variation of FEMALE which already satisfies MR) can also satisfy CCAML2 (where the small l reflects partial leakages) and therefore AEML-VP (in addition to AEML-VF) is an interesting open problem.

Our 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 leakages. In contrast, these leakages are the central concern of our security definitions, and they are our motivation for departing from traditional misuse-resistance in the VF leakage model.

Eventually, we mention the line of works about “after-the-fact” leakages which is complementary to ours [20] 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 [19], 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 the leakages during the challenge phase, as will be available in the context of a side-channel attack based on power consumption leakages, which is our main concern here.

1.4 Paper structure
After introducing preliminary notions in Sec. 2, we introduce, in Sec. 3, our definitions of CCAML2, AEML and AEmL security. These definitions are analyzed and compared with other leaking variants of standard authentication, integrity and confidentiality guarantees in Section 4. (These results are extended to the multi-challenge setting in Appendix E). We then discuss our model and assumptions on leakages in Section 5, and conclude by describing and proving the AEML (resp., AEmL) security of FEMALE (resp., AEDT) in Section 6.

2 Preliminaries
Throughout the paper n denotes the security parameter. Cryptographic primitives specify some family of sets: key-space family, message-space family, … We implicitly assume that n pinpoints a member within these family: a key-space, a message-space, … For any set S, we define $S^* = \bigcup_{i=1}^{\infty} S^i$ so that $s \in S^*$ if and only if it exists a non negative integer $\ell$ such that $s \in S^\ell$. This notation is convenient for instance for variable block-length (authenticated) encryption with message space $M^*$, where messages of $M$ consist of one block.

2.1 Notations
Adversary. We denote by a $(q_1, \ldots, q_\omega, t)$-bounded adversary a probabilistic algorithm that has access to $\omega$ oracles, can make at most $q_i$ queries to its $i$-th oracle, and can perform computations bounded by running
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 \( L \).

**Leaking algorithm.** For an algorithm \( \text{Algo} \), a leaking version is denoted \( \text{LAlgo} \). It runs both \( \text{Algo} \) and a leakage function \( L_{\text{algo}} \) which captures the additional information given by an implementation of \( \text{Algo} \) during its execution. \( \text{LAlgo} \) returns the outputs of both \( \text{Algo} \) and \( L_{\text{algo}} \) which all take the same input.

### 2.2 Definitions of primitives

We first need the following definition of collision-resistant hash function.

**Definition 1 (Collision-Resistant Hash Function).** A \((t, \epsilon_{cr})\)-collision resistant hash function \( H : S \times \{0, 1\}^* \rightarrow B \) for security parameter \( n \) is a function that is such that, for every \( t \)-bounded adversary \( A \), the probability that \( \mathcal{A}(1^n, s) \) outputs a pair of distinct bit-strings \((m_0, m_1)\) such that \( H_s(m_0) = H_s(m_1) \) is bounded by \( \epsilon_{cr} \), where \( s \in S \) is selected uniformly at random.

We next need the following definition of range-oriented preimage resistance.

**Definition 2 (Range-Oriented Preimage-Resist. Hash Function [3]).** A \((t, \epsilon_{pr})\)-range-oriented preimage resistant hash function \( H : S \times \{0, 1\}^* \rightarrow B \) for security parameter \( n \) is a function such that, for every adversary \( A \) running in time \( t \), the probability that \( \mathcal{A}(1^n, s, y) \) outputs a bit-string \( m \) such that \( H_s(m) = y \) is bounded by \( \epsilon_{pr} \), where \( s \in S \), \( y \in B \) are selected uniformly at random.

In the following, we assume that the key \( s \) is not private, and refer to the hash function simply as \( H \) for simplicity, the key \( s \) being implicit.

We also need the following definitions of pseudorandom permutation.

**Definition 3 (Pseudorandom Function).** A function \( F : K \times M \rightarrow M \) is a \((q, t, \epsilon_E)\)-pseudorandom function (PRF) for a security parameter \( n \) if, for all \((q, t)\)-bounded adversaries \( A \), we have:

\[
\left| \Pr[k \in K : A^F_k(1^n) \Rightarrow 1] - \Pr[f \in F : A^f(1^n) \Rightarrow 1] \right| \leq \epsilon_E,
\]

where \( F \) denotes the set of all functions from \( M \) to \( M \).

The strong pseudorandom permutation notion is as follows.

**Definition 4 (Strong Pseudorandom Permutation).** A function \( E : K \times M \rightarrow M \) is a \((q, t, \epsilon_E)\)-strong pseudorandom permutation (SPRP) for a security parameter \( n \) if, for all \((q, t)\)-bounded adversaries \( A \), we have:

\[
\left| \Pr[k \in K : A^{E_k, E_k^{-1}}(1^n) \Rightarrow 1] - \Pr[P \in P : A^{P, P^{-1}}(1^n) \Rightarrow 1] \right| \leq \epsilon_E,
\]

where \( P \) denotes the set of all permutations on \( M \).

We will focus on authenticated encryption with the following formalism.

**Definition 5 (Nonce-Based AEAD [34]).** A nonce-based authenticated encryption scheme with associated data is a tuple \( \text{AEAD} = (\text{Gen}, \text{Enc}, \text{Dec}) \) such that, for any security parameter \( n \), and keys in \( K \) generated from \( \text{Gen}(1^n) \):

- \( \text{Enc} : K \times N \times AD \times M \rightarrow C \) deterministically maps a key selected from \( K \), a nonce value from \( N \), some blocks of associated data selected from \( AD \), and a message from \( M \) to a ciphertext in \( C \).
- \( \text{Dec} : K \times N \times AD \times C \rightarrow \mathcal{M} \cup \{\perp\} \) deterministically maps a key from \( K \), a nonce from \( N \), some associated data from \( AD \), and a ciphertext from \( C \) to a message in \( M \) or to a special symbol \( \perp \) if integrity checking fails.
The sets $\mathcal{K}, \mathcal{N}, \mathcal{AD}, \mathcal{M}, \mathcal{C}$ are completely specified by $n$. Given a key $k \leftarrow \text{Gen}(1^n)$, $\text{Enc}_k(N, A, M) := \text{Enc}(k, N, A, M)$ and $\text{Dec}_k(N, A, M) := \text{Dec}(k, N, A, M)$ are deterministic functions whose implementations may be probabilistic.

Since we only focus on nonce-based authenticated encryption with associated data in this paper, we will simply refer to it as authenticated encryption.

2.3 Security definitions

We first recall the definition of Misuse-Resistance (MR) formalized in [35].

**Definition 6 (MR).** A nonce-based authenticated encryption scheme with associated data $\text{AEAD} = (\text{Gen}, \text{Enc}, \text{Dec})$ is $(q_e, q_d, t, \varepsilon)$ misuse resistant for a security parameter $n$ if, for all $(q_e, q_d, t)$-bounded adversaries $A$,

$$\left| \Pr[k \leftarrow \text{Gen}(1^n) : A^{\text{Enc}, \text{Dec}}(1^n) \Rightarrow 1] - \Pr[A^{\bot}(1^n) \Rightarrow 1] \right| \leq \varepsilon,$$

where $\mathcal{S}(N, A, M)$ outputs and associates a fresh random ciphertext $C \leftarrow C$ to fresh inputs², and the associated $C$ otherwise, and $\bot(N, A, C)$ outputs $\bot$ except if $C$ was associated to $(N, A, M)$ for some message $M$, in which case it returns $M$.

3 AE with full vectorial leakages

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 IND-CPA and the IND-CCA confidentiality notions of nonce-based authenticated encryption in the setting of nonce misuse and leakages. We focus then on the extension of the INT-PTXT and INT-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 CCAmL2. Then, we derive the weaker notion of security against chosen-plaintext attacks with misuse resilience and leakages, denoted CPAml2.

**Chosen-ciphertext security with misuse and leakages.** To capture the CCAmL2 security, as it is motivated in the introduction, we define the game PrivK_{\text{CCAmL2}}^{b} b detailed in Figure 1. This game takes as parameters an adversary $A$, a nonce-based authenticated encryption AEAD and a (possibly probabilistic) leakage function pair $L = (L_{\text{enc}}, L_{\text{dec}})$ resulting from the implementation of the scheme. During PrivK_{A,\text{AEAD},L}^{b}, the adversary $A$ has to guess the bit $b$ of which depend the challenge ciphertext $C^b$ and the encryption leakages $\text{leak}^{b}_{\text{enc}}$, computed as a leaking encryption of $(N_{b}, A_{b}, M^b)$, where the nonce $N_{b}$, the associated data $A_{b}$ and the messages $M^0, M^1$ are chosen by $A$ under certain conditions. All along this game $A$ is also granted unbounded and adaptive access to three types of oracles: LEnc, a leaking encryption oracle; LDec, a leaking decryption oracle; and $L_{\text{decch}}$, a challenge decryption leakage oracle.

Overall, this definition follows the general pattern of CCA security. In terms of misuse-resilience and forbids the adversary to reuse an old nonce $N_{b}$ in its challenge query. In terms of leakage-resilience, all

---

² This slight extension allows AEAD with polynomial size $C$ to be misuse-resistant.
PrivK\textsuperscript{CCAmL2,b}\textsubscript{\texttt{AEAD.L}}(1^n) is the output of the following experiment:

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

\textbf{Pre-challenge queries:} \( A \) gets adaptive access to \texttt{LEnc} and \texttt{LDec}.

1. \texttt{LEnc}(\( N, A, M \)) computes \( C \leftarrow \text{Enc}_k(N, A, M) \) and \( \text{leak}_k \leftarrow \text{Lenc}(k, N, A, M) \)
   \( \) updates \( \mathcal{E} \leftarrow \mathcal{E} \cup \{N\} \) and finally returns \( (C, \text{leak}_k) \)

2. \texttt{LDec}(\( N, A, C \)) computes \( M \leftarrow \text{Dec}_k(N, A, C) \) and \( \text{leak}_d \leftarrow \text{Ldec}(k, N, A, C) \)
   \( \) and returns \( (M, \text{leak}_d) \) – we stress that \( M = \perp \) may occur

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

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

   Else compute \( C^b \leftarrow \text{LEnc}_k(N_{ch}, A_{ch}, M^b) \) and \( \text{leak}_k^b \leftarrow \text{Lenc}(k, N_{ch}, A_{ch}, M^b) \)

   and return \((C^b, \text{leak}_k^b)\)

\textbf{Post-challenge queries:} \( A \) can keep accessing \texttt{LEnc} and \texttt{LDec} with some restrictions

   but it can also get an unlimited access to \texttt{Ldec}

3. \texttt{LEnc}(\( N, A, M \)) returns \( \perp \) if \( N = N_{ch} \) otherwise computes \( C \leftarrow \text{Enc}_k(N, A, M) \)

   and \( \text{leak}_k \leftarrow \text{Lenc}(k, N, A, M) \) and finally returns \( (C, \text{leak}_k) \)

4. \texttt{LDec}(\( N, A, C \)) returns \( \perp \) if \( (N, A, C) = (N_{ch}, A_{ch}, C^0) \) otherwise computes

   \( M \leftarrow \text{Dec}_k(N, A, C) \) and \( \text{leak}_d \leftarrow \text{Ldec}(k, N, A, C) \) and returns \( (M, \text{leak}_d) \)

5. \texttt{Ldec} outputs the leakage trace \( \text{leak}_d^n \leftarrow \text{Ldec}(k, N_{ch}, A_{ch}, C^b) \)

\textbf{Finalization:} \( A \) outputs a guess bit \( b' \) which is defined as the output of the game

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{image1.png}
\caption{The \text{PrivK}\textsuperscript{CCAmL2,b}\textsuperscript{\texttt{AEAD.L}}(1^n) game.}
\end{figure}

encryption and decryption oracles leak (hence the “2” of \texttt{CCAmL2} for the two leaking oracles), including during the challenge query. We go one step further with the \texttt{Ldec} oracle, which offers the leakages corresponding to the decryption of the challenge ciphertext (but of course 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 through side-channels, which might be valuable in applications such as secure bootloading or firmware update with a device controlled by \( A \) \cite{30} (see \cite{10} for more discussion). We let the adversary query the \texttt{Ldec} oracle multiple times, as leakages can be non-deterministic (e.g., contain measurement noise), and \( A \) may benefit from the observation of leakages from multiple decryptions of the same plaintext.

Here, we focus on a single challenge definition but we extend it to the multi-challenge setting in Appendix E.1 where we establish their equivalence.

\textbf{Definition 7 (CCAmL2).} A nonce-based authenticated encryption with associated data \texttt{AEAD} = \( \text{AEAD} = (\text{Gen}, \text{Enc}, \text{Dec}) \) with leakage function pair \( L = (\text{Lenc}, \text{Ldec}) \) is \( (q_e, q_d, q_c, q_l, t, \varepsilon)\)\texttt{-CCAmL2} secure for a security parameter \( n \) if, for every \((q_e, q_d, q_c, q_l, t)\)-bounded adversary \( A \),\footnote{The notation of \( A \) indicates that the adversary may query \( L \) on chosen inputs including chosen keys selected and known by \( A \).} we have:

\[ \Pr[\text{PrivK}_{A,\text{AEAD.L}}^{\text{CCAmL2,0}}(1^n) \Rightarrow 1] = \Pr[\text{PrivK}_{A,\text{AEAD.L}}^{\text{CCAmL2,1}}(1^n) \Rightarrow 1] \leq \varepsilon, \]

where the adversary \( A \) 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.

We will sometimes refer to \texttt{CCAmL2}” as a weakened version of \texttt{CCAmL2} where we drop the challenge decryption leakage oracle \texttt{Ldec}, from the game of Figure 1.

\textit{Chosen-plaintext security with misuse and leakages.} Derived from \texttt{CCAmL2}, \texttt{CPAmL2} is defined by a game \text{PrivK}_{A,\text{AEAD.L}}^{\text{CCAmL2,b}} which is exactly as \text{PrivK}_{A,\text{AEAD.L}}^{\text{CCAmL2,b}} except that we remove \( A \)’s access to the leaking decryption oracle \texttt{LDec} in Figure 1 (Items 2,4). Yet, \( A \) is still able to get challenge decryption leakages \( \text{leak}_d^b \) – this makes \texttt{CPAmL2} a bit theoretical, and only fit into specific settings. We remark that a more natural notion \texttt{CPAmL1}, which can be seen as the \texttt{CPAmL2} variant without \texttt{Ldec}, is subsequently given in section 4.1.
Furthermore, even if the traditional ones in the black-box setting, which is based on the combination of CPA and INT-CTXT security, which we call CIML2, comes from [10] and is an earlier proposal CIML1 [9] extended with decryption leakage. Based on this definition, we propose here the corresponding extension of INT-PTXT security, which we call PIML2.

**Definition 8 (CPAmL2).** A nonce-based authenticated encryption with associated data AEAD = (Gen, Enc, Dec) with leakage function pair \( L = (L_{\text{enc}}, L_{\text{dec}}) \) is \((q_e, q_c, q_l, 1, \varepsilon)-\text{CPAmL2 secure}\) for every \((q_e, q_c, q_l, 1, \varepsilon)\)-bounded adversary \( A \), we have:

\[
\left| \text{Pr}_{A, L\leftarrow\{0,1\}^n} \left[ \text{Priv}_{A,\text{AEAD},L}(1^n) \Rightarrow 1 \right] - \text{Pr}_{A, L\leftarrow\{0,1\}^n} \left[ \text{Priv}_{A,\text{AEAD},L}(1^n) \Rightarrow 1 \right] \right| \leq \varepsilon,
\]

where the adversary \( A^L \) 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 INT-CTXT and INT-PTXT to nonce-misuse resistance and (full) leakages in encryption and decryption. The INT-CTXT extension, called Ciphertext Integrity with Misuse and Leakages, noted CIML2, comes from [10] and is an earlier proposal CIML1 [9] extended with decryption leakage. Based on this definition, we propose here the corresponding extension of INT-PTXT security, which we call PIML2.

**Definition 9 (CIML2, PIML2).** An authenticated encryption AEAD = (Gen, Enc, Dec) with leakage function pair \( L = (L_{\text{enc}}, L_{\text{dec}}) \) provides \((q_e, q_d, q_l, t, 1, \varepsilon)\)-ciphertext (resp. plaintext) integrity with nonce misuse and leakages for security parameter \( n \) if, for all \((q_e, q_d, q_l, t, 1, \varepsilon)\)-bounded adversaries \( A^L \), we have:

\[
\text{Pr}_{A, L\leftarrow\{0,1\}^n} \left[ \text{Priv}_{A,\text{AEAD},L}(1^n) \Rightarrow 1 \right] \leq \varepsilon,
\]

(resp. \( \text{Pr}_{A, L\leftarrow\{0,1\}^n} \left[ \text{Priv}_{A,\text{AEAD},L}(1^n) \Rightarrow 1 \right] \leq \varepsilon \)),

where the security game \( \text{Priv}_{A,\text{AEAD},L} \) (resp. \( \text{Priv}_{A,\text{AEAD},L} \)) is defined in the left part of Table 2 (resp. right part) when \( A^L \) makes at most \( q_e \) leaking encryption queries, \( q_d \) leaking decryption queries and \( q_l \) leakage evaluation queries.

### 3.2 Overall requirement on AE

We eventually require that a secure AE intended to support nonce misuse and full leakages 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 [24]. 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.
Fig. 2: Relations among notions. Arrows (resp., barred arrows) denote implications (resp., separations). Dotted arrows are trivially implied by other relations.

**Definition 10 (AEML-VF).** An AE scheme with security against nonce misuse and full vectorial leakages (denoted as AEML-VF) is an AE scheme \( \text{AEAD} = (\text{Gen}, \text{Enc}, \text{Dec}) \) with a leakage function pair \( L = (L_{\text{enc}}, L_{\text{dec}}) \) satisfying the following assertions: (i) \( \text{AEAD} \) is misuse resistant; (ii) \( \text{AEAD} \) is CIML\(^2\) secure with leakage function \( L \); (iii) \( \text{AEAD} \) is CCAmL\(^2\) 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, CCamL\(^2\) and CIML\(^2\) 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.

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, CCAmL\(^2\) ∧ PIML\(^2\) ↠ CPAmL\(^2\) ∧ CIML\(^2\).

\( \text{MR} \land \text{CPAmL}^2 \land \text{CIML}^2 \not\Rightarrow \text{CCAmL}^2 \). It is not surprising that MR does not imply CCamL\(^2\) since the leakage function \( L \) is absent from the black-box notion. Contrarily, \( L \) appears both in CPAmL\(^2\) and CIML\(^2\). 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 CCamL\(^2\) where the challenge decryption leakage oracle \( L_{\text{decch}} \) is unavailable. While the latter leakages are motivated in the context of side-channel attacks [10], it 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, CPAmL\(^2\), and CIML\(^2\) in the unbounded leakage setting, then there exists an AE with the same security properties but which fails to achieve CCamL\(^2\), even without challenge decryption leakages (i.e., CCamL\(^2\)\(^\ast\)).

The proof, in Appendix A, 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 CPAmL\(^2\) (discussed in Appendix E).

\( \text{MR} \land \text{CCAmL}^2 \land \text{PIML}^2 \not\Rightarrow \text{CIML}^2 \). As for the previous assertion, being MR does not say anything about leakages, so not being CIML\(^2\) is obviously compatible. The most interesting part comes from CCamL\(^2\) and PIML\(^2\) 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, CCamL\(^2\), and PIML\(^2\) in the unbounded leakage model, then there exists an AE which satisfies the same security properties but which fails to achieve CIML\(^2\) (even if not in the unbounded leakage model).

The proof is in Appendix B. It proceeds by building a \( \neg\text{CIML}^2 \) scheme \( \text{AEAD}' \) from an MR ∧ CCamL\(^2\) ∧ PIML\(^2\) scheme \( \text{AEAD} \). An interesting feature is that this counterexample \( \text{AEAD}' \) preserves the tidiness of \( \text{AEAD} \). This deviates from Bellare and Namprempre’s well-known approach for establishing INT-PTXT \( \not\Rightarrow \) INT-CTX, 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 Appendix E.

4.1 Security definition list (single challenge setting)

The CCAmL2 security game PrivK_{\text{AEAD,L}}^{\text{CCAmL2}} 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 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 CIML2 security game PrivK_{\text{AEAD,L}}^{\text{CIML2}} defined in Section 2.3, Table 2. We next formalize these variants.

Prefix-suffix definitions. In all the notions derived from CCAmL2 and CIML2 we only focus on those capturing full leakages. Therefore all the definitions below keep the large L in their notation. This leads us to consider 16 different notions denoted as “pre-suf” with prefix pre ∈ \{CCA, CPA, CI, PI\} and suffix suf ∈ \{M1L1, M2L2, mL1, L1, L2, L1\}: a large “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 pre ∈ \{CCA, CPA\} we obtain the following 8 notions, by starting from CCAmL2 and by removing one security layer at a time:

\[
\text{CCAmL2} \rightarrow \text{CCAmL1}, \text{CCAL2}, \text{CPAmL2} \rightarrow \text{CPAmL1}, \text{CPAL2}, \text{CCAL1} \rightarrow \text{CPAL1}.
\]

**Definition 11.** A nonce-based authenticated encryption with associated data AEAD = (Gen, Enc, Dec) with leakages \(L = (L_{\text{enc}}, L_{\text{dec}})\) is \((q_{\text{pre-suf}}, q_i, t, \varepsilon)\)-pre-suf secure for a security parameter n if, for every \((q_{\text{pre-suf}}, q_i, t)\)-bounded adversary \(A\), we have pre ∈ \{CCA, CPA\} and:

\[
\Pr \left[ \text{PrivK}_{\text{AEAD,L}}^{\text{pre-suf},0}(1^n) = 1 \right] - \Pr \left[ \text{PrivK}_{\text{AEAD,L}}^{\text{pre-suf},1}(1^n) = 1 \right] \leq \varepsilon,
\]

where the adversary \(A\) makes at most \(q_{\text{pre-suf}}\) queries defined in PrivK_{\text{AEAD,L}}^{\text{pre-suf},b} below, and \(q_i\) leakage evaluation queries on arbitrarily chosen keys.

1. PrivK_{\text{CCAmL2,b}}^{\text{CCAmL2}}; \(q_{\text{CCAmL2}} = (q_e, q_d, q_c)\) with the CCAmL2 game in Figure 1.
2. PrivK_{\text{CCAmL1,b}}^{\text{CCAmL1}}; \(q_{\text{CCAmL1}} = (q_e, q_d)\) and the CCAmL1 security game “removes 2” from the CCAmL2 game, meaning that \(L_{\text{dec}}\) is removed from all the oracles. In other words items (2) and (4) become black-box and (5) disappears.
3. PrivK_{\text{CCAL2,b}}^{\text{CCAL2}}; \(q_{\text{CCAL2}} = (q_e, q_d, q_c)\) and the CCAL2 security game “removes M” from the CCAmL2 game which becomes a nonce-respecting.
4. PrivK_{\text{CPAmL2,b}}^{\text{CPAmL2}}; \(q_{\text{CPAmL2}} = (q_e, q_c)\) and no decryption oracle access is given in Figure 1: items (2) and (4) are removed but not item (5), hence the 2.
5. PrivK_{\text{CPAL2,b}}^{\text{CPAL2}}; \(q_{\text{CPAL1}} = (q_e, q_c)\), the CPAL2 game only keeps items (1) and (3) from the CCAmL2 game, (like the CPAmL2 game without \(L_{\text{decch}}\)).
6. PrivK_{\text{CPAL2,b}}^{\text{CPAL2}}; \(q_{\text{CPAL1}} = (q_e, q_c)\), the CPAL2 game is a nonce-respecting version of the CPAmL2 and \(L_{\text{decch}}\) is still available in item (5).
7. PrivK_{\text{CCAL1,b}}^{\text{CCAL1}}; \(q_{\text{CCAL1}} = (q_e, q_d)\) and the CCAL1 is a nonce-respecting version of CCAmL1 (with black-box dec. and nonce-respecting leaking enc.).
8. PrivK_{\text{CPAL1,b}}^{\text{CPAL1}}; \(q_{\text{CPAL1}} = (q_e)\) with only nonce-respecting leaking encryption.
Zoo of integrity notions. For $\text{pre} \in \{\text{CI}, \text{PI}\}$ we obtain the following 8 notions, by starting from $\text{CIML}_2$ and by removing one security layer at a time:

$$\text{CIML}_2 \rightarrow \text{CIML}_1, \text{CIL}_2, \text{PIML}_2 \rightarrow \text{PIML}_1, \text{PIL}_2, \text{CIL}_1 \rightarrow \text{PIL}_1.$$  

**Definition 12.** A nonce-based authenticated encryption with associated data $\text{AEAD} = (\text{Gen}, \text{Enc}, \text{Dec})$ with leakages $L = (\text{L}_{\text{enc}}, \text{L}_{\text{dec}})$ is $(q_e, d_d, q_i, t, \varepsilon)$-pre-suf secure for a security parameter $n$ if, for every $(q_e, q_d, q_i, t)$-bounded adversary $\mathcal{A}^L$, we have $\text{pre} \in \{\text{CI}, \text{PI}\}$ and:

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

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

(i) $\text{Priv}^\text{CIML}_2_{\mathcal{A},\text{AEAD},L}$: the $\text{CIML}_2$ game, see Table 2.
(ii) $\text{Priv}^\text{CIML}_1_{\mathcal{A},\text{AEAD},L}$: the $\text{CIML}_1$ game removes $\text{L}_{\text{dec}}$ (i.e., decryption is black-box).
(iii) $\text{Priv}^\text{CIL}_2_{\mathcal{A},\text{AEAD},L}$: the $\text{CIL}_2$ game is a nonce-respecting version of $\text{CIML}_2$.
(iv) $\text{Priv}^\text{PIML}_2_{\mathcal{A},\text{AEAD},L}$: in the $\text{PIML}_2$ game the winning condition changed (Table 2).
(v) $\text{Priv}^\text{PIML}_1_{\mathcal{A},\text{AEAD},L}$: the $\text{PIML}_1$ game removes $\text{L}_{\text{dec}}$ from $\text{PIML}_2$.
(vi) $\text{Priv}^\text{PIL}_2_{\mathcal{A},\text{AEAD},L}$: the $\text{PIL}_2$ game is a nonce respecting version of $\text{PIML}_2$.
(vii) $\text{Priv}^\text{PIL}_1_{\mathcal{A},\text{AEAD},L}$: the $\text{PIL}_1$ game is a nonce respecting version of $\text{PIML}_2$ free of $\text{L}_{\text{dec}}$.
(viii) $\text{Priv}^\text{CIL}_1_{\mathcal{A},\text{AEAD},L}$: the $\text{CIL}_1$ game is a nonce-respecting version of $\text{CIML}_2$ free of $\text{L}_{\text{dec}}$.

**Connection with previous works.** Among the above sixteen notions, three of them are equivalent to already defined ones: CPAL1 appeared in [32] under the name of LMCPA, CIML1 was introduced in [9] (under the name CIML) and CIML2 was introduced in [10].

4.2 Relations within the zoo (single challenge setting)

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

![Image](image.png)

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.

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

$$\text{CCAmL}_1 \not\equiv \text{CPAL}_2 \quad \text{CCAL}_2 \not\equiv \text{CPAmL}_1 \quad \text{CPAmL}_2 \not\equiv \text{CCAL}_1 \quad \text{CIML}_1 \not\equiv \text{PIL}_2 \quad \text{CIL}_2 \not\equiv \text{PIML}_1 \quad \text{PIML}_2 \not\equiv \text{CIL}_1.$$  

As a corollary, all the arrows of Figure 3 are strict. The proof is given in Appendix C. It only requires to show 4 of the 6 assertions.
Remark #1. By revisiting the proof for $\text{MR} \land \text{CCAmL2} \land \text{PIML2} \nRightarrow \text{CIML2}$ in subsection 3.3, it can be seen that the exhibited CIML2 adversary only relies on the leaking encryption. This means that it also breaks the CIML1 security. Therefore, we already know that $\text{MR} \land \text{CCAmL2} \land \text{PIML2} \nRightarrow \text{CIML1}$.

Remark #2. We discuss the relations between these notions and the Eavesdropper Security with Decryption Leakages (EavDL) introduced in [10] (which is implied by AEML) in Appendix D. We also recall that the treatment of the multi-challenge setting is given in Appendix E.

5 Model and assumptions

Before moving to the description and proof of first modes of operations satisfying the previous security definitions, we now discuss the physical assumptions that will be required for this purpose in more detail. We use this discussion to put forward their similarities with the assumptions required in BMOS.

5.1 Assumptions for CIML2

Starting with the authentication and integrity guarantees, the only assumption required for proving the CIML2 of the following designs is a strongly protected block cipher, that we will formalize as a leak-free SPRP hiding the long-term key, and that will be used a small constant number of times (i.e., 2) per message to encrypt, independently of the length of the message. In our current model, we assume this is perfectly achieved but it is an interesting open problem to investigate how gracefully the security degrades in case of moderate key leakages. For the rest, CIML2 is proven in the very liberal unbounded leakage model introduced in [9]: all the intermediate values computed by our AE designs (excepted the long term-key) can be leaked in full. Note that the need of a PRP (and not a PRF) is a strict one. As discussed in [10], this allows avoiding attacks where valid tags are produced thanks to the decryption leakages of invalid ciphertexts.\footnote{We note that such attacks may even prevent to achieve PIL2.}

Concretely, it is expected that the leak-free block is protected with state-of-the-art countermeasures such as masking, which reduces the amount of leakages at a level depending on the number of shares selected by the implementer. For example, [23,18] described how security levels of up to 128 bits can be achieved on ARM Cortex M4/M8 devices, at the cost of significant performance overheads that our limited use of this strongly protected component aim to limit.

5.2 Assumptions for CCAmL2

Confidentiality guarantees are much harder to obtain with leakages. This was already put forward by Micali and Reyzin in their seminal paper on physically observable cryptography [29], and is also reflected by the combination of stronger assumptions that we will use to prove the CCAmL2 of our AE designs.

Precisely, we will need a (slightly) stronger version of the leak-free block cipher required for CIML2, together with simulatable leakages for the other block ciphers, and will reduce the security of a full AE implementation to the eavesdropper security of a leaking single-block encryption scheme (which is therefore part of our assumptions).

Strongly protected block cipher. Our proofs of CCAmL2 require a strongly protected SPRP that hides its master key, as in our proofs of CIML2. Yet, in this case, we will additionally need that this SPRP hides its output. More precisely, one out of the two strongly protected block ciphers in our following modes of operation will need to enforce this additional feature, which we can explain as follows.\footnote{For this “output-hidden” ideal component, a PRF would be sufficient.} Given a nonce $N$, associated data $A$ and message $M$, the proposals in the next section start by processing a hash of $(N, A)$ which is then sent to the strongly protected SPRP in order to generate a pseudorandom ephemeral key. This ephemeral key must be strongly protected as well, since it will be used to derive all the other keys of the scheme (by contrast, the leakage-resilience of these other keys used in the “re-keying” parts of the modes will only rely on the simulatability assumption).
Note that, if the strongly protected block cipher is implemented as a masked design, this stronger requirement is not expected to lead to improved concrete attacks. For example, one could consider a setup where this cipher outputs several key shares, that are then re-combined as part of the key scheduling of the following block cipher in our encryption schemes. From the DPA viewpoint, it is unlikely that XORing these shares provides significantly more information than loading an unprotected key.

Simulatable leakages. Besides the previous idealized components that are, by far, the most expensive in terms of side-channel countermeasures, our leakage-resilient modes of operation also require that the bulk of the computation does not leak too much (e.g., does not leak the ephemeral keys in full in just two executions). For this part of the computation, weaker and cheaper countermeasures are expected to be sufficient. Various types of limitations on leakages have been proposed in the literature – see Fuller and Hamlin [17] for a review and a comparison (excluding idealized assumptions such as used in [38]). For instance, leakages may be required to be limited in size at each round of computation, as used by Dziembowski and Pietrzak [14] and follow up works [33,12,15], or required to be simulatable [36] as we exploit in the next section.

Admittedly, none of these assumptions is fully convincing. Assuming the leakages to be bounded in size is known to be unrealistic (i.e., it is hopeless to bound the output of an oscilloscope), extensions based on preserving the HILL pseudointeroperty of the leaking secrets are hard to quantify [37], while the first instance of simulator proposed in [36] has been falsified in [26] and it remains an open problem to propose new (implementations of) simulators that withstand the correlation distinguisher put forward by Galea et al.

We selected the simulatability assumption since it provides an elegant way to deal with unrealistic pre-computation / future computation attacks (where the leakages in one round provide exploitable information about a later round) [14,37], and it seems the only standard model assumption for arguing security of the single-pass fresh rekeying schemes without public randomness. In a departure, as far as our main goal of minimizing the adversary’s surface is concerned, proofs in the bounded leakage model (or even proofs using idealized assumptions such as in [38]) would provide good hints that a mode of operation is well suited to resist side-channel attacks as well. In general, we leave the security analysis of our constructions under these complementary leakage assumptions as an important scope for further research.

Formally, the leakage-resilience of FEMALE and AEDT relies on the assumption that leakages satisfy \((p,q)\)-recyclable-simulatability defined below, and based on the \((p,q)\)-\(rsim\)-game in Table 3. This assumption is an extension of the \(q\)-simulatability [36]: \((p,q)\)-recyclable-simulatability is defined as \(q\)-simulatability where each of the \(q\) leakages can be obtained \(p\) times.

<table>
<thead>
<tr>
<th>Game ((p,q))-(rsim) ((\mathcal{A},E,L,S,b)).</th>
</tr>
</thead>
<tbody>
<tr>
<td>The challenger selects two random keys (k,k^* \in_k K). The output of the game is a bit (b') computed by (\mathcal{A}^k) based on the challenger responses to a total of at most (q) adversarial queries of the following type, each repeated at most (p) times:</td>
</tr>
<tr>
<td>Query</td>
</tr>
<tr>
<td>Enc((x))</td>
</tr>
<tr>
<td>and one query of the following type, repeated at most (p) times:</td>
</tr>
<tr>
<td>Query</td>
</tr>
<tr>
<td>Gen((k_{pre}, x))</td>
</tr>
<tr>
<td>Table 3. The ((p,q))-(rsim)-game.</td>
</tr>
</tbody>
</table>

**Definition 13 \(((p,q)\)-recyclable-simulatability of leakages).** Let \(E\) be a PRP with \(L\) as its leakage function. Then the leakages of \(E\) are said to have \((q_S,t_S,q_t,t,\varepsilon_{(p,q)\text{-}rsim})\) \((p,q)\)-recyclable-simulatability, if there exists a \((q_S,t_S)\)-bounded simulator \(S^L\) such that, for every \((q_t,t)\)-bounded adversary \(\mathcal{A}^L\) (making at most \(q_t\) queries to \(L\) and running in time \(t\)), we have:

\[
\Pr[(p,q)\text{-}sim(\mathcal{A},E,L,S,1) \Rightarrow 1] - \Pr[(p,q)\text{-}sim(\mathcal{A},E,L,S,0) \Rightarrow 1] \leq \varepsilon_{(p,q)\text{-}rsim}.
\]

The necessity of the recyclable assumption stems from the CCAmL2 game which offers the challenge decryption leakage oracle \(L_{\text{decch}}\) that was not considered in previous works [9,10] and (in cooperation with
the challenge encryption leakages) may result in the same set of leakages traces being generated more than once. Recyclable simulatability is a stronger assumption than simulatability, since the typical $q$ one considers in leakage-resilient constructions is 2 and the $p$ values allowed by decryption leakages are polynomial.

We note that if only $CCAmL1$ or $CCAmL2^*$ (i.e., the weakened version of $CCAmL2$ from Section 3.1) is required, in which only challenge encryption leakages are generated (and $L_{decch}$ oracle is not available), the original $q$-simulatability is sufficient (with $q = 2$). So overall, what our proofs demonstrate is that security with only challenge encryption leakages for multiple messages and blocks can be reduced to the security of a single block with (noisy, unrepeated) leakages under the assumption that 2 (noisy) leakages can be simulated. By contrast, security with challenge decryption leakages for multiple messages and blocks can only be reduced to the security of a single block with (noise-free, repeated) leakages under the assumption that 2 (noise-free) leakages can be simulated.

Concretely, the recommended designs for (recyclable) simulatable block cipher implementations can be quite different from the strongly protected ones. As already mentioned, leak-free PRPs will typically be implemented with very high-order masking to resist DPA (and thus can be very expensive). By contrast, simulatability is expected to be achievable with weakly protected implementations and essentially requires resistance against SPA with noisy measurements. Similarly, recyclable simulatability requires resistance against SPA with noise-free measurements and could for example be obtained with very low latency hardware (e.g., an unrolled AES implementation in one or two clock cycles [25], limiting the number of noise-free samples that the adversary can collect).

Reduction to a single message block & gradual security degradations. Eventually, our proofs reduce the security of multiple (long) messages to the eavesdropper security of the single-block encryption scheme against CPA, which is not expected to hold with negligible adversarial advantage. This is because the current state-of-the-art does not provide tools to prevent some minimum leakages on plaintext/ciphertext manipulations that inevitably occur when fully leakage-resilient encryption schemes are considered (i.e., when the leakages of the challenge query is given to the adversary).

Yet, as mentioned in the Introduction, we achieve security-preserving domain extension of the single-block encryption (an important direction in theory), and we add mechanisms to render decryption leakage harmless (to resist CCA rather than mere CPA). There is (clearly) a significant theoretical difference between such security-preserving modes of operation (as considered next) and the non-preserving ones. In practice, the former enforce that DPA attacks are hard(er) to mount, for example thanks to re-keying which is usually instrumental to enable the reductions. This is what we informally denote as minimizing the adversary’s surface.

In this respect, it is important to mention that the stronger nature of these confidentiality assumptions is actually the main motivation for the gradual security degradations that our aggregate definition of $AEML$ allows. That is, even if the confidentiality of a message is reduced due to a powerful side-channel attack against the single-block encryption scheme or the (recyclable) simulatability assumption, it will have no impact on other messages.

5.3 Leveled leakage setting

The previous descriptions allow us to clarify the meaning of leveled implementations. In general and informally, leveled implementations mix different types of cipher designs, with different levels of protections against side-channel attacks. In our specific case, leveled implementations mix two types of block cipher designs: a minimally used and strongly protected one that has to resist side-channel adversaries able to collect the leakages corresponding to polynomial number of inputs (i.e., a DPA), and an intensely used and weakly protected one that only has to resist side-channel adversaries able to collect the leakages corresponding to a small constant number of inputs (equivalent to an SPA).

5.4 Relation with BMOS

Despite it is hard to prove that our strongly protected SPRP and our leakage assumptions are necessary (which is another important and challenging open problem), we observe that the BMOS work requires very similar ones.
**Strongly protected block.** The BMOS [5] security proofs are based on very similar assumptions, and require an ideal component that essentially solves the same challenges as ours which can be viewed as a counterpart to our leak-free SPRP, as we explain next.

Given \((N, A, M)\), the BMOS construction combines an IV-based encryption scheme with a pseudorandom Hash-then-MAC. To verify a tag \(T\), the proposed implementation described in the appendices of the long version of [5] recomputes fresh shares, say \(T_1\) and \(T_2\), of the right tag \(T_1 \cdot T_2\) by applying a PRF on its hashed value, leading to the following observations.

First, the key of this PRF has to be “well protected”. The BMOS authors propose an instantiation based on pairings which is proven in the generic group model for this purpose. Critically, these shares are directly processed to test whether \(H'(T/T_1) = H'(T_2)\), and the proof requires \(H'\) to be modeled as a random oracle. This allows its output to be pseudorandom which prevents attacks targeting the correct tag based on incorrect verifications. However, it implicitly requires that the output of the well protected PRF and the input of the random oracle must be leak-free (since there are no other arguments ruling out attacks exploiting the leakages of this intermediate value) because their definition requires ciphertexts (and hence, tags) to remain pseudorandom even if the adversary keeps a real leaking decryption (and hence, verification) access.

This second issue is actually identical to the one mentioned earlier that one of the strongly protected block cipher in our AE schemes must hide its output. In fact, by tweaking the verification process of BMOS as proposed in [10], one could even replace the leak-free random oracle requirement by a leak-free PRF protecting its input (which would then be combined with their first leak-free PRF protecting its output, which is the one we also need).

As a result, both BMOS’ and our following constructions require the same type of ideal component. Yet, it is worth observing that only our model and constructions allow a gradual security degradation. That is, the stronger requirement to perfectly hide one of our leak-free PRF’s output only impacts the \(\text{CCAmL2}\) security of our schemes. By contrast, it is already required for integrity guarantees in BMOS.

**Bounded leakage.** For the rest, the BMOS proofs require that each execution of their well protected PRF only leaks a bounded amount of bits per call (i.e., bounded leakages). As already mentioned, this requirement is conceptually similar to our requirement of simulatable leakages (i.e., it aims at avoiding that ephemeral secrets are leaked in full) and analyzing our following constructions under this complementary leakage assumption is an interesting open problem.\(^6\)

### 6 First instantiations: FEMALE and AEDT

We finally present FEMALE and AEDT, two AE schemes making only two calls to a strongly protected block cipher and enabling leveled implementations. FEMALE is AEML. AEDT only offers both CIML2 and \(\text{CCAmL2}\) but is more efficient.

#### 6.1 FEMALE design and analysis

**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.

**Description.** Given a hash function \(H : \{0, 1\}^* \rightarrow \mathcal{B}\) 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:

(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 \(M = (m_1, \ldots, m_t)\) are “pre-encrypted” as \(d_i\), resulting in \(d = (d_1, \ldots, d_t)\), where \(d_i\) depends on \((N, A, m_1, \ldots, m_i)\);

---

\(^6\) In the case of BMOS, the systematic exploitation of shared computations (based on pairings) also prevents the issue of precomputation / future computation attacks that is typical of symmetric constructions (which is an additional motivation for using the simulatability assumption in our analyzes).
(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 \(R = H(0\|N\|A)\), 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.

**Security Analysis.** We show that **FEMALE** satisfies AEML security. In the black-box setting, MR follows from the fact that \(V\) is pseudorandom on \((N, A, M)\) even for a variable-length \(M\). This is because \(W\) remains secret and already depends on all the (ordered) input. Computing and giving \(V = E_w(W)\) rather than \(W\) prevents continuing the computation of the first stage of the encryption algorithm from multiple encryption queries (and captures the message length).

The **CIML2** property mainly stems from the last stage of the encryption algorithm and how this stage is verified from the tag in the decryption algorithm. The result follows the technique used in [10] where tag verifications are made by inverting the SPRP (which is useless to produce valid fresh tags).

Therefore, our main focus is on **CCAmL2** security. To make it formal, we define the leakage function \(L = (L_{\text{enc}}, L_{\text{dec}})\) of **FEMALE** as:

- \(L_{\text{enc}}\) consists of the leakages that are generated during the encryption:
  - the leakages \(L_E(s, x)\) generated by all the internal calls to \(E_s(x)\), and
  - the leakages \(L_{\oplus}(a, b)\) generated by all the internal actions \(a \oplus b\), and
  - all the intermediate values involved in the computations of the hash functions (i.e., hash functions are non-protected, and leak everything).

- \(L_{\text{dec}}\) consists of the above that are generated during the decryption.

Our security reduction is made against (i) the simulatability of the leaking blocks, (ii) the security of the encryption of one single block with a fresh key.

Following Pereira et al.'s approach [32], we consider a Leaking Real Single-block Encryption scheme **LRSE** defined in Figure 6 as the basic unit of **FEMALE**. As motivated in the previous section, it is introduced to
where secure in the sense that for any \((L_m LRSE)\)
\[ \text{Description of } RSDec \sigma \text{ with leakage function } p \]
\[ \text{Theorem 4 (informal).} \]
\[ \text{Let } \text{result is informally given below. The precise bounds for both theorems below can be found in Appendix F.} \]
\[ \text{Description of LRSE scheme:} \]
\[ \text{RSGen}(1^n) \text{ picks } k_{ch} \in \{0,1\}^n, M, C = \{0,1\} \ltimes (p_0, p_1) \in \{0,1\} \text{ can be chosen by the adversary with } p_0 \neq p_1 \]
\[ \text{RSEnc}_{k_{ch}}(m) \text{ returns } (k_{up}, c), \text{ where } c = y_{ch} \oplus m, y_{ch} = E_{k_{ch}}(p_1), \text{ and } k_{up} = E_{k_{ch}}(p_0). \]
\[ \text{RSDec}_{k_{ch}}(c) \text{ proceeds in the natural way.} \]
\[ \text{The leakage } L_{RSE} = (L_{senc}, L_{udec}, k_{pre}) \text{ resulting from the LRSE implementation is defined as } \]
\[ L_{senc}(k_{ch}, m) = (L_{E}(k_{ch}, p_0), L_{E}(k_{ch}, p_1), L_{S}(y_{ch}, m), S^c(k_{pre}, p_0, k_{ch})), \]
\[ L_{udec}(k_{ch}, c) = (L_{E}(k_{ch}, p_0), L_{E}(k_{ch}, p_1), L_{S}(y_{ch}, c), S^c(k_{pre}, p_0, k_{ch}))) \text{ for a fixed random } k_{pre} \in \{0,1\}^n. \]
\[ \text{As usual we denote } \text{LRSE}_{k_{ch}}(m) = (\text{RSEnc}_{k_{ch}}(m), L_{senc}(k_{ch}, m)). \]
\[ \text{Fig. 5: The FEMALE AEAD scheme.} \]
\[ \text{Fig. 6: Basic unit: the single-block encryption scheme LRSE.} \]
\[ \text{determine the CCAmL2 security bound. Since for each generated key } k_{ch}. \]
\[ \text{LRSE is used to encrypt a single message } m \text{ composed of a single block, we assume that given a security parameter } n, \text{ LRSE is } (p, q_t, t, \varepsilon_{s\text{-block}}) \text{ secure in the sense that for any } (q_t, t) \text{-bounded eavesdropper adversary } A^{RSE} \text{ choosing } m^0, m^1 \in M, \text{ we have:} \]
\[ \text{Pr}[A^{RSE}(\text{LRSE}_{k_{ch}}(m^0)) \Rightarrow 1] - \text{Pr}[A^{RSE}(\text{LRSE}_{k_{ch}}(m^1)) \Rightarrow 1] \leq \varepsilon_{s\text{-block}}, \]
\[ \text{where } \text{LRSE}_{k_{ch}}(m^0) = (\text{LRSE}_{k_{ch}}(m^0), [L_{sdec}(k_{ch}, c^p)])^{p-1}, k_{pre}) \text{ for } (c^p, k_{up}) = \text{RSEnc}_{k_{ch}}(m^0) \text{—here the superscript } p-1 \text{ indicates the adversary could measure the trace } p-1 \text{ times (this typically doesn't give identical traces). The reason why the adversary also gets the auxiliary outputs } k_{pre} \text{ and } k_{up} \text{ is for composability purpose (which appear in the proof).} \]
\[ \text{With the above, our CCAmL2 result is informally given below. The precise bounds (for both theorems below) can be found in Appendix F.} \]
\[ \text{Theorem 4 (informal).} \]
\[ \text{Let } H : \{0,1\}^* \rightarrow B \text{ be a } (0, t', \varepsilon_{r}) \text{-collision-resistant and } (q_d, t', \varepsilon_{pr}) \text{-range-oriented preimage resistant hash function. And let } E : M \times M \rightarrow M \text{ be a } (2q_e + 2q_d + 2, t', \varepsilon_{-SPR}) \text{ with two implementations: a strongly protected implementation } E' \text{ is leak free, and a plain implementation } E \text{ have leakage function } L_{E} \text{ that is } (q_s, t, q_t, t, \varepsilon_{(p, 2)-Rsim}) \text{ (p, 2)-R-simulatable. Then the FEMALE implementation with leakage function } L = (L_{senc}, L_{sdec}) \text{ is } (q_s, q_d, p - 1, q_t, t, \varepsilon_{CCAmL2}) \text{ CCAmL2-secure, with } \]
\[ \varepsilon_{CCAmL2} \leq O \left( \frac{t' \ldots q_t}{2^n} + O \left( \frac{(t' \ldots q_t + q_d)}{2^n} \right) + O \left( \frac{t \cdot t' + \sigma^2}{2^n} \right) + O \left( \sigma \varepsilon_{(p, 2)-Rsim} + \varepsilon_{s\text{-block}} \right) \right), \]
\[ \text{where } \sigma \text{ denotes the total number of blocks in the challenge messages.} \]
The proof is given in Appendix F. The term $O(\sigma(\varepsilon_{(p,2)},rsim+\varepsilon_{s\text{-block}}))$ corresponds to the security degradation in the presence of side-channel leakage & the reduction to a single block encryption. The bound is roughly of birthday type, and it shows FEMALE is secure up to $t = 2^{n/2} \cdot c^{-1}$ computation and encrypting $2^{n/2} \cdot c^{-1}$ message blocks, where $c \geq 1$ depends on the side-channel strength of the concrete (plain) implementation $E$.

The CIML2 and MR security of FEMALE are finally presented below.

**Theorem 5 (informal).** Let $H : \{0,1\}^* \to B$ be a $(0,t',\varepsilon_{cr})$-collision resistant and $(q_d + 1, t', \varepsilon_{pr})$-range-oriented preimage resistant hash function. And let $E : \mathcal{M} \times \mathcal{M} \to \mathcal{M}$ be a $(2q_e + 2q_d + 2, t', \varepsilon_E)$-SPRP, with $E^*$ being its strongly protected leak free implementation. Then:

1. **FEMALE** provides $(q_e, q_d, t, \varepsilon_{CIML2})$-ciphertext integrity with coin misuse and unbounded leakage on encryption and decryption as long as $t \leq t' - O(q_e + q_d)$, where $\varepsilon_{CIML2} \leq O\left(\frac{t^2}{2^n} + \frac{q_e + q_d}{n^2}\right)$;
2. **FEMALE** scheme is $(q_e, q_d, t, \varepsilon_{mr})$-MR as long as $t \leq t' - O(q_e + q_d)$, where $\varepsilon_{MR} \leq O\left(\frac{t^2}{2^n} + \frac{q_e + q_d}{n^2}n^2\right)$, $\sigma$ denotes the total number of blocks in the encryption queries.

The proofs are in Appendices F.3 and F.4. Both bounds are roughly birthday.

### 6.2 AEDT design and analysis

AEDT is a simplified version of FEMALE achieving only AEml security, that is, dropping the black-box MR requirement. The AEDT encryption algorithm has only two stages: given $(N, A, M)$ with an $\ell$-block message $M$, it computes the hash $R = H(0\|N\|A)$, and then just uses $E^*_1(0\|R)$ as the initial key to “start” a one-time encryption of $M$ (in the nonce-respecting case this process will indeed occur once). Then, AEDT applies the same authentication principle as FEMALE. The encryption algorithm is depicted in Figure 7.

AEDT uses 2 leak-free blocks, just as FEMALE, but is twice more efficient in terms of number of weakly protected blocks to evaluate. Furthermore, its encryption can be performed “on-the-fly” by pushing each ciphertext block $c_i$ directly into the hash function (if it proceeds block by block).

Fig. 7: (Top) Initialization and encryption parts of AEDT, with notations to be used in its specification and security analysis. The meaning of $E^{*\text{h}}$ is similar to Figure 4. (Bottom) Authentication part of AEDT, with notations consistent with the top half. The final output is $C = (c, T)$.

The CIML2 security of AEDT follows from the one of the EDT mode in [10]. Its CCAmL2 security is is analyzed in a similar way as FEMALE and is deferred to Appendix G, resulting in similar bounds.

### References


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


A Proof of Theorem 1

Let $A_{\text{AEAD}} = (\text{Gen}, \text{Enc}, \text{Dec})$ with leakage function $L = (L_{\text{enc}}, L_{\text{dec}})$ be MR, CPA_{\text{AmL2}} with respect to $L$ and CIML2 with respect to $L^*$ such that, for a fixed message $M^1 \in \mathcal{M}$:

- $\text{Gen}^{(1^n)}$: returns $k \leftarrow \text{Gen}^{(1^n)}$ and $k' \leftarrow \text{Gen}^{(1^n)}$;
- $L'_{\text{enc}}((k, k'), N, A, M)$; outputs $(\text{leak}_e, C', \text{leak}_r)$ where $\text{leak}_e = L_{\text{enc}}(k, N, A, M)$ (comes from the computation of $C' \leftarrow \text{Enc}_k(N, A, M)$ and consequently $\text{leak}_r = L_{\text{dec}}(k', N, A, M)$); $L'_{\text{dec}}((k, k'), N, A, C)$; outputs $\text{leak}_d = L_{\text{dec}}(k, N, A, C)$ if $M \neq \perp$ (which comes from the computation of $M \leftarrow \text{Enc}_k(N, A, C)$) and outputs $(\text{leak}_d, C^l, \text{leak}_e^l)$ otherwise, where $\text{leak}_d = L_{\text{dec}}(k, N, A, C), C^l \leftarrow \text{Enc}_{k'}(N, A, M^1)$ and consequently also $\text{leak}_e^l = L_{\text{enc}}(k', N, A, M^1)$.

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

**CPA_{\text{AmL2}}.** In the $\text{PrivK}^\text{C_{\text{AmL2}, b}}$ game, the adversary $A'$ does not have access to $L'_{\text{dec}}$ except from the challenge decryption leakage through $L'_{\text{dec}}$. But since the challenge ciphertext is valid, $L_{\text{dec}} = L_{\text{dec}}$ which returns $L_{\text{dec}}(k, N, A, C, \cdot).$ Consequently, an adversary $A$ in $\text{PrivK}^\text{C_{\text{AmL2}, b}}$ can easily simulate the view of $A'$ simply by picking $k' \leftarrow \text{Gen}^{(1^n)}$ transmitting all the queries to its own oracles and just add the encryption leakage $(C', \text{leak}_e)$ if necessary.

**CIML2.** In the $\text{PrivK}^\text{C_{\text{AmL2}, b}}((L')^*)$ game, the adversary $A'$ still needs to forge a fresh ciphertext of AEAD with key $k$ while the additional unbounded leakage given by $(L')^*$ only depends $k'$. Then, building a reduction to $\text{PrivK}^\text{C_{\text{AmL2}, b}}$ is straightforward.

**CC_{\text{AmL2}, b}**. We build a distinguisher $A'$ against $A_{\text{AEAD}}$. In the security game $\text{PrivK}^\text{C_{\text{AmL2}, b}}$, the adversary queries leaking decryption of $(N, A, C)$ for any chosen $N, A$ 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 $(\perp, (\text{leak}_e, C^l, \text{leak}_r^l))$ from $L_{\text{dec}}(N, A, C)$ and sets $(M^0, C^0) = (M^1, C^1)$. In the challenge phase, $A'$ sends $(N, A, M^0, M^1)$ for any distinct $M^1$ than $M^0$. Since the pair $(N, A)$ has never been queried for (leaking) encryption, $A'$ does not receives $\perp$. In the answer $L_{\text{enc}}(N, A, M^0), A'$ gets $C^0$. If $C^0$ equals the known $C^0$, $A'$ outputs 0, otherwise it outputs 1. Obviously the distinction holds with probability 1.

Now, it is easy to see that $A_{\text{AEAD}}'$ with leakage $L'$ fulfills all the desired requirements of the theorem based on the existence of $A_{\text{AEAD}}$.

B Proof of Theorem 2

Let $A_{\text{AEAD}} = (\text{Gen}, \text{Enc}, \text{Dec})$ with leakage function $L = (L_{\text{enc}}, L_{\text{dec}})$ be MR, CC_{\text{AmL2}} with respect to $L$ and PI_{\text{M2}} with respect to $L^*$ as such authenticated encryption exists by assumption. Then we build $A_{\text{AEAD}}' =$
(Gen', Enc, Dec) with leakage L' = (L'_{enc}, L_{dec}) as follows, where N^\dagger, A^\dagger, M^\dagger, N^\circ, A^\circ, M^\circ below are the outputs of a publicly samplable distribution parametrized by n:

Gen'(1^n): generates k ← Gen(1^n). Then, it selects distinct nonces N^\dagger, N^\circ ∈ N, distinct A^\dagger, A^\circ ∈ AD and distinct messages M^\dagger, M^\circ ∈ M. It computes the ciphertext C^\dagger ← Enc_k(N^\dagger, A^\dagger, M^\dagger) and splits it into four random shares: it samples three |C^\dagger|-bit strings R_0, R_1, S_0 and sets S_1 = C^\dagger ⊕ R_0 ⊕ R_1 ⊕ S_0. It outputs (k, sh) where sh = (R_0, R_1, S_0, S_1).

L'_{enc}((k, sh), N, A, M): outputs leak_e = L_{enc}(k, N, A, M) (which comes from the computation of the ciphertext C ← Enc_k(N, A, C)) as well as the additional value B but only in four cases:
- Case 1.1: (N, A) = (N^\dagger, A^\dagger), M ≠ M^\circ: B = R_0;
- Case 1.2: (N, A) = (N^\dagger, A^\circ), M = M^\circ: B = R_1;
- Case 2.1: (N, A) = (N^\circ, A^\dagger), M ≠ M^\dagger: B = S_0;
- Case 2.2: (N, A) = (N^\circ, A^\circ), M = M^\dagger: B = S_1.

If we drop the leakage functions AEAD' shows no deviation from AEAD and thus is still MR. It remains to establish the desired leakage-related claims:

CCAmL2. We prove that if there is a CCAmL2 adversary A' against AEAD', then there is an adversary A which uses A' to break the CCAmL2 security of AEAD. In detail, once PrivK_{A,A'AEAD,L'}^\text{CAmL2,b} (1^n) is setup, A(1^n) publicly sample N^\dagger, A^\dagger, M^\dagger, N^\circ, A^\circ, M^\circ and sends to A' whatever specified by AEAD'. Even if the ciphertext C^\dagger = Enc_k(N^\dagger, A^\dagger, M^\dagger) has never been computed its length is fully determined by n and (N^\dagger, A^\dagger, M^\dagger). Therefore, A picks random |C^\dagger|-bit strings R_1, S_0, S_1. Then it runs A' and simulates the game PrivK_{A,A'AEAD,L'}^\text{CAmL2,b} using its own interaction PrivK_{A,A,AEAD,L'}^\text{CAmL2,b}. For each query from A', the actions of A are as follows:

Leaking (non-challenge) encryption queries: On input (N, A, M),
1. If N = N^\dagger appears for the first time in a leaking encryption query (and so not in the challenge query) A queries a leaking encryption involving (N^\dagger, A^\dagger, M^\dagger) to its own oracle and gets back C^\dagger and then computes R_0 = C^\dagger ⊕ R_1 ⊕ S_0 ⊕ S_1;
2. A queries its own leaking encryption oracle on (N, A, M) and gets back some (C, leak_e) or possibly ⊥ (for forbidden queries);
3. A checks whether it should append an additional leakage B to (C, leak_e) depending on (N, A, M) falls into one of the four cases described above.

A respectively returns (C, leak_e) or ⊥ or even (C, leak_e, B) to A' according to the above situations.

Leaking decryption queries: on input (N, A, C), A simply calls its own oracle and reply to A' with the answer it received.

Leaking challenge query: on input (N_{ch}, A_{ch}, M^0, M^1), A sends it to its leaking challenge oracle and gets the tuple (C^b, leak_{cb}) or possibly ⊥. If not ⊥,
1. if (N_{ch}, A_{ch}) = (N^\dagger, A^\dagger), A returns (C^b, leak_{cb}, R_1).
2. if (N_{ch}, A_{ch}) = (N^\circ, A^\circ), A returns (C^b, leak_{cb}, S_0) regardless of M^0, M^1;
3. else, A just returns (C^b, leak_{cb}).

Challenge decryption leakage: returns the corresponding decryption leakage of the challenge ciphertext.

Eventually, A outputs the bit returned by A'.

Now we explain why A properly emulates the CCAmL2 game in front of A'. First, as long as C^\dagger is returned to A in a leaking encryption queries A' can not make a leaking challenge query with N_{ch} = N^\dagger and then case (i) in the challenge phase will not occur since A will receive ⊥ and will send it to A' as expected by the game. Therefore, B = R_0 or B = R_1 cannot be among the encryption leakage of the challenge ciphertext. Second, if N^\circ appears at first in a leaking encryption query, case (ii) of the challenge phase will not occur and B = S_0 or B = S_1 will never be among the encryption leakage of the challenge ciphertext as in AEAD'. We stress that R_0, R_1, S_0, S_1 might appear several times in next responses to leaking encryption queries but exactly as in the honest run of PrivK_{A,A'AEAD,L'}^\text{CAmL2,b}. Third, if N_{ch} = N^\dagger appears for the first time in the challenge phase, case (i) of the leaking encryption query phase will not occur and A will receive ⊥ and send it to A' if A' queries a leaking encryption involving N^\dagger afterwards, which corresponds to the right view of the game.
Therefore, $R_0$ will never be defined and if we also have $A_{ch} = A^0$ the additional leakage $B^0 = R_1$ will be given in the encryption leakage of the challenge ciphertext. Here it is easy to see, since $A'$ will receive a single share among those specified in Case 1.1 and Case 1.2 of $L^{'\text{enc}}$: that this distribution is exactly as the one expected from $\text{AEAD}'$. Four, if $N_{ch} = N^0$ appears for the first time in the challenge phase, a similar argument shows that $A'$ will only get a single share among those specified in Case 1.1 and Case 1.2 of $L^{'\text{enc}}$ and only if $A_{ch} = A^1$ independently of the choice $M^0, M^1$. Once again, this is exactly the right distribution where $B^0 = S_0$ is random and remains independent of the rest of the game. Five, all the previous arguments show that in any of these situations the potential additional leakage $B$ ($B^0$ included) can be perfectly emulated by $A$ so that we have

$$\Pr[PrivK^{CCAmL_2,b}_{A,\text{AEAD},L}(1^n) \Rightarrow 1] = \Pr[PrivK^{CCAmL_2,b}_{A',\text{AEAD}',L}(1^n) \Rightarrow 1].$$

Finally, in any other cases the above conclusion obviously holds as well.

\textbf{PIML2.} We prove that if there is a PIML2 adversary $A'$ against $\text{AEAD}'$, then there is an adversary $A$ which uses $A'$ to break the PIML2 security of $\text{AEAD}$. In detail, once $PrivK^{PIML_2,b}_{A,\text{AEAD},L}(1^n)$ is setup, $A(1^n)$ publicly sample $N^1, A^1, M^1, N^0, A^0, M^0$ and sends to $A'$ whatever is specified by $\text{AEAD}'$. Since it is assumed that the size of $C^1 = \text{Enc}_k(N^1, A^1, M^1)$ is known, $A$ can pick random $|C|^\text{-bit strings}$ $R_0, R_1, S_0$. Then it runs $A'$ and simulates $PrivK^{PIML_2,b}_{A',\text{AEAD}',L}$ using its own interaction $PrivK^{PIML_2,b}_{A,\text{AEAD},L}$ such that any query made by $A'$ is simply relayed by $A$ to its oracles and answered back with the answer possibly augmented with an additional encryption leakage according to the different four cases defined in $L^{'\text{enc}}$ except in Case 2.2, namely if $A'$ requests an encryption of $(N^0, A^1, M^1)$. In the latter case, $A$ first queries an encryption of $(N^1, A^1, M^1)$ and gets back $C^1$. Then, $A$ proceeds as in the other cases but adds the leakage $S_1 = C^1 \oplus R_0 \oplus R_1 \oplus S_0$.

It should be straightforwards that the simulation is perfect. Furthermore, as long as $A'$ does not get the share $S_1$ all the information in its view is exactly the same as the information $A$ gets in its PIML2 view and therefore any associated-data/plaintext forgery computed by $A'$ is a valid forgery against $\text{AEAD}$. In order to get some information about $C^1$, $A'$ must receive the four shares $R_0, R_1, S_0, S_1$ but it means that it queried $(N^0, A^1, M^1)$ but then $C^1 = R_0 \oplus R_1 \oplus S_0 \oplus S_1$ is no more considered as a valid forgery since the pair $(A^1, M^1)$ is already involved in a leaking encryption query. As a conclusion $A$ never makes a leaking encryption query involving $(A, M)$ which has never been among a leaking encryption query made by $A'$ at the first place.

Note that this reduction works in the unbounded leakage setting as well.

\textbf{–CiML2.} We mount a ciphertext forgery attack against $\text{AEAD}'$. Let $A'$ be an adversary which given $N^1, A^1, M^1, N^2, A^2, M^2$ sequentially queries a leaking encryption on $(N^1, A^1, M^1), (N^1, A^2, M^2), (N^2, A^1, M^2)$, and $(N^2, A^2, M^1)$ which fall into the four cases where each one of the shares $R_0, R_1, S_0, S_1$ are additionally given in the encryption leakage. Then, $A'$ output $C^1 = R_0 \oplus R_1 \oplus S_0 \oplus S_1$ which is a valid encryption of $(N^1, A^1, M^1)$ which has never been received as an answer to some leaking encryption query. \hfill $\square$

As a side note, (i) it can be seen the leakage functions $L' = (L^{\text{enc}}, L^{\text{dec}})$ of the counterexample $\text{AEAD}'$ preserves the determininsticness of the original one $L = (L_{\text{enc}}, L_{\text{dec}})$; (ii) we never require that $|M| > 2$.

\section{Proof of Theorem 3}

Recall that we are to prove 12 non-implications:

(i) $\text{CCAmL1} \not\Rightarrow \text{CPAL2}$,
(ii) $\text{CPAL2} \not\Rightarrow \text{CCAmL1}$,
(iii) $\text{CCAL2} \not\Rightarrow \text{CPAmL1}$,
(iv) $\text{CPAmL1} \not\Rightarrow \text{CCAL2}$,
(v) $\text{CPAmL2} \not\Rightarrow \text{CCAL1}$,
(vi) $\text{CCAL1} \not\Rightarrow \text{CPAmL2}$,
(vii) $\text{CiML1} \not\Rightarrow \text{PIL2}$,
(viii) $\text{PIL2} \not\Rightarrow \text{CiML1}$,
(ix) $\text{CIL2} \not\Rightarrow \text{PIML1}$,
(x) $\text{PIML1} \not\Rightarrow \text{CIL2}$,
We first show that a security notion \( X_1 \) without decryption leakages cannot imply the corresponding notion \( X_2 \) with decryption leakages. This would establish six separations (i), (iv), (vi), (vii), (x), and (xii). For this, assume that AEAD is a \( X_1 \) secure scheme with master-key \( K \). We define a new scheme \( \text{AEAD}^* \), which is the same as AEAD except that its leakages for decryption queries explicitly include the master-key \( K \). In this way, \( \text{AEAD}^* \) is clearly not \( X_2 \) secure (as the key is leaked). But it remains \( X_1 \) secure, since this enhancement of decryption leakage cannot be observed in the \( X_1 \) security game.

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

1. \( \text{Gen}'(1^n) \): generates two keys \( k \leftarrow \text{Gen}(1^n) \) and \( k' \leftarrow \text{Gen}(1^n) \), and selects a public pair \( (N^\dagger, A^\dagger) \).
2. \( L_{\text{enc}}((k, k'), N, A, M) \): outputs \( \text{leak}_{L_{\text{enc}}} = L_{\text{enc}}(k, N, A, M) \) 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, \( \text{AEAD}^* \) 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 \( \text{CPAmL}_2 \Rightarrow \text{CCAL}_1 \) and \( \text{PIML}_2 \Rightarrow \text{CIL}_1 \). For this we follow the standard idea of showing \( \text{CCA} \Rightarrow \text{CPA} \) and \( \text{INT-PTXT} \Rightarrow \text{INT-CTXT} \). In detail, consider \( \text{CPAmL}_2 \Rightarrow \text{CCAL}_1 \) first, and assume that \( \text{AEAD} = (\text{Gen}, \text{Enc}, \text{Dec}) \) is \( \text{CPAmL}_2 \) secure. We define a new scheme \( \text{AEAD}^* = (\text{Gen}, \text{Enc}, \text{Dec'}) \) as follows:

1. \( \text{Dec}_k'(N, A, C) \): outputs \( \text{Dec}_k(N, A, C) \| k \), 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 \( \text{CPAmL}_2 \Rightarrow \text{CCAL}_1 \).

For \( \text{PIML}_2 \Rightarrow \text{CIL}_1 \), assume that \( \text{AEAD} = (\text{Gen}, \text{Enc}, \text{Dec}) \) is \( \text{PIML}_2 \) secure. We define a new scheme \( \text{AEAD}^* = (\text{Gen}, \text{Enc'}, \text{Dec'}) \) as follows:

1. \( \text{Enc}_k'(N, A, M) \): outputs \( \text{Enc}_k(N, A, M) \| 0|0 \), i.e., two bits are appended to the ciphertext.
2. \( \text{Dec}_k(N, A, C) \): parses \( C = C'' \| b|b' \), and outputs \( \text{Dec}_k(N, A, C') \) if and only if \( b = b' \).

Then it’s clear that \( \text{AEAD}^* \) is not \( \text{CIL}_1 \) 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 \( \text{PIML}_2 \) secure.

## D EavDL and AEML

The notion EavDL was introduced by Berti et al. [9]. It formalizes message confidentiality in a context where an adversary can observe decryption leakages but not the corresponding messages. This setting is motivated by applications such as secure bootloading and bitstream decryption. In this section, we recall this notion, and make discussion on the links between this notion and the other ones.

### D.1 mEavDL and its extension mEavDL2

The original definition given by Berti et al. is of a single-challenge form. To save space, we concentrate on its multi-challenge variant \( \text{mEavDL} \), which is based on the experiment in Fig. 8.

Note that the challenge nonce has to be respecting for both \( \text{mEavDL} \) and \( \text{mEavDL2} \). If we restrict the number of challenges to 1 in \( \text{mEavDL} \) then we recover the original EavDL notion of Berti et al. [9].
and runs in time $t$ secure for short, if for any adversary $\mathcal{A}^{E}$

against eavesdropping with differential leakage attacks for a security parameter $\lambda$. Definition 14 (mEavDL: Eavesdropper Security with Decryption Leakage). An authenticated encryption AEAD = (Gen, Enc, Dec) with decryption leakage function $\mathcal{L}_{\text{dec}}$ provides $(q_{m}, q_{d}, t, \varepsilon)$-indistinguishability of ciphertexts against eavesdropping with differential leakage attacks for a security parameter $\lambda$, or is $(q_{m}, q_{d}, t, \varepsilon)$-mEavDL secure for short, if for any adversary $\mathcal{A}$ that makes at most $q_{m}$ challenge queries, $q_{d}$ queries to $\mathcal{L}_{\text{dec}}$, and runs in time $t$,

\[
\Pr[PrivK^{mEavDL,b}_{\mathcal{A}^{E},\text{AEAD}}(1^{\lambda}) \Rightarrow 1] \leq \frac{1}{2} + \varepsilon
\]

for the $PrivK^{mEavDL,b}_{\mathcal{A}^{E},\text{AEAD}}$ game defined in Fig. 8.

**Stronger variant: mEavDL2.** We define a strengthened variant mEavDL2, which is mEavDL enhanced with challenge encryption leakages so that the “2” means that both the leaking oracles in encryption and decryption are available.

**Definition 15 (mEavDL2: Eavesdropper Security with Encryption & Decryption Leakage).** An authenticated encryption AEAD = (Gen, Enc, Dec) with leakage function pair $L = (\mathcal{L}_{\text{enc}}, \mathcal{L}_{\text{dec}})$ provides $(q_{m}, q_{d}, t, \varepsilon)$-indistinguishability of ciphertexts against eavesdropping with differential leakage attacks in encryption and decryption for a security parameter $\lambda$, or is $(q_{m}, q_{d}, t, \varepsilon)$-mEavDL2 secure for short, if for any adversary $\mathcal{A}$ that makes at most $q_{m}$ leaking challenge queries, $q_{d}$ queries to $\mathcal{L}_{\text{dec}}$, and runs in time $t$,

\[
\Pr[PrivK^{mEavDL2,b}_{\mathcal{A}^{E},\text{AEAD}}(1^{\lambda}) \Rightarrow 1] \leq \frac{1}{2} + \varepsilon
\]

for the $PrivK^{mEavDL2,b}_{\mathcal{A}^{E},\text{AEAD}}$ game defined in Fig. 8.

mCCAmL2 $\Rightarrow$ mEavDL2. Our definition of mCCAmL2 almost explicitly incorporates the elements of mEavDL2, and thus mCCAmL2 $\Rightarrow$ mEavDL2. This means CCAmL2 captures all pre-existing confidentiality notions.

**D.2 mCCAmL2* $\Rightarrow$ EavDL**

In this subsection, we show the necessity of including challenge decryption leakage, by showing that the weakened version mCCAmL2* does not imply EavDL (neither EavDL2, of course). The idea is that if the decryption leakages always leak the decrypted plaintext, then it remains possible to retain mCCAmL2* security; yet, this feature immediately rules out the possibility of EavDL, since in the EavDL security game, the challenge plaintext $M^{\ast}$ has to be hidden from $A$.

More clearly, let AEAD = (Gen, Enc, Dec) with leakage function $L = (\mathcal{L}_{\text{enc}}, \mathcal{L}_{\text{dec}})$ be mCCAmL2* secure with respect to $L$. Then we build AEAD' = (Gen, Enc, Dec) with leakage $L' = (\mathcal{L}_{\text{enc}}, \mathcal{L}_{\text{dec}}')$ such that:

$L'_{\text{dec}}(k, N, A, C)$: outputs leakd = $L_{\text{dec}}(k, N, A, C)$ if Dec$_{k}(N, A, C) = \bot$ and outputs (leakd, $M$) otherwise, where leakd = $L_{\text{dec}}(k, N, A, C)$, $M = $ Dec$_{k}(N, A, C)$.
It’s not hard to see $\text{AEAD}^*$ remains $\text{mCCAmL}^2$ secure, since the additional decryption leakage (the correct message) essentially contains no new information. However, during the $\text{EavDL}$ security game, the challenge plaintext $M^b$ is directly given by this leakage, allowing to precisely determine the value of $b$.

D.3 $\text{MR} \land \text{mCPAmL}^2 \land \text{CIML}^2 \land \text{mEavDL}^2 \Rightarrow \text{CCAmL}^2$

In subsection 3.3 we have proved $\text{MR} \land \text{CPAmL}^2 \land \text{CIML}^2 \Rightarrow \text{CCAmL}^2$. In this subsection we prove an even stronger claim of $\text{MR} \land \text{mCPAmL}^2 \land \text{CIML}^2 \land \text{mEavDL}^2 \Rightarrow \text{CCAmL}^2$. The idea stems from the following observations:

– in the $\text{MR}$ game, no leakage traces are given;
– in the $\text{mCPAmL}^2$ game, decryption leakage is only available for challenge ciphertexts, and thus “nonce respecting”;
– in the $\text{CIML}^2$ game, additional information unrelated to the key $k$ but only to messages $M$ are irrelevant;
– in the $\text{mEavDL}^2$ game, assuming $\text{CIML}^2$ holds, the decryption leakage oracle will essentially be “nonce respecting”;
– on the other hand, in the $\text{CCAmL}^2$ game, many valid decryption leakages with respect to a single are available through trivial leaking decryption query, i.e. from ciphertext returned by the leaking encryption oracle.

Let $\text{AEAD} = (\text{Gen, Enc, Dec})$ be $\text{MR}$, $\text{mCPAmL}^2 \land \text{mEavDL}^2$ with respect to leakage function $L = (L_{\text{enc}}, L_{\text{dec}})$ and $\text{CIML}^2$ with respect to $L^*$. Then we build $\text{AEAD}' = (\text{Gen}', \text{Enc}, \text{Dec})$ with leakage function $L' = (L'_{\text{enc}}, L'_{\text{dec}})$ as follows, where $N^\dagger, N^\circ, M^\dagger$ below are the outputs of a publicly samplable distribution parametrized by $n$:

$\text{Gen}'(1^n)$: generates $k \leftarrow \text{Gen}(1^n)$ and samples two random bits $t_0, t_1$. It returns $(k, sh)$ where $sh = (t_0, t_1)$.

$L'_{\text{enc}}((k, sh), N, A, M)$: outputs $\text{ leak}_e = L_{\text{enc}}(k, N, A, M)$ as well as the additional value $B$ but only in two cases:

– Case 1: $N = N^\dagger$ and $M = M^\dagger$, then $B = t_0 \oplus t_1 \oplus 1$;
– Case 2: $N = N^\circ$ and $M \neq M^\dagger$, then $B = t_0 \oplus t_1$.

$L'_{\text{dec}}((k, sh), N, A, C)$: outputs $\text{ leak}_d = L_{\text{dec}}(k, N, A, C)$ as well as the additional value $B$ if $\text{Dec}_k(N, A, C) = M \neq \bot$ and:

– Case 1: $N = N^\circ$ and $M = M^\dagger$, then $B = t_0$;
– Case 2: $N = N^\circ$ and $M \neq M^\dagger$, then $B = t_1$.

We establish the desired claims in turn:

$\text{MR}$. Both scheme are the same from a black-box perspective.

$\text{CIML}^2$. Given a $\text{CIML}^2$ adversary $A'$ against $\text{AEAD}'$, it’s not hard to see a $\text{CIML}^2$ adversary $A$ could use the oracles of $\text{AEAD}$ and internally sampled bits $t_0$ and $t_1$ to perfectly simulate the oracles of $\text{AEAD}'$ in front of $A'$. By this, $\text{AEAD}'$ is $\text{CIML}^2$.

$\text{mCPAmL}^2$. We show that if there is a $\text{mCPAmL}^2$ adversary $A'$ against $\text{AEAD}'$, then there is an adversary $A$ which uses $A'$ to break the $\text{mCPAmL}^2$ security of $\text{AEAD}$. In detail, once $\text{Priv}_{\text{mCPAmL}^2, b}^{\text{CCAmL}^2, \text{AEAD}, L}(1^n)$ is setup, $A(1^n)$ publicly samples $N^\dagger, N^\circ, M^\dagger$ and sends to $A'$ whatever is specified by $\text{AEAD}'$. $A$ also picks two random bits $t_0, t_1$. Then it runs $A'$ and simulates $\text{Priv}_{\text{mCPAmL}^2, b}^{\text{CCAmL}^2, \text{AEAD}, L}$, using its own interaction $\text{Priv}_{\text{mCPAmL}^2, b}^{\text{CCAmL}^2, \text{AEAD}, L}$. For each query from $A'$, the actions of $A$ are as follows:

$\text{Leaking (non-challenge) encryption queries}$: On input $(N, A, M)$,

(i) $A$ queries its own leaking encryption oracle on $(N, A, M)$ and gets back some $(C, \text{ leak}_e)$ or possibly $\bot$;

(ii) If not $\bot$, $A$ checks whether it should append an additional leakage $B$ to $(C, \text{ leak}_e)$ in the case $(N, A, M)$ falls into one of the two cases described above.

$A$ respectively returns $(C, \text{ leak}_e)$ or $\bot$ or even $(C, \text{ leak}_e, B)$ to $A'$ according to the above situations.

$\text{Leaking challenge query}$: on input $(N_{ch}, A_{ch}, M^0, M^1)$, $A$ sends it to its leaking challenge oracle and gets the tuple $(C^b, \text{ leak}_e^b)$ or possibly $\bot$. In the latter case it returns $\bot$ as well, otherwise
(i) if \( N_{ch} = N^1 \), regardless of what is encrypted by the challenge oracle, \( \mathcal{A} \) uniformly samples a new random bit \( s^* \) and returns \( (C^b, \text{leak}^b_d, s^*) \);
(ii) else, \( \mathcal{A} \) just returns \( (C^b, \text{leak}^b_d) \).

**Challenge decryption leakage:** on input \( i \), \( \mathcal{A} \) sends \( i \) to its challenge decryption leakage oracle and gets \( \text{leak}^b_d \). Then,
(i) if the \( i \)-th challenge ciphertext contains \( N_{ch} = N^0 \), \( \mathcal{A} \) returns \( (\text{leak}^b_d, t_0) \);
(ii) else, \( \mathcal{A} \) just returns \( \text{leak}^b_d \).

Eventually, \( \mathcal{A} \) outputs the bit returned by \( \mathcal{A}' \).

Now we explain why \( \mathcal{A} \) properly emulates the \( \text{mCPAmL2} \) game in front of \( \mathcal{A}' \). As long as \( N^1 \) is never involved in a challenge query, the additional decryption leakage associated to the nonce \( N^0 \) is independent of the involved message. Therefore, the simulation is of no deviation.

We next concentrate on the case \( N^1 \) is first involved in a challenge query. Since \( N^1 \) can only appear once there it means that \( \mathcal{A} \) will never sees either \( t_0 \oplus t_1 + 1 \) or \( t_0 \oplus t_1 \). By the definition of \( \text{mCPAmL2} \) we know there is at most one challenge query under the nonce \( N^0 \). This means that even \( \mathcal{A}' \) would have been received either \( t_0 \) or \( t_1 \) in the real game both equally remains uniform. So when \( \mathcal{A} \) always gives \( t_0 \) it makes no difference in the view of \( \mathcal{A}' \). Therefore, the distributions of the transcript of queries and answers obtained in the game \( \text{PrivK}^{m\text{CPAmL2},b}_\mathcal{A},\mathcal{L}^{CIML2} (1^n) \) and the game simulated by \( \mathcal{A} \) are the same, and thus

\[
\Pr \left[ \text{PrivK}^{m\text{CPAmL2},b}_\mathcal{A},\mathcal{L}^{CIML2} (1^n) \Rightarrow 1 \right] = \Pr \left[ \text{PrivK}^{\mathcal{A}'},\mathcal{L}^{CIML2} (1^n) \Rightarrow 1 \right].
\]

**mEavDL2.** The proof just follows the same line as the proof for \( \text{mCPAmL2} \), except that the adversary \( \mathcal{A} \) aborts if the internally ran \( \mathcal{A}' \) manages to call the decryption leakage oracle on a valid ciphertext which is not a challenge ciphertext. Since the probability to abort is bounded by the probability to create a forgery, the abort probability is at most \( \Pr [\text{PrivK}^{\text{CIML2}}_\mathcal{A},\mathcal{L}^{CIML2} (1^n) \Rightarrow 1] \), which is small since \( \text{AEAD}^{\text{CIML2}} \) secure. Now, assuming that abort does not occur we are in the same situation than in the \( \text{mCPAL1} \) argument above.

\(-\text{CCAmL2}^*\). Let \( \mathcal{A}' \) be an adversary which given \( N^1, N^0, M^1 \) sequentially makes the following queries for \( M \neq M' \):
(i) \( (N^0, A, M^1) \) to the leaking encryption oracle and get \( C^l \); 
(ii) \( (N^0, A, M) \) to the leaking encryption oracle and get \( C^l \); 
(iii) \( (N^0, A, C^l) \) to the leaking decryption oracle and get \( t_0 \); 
(iv) \( (N^0, A, C) \) to the leaking decryption oracle and get \( t_1 \).

\( \mathcal{A}' \) finally submits \( (N^1, A, M^1, M) \) and obtains ciphertext \( C^b \) and (the additional) leakage bit \( B \). Now whether \( B = t_0 \oplus t_1 \) or not allows distinguishing.

This concludes our proof for \( \text{MR} \land \text{mCPAmL2} \land \text{CIML2} \land \text{mEavDL2} \Rightarrow \text{CCAmL2}^* \). \( \square \)

## E Extention to the multi-challenge setting

We next revisit the confidentiality against chosen-ciphertext and chosen-plaintext adversaries with partial misuse and full leakages in the multi-challenge setting. We show that the single-challenge and multi-challenge settings are equivalent for chosen-ciphertext notions, but not for chosen-plaintext notions.

### E.1 Multi-challenge chosen-ciphertext security

A natural extension of the \( \text{CCAmL2} \) experiment of Figure 1 to the multi-challenge setting is given by the \( \text{mCCAmL2} \) experiment in Figure 9. It leads to the following definition.

**Definition 16 (mCCAmL2).** (For a security parameter \( n \)) a nonce-based authenticated encryption with associated data \( \text{AEAD} = (\text{Gen}, \text{Enc}, \text{Dec}) \) with leakage function pair \( \mathcal{L} = (L_{\text{enc}}, L_{\text{dec}}) \) is \( (q_e, q_d, q_{c}, q_{m}, q_{l}, t, \varepsilon) \)-\( \text{mCCAmL2} \) secure if for every \( (q_{t}, q_{d}, q_{c}, q_{m}, q_{l}, t) \)-bounded adversary \( \mathcal{A}^L \), we have:

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

where the adversary \( \mathcal{A}^L \) makes at most \( q_{e} \) leaking encryption queries, \( q_{d} \) leaking decryption queries, \( q_{c} \) challenge decryption leakage queries, \( q_{m} \) leaking challenge queries and \( q_{l} \) leakage evaluation queries on arbitrarily chosen keys.
If the adversary \( \text{Priv}_K \) implies bounded adversary \( m \text{CCA}_m^L_2 \) game in front of \( \text{Proof} \).

**Initialization:** generates a secret key \( k \leftarrow \text{Gen}(1^n) \) and sets \( E, E_{ch} \leftarrow \emptyset \).

**Leaking encryption queries:** \( A^k \) gets adaptive access to \( \text{LEnc}(\cdot, \cdot, \cdot) \), \( \text{LEnc}(N, A, M) \) outputs \( \bot \) if \( (N, *, *) \in E_{ch} \), else computes \( C \leftarrow \text{Enc}_k(N, A, M) \) and \( \text{leak}_e \leftarrow \text{LEnc}(k, N, A, M) \), updates \( E \leftarrow E \cup \{N\} \) and finally returns \( (C, \text{leak}_e) \).

**Leaking decryption queries:** \( A^k \) gets adaptive access to \( \text{LDec}(\cdot, \cdot, \cdot) \), \( \text{LDec}(N, A, C) \) outputs \( \bot \) if \( (N, A, C) \in E_{ch} \), else computes \( M \leftarrow \text{Dec}_k(N, A, C) \) and \( \text{leak}_d \leftarrow \text{LDec}(k, N, A, C) \) and returns \( (M, \text{leak}_d) \); (Where \( M = \bot \) may occur.)

**Challenge queries:** on possibly many occasions \( A^k \) submits \( (N_{ch}, A_{ch}, M^0, M^1) \),

If \( M^0 \) and \( M^1 \) have different (block) length or \( N_{ch} \notin E \) or \( (N_{ch}, *, *) \in E_{ch} \), returns \( \bot \); else computes \( C^b \leftarrow \text{Enc}_k(N_{ch}, A_{ch}, M^b) \) and \( \text{leak}_e^k \leftarrow \text{LEnc}(k, N_{ch}, A_{ch}, M^b) \), updates \( E_{ch} \leftarrow E_{ch} \cup \{(N_{ch}, A_{ch}, C^b)\} \) and finally returns \( (C^b, \text{leak}_e^k) \);

**Decryption challenge leakage queries:** \( A^k \) gets adaptive access to \( \text{LDec}(\cdot, \cdot, \cdot) \), \( \text{LDec}(N, A, C) \) outputs \( \bot \) if \( (N, A, C) \in E_{ch} \), else computes \( M \leftarrow \text{Dec}_k(N, A, C) \) and outputs a leakage trace \( \text{leak}_d^k \leftarrow \text{LDec}(k, N_{ch}, A_{ch}, C^b) \);

**Finalization:** \( A^k \) outputs a guess bit \( b' \) which is defined as the output of the game.

**Fig. 9:** The \( \text{Priv}_{K,\text{AEAD}_m}^m(1^n) \) game.

**Equivalent notions.** If the adversary \( A \) in the multi-challenge experiment above is restricted to \( q_m \leq 1 \) challenge query then the \( m \text{CCA}_m^L \) security and the \( \text{CCA}_m^L \) security collide. Therefore, \( m \text{CCA}_m^L \) security implies \( \text{CCA}_m^L \) security. The next theorem states that the both notions are equivalent.

**Theorem 6.** The \( \text{CCA}_m^L \) security is equivalent to the \( m \text{CCA}_m^L \) security. Formally, for any \((q_e, q_d, q_c, q_m, q_t, \varepsilon)\)-bounded adversary \( A^k \) against the \( m \text{CCA}_m^L \) security of \( \text{AEAD} \) for a security parameter \( n \), if \( \text{AEAD} \) is \((q_e + q_m - 1, q_d + q_e, q_c, q_t, \varepsilon)\)-\( \text{CCA}_m^L \)-secure for a security parameter \( n \), we have:

\[
\left| \Pr[\text{Priv}_{K,\text{AEAD}_m}^m(1^n) \Rightarrow 1] - \Pr[\text{Priv}_{K,\text{AEAD}_m}^m(1^n) \Rightarrow 1] \right| \leq q_m \times \varepsilon_{\text{CCA}_m^L}.
\]

**Proof.** Given an adversary \( A^k \) against the \( m \text{CCA}_m^L \) security of \( \text{AEAD} \) making \( q_m \) leaking challenge queries, we build \( q_m \) adversaries \( A'_i \) against the \( \text{CCA}_m^L \) security of \( \text{AEAD} \), for \( i = 1 \) to \( q_m \). To simulate the \( m \text{CCA}_m^L \) game in front of \( A^k \), each \( A'_i \) has to emulate the responses to the different types of queries in \( \text{Priv}_{K,\text{AEAD}_m}^m(1^n) \). Therefore, \( A'_i \) proceeds as follows with \( E_i = \emptyset \):

**Leaking encryption queries:** when \( A^k \) queries \( \text{LEnc}(N, A, M) \),

If \( (N, *, *) \in E_i \), \( A'_i \) returns \( \bot \), else \( A'_i \) queries \( \text{LEnc}(N, A, M) \) and gets back either \( \bot \) or \( (C, \text{leak}_e) \) which it sends to \( A^k \).

**Leaking decryption queries:** when \( A^k \) queries \( \text{LDec}(N, A, C) \),

If \( (N, A, C) \in E_i \), \( A'_i \) returns \( \bot \), else \( A'_i \) queries \( \text{LDec}(N, A, C) \) and gets back either \( \bot \) or \( (M, \text{leak}_d) \), where \( M = \bot \) may occur, and sends it to \( A^k \).

**Challenge queries:** for \( j = 1 \) to \( q_m \), \( A^k \) queries \( (N_{ch}^j, A_{ch}^j, M_0^j, M_1^j) \),

If \( (N_{ch}^j, A_{ch}^j, M_0^j, M_1^j) \in E_i \), \( A'_i \) returns \( \bot \), else:

\( j < i \), \( A'_i \) queries its leaking encryption oracle \( \text{LEnc}(N_{ch}^j, A_{ch}^j, M^1) \) and gets either \( \bot \) or \( (C^j, \text{leak}_e^j) \) which it sends to \( A^k \), \( A'_i \) updates \( E_i = E_i \cup (N_{ch}^j, A_{ch}^j, C^j) \);

\( j = i \), \( A'_i \) queries its leaking challenge oracle \( (N_{ch}^j, A_{ch}^j, M_0^j, M_1^j) \) and gets either \( \bot \) or \( (C^j, \text{leak}_e^j) \) which it sends to \( A^k \);

\( j > i \), \( A'_i \) queries its leaking encryption oracle \( \text{LEnc}(N_{ch}^j, A_{ch}^j, M^0) \) and gets either \( \bot \) or \( (C^j, \text{leak}_e^j) \) which it sends to \( A^k \), \( A'_i \) updates \( E_i = E_i \cup (N_{ch}^j, A_{ch}^j, C^j) \).

**Decryption challenge leakage queries:** when \( A^k \) queries \( \text{LDec}(j) \), for \( j = 1 \) to \( q_m \),

\( j \neq i \), if \( A'_i \) sent \( \bot \) to the \( j \)-th challenge query made by \( A^k \) it sends \( \bot \), else it queries its leaking decryption oracle \( (N_{ch}^j, A_{ch}^j, C_{j}^{<j}) \) and gets \( \text{leak}_{d,j}^{[j<]} \) which it sends to \( A^k \);
efficiency of the adversaries. Proof easy follows from standard hybrid arguments, the assumption made on AEAD secret key \( k \) that would not be deemed as such in the CCAmL2 experiment played by \( \mathcal{A}' \). The conclusion of the proof easy follows from standard hybrid arguments, the assumption made on AEAD and by evaluating the efficiency of the adversaries.

By removing the decryption challenge leakage oracle \( L_{\text{dec}} \) from the game in Figure 9, we define the security notion of mCCA mL2*. It is easy to see that CCA mL2* security is equivalent to mCCA mL2* security as well.

E.2 Multi-challenge chosen-plaintext case

A natural extension of the CPA mL2 experiment to the multi-challenge setting is obtained by removing the leaking decryption oracle of the above mCCA mL2 experiment. The resulting experiment is called the CPA mL2 experiment, where the 2 highlights the access to the decryption challenge leakage oracle.

**Definition 17 (mCPA mL2).** A nonce-based authenticated encryption scheme with associated data AEAD = (Gen, Enc, Dec) with leakage function \( L = (L_{\text{enc}}, L_{\text{dec}}) \) is \((q_e, q_c, q_m, q_l, \varepsilon)\)-mCPA mL2 secure for a security parameter \( n \) if, for every \((q_e, q_c, q_m, q_l, \varepsilon)\)-bounded adversary \( \mathcal{A} \), we have:

\[
\Pr[\text{PrivK}_{\mathcal{A}, \text{AEAD}, L}^{\text{mCPA mL2, 0}}(1^n) \Rightarrow 1] - \Pr[\text{PrivK}_{\mathcal{A}, \text{AEAD}, L}^{\text{mCPA mL2, 1}}(1^n) \Rightarrow 1] \leq \varepsilon,
\]

when the adversary \( \mathcal{A} \) makes at most \( q_e \) leaking encryption queries, \( q_c \) challenge decryption leakage queries, \( q_m \) leaking challenge queries and \( q_l \) leakage evaluation queries on arbitrarily chosen keys.

**Non equivalent notions.** In general, while mCPA mL2 security obviously implies CPA mL2 security, the converse is false.

**Theorem 7.** Assuming there exists a CPA mL2 secure AE, then there exists CPA mL2 secure authenticated encryption which is not mCPA mL2 secure.

Essentially, in the mCPA mL2 game an adversary can obtain decryption leakages for multiple tuples \((N, A, C)\) via the multiple challenge queries, whereas in the CPA mL2 game she can obtain such leakages for only one (challenge) tuple \((N, A, C)\). This difference establishes the separation. By removing the \( L_{\text{dec}} \) oracle as well, the aforementioned difference disappears, and we find back the equivalence of CPA mL1 := CPA mL2, for the single-challenge notion, with mCPA mL1 := mCPA mL2, for the multi challenge notion.

**Proof.** Let AEAD = (Gen, Enc, Dec) be a CPA mL2 secure authenticated encryption with leakage function pair \( L = (L_{\text{enc}}, L_{\text{dec}}) \). Then, we build the following authenticated encryption AEAD' = (Gen', Enc, Dec) with leakage function pairs \( L' = (L_{\text{enc}}, L'_{\text{dec}}) \), where \( N_0, N_1 \) below are the outputs of a publicly samplable distribution parametrized by \( n \):

- \( \text{Gen}'(1^n) \): runs \( k \leftarrow \text{Gen}(1^n) \), \( k_0 \leftarrow \text{Gen}(1^n) \) and computes \( k_1 = k \oplus k_0 \). The secret key is defined as \((k, k_0, k_1)\).
- \( \text{L}'_{\text{dec}}((k, k_0, k_1), N, A, C) \) outputs \( L_{\text{dec}}(k, N, A, C) \) and possibly the following additional decryption leakage:
  - If \( N = N_0 \), gives \( k_0 \);
  - If \( N = N_1 \), gives \( k_1 \).

Since an adversary against the CPA mL2 security of AEAD' will never receive leakage traces for more than one ciphertext of the form \((N, A, C)\), it will never get both \( k_0 \) and \( k_1 \). Since they are random shares of the key \( k \), the CPA mL2 security still holds from AEAD because we can simulated the additional leakage by picking \( k' \leftarrow \text{Gen}(1^n) \) and set \( k_0 = k' \) or \( k_1 = k' \) on-the-fly when needed. However, an mCPA mL2 adversary simply has to make two challenge queries involving \( N_0 \) and \( N_1 \) and then to call the decryption challenge leakage oracle \( L_{\text{dec}} \) on the corresponding challenge ciphertexts received as answers to get \( k_0 \) and \( k_1 \), from which the secret key \( k = k_0 \oplus k_1 \) can be efficiently computed.
F FEMALE Security Proofs

We first analyze the CCA security of FEMALE in sections F.1 and F.2, and then prove CCIML2 and MR in sections F.3 and F.4 resp.

F.1 Preparations for CCAML2 Proof

We are also interested in the mCCAML2 security, namely our strongest confidentiality notion in the multi-challenge setting introduced in Appendix E.1. An approach is to first prove the CCAML2 bound claimed in Theorem 4 and then use the generic result of Theorem 6 to derive the mCCAML2 bound. However, we prefer an “inverse” direction: we directly prove a mCCAML2 bound, and then obtain the CCAML2 bound by setting \(q_m = 1\). This approach could produce a slightly better bound without the factor \(q_m\) in some terms. Formally, sections F.1 and F.2 devote to prove the following theorem.

Theorem 8. Let \(H : \{0, 1\}^* \to B\) be a \((0, t', \varepsilon_{cr})\)-collision resistant and \((q_d, t', \varepsilon_{pr})\)-range-oriented preimage resistant hash function. And let \(E : M \times M \to M\) be a \((2q_c + 2q_d + 2q_m, t', \varepsilon_E)\)-SPRP with two implementations: a strongly protected implementation \(E^*\) is leak free, and a plain implementation \(E\) have leakage function \(L_E\) that is \((q_S, t_S, q_i, t, \varepsilon_{(p,2)\cdot RSIM}(p), 2)-R\)-simulatable. Then the FEMALE implementation with leakage function \(L = (L_{enc}, L_{dec})\) defined before is \((q_c, q_d, p - 1, q_m, q_i, t, \varepsilon_mCCAML2)\) mCCAML2-secure, where

\[
\varepsilon_mCCAML2 \leq 2\varepsilon_E + \varepsilon_{cr} + \varepsilon_{pr} + \sum_{i=1}^{2m} \varepsilon_{FEMALE-cav}(\ell_i),
\]

\(\ell_i\) is the number of blocks in the \(i\)-th challenge message, and \(\varepsilon_{FEMALE-cav}(\ell_i)\) is the upper bound on the eavesdropper advantage of \((q_i, t')\)-bounded adversaries against FEMALE with \(\ell_i\) blocks. Concretely, we have:

\[
\varepsilon_{FEMALE-cav}(\ell_i) \leq (6\ell_i + 4)(\varepsilon_E + \varepsilon_{(p,2)\cdot RSIM}) + \ell_i \cdot \varepsilon_{s\cdot block} + \frac{6\ell_i + 4}{2n}.
\]

Here \(t' = (q_c + q_d + q_m)(t_S + t_{1-pass}), t_{1-pass}\) is the maximum running time of FEMALE upon a single (encryption or decryption) query, and \(t_S\) is the time needed for randomly sampling a value from \(M\).

As mentioned, setting \(q_m = 1\) yields the exact CCAML2 bound which was interpreted as

\[
O \left( \frac{t^2}{2^n} \right) + O \left( \frac{t(q_c + q_d)}{2^n} \right) + O \left( \frac{\sigma \cdot t + \sigma^2}{2^n} \right) + O \left( \sigma (\varepsilon_{(p,2)\cdot RSIM} + \varepsilon_{s\cdot block}) \right)
\]

in Theorem 4. To further clarify, the first two terms correspond to \(\varepsilon_{cr}\) and \(\varepsilon_{pr}\). The term \(\frac{\sigma \cdot t + \sigma^2}{2^n}\) is different from the classical non-rekeying modes (which typically only contains the term \(\frac{t}{2^n}\)), and it stems from \(O(\sigma \cdot \varepsilon_E)\) in \(\sum_{i=1}^{2m} \varepsilon_{FEMALE-cav}(\ell_i)\): since \(t' = t + O(\sigma)\), we have \(\varepsilon_E = \frac{t + O(\sigma)}{2^n}\), and thus \(O(\sigma \cdot \varepsilon_E) = O(\frac{t + \sigma^2}{2^n})\). As mentioned, the term \(O(\sigma (\varepsilon_{(p,2)\cdot RSIM} + \varepsilon_{s\cdot block}))\) corresponds to the security degradation due to side-channel & the reduction to a single block encryption.

It will be apparent in the proof that FEMALE actually satisfies an even stronger notion of mCCAML2. Since the encryption starts with \(R \leftarrow H(0||N||A)\), only the pairs \((N, A)\)’s must be fresh in the challenge phase to derive the security instead of each of these nonces \(N\)’s.

The analysis proceeds in five steps, each corresponding to a subsubsection. As the first step, we prove a useful “indistinguishability-like” lemma for our leaking setting. Then is a preparation: we define a model named Leaking, Idealized, Single-block Encryption scheme LISE with encryption and decryption leakages. This is actually the idealized version of the LRSE scheme defined in Fig. 6, and could be proved indistinguishable from LRSE (Lemma 2). It will be used in the 4th step (see below), constituting a bridge in the reduction.

Third, (informally speaking) we prove indistinguishability for the two systems \((\text{FEMALE}(M), L_{\text{FEMALE}(M)})\) and \((\$^*, S_{\text{FEMALE}}(M))\), for a single message \(M\) (Lemma 3). In other words, \((\text{FEMALE}(M), L_{\text{FEMALE}(M)})\), the process of using FEMALE to encrypt a single message, is indistinguishable from an idealized process that produces random outputs \$ and simulated leakages \(S_{\text{FEMALE}}(M)\). This shows the design of FEMALE is good in the sense that it achieves nice confusion and diffusion (so that it produces somewhat pseudorandom outputs).
Yet, the conclusion of step 3 says nothing about the message confidentiality—or eavesdropper security—of \( \text{FEMALE}(M) \), since the leakages \( L_{\text{FEMALE}(M)} \) or its indistinguishable counterpart \( S_{\text{FEMALE}(M)} \) may leak \( M \) completely. To remedy this, we focus on the idealized process \( (S, S_{\text{FEMALE}(M)}) \), and show how to relate its eavesdropper security to the eavesdropper security of applying \( \text{LISE} \)—the single-block encryption scheme—to independently encrypt \( |M| \) blocks. Since we’ve established the indistinguishability of \( \text{LISE} \) and LRSE, the eavesdropper security of \( (\text{FEMALE}(M), L_{\text{FEMALE}(M)}) \) can be established via the following chain:

\[
\begin{align*}
\text{eavesdropper security of LRSE} & \quad (\text{our assumption, see (1)}) \\
\Rightarrow & \quad \text{eavesdropper security of LISE (using indistinguishability of LRSE and LISE)} \\
\Rightarrow & \quad \text{eavesdropper security of the idealized process} \quad (S, S_{\text{FEMALE}(M)}) \\
\Rightarrow & \quad \text{eavesdropper security of} \quad (\text{FEMALE}(M), L_{\text{FEMALE}(M)}). 
\end{align*}
\]

Eventually, based on the eavesdropper security of \( (\text{FEMALE}(M), L_{\text{FEMALE}(M)}) \), we establish the \( \text{mCCAmL2} \) security (this step is in subsection F.1). Roughly, the proof relies on the following features of \( \text{FEMALE} \):

1. Every invalid decryption query only leaks a pseudorandom value, i.e. \( (E_k)^{-1}(T) \) for some \( T \). So the encryption can be seen as independent from these values;
2. For each challenge encryption query, since the nonce is used only once during the experiment, the process starts from a ephemeral key \( s_0 \) that is different from any other ephemeral key of the other encryption queries. By this, encryption of this challenge is quite independent from the other encryption queries, and we can view the entire experiment as an eavesdropper adversary against \( (\text{FEMALE}(M), L_{\text{FEMALE}(M)}) \) with a lot of offline computations (i.e. all the other encryptions are turned into offline computations).

**Indistinguishability of Real-Leaking and Ideal-Simulating Worlds.** Standaert et al. proved that based on the pseudorandomness of \( E \) and the simulatability of the leakage, (roughly) the “real-leaking world” \( (E_k(p), L(k, p)) \) is indistinguishable from the “ideal-simulating” world \( (S, S^L(k, p, S)) \) [36]. A similar intermediate result could be obtained in our \( R \)-simulatability framework. For convenience of applying later in our analysis, we focus on the case \( q = 2 \). Moreover, we write \( [\text{leak}_1, \ldots, \text{leak}_p]^p \) for the vector of \( \ell p \) leakages, which consist of \( \ell \) (probably distinct) leakages, and each is obtained \( p \) times. We stress that trying to obtain the same leakage for \( p \) times would not result in completely identical traces: each time \( L(x) \) is queried for some input \( x \), the trace would be mixed with random noise, and would probably deviate from the traces generated by previous queries to \( L(x) \).

**Lemma 1.** Let \( E : \mathcal{M} \times \mathcal{M} \to \mathcal{M} \) be a \( (2, t, \epsilon_E) \)-PRP, whose implementation has a leakage function \( L_E \) having \( (q_S, t_S, q_t, t, \epsilon_{(p, 2) \text{-rsim}}) \) \( (p, 2) \)-\( R \)-simulatable leakages, and let \( S^L \) be an appropriate \( (q_S, t_S) \)-bounded leakage simulator. Then, for every \( k_{pre}, p_0, p_1, z \in \mathcal{M} \) and every \( (q_l - q^*, t - t^*) \)-\( \text{-distinguishable} \) \( D^1 \), the following holds:

\[
\Pr[k_A : \mathcal{D}^1(E_{k_A}(p_0), E_{k_A}(p_1), [L_E(k_{ch}, p_0), L_E(k_{ch}, p_1)], S^L(k_{pre}, z, k_{ch})]^p) \Rightarrow 1] - \Pr[k_{ch}, c_A, c_B \in \mathcal{M}, c_A \neq c_B \text{ iff } p_0 \neq p_1 : \mathcal{D}^1(c_A, c_B, [S^L(k_{ch}, p_0, c_A), S^L(k_{ch}, p_0, c_B), S^L(k_{pre}, z, k_{ch})]^p) \Rightarrow 1] | \leq \epsilon_E + \epsilon_{(p, 2) \text{-rsim}}.
\]

Here \( q^* = 3p \cdot q_S \), while \( t^* = \text{Max}\{t_r, t_{sim}\} \), in which \( t_r \) is equal to \( 3p \cdot t_S \) augmented with the time needed to make 2 \( \epsilon_E \)-queries to the PRP challenger and select a uniformly random key in \( \mathcal{M} \), and \( t_{sim} \) is the time needed to relay the content of \( 2p \) \( \text{Enc} \) and \( p \) \( \text{Gen} \) queries from and to a \((p, 2)\)-\( \text{rsim} \) challenger.

**Proof.** The proof consists of two simple transitions: we first replace the real leakages by with simulated ones, relying on the recyclable-simulatability assumption, then replace \( E_{k_A}(p_0) \) and \( E_{k_A}(p_1) \) by two distinct random values to obtain the target inputs, relying on the assumption that \( E \) is a PRP.

It’s not hard to see the claim remains valid if \( E \) is a PRF rather than PRP. To save space we concentrate on the latter case (that will be used in our proof).

**Single-Block One-Time Encryption Scheme.** This is actually an extension of the analogue introduced in [32]. In detail, the use of \( E \) for computing \( k_{up} \) and \( y_{ch} \) is replaced by sampling random values. And we adapt the corresponding leakage traces using \( S^L \). The resulted algorithm is defined in Fig. 10.

We also define \( \text{LISEdec}_{k_{ch}}^+(m) = (\text{LISEdec}_{k_{ch}}(m), [L_{\text{idec}}(k_{ch}, c)]^{p-1}, k_{pre}) \) for \((c, k_{up}) = \text{ISEnc}_{k_{ch}}(m) \). Similarly to Pereira et al. [32], our \( \text{ISEnc} \) scheme is indistinguishable from its real version \( \text{RSEnc} \).
\textbf{Description of LISE:} (tool for the proof)

\begin{itemize}
\item \textbf{ISGen} \(^{(1^n)}\) picks \(k_{ch}, \ell \in \mathcal{M}, \mathcal{M}, \mathcal{C} = \mathcal{M} \) (\(p_0, p_1 \in \mathcal{M}\) can be chosen)
\item \textbf{ISEnc}_{ch}(m) \ returns \((k_{up}, c), \) where \(c = y_{ch} \oplus m, \) and \(k_{up}, y_{ch} \in \mathcal{M}, k_{up} \neq y_{ch}\) as long as \(p_1 \neq p_0 \) (and \(k_{up} = y_{ch}\) otherwise).
\item \textbf{ISDec}_{k_{ch}}(c) \ proceeds in the natural way.
\end{itemize}

The leakage \(L_{\text{ISE}} = (L_{\text{ISEnc}}, L_{\text{ISdec}, k_{pre}})\) resulting from the LISE implementation is defined as
\[
L_{\text{ISEnc}}(k_{ch}, m) = (\mathcal{S}^L(k_{ch}, p_1, k_{up}), \mathcal{S}^L(k_{ch}, p_0, y_{ch}), L_{\text{ISdec}}(y_{ch}, m), \mathcal{S}^L(k_{pre}, p_1, k_{ch})), \quad L_{\text{ISdec}}(k_{ch}, c) = (\mathcal{S}^L(k_{ch}, p_1, k_{up}), \mathcal{S}^L(k_{ch}, p_0, y_{ch}), L_{\text{dec}}(y_{ch}, c), \mathcal{S}^L(k_{pre}, p_1, k_{ch}))
\]
for a fixed random \(k_{pre} \in \mathcal{E}, \mathcal{M}.\)

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig10}
\caption{The ideal single-block encryption scheme LIEnc.}
\end{figure}

\textbf{Lemma 2.} Let \(E : \mathcal{M} \times \mathcal{M} \rightarrow \mathcal{M}\) be a \((2, t, \varepsilon_E)\)-PRP, whose implementation has a leakage function \(L_E\) having \((q_S, t_S, q_t, t, \varepsilon(p, 2)\text{-rsim}) \) \((p, 2)\text{-R-simulatable leakages, and let} \mathcal{S}^L\) be an appropriate \((q_S, t_S)\)-bounded leakage simulator. Then, for every \(p_0, p_1 \in \mathcal{M}, p_0 \neq p_1, \) and every \((q_l - q^*, t - t^*)\)-bounded distinguisher \(D^L,\) the following holds:
\[
\left| \Pr[D^L_{\text{ISEnc}}(m, \text{LRSEnc}_{k_{ch}}^+(m)) \Rightarrow 1] - \Pr[D^L_{\text{ISEnc}}(m, \text{LIEnc}_{k_{ch}}^+(m)) \Rightarrow 1] \right| \leq \varepsilon_E + \varepsilon(p, 2)\text{-rsim}.
\]

Here \(q^* = 3p \cdot q_S + p, \) while \(t^* = \text{Max}\{t_r, t_{\text{rsim}}\},\) in which \(t_r\) is equal to \(3p \cdot t_S + 2t_{\oplus}\) augmented with the time needed to make 2 oracle queries to the PRP challenger and select a uniformly random key in \(\mathcal{M}, \) \(t_{\oplus}\) is the time needed to evaluate the \(\oplus\) action on an \(n\)-bit input, and \(t_{\text{rsim}}\) is the time needed to relay the content of four \(\text{Enc}\) and two \(\text{Gen}\) queries from and to a \((p, 2)\text{-rsim}\) challenger.

\textbf{Proof.} The proof just follows the same line as Lemma 1. Note that to generate the leakage \(L_{\oplus}(y_{ch}, c) \) \(p - 1\) times, one does not need to evaluate \(\oplus\) for that many times; instead, one just needs to make \(p-1\) queries \(L.\)

It’s not hard to see Lemma 2 actually holds even if \(p_0 = p_1.\) However, as we remarked before, when the input \(p_1\) equals \(p_0,\) neither \(\text{REnc}\) nor \(\text{ISenc}\) ensures eavesdropper security. To highlight this issue and avoid confusion, we put this restriction in Lemma 2.

On the other hand, for the scheme \(\text{ISEnc}\) we do not enforce the constraint \(p_1 \neq p_0,\) since the case of \(p_1 = p_0\) would be used in some of our arguments below (of course, these arguments do not rely on the eavesdropper security of \(\text{ISEnc}\) in the case of \(p_1 = p_0).\)

\textbf{LRFSM and LIFSM: FEMALE on a Single Message, and its Ideal Version.} We first formally describe two algorithms (L)RFSM and (L)IFSM, which denote the processes of using (Leaking) Real/Idealized FEMALE to encrypt a Single Message respectively (without generating the tag). The algorithm LRFSM is described in Fig. 11, while LIFSM is in Fig. 12.

We then prove that LRFSM and LIFSM are indistinguishable, by relying on the \((p, 2)\)‐recyclable-simulatability assumption and the cryptographic strength of \(E.\)

\textbf{Lemma 3.} Let \(E : \mathcal{M} \times \mathcal{M} \rightarrow \mathcal{M}\) be a \((2, t, \varepsilon_E)\)-PRP, whose implementation has a leakage function \(L_E\) having \((q_S, t_S, q_t, t, \varepsilon(p, 2)\text{-rsim}) \) \((p, 2)\text{-R-simulatable leakages, and let} \mathcal{S}^L\) be an appropriate \((q_S, t_S)\)-bounded leakage simulator. Then, for every \(\ell\)-block message \(m,\) every \(p_A \neq p_B,\) and every \((q_l - p \cdot q_r, t - p \cdot t_r - t^*)\)-bounded distinguisher \(D^L\) (that makes at most \(q_l - p \cdot q_r - q^*\) queries to \(L\) and runs in time \(t - p \cdot t_r - t^*\)), the following holds:
\[
\left| \Pr[D^L(m, \text{LRFSM}_{\text{sh}}(m)) \Rightarrow 1] - \Pr[D^L(m, \text{LIFSM}_{\text{sh}}(m)) \Rightarrow 1] \right| \leq 2(\ell + 2)(\varepsilon_E + \varepsilon(p, 2)\text{-rsim}) + \frac{2\ell + 2}{2^n}.
\]

Here \(q_r = (4\ell + 5)(q_S + 1) + 2\ell, q^*\) and \(t^*\) are as defined in Lemma 1, and \(t_r = (4\ell + 5)(t_E + t_S + t_S) + 2\ell \cdot t_{\oplus},\) where \(t_E\) is the time needed for evaluating \(E\) once, \(t_S\) is the time needed for randomly sampling a value from \(\mathcal{M},\) and \(t_{\oplus}\) is the time needed for evaluating \(\oplus\) once.

\textbf{Proof.} We define \(G_{0,1}\) as the security game in which \(D^L\) receives \(\text{LRFSM}_{\text{sh}}(m)\) as the input, and \(G^*_\ell\) as the game in which \(D^L\) receives \(\text{LIFSM}_{\text{sh}}(m)\) as the input.

33
Description of RFSM:

- Gen picks \( s_0 \in \mathcal{M} \)
- \( \text{RFSM}_{k_0}(m_1, \ldots, m_\ell) \) proceeds in four steps:
  (1) Initializes an empty list leak for the leakage;
  (2) Computes \( s_1 \leftarrow E_s(p_A) \) and \( v \leftarrow E_{s_0}(p_B) \), and adds \( L_E(s_0, p_A) \) and \( L_E(s_0, p_B) \) to the list leak. Here the two constants \( p_A \) and \( p_B \) are those used in FEMALE.
  (3) For \( i = 1, \ldots, \ell \), computes \( s_{i+1} \leftarrow E_s(p_A), y_i \leftarrow E_{i_1}(d_{i-1}), \) and \( d_i \leftarrow y_i \oplus m_i \), and adds \( L_E(s_i, p_A), L_E(s_i, d_{i-1}), L_\oplus(y_i, m_i) \), and \( L_E(s_i, p_A), L_E(s_i, d_{i-1}), L_\oplus(y_i, d_i) \) to the list leak;
  (4) Computes \( U \leftarrow E_{s_{i+1}}(p_A), W \leftarrow E_{s_{i+1}}(d_i), V \leftarrow E_{s_i}(W), \) and \( k_1 \leftarrow E_U(V) \); and adds \( L_E(s_{i+1}, p_A), L_E(s_i, d_i), L_E(w, W), L_E(U, V), \) and \( L_E(s_{i+1}, p_A), L_E(U, V) \) to the list leak;
  (5) for \( i = 1, \ldots, \ell \), computes \( k_{i+1} \leftarrow E_s(p_A), z_i \leftarrow E_{k_i}(p_B), \) and \( c_i \leftarrow z_i \oplus d_i \), and adds \( L_E(k_i, p_A), L_E(k_i, p_B), L_\oplus(z_i, c_i) \) to the list leak.

We define \( \text{LRFSM}_{k_0}(m) = (\text{RFSM}_{k_0}(m), \text{leak}) \), where leak is the list of traces standing at the end of the computation.

Fig. 11: The RFSM scheme and the involved leakages.

Description of IFSM:

- \( \text{IFSM}_{k_0}(m_1, \ldots, m_\ell) \) proceeds in four steps:
  (1) Initializes an empty list leak for the leakage;
  (2) Samples \( s_1 \in \mathcal{M} \), and adds \( [S^i(s_0, p_A, s_1)]^\ell \) to the list leak;
  (3) For \( i = 1, \ldots, \ell \), samples \( s_{i+1} \in \mathcal{M} \) and \( y_i \in \mathcal{M} \) such that \( s_{i+1} \neq y_i \) as long as \( d_{i-1} \neq p_A \)
    (\( s_{i+1} = y_i \) otherwise), sets \( d_i \leftarrow y_i \oplus m_i \), and adds \( S^i(s_i, p_A, s_{i+1}), S^i(s_i, d_{i-1}, y_i), L_\oplus(y_i, m_i) \), and \( [S^i(s_i, p_A, s_{i+1}), S^i(s_i, d_{i-1}, y_i), L_\oplus(y_i, d_i)]^{\ell-1} \) to the list leak;
  (4) Samples \( U \leftarrow M, W \leftarrow M, U \neq W \iff d_i \neq p_A \); \( V \leftarrow M, w \neq s_1, V \leftarrow M, \) and \( k_1 \in \mathcal{M} \); and adds \( S^i(s_{i+1}, p_A, U), S^i(s_{i+1}, d_i, W), S^i(s_0, p_B, w), S^i(w, W, V), S^i(U, V, k_1), [S^i(s_{i+1}, p_A, U)]^{\ell-1} \) to the list leak;
  (5) For \( i = 1, \ldots, \ell \), samples \( k_{i+1} \in \mathcal{M}, z_i \in \mathcal{M}, \) and \( c_i \leftarrow z_i \oplus d_i \), and adds \( S^i(k_i, p_A, k_{i+1}), S^i(k_i, p_B, z_i), L_\oplus(z_i, c_i)]^{\ell-1} \) to the list leak.

IFSM_{k_0}(m_1, \ldots, m_\ell) eventually returns \((V, c), \) where \( c = (c_1, \ldots, c_\ell) \).

We define \( \text{LIFSM}_{k_0}(m) = (\text{IFSM}_{k_0}(m), \text{leak}) \) for the list leak standing at the end of the computation.

Fig. 12: The IFSM scheme and the involved leakages.
We show that \( G_{0,1} \) could be transited to \( G_\ell^* \) via a sequence of games

\[ G_0, G_1, \ldots, G_\ell, G_{\ell+1}, G_{\ell+2}, G_{\ell+3} = G_0^*, G_1^*, \ldots, G_{\ell-1}^*. \]

The first half sequence \( G_0, \ldots, G_\ell, G_{\ell+1}, G_{\ell+2}, \) and \( G_{\ell+3} \) “idealizes” the Ephemeral key-IV generation phase of the encryption. In detail, we first consider \( G_{0,1} \), and replace the two intermediate values \( E_o(p_A) \) and \( E_o(p_B) \) by two distinct random values \( s_1 \) and \( w \). We also replace the leakages \( \lfloor L_E(s_0, p_A) \rfloor^p \) and \( L_E(s_0, p_B) \) with \( \lfloor S^l(s_0, p_A, s_1) \rfloor^p \) and \( S^l(s_0, p_B, w) \). This yields the game \( G_0 \).

We next derive an upper bound for \( \Pr[ (D_L^L; G_0 \Rightarrow 1) - \Pr[ (D_L^L; G_{0,1} \Rightarrow 1) ] \). For this, we assume a \((q_r \cdot q_s \cdot \ell \cdot t \cdot t_r \cdot t^*)\)-bounded distinguisher \( D^L \) against \( G_0 \) and \( G_{0,1} \), and we build a distinguisher \( D^L \) against the real-leaking-world and the ideal-simulation-world. Concretely, \( D^L \) uses \( p_o = p_A, p_1 = p_B, \) and \( z = p_A \) to get his challenge tuple, and we assume that \( D^L \) receives

\[ (c_A, c_B, |\text{leak}_1, \text{leak}_2, S^l(s, p_A, k_{ch}))^p) \]
as inputs, with \( c_A \neq c_B \). \( D^L \) proceeds in two steps:

1. \( D^L \) first uses \( |\text{leak}_1|^p \) and \( \text{leak}_2 \) as the leakages of the first iteration;
2. \( D^L \) then sets \( s_1 \leftarrow c_A \) and \( w \leftarrow c_B \), and emulates all the remaining actions of LRFSM encryption.

Eventually, it serves the obtained ciphertext \( V |c_1| \ldots |c_\ell \) as well as the leakage traces to \( D^L \), and outputs whatever \( D^L \) outputs.

It can be seen depending on whether the inputs to \( D^L \) follow real-leaking or ideal-simulating distribution, \( D^L \) is eventually interacting with \( G_{0,1} \) or \( G_0 \). We show that to perform the additional operations, \( D^L \) makes at most \( p \cdot s_r \) additional queries to \( L \) and spend \( p \cdot t_r \) additional time. To this end, we note that the encryption process of LRFSM involves \( 4\ell + 5 \) calls to \( E \) and \( 2\ell \) XOR operations. Moreover,

- in the real world, each call to \( E \) costs 1 query to \( L \) and \( t_E \) running time;
- in the ideal world, each “idealized” call to \( E \) is translated into sampling a random value and making a call to \( S \), which costs \( q_S \) queries to \( L \) and \( (t_E + t_S) \) running time;
- each XOR operation costs 1 query to \( L \) and \( t_\oplus \) running time.

Therefore, to emulate the “hybrid” encryption process once, \( D^L \) needs at most \((4\ell + 5)(q_S + 1) + 2\ell = q_r \) queries to \( L \) and \((4\ell + 5)(t_E + t_S + t_r + t_\oplus) + 2\ell \cdot t_\oplus = t_r \) running time. To obtain the required decryption leakage traces, \( D^L \) has to additionally perform the “hybrid” decryption process for \( p - 1 \) times, which contributes to \((p - 1)q_r \) more queries and \((p - 1)\ell \) more time. Therefore, as claimed, \( D^L \) makes at most \( p \cdot q_r \) additional queries to \( L \) and spends \( p \cdot t_r \) additional time for the additional operations. By the above and Lemma 1 we have

\[ |\Pr[(D_L^L; G_0 \Rightarrow 1) - \Pr[(D_L^L; G_{0,1} \Rightarrow 1)] | \leq \varepsilon \varepsilon_E + \varepsilon_{(p,2)-rsim}. \]

Then, for \( 1 \leq i \leq \ell \), to obtain \( G_i \), we modify the game \( G_{i-1} \) by replacing the two intermediate values \( E_o(p_A) \) and \( E_o(d_{i-1}) \) with two values \( s_{i+1} \) and \( y_i \), such that \( s_{i+1} \) is uniform, and \( y_i \) is uniform and \( y_i \neq s_{i+1} \) when \( d_{i-1} \neq p_A \), and \( y_i = s_{i+1} \) otherwise; and further replacing the leakages \( \lfloor L_E(s_i, p_A) \rfloor^p, L_E(s_i, d_{i-1})^p \) with \( \lfloor S^l(s_i, p_A, s_{i+1}) \rfloor^p, S^l(s_i, d_{i-1}, y_i)^p \). To show the indistinguishability of \( G_0 \) and \( G_{i-1} \), we build \( D^L \) which proceeds in five steps (to ease description, we define \( d_{-1} = p_B \) and \( y_0 = w \)):

1. \( D^L \) first uniformly samples \( s_0 \);
2. For \( j = 0, \ldots, i - 2 \), \( D^L \) uniformly samples random values \( y_j \) such that \( s_{i+1} \neq y_j \), computes \( d_j \leftarrow y_j \oplus m_j^0 \) and \( m_j^0 \leftarrow y_j \oplus d_j \), and obtains the traces \( L_{\oplus}(y_j, m_j^0)^{p-1}, L_{\oplus}(y_j, d_j)^{p-1} \);
3. Then, if \( d_{i-2} \neq p_A \), it uniformly samples \( y_{i-1} \), computes \( d_{i-1} \leftarrow y_{i-1} \oplus m_{i-1}, m_{i-1} \leftarrow y_{i-1} \oplus d_{i-1} \);
4. When \( y_j = 0 \) the second part of the leakages is \( S^l(s_0, p_B, w) \) without repeating;
5. These XOR actions are omitted when \( y_j = 0 \).
(4) Uses \( p_0 = p_A, p_1 = p_B, \) and \( z = p_A \) to get the challenge tuple
\[
(c_A, c_B, [\text{leak}_1, \text{leak}_2, S^L(\cdot, p_A, k_{ch})]^p)
\]

with \( c_A \neq c_B \). Then, it sets \( s_{i+1} \leftarrow c_A \) and \( y_i \leftarrow c_B \), computes \( d_i \leftarrow y_i \oplus m_i \), and uses the simulated leakages \([S^L(s_i, p_A, k_{ch}), S^L(s_i, d_i, y_i, 1)]^p, L_{\oplus}(y_i, m_i)]^{p-1} as the traces of the \( i \)-th iteration., while \([\text{leak}_1, \text{leak}_2]^p, L_{\ominus}(y_i, m_i)]^{p-1} as the leakage of the \( i+1 \)-th iteration in the first pass.

(5) Takes \( s_{i+1} \) and \( d_i \) as the starting points and emulates the remaining part of the execution of LRFSM encryption. Eventually, \( D^U \) serves the obtained ciphertext \( V|c_1|\ldots|c_\ell \) as well as the leakage traces to \( D^L \), and outputs whatever \( D^L \) outputs.

It can be seen that depending on the input tuple received by \( D^U \) is real-leaking or ideal-simulation, \( D^L \) is interacting with \( G_{i-1} \) or \( G_i \), unless:

- \( d_{i-2} \neq p_A \), yet \( y_{i-1} = k_{ch} \), or
- \( d_{i-2} = p_A \).

Therefore, denoting this event by \( \text{Bad}_i \), we have
\[
\Pr[(D^L)^{G_i} \Rightarrow 1] - \Pr[(D^L)^{G_{i-1}} \Rightarrow 1] \leq \Pr[\text{Bad}_i] + \varepsilon_E + \varepsilon_{(p,2)-rsim}.
\]

It can be seen that:

- for \( i = 3, \ldots, \ell \), \( \Pr[d_{i-2} = p_A] = \frac{1}{2^n} \), and \( \Pr[y_{i-1} = k_{ch}] = \frac{1}{2^n} \). Therefore, \( \Pr[\text{Bad}_i] = \frac{2}{2^n} \); and
- for \( i = 1, 2, d_{i-2} = p_B \neq p_A \), while \( \Pr[y_{i-1} = k_{ch}] = \frac{1}{2^n} \). So \( \Pr[\text{Bad}_i] = \frac{1}{2^n} \).

The hop from \( G_i \) to \( G_{i+1} \) is slightly different, as no message block would be xored with \( E_{s_{i+1}}(d_i) \). In detail, we replace the two intermediate values \( E_{s_{i+1}}(p_A) \) and \( E_{s_{i+1}}(d_i) \) by two random values \( U \) and \( W \) such that \( W = U \) iff. \( d_i = p_A \), and replace the leakage \([L_E(s_{i+1}, p_A)]^p \) and \( L_E(s_{i+1}, d_i) \) by \([S^L(s_{i+1}, p_A, U)]^p \) and \( S^L(s_{i+1}, d_i, W) \). By an analysis similar to the argument above, \( D^U \) could consistently simulate \( G_{i-1} | G_i \) against \( D^L \) unless \( d_{i-1} = p_A \) or \( y_i = k_{ch} \) (denoted \( \text{Bad}_{i+1} \)). Therefore,
\[
\frac{2}{2^n} + \varepsilon_E + \varepsilon_{(p,2)-rsim}.
\]

We then replace the intermediate value \( E_w(W) \) by a random \( V \), and replace the leakage \( L_E(w, W) \) by \( S^L(w, W, V) \). This yields \( G_{i+2} \). In a similar vein to the above, we have
\[
\Pr[D_{G_{i+2}} \Rightarrow 1] - \Pr[D_{G_{i+1}} \Rightarrow 1] \leq \frac{1}{2^n} + \varepsilon_E + \varepsilon_{(p,2)-rsim}.
\]

We then replace the intermediate value \( E_{(V)}(V) \) by a random \( k_1 \), and replace the leakage \([L_E(U, V)]^p \) by \([S^L(U, V, k_1)]^p \). This yields \( G_{i+3} = G^0 \). We have
\[
\Pr[D_{G_{i+3}} \Rightarrow 1] - \Pr[D_{G_{i+2}} \Rightarrow 1] \leq \frac{1}{2^n} + \varepsilon_E + \varepsilon_{(p,2)-rsim}.
\]

Therefore,
\[
\frac{1}{2^n} + \varepsilon_E + \varepsilon_{(p,2)-rsim}.
\]

\text{When } i = 1, \text{ these xor operations are omitted. More clearly, } D^U \text{ uniformly samples } w, \text{ and uses } [S^L(s_0, p_A, k_{ch})]^p \text{ and } S^L(s_0, p_B, w) \text{ as the traces of the 1st iteration.}
Then, for \( j \) from 1 to \( \ell \), we modify the game \( G_{j-1}^* \) to obtain \( G_j^* \). The modifications are similar to those made for \( G_{j-1}^* \): we replace the two intermediate values \( E_j(p_A) \) and \( \hat{E}_j(p_B) \) by two distinct random values \( k_{j+1} \) and \( z_j \) (note that we always have \( p_B \neq p_A \)), and replace the leakage \( L_E(k_j, p_A), L_E(k_j, p_B) \), \( L_\oplus(E_j(p_B), d_j) \), and \( L_\oplus(E_j(p_B), c_j) \) by \( \mathcal{S}^L(k_j, p_A, k_{j+1}), \mathcal{S}^L(k_j, p_B, z_j), L_\oplus(z_j, d_j), \) and \( L_\oplus(z_j, c_j) \) respectively. We start by building a sequence of \( \ell \)-block security. For this, assuming a \((q, t)\)-bounded adversary \( A^L \), there exists a \((q + p \cdot q_r, t + p \cdot t_r)\)-bounded adversary \( A^{L'} \) such that

\[
| \Pr[A^{L'}(\text{LIFSM}_{s_0}(m^0)) \Rightarrow 1] - \Pr[A^{L'}(\text{LIFSM}_{s_0}(m^1)) \Rightarrow 1] | \\
\leq 2\ell \left( \varepsilon_E + \varepsilon_{(p, 2) \cdot \text{rsim}} \right) + \frac{2\ell + 2}{2n}
\]

as claimed.

**From 1-Block to \( \ell \)-Block Security.** We now evaluate the (eavesdropper) security of an \( \ell \)-block encryption with LIFSM by comparison with the security of \( \ell \) encryptions with \( \text{LISEnc} \) performed with independent keys, block by block.

**Lemma 4.** For every pair of \( \ell \)-block messages \( m^0 \) and \( m^1 \) and \((q, t)\)-bounded adversary \( A_L \), there exists a \((q + p \cdot q_r, t + p \cdot t_r)\)-bounded adversary \( A^{L'} \) such that

\[
| \Pr[A^{L'}(\text{LIFSM}_{s_0}(m^0)) \Rightarrow 1] - \Pr[A^{L'}(\text{LIFSM}_{s_0}(m^1)) \Rightarrow 1] | \\
\leq 2\ell \left( \varepsilon_E + \varepsilon_{(p, 2) \cdot \text{rsim}} \right) + \frac{2\ell + 2}{2n}
\]

where \( s_0, \ldots, s_{\ell-1} \) are chosen uniformly at random, the \( p_1 \) value of \( \text{LISEnc}_{s_{i-1}}^+(m^h_i) \) is \( d_{i-1}, \) \( d_0 = p_1, \) \( d_1, \ldots, d_{\ell-1} \) are chosen uniformly at random and \( d_1, \ldots, d_{\ell-1} \neq p_0, \) and \( m^0_i \) and \( m^1_i \) are the \( i \)-th block of \( m^0 \) and \( m^1 \) respectively. Here \( q_r = (4\ell + 5)q_S + 2\ell \) and \( t_r = (4\ell + 5)(t_S + t_S) + 2\ell \cdot t_S, \) where \( t_S, t_S, \) and \( t_S \) are as assumed in Lemma 3.

**Proof.** We start by building a sequence of \( \ell + 1 \) messages \( m_{h,0}, \ldots, m_{h,\ell} \) starting from \( m^0 \) and modifying its blocks one by one until obtaining \( m^1 \). That is, \( m_{h,i} := m^0_i \| \ldots \| m^0_{i-1} \| m^1_{i-1} \| \ldots \| m^1_{\ell}, \)

We proceed to argue there exists a \((q + p \cdot q_r, t + p \cdot t_r)\)-bounded adversary \( A' \) such that

\[
| \Pr[A'^L(\text{LIFSM}_{s_0}(m_{h,i-1})) \Rightarrow 1] - \Pr[A'^L(\text{LIFSM}_{s_0}(m_{h,i})) \Rightarrow 1] | \\
\leq 2\ell \left( \varepsilon_E + \varepsilon_{(p, 2) \cdot \text{rsim}} \right) + \frac{2\ell + 2}{2n}
\]

This along with a simple summation would imply the main claim.

The arguments on \( m_{h,i-1} \) and \( m_{h,i} \) for all \( i \) are similar in general. To make it clearer, we take the first two messages, i.e.,

\[
m_{h,0} = m^0_0 \| \ldots \| m^0_{\ell-1} \| m^1_{\ell}, \text{ and } m_{h,1} = m^0_0 \| \ldots \| m^0_{\ell-1} \| m^1_\ell,
\]
as example. For this, assuming a \((q, t)\)-bounded adversary \( A^L \) against \( \text{LIFSM}_{s_0}(m_{h,0}) \) and \( \text{LIFSM}_{s_0}(m_{h,1}) \), we build a \((q + p \cdot q_r, t + p \cdot t_r)\)-bounded adversary \( A'^L \) against \( \text{LISEnc} \). In detail, \( A'^L \) proceeds in six steps:

1. \( A'^L \) uniformly samples \( s_0, s_1, w \), such that \( s_1 \neq w \), and obtains the simulated leakage traces \( \mathcal{S}^L(s_0, p_A, s_1)^p \) and \( \mathcal{S}^L(s_0, p_B, w)^p \);
2. for \( j = 1, \ldots, \ell - 3, \) \( A'^L \) uniformly samples random values \( s_{j+1}, y_j \) such that \( s_{j+1} \neq y_j \) iff. \( d_{j-1} \neq p_A \), obtains traces \( \mathcal{S}^L(s_j, p_A, s_j), \mathcal{S}^L(s_j, d_{j-1}, y_j)^p \), computes \( d_j \leftarrow y_j \oplus m^0_j \) and obtains the traces \( L_\oplus(y_j, m^0_j) \) and \( L_\oplus(y_j, d_j)^p \).
(3) \( A' \) samples \( d_{\ell - 1} \neq p_A \), and computes \( y_{\ell - 1} \leftarrow d_{\ell - 1} \oplus m_0^{d_{\ell - 1}} \). \( A' \) then sets the \( p_0 \) value of its eavesdropper security challenger of \( \text{LISEnc} \) to \( p_A \) while \( p_1 \) to \( d_{\ell - 1} \), and submits \( m_0^p \), and \( m_1^p \) to the challenger to obtain the tuple

\[
((k_{up}, c_{ch}^0), ([\text{leak}_1, \text{leak}_2]^p), L_{\oplus}(y_{ch}, m_0^{c_{ch}}), [L_{\oplus}(y_{ch}, c_{ch}^0)]^{p-1}, [S^L(k_{pre}, p_A, k_{ch})]^p, k_{pre})
\]

as outputs (which is produced by either \( \text{LISEnc}_{k_{ch}}^+ (m_0^p) \) or \( \text{LISEnc}_{k_{ch}}^+ (m_1^p) \)).

(4) if \( d_{\ell - 3} = p_A \) then \( A' \) sets \( y_{\ell - 2} \leftarrow k_{pre} \), otherwise samples \( y_{\ell - 2} \) such that \( y_{\ell - 2} \neq k_{pre} \). It then computes \( d_{\ell - 2} \leftarrow y_{\ell - 2} \oplus m_0^{d_{\ell - 2}} \). At this stage, \( A' \) aborts, if either of the following two conditions is fulfilled:

- \( d_{\ell - 2} = p_A \), yet \( y_{\ell - 1} \neq k_{ch} \);
- \( d_{\ell - 2} \neq p_A \), yet \( y_{\ell - 1} = k_{ch} \).

Otherwise, \( A' \) uses the leakages \( [S^L(s_{\ell - 2}, p_A, k_{pre})], S^L(s_{\ell - 2}, d_{\ell - 3}, y_{\ell - 2})]^p, L_{\oplus}(y_{\ell - 2}, m_0^{d_{\ell - 2}}), [L_{\oplus}(y_{\ell - 2}, d_{\ell - 2})]^{p-1} \) as leakages in \( \ell - 1 \) th iteration, \( [S^L(k_{pre}, p_A, k_{ch})], S^L(k_{pre}, d_{\ell - 2}, y_{\ell - 1})]^p, L_{\oplus}(y_{\ell - 1}, m_0^{d_{\ell - 1}}), [L_{\oplus}(y_{\ell - 1}, d_{\ell - 1})]^{p-1} \) as leakages in the \( \ell \) th iteration in the first pass;

(5) sets \( s_{\ell + 1} \leftarrow k_{up} \) and \( d_{\ell} \leftarrow c_{ch}^{d_{\ell}} \), and uses \([\text{leak}_1, \text{leak}_2]^p, L_{\oplus}(y_{ch}, m_0^{c_{ch}}), [L_{\oplus}(y_{ch}, c_{ch}^0)]^{p-1}\) as the corresponding leakage traces (now the message absorbing phase of the first pass has been completed);

(6) Then, \( A' \) takes \( s_{\ell + 1} \) and \( d_{\ell} \) as the starting points, and emulates the remaining actions of \( \text{LIFSM} \) encrypting the message \( m_0^\ell \cdots m_0^{\ell - 1} m_0^{c_{ch}} \) to obtain \( c_1^d, \ldots, c_{\ell - 1}^d \). Note that this requires \( A' \) to uniformly sample \( z_{\ell} \) and further compute \( c_\ell^d \leftarrow z_{\ell} \oplus d_{\ell} \) and generates the traces \( L_{\oplus}(z_{\ell}, c_{\ell}^0) \) and \( [L_{\oplus}(z_{\ell}, c_{\ell}^d)]^{p-1} \) at the end of the second pass. Eventually, \( A' \) serves the ciphertext \( V||c_1^d||\cdots||c_{\ell - 1}^d||c_\ell^d \) as well as all the generated simulated leakages to \( A \), and outputs whatever \( A' \) outputs.

It can be seen that as long as \( A' \) does not abort (the probability of which is at most \( 2/2^n \) since both \( d_{\ell - 2} \) and \( y_{\ell - 1} \) are uniform), depending on whether the input tuple received by \( A' \) captures \( \text{LISEnc} \) encrypting \( m_0^p \) or \( m_1^p \), the inputs to \( A' \) capture \( \text{LIFSM} \) encrypting \( m_0^\ell \cdots m_0^{\ell - 1} m_0^t \) or \( m_1^\ell \cdots m_1^{\ell - 1} m_1^t \). Moreover, \( A' \) is indeed \((q_t + p \cdot q_r, t + p \cdot t_r - t^*)\)-bounded if \( A \) is \((q_t, t)\)-bounded. These complete the proof.

Gathering Lemmas 2, 3, and 4, we obtain Lemma 5 which shows the eavesdropper security bound of \( \text{LRFSM} \) (which, in fact, is also the eavesdropper bound of \( \text{FEMALE} \)).

**Lemma 5.** Let \( E : \mathcal{M} \times \mathcal{M} \rightarrow \mathcal{M} \) be a \((2, t, \varepsilon_E)\)-PRP, whose implementation has a leakage function \( L_E \) having \((q_s, t_s, q_t, t, \varepsilon_{(p,2),-r-sim})\) \((p,2)\)-R-simulatable leakages, and let \( S^L \) be an appropriate \((q_s, t_s)\)-bounded leakage simulator. Then, for every pair of \( \ell \)-block messages \( m_0^\ell \) and \( m_1^\ell \) and \((q_t - p \cdot q_r - q^*, t - p \cdot t_r - t^*)\)-bounded adversary \( A \), it holds

\[
\frac{\Pr[A^L(\text{LRFSM}_{s_0}(m_0^\ell) \Rightarrow 1)] - \Pr[A^L(\text{LRFSM}_{s_0}(m_1^\ell) \Rightarrow 1)]}{\leq (6\ell + 8)(\varepsilon_E + \varepsilon_{(p,2),-r-sim}) + \ell \cdot \varepsilon_{-\text{block}} + \frac{6\ell + 4}{2^n},}
\]

where \( q_r, t_r \) are as defined in Lemma 3, and \( q^*, t^* \) are as defined in Lemma 2.

**Proof.**

\[
\Pr[[A^L(\text{LRFSM}_{s_0}(m_0^\ell) \Rightarrow 1)] - \Pr[A^L(\text{LRFSM}_{s_0}(m_1^\ell) \Rightarrow 1)]] \\
\leq \sum_{b=0,1} \Pr[[A^L(\text{LRFSM}_{s_0}(m_b^\ell) \Rightarrow 1)] - \Pr[A^L(\text{LRFSM}_{s_0}(m_b^\ell) \Rightarrow 1)]] \\
\leq 2(2\ell + 4)(\varepsilon_E + \varepsilon_{(p,2),-r-sim}) + \frac{2\ell + 2}{2^n} \ (\text{by Lemma 3})
\]
For the involved term \( A \), by Lemma 4 there exists a \((q_t - q^*, t - t^*)\)-bounded adversary \( \mathcal{A}^L \) that satisfies

\[
A \leq \frac{2\ell}{2^n} + \sum_{i=1}^{\ell} |\Pr[\mathcal{A}^L(\text{LISEnc}_{i-1}^i(m^0_t)) \Rightarrow 1] - \Pr[\mathcal{A}^L(\text{LISEnc}_{i-1}^i(m^1_t)) \Rightarrow 1]| \\
\leq \sum_{i=1}^{\ell} |\Pr[\mathcal{A}^L(\text{LRSEnc}_{i-1}^i(m^0_t)) \Rightarrow 1] - \Pr[\mathcal{A}^L(\text{LRSEnc}_{i-1}^i(m^1_t)) \Rightarrow 1]| \\
\leq \ell \epsilon_{\text{block}} \quad \text{(by (1))}
\]

\[
+ 2\ell(q_E + \epsilon_{(p,2)\text{-rsim}}) \quad \text{(by Lemma 2)}.
\]

The claim thus follows.

Theorem 4, the CCAmL2 security of FEMALE, could then be derived from Lemma 5. The argument is in the next subsection.

### F.2 Proof of Theorem 8

We define trivial decryption queries: if the adversary queries \( \text{Enc}_k(N, A, M) \rightarrow C \) and makes the decryption query \( \text{Dec}_k(N, A, C) \) later on, the latter is called trivial. Trivial decryption queries are usually deemed useless in black-box models. However, with leakage, such queries may give new information to the adversary. We will have explicit arguments for handling such queries.

Then we step into the proof. We start by defining \( \mathcal{G}_0 \) as the game \( \text{PrivK}^{\text{CCAmL2},0}_k, \) and \( \mathcal{G}_0^* \) as the game \( \text{PrivK}^{\text{CCAmL2},1}_k, \). We show that they are indistinguishable from two games \( \mathcal{G}_1 \) and \( \mathcal{G}_1^* \) respectively, the gap between which is the eavesdropper security bound of LRFSM plus an additional loss.

To this end, we first introduce the game \( \mathcal{G}_1 \), which is obtained from \( \mathcal{G}_0 \) by replacing all the occurrences of \( E_k^* \) and its inverse function by a random permutation \( \pi \) and its inverse. To upper bound \( |\Pr[\mathcal{G}_1 \Rightarrow 1] - \Pr[\mathcal{G}_0 \Rightarrow 1]| \), we build a \((2q_t + 2q_d + 2q_m, t_D)\)-bounded distinguisher \( \mathcal{D}_{\text{SPRP}} \) against the SPRP security of \( E^* \) (more precisely, \( E \), since the crypto strength of \( E^* \) stems from \( E \)). The challenger \( \mathcal{D}_{\text{SPRP}} \) picks all the necessary constants itself, and emulates the encryption and decryption oracles to interact with \( \mathcal{A}^L \) as follows. On each query made by \( \mathcal{A}^L \), \( \mathcal{D}_{\text{SPRP}} \) emulates all the actions described by \( \text{FEMALE} \), except for the computations involving \( E^* \), which is replaced by calls to its own permutation oracle \( O \) (which is either \( E_k \) or \( \pi \)). When \( \mathcal{A}^L \) outputs its guess bit, \( \mathcal{D}_{\text{SPRP}} \) returns that bit as its own guess. It’s clear that depending on whether \( \mathcal{D}_{\text{SPRP}} \) is interacting with \( E_k \) or \( \pi \), \( \mathcal{A}^L \) is playing \( \mathcal{G}_0 \) or \( \mathcal{G}_1 \). Therefore, any difference between \( \Pr[\mathcal{G}_1 \Rightarrow 1] \) and \( \Pr[\mathcal{G}_0 \Rightarrow 1] \) leads to the same difference in \( \mathcal{D}_{\text{SPRP}} \) distinguishing \( E_k \) from \( \pi \). During the emulation, the simulated \( \text{FEMALE} \) scheme receives \( q_t + q_m \) encryption queries and \( q_d \) decryption queries. Therefore, \( \mathcal{D}_{\text{SPRP}} \) is a \((2q_t + 2q_d + 2q_m, t_D)\)-bounded adversary against \( E \), making at most \( 2q_t + 2q_d + 2q_m \) queries to \( O \) and running in time at most \( t + (q_t + q_d + q_m) t^*_{\text{pass}} \leq t' \). Thus \( |\Pr[\mathcal{G}_1 \Rightarrow 1] - \Pr[\mathcal{G}_0 \Rightarrow 1]| \leq \epsilon_{\text{E}} \)

follows from the assumption that \( E \) is a \((2q_t + 2q_d + 2q_m, t', \epsilon_{\text{E}})\)-SPRP.

Similarly, we replace \( E^* \) by \( \pi \) to turn \( \mathcal{G}_0 \) into \( \mathcal{G}_1^* \), which also introduces a gap of \( \epsilon_{\text{E}} \).

We then prove

\[
|\Pr[(\mathcal{A}^L)^{\mathcal{G}_1} \Rightarrow 1] - \Pr[(\mathcal{A}^L)^{\mathcal{G}_1^*} \Rightarrow 1]| \leq \epsilon_{\text{cr}} + \epsilon_{\text{sp}} + \sum_{i=1}^{\ell} \epsilon_{\text{FEMALE-eav}}(\ell_i),
\]

where \( \ell_i \) is the number of blocks in the \( i \)th challenge message, and the bound on \( \epsilon_{\text{FEMALE-eav}}(\ell_i) \) is as defined in Theorem 5 but where \( \ell_i \) corresponds to the block-length of the \( i \)-th challenge messages. This plus the above gap \( 2\epsilon_{\text{E}} \) yields the claimed bound. To this end, we denote the \( q_m \) challenge tuples by

\[
(N_{c_1}, Ac_1, M_{c_1}^0, M_{c_1}^1), \ldots, (N_{c_m}, Ac_m, M_{c_m}^0, M_{c_m}^1).
\]

Then, we note that in \( \mathcal{G}_1 \), the \( q_m \) messages being encrypted by the challenge encryption oracle are \( M_{c_1}^0, \ldots, M_{c_m}^0 \), while those encrypted in \( \mathcal{G}_1^* \) are \( M_{c_1}^1, \ldots, M_{c_m}^1 \). We use \( q_m \) hops to replace \( M_{c_1}^0, \ldots, M_{c_m}^0 \) by \( M_{c_1}^1, \ldots, M_{c_m}^1 \) in turn, to show that \( \mathcal{G}_1 \) can be transited to \( \mathcal{G}_1^* \). For convenience, we define \( \mathcal{G}_{2,0} = \mathcal{G}_1 \), and define a sequence of games

\[
\mathcal{G}_{2,1}, \mathcal{G}_{2,2}, \ldots, \mathcal{G}_{2,q_m},
\]

39
such that in the $i$-th system $G_{2,i}$, the first $i$ messages processed by the challenge encryption oracle are $Mc_1^i, \ldots, Mc_{i}^i$, while the remaining $q_m-i$ messages being processed are $Mc_{i+1}^1, \ldots, Mc_{q_m}^1$. It can be seen actually $G_{2,q_m} = G_{2}^*$.

We then show that for $i = 1, \ldots, q_m$, $G_{2,i-1}$ and $G_{2,i}$ are indistinguishable for $A^i$. For this, from $A^i$ we build an adversary $A^{i'}$, such that if $A^i$ distinguishes $G_{2,i-1}$ and $G_{2,i}$ then $A^{i'}$ breaks the eavesdropper security of LRFSM. In detail, $A^{i'}$ keeps a pair of tables $(I, I^{-1})$ to simulate the random permutation $\pi$ (via lazy sampling). And it runs $A^i$, and reacts as follows:

- Upon an encryption query $(N_j, A_j, M_j)$ from $A^i$, $A^{i'}$ computes $R_i \leftarrow H(0||N_j||A_i)$. Then:
  - if $0||R_i \notin I$, $A^{i'}$ samples $s_0^{(i)} \leftarrow \mathcal{M}\setminus I^{-1}$, sets $I(0||R_i) \leftarrow s_0^{(i)}$ and $I^{-1}(s_0^{(i)}) \leftarrow 0||R_i$, and then runs $LRFSM_{s_0^{(i)}}(M_j)$ to get the ciphertext $(V_i, c_i)$ and leakages. $A^{i'}$ then computes $h_i \leftarrow H(1||R_i||V||c_i)$ and $T_i \leftarrow I(1||h_i)$ (if $1||h_i \notin I$ then $A^i$ defines $I(1||h_i)$ as a value newly sampled from $\mathcal{M}\setminus I^{-1}$).
  - Finally, $A^{i'}$ returns the outputs $(V_i, c_i, T_i)$ and the leakages to $A^i$;
  - if $0||R_i \in I$, $A^{i'}$ simply runs $LRFSM_{I(0||R_i)}(M_j)$, performs the tag generation action $h_i \leftarrow H(1||R_i||V_i||c_i)$ and $T_i \leftarrow I(1||h_i)$ on the obtained $V_i$ and $c_i$, and returns $(V_i, c_i, T_i)$ and the leakages to $A^i$.

- Upon a trivial decryption query $(N_j, A_j, C_j)$ from $A^i$, $A^{i'}$ parses $C_j = (V_j, c_j, T_j)$, and computes $R_i \leftarrow H(0||N_j||A_j)$ and $h_j \leftarrow H(1||R_i||V_j||c_j)$. Then:
  - if $T_j \notin I^{-1}$, $A^{i'}$ samples $Z_j \leftarrow \mathcal{M}\setminus I$, and sets $I(Z_j) \leftarrow T_j$ and $I^{-1}(T_j) \leftarrow Z_j$;
  - if $T_j \in I^{-1}$, $A^{i'}$ simply sets $Z_j \leftarrow I^{-1}(T_j)$.
Now $A^{i'}$ defines $h_j = \text{trunc}(Z_j)$, aborts if $h_j = h_j^*$ (this type of abortion is defined as BadDec), and returns $(L, h_j^*)$ to $A^i$.

- Upon $A^i$ submitting the $j$-th challenge tuple $(N_{C_j}, A_{C_j}, M_{C_j}, M_{C_j}^0)$, $A^{i'}$ computes $R_{C_j} \leftarrow H(0||N_{C_j}|A_{C_j})$. Then, if $0||R_{C_j} \in I$, $A^{i'}$ aborts (this type of abortion is defined as BadChall); otherwise, its action depends on $j$:
  - When $j < i$, it simply encrypts $M_{C_j}^0$ and returns. In detail, $A^{i'}$ samples $s_0^{(j)} \leftarrow \mathcal{M}\setminus I^{-1}$, sets $I(0||R_{C_j}) \leftarrow s_0^{(j)}$ and $I^{-1}(s_0^{(j)}) \leftarrow 0||R_{C_j}$, and then runs $LRFSM_{I(0||R_{C_j})}(M_{C_j}^0)$, performs the tag generation action $h_{C_j} \leftarrow H(1||R_{C_j}||V_{C_j}||c_{C_j})$ and $T_{C_j} \leftarrow I(1||h_{C_j})$ on the obtained $V_{C_j}$ and $c_{C_j}$, and returns $(V_{C_j}, c_{C_j}, T_{C_j})$ and the leakages to $A^i$;
  - When $j = i$, it relays $M_{C_j}^0$ and $M_{C_j}$ to its eavesdropper security challenger to obtain $(V_{C_j}^b, c_{C_j}^b)$ and leakages $\text{leak}_{ac}$ and $[\text{leak}_{dec}]^{p-1}_b$, and then computes $T_{C_j} \leftarrow H(1||H(1||R_{C_j}||V_{C_j}^b||c_{C_j}^b))$ (possibly defines a new entry in $I$) and returns $C_{C_j}^b = (V_{C_j}^b, c_{C_j}^b, T_{C_j})$ to $A^i$. Note that this means the relation $I(0||R_{C_j}) = s_0^{(b)}$ is implicitly fixed, where $s_0^{(b)}$ is the secret key generated inside the eavesdropper security challenger;
  - When $j > i$, it simply encrypts $M_{C_j}^i$ and returns. The details are similar to the described case $j < i$.
  - Upon $A^i$ making the $\lambda$-th query to $L_{\text{decch}}(j)$ $(1 \leq \lambda \leq p-1)$,
    - When $j \neq i$, $A^{i'}$ performs a corresponding decryption process (decrypting $(N_{C_j}, A_{C_j}, C_{C_j})$ when $j < i$, and $(N_{C_j}, A_{C_j}, C_{C_j}^b)$ when $j > i$) and returns the obtained leakages to $A^i$;
    - When $j = i$, $A^{i'}$ simply returns the $\lambda$-th trace in the vector $[\text{leak}_{dec}]^{p-1}_b$ as the answer.

Moreover, whenever new entries are added to $I$, $A^{i'}$ aborts if $0||R_{C_j} \notin I$ (so that the implicit relation $I(0||R_{C_j}) = s_0^{(b)}$ never causes inconsistency).

It can be seen that the whole process is the same as either $G_{2,i-1}$ or $G_{2,i}$ depending on whether $b = 0$ or $1$, given that:

(i) BadDec never occurs, and
(ii) BadChall never occurs.

On the other hand, besides running $A^i$, $A^{i'}$ samples at most $2(q_e + q_d + q_m)$ random values (to emulate $\pi$) and internally processes $q_e + q_d + q_m$ queries. Therefore, the running time of $A^{i'}$ is at most $t_{A^{i'}} \leq t + (q_e + q_d + q_m)(2t_b + t_{\text{pass,ss}})$ (while $A^{i'}$ makes $q_i$ queries to $L$, which is the same as $A^i$).

We need to bound the probabilities of the defined events to complete the proof. They are as follows.
Lemma 6. \( \Pr[\text{BadDec} \lor \text{BadChall}] \leq \varepsilon_{cr} + \varepsilon_{pr}. \)

Proof. To show this, we introduce two other events as follows:

- \text{CollH:} during the interaction between \( \mathcal{A}^L \) and the eavesdropper security challenger, there appears two calls to \( H(x) \) and \( H(x') \) such that \( x \neq x' \) while \( H(x) = H(x') \);
- \text{PreimgH:} during the interaction between \( \mathcal{A}^L \) and the eavesdropper security challenger, there appears a decryption query \((N_j, A_j, (V_j, c_j, T_j))\) such that \( T_j \notin \Pi^{-1} \) before this query is made, yet after \( \Pi^{-1}(T_j) \) is defined, it holds \( H(x) = \text{trunc}(\Pi^{-1}(T_j)) \) for any hash-call \( H(x) \) occurring during the game.

If \text{CollH} happens, then from \( \mathcal{A}^L \) we are able to build a collision adversary \( \mathcal{A}_{cr} \) against \( H \): \( \mathcal{A}_{cr} \) just simulates the eavesdropper security challenger and interacts with \( \mathcal{A}^L \), and waits for \text{CollH} to occur. It’s clear that the running time of \( \mathcal{A}_{cr} \) is no more than \( t_{\mathcal{A}^L} \) plus the time needed to emulate the eavesdropper challenger, which turns out \( t + (q_e + q_d + q_m)(2t_8 + t_{1-pass}) = t' \) in total. Therefore, by the assumption on the collision resistance of \( H \), we obtain \( \Pr[\text{CollH}] \leq \varepsilon_{cr} \).

On the other hand, if \text{PreimgH} happens, then from \( \mathcal{A}^L \) we are able to build a preimage adversary against \( H \). To this end, note that by the definition of \text{PreimgH} if it happens with respect to a decryption query \((N_j, A_j, (V_j, c_j, T_j))\), then \( T_j \notin \Pi^{-1} \) before this query is made. According to the description of \( \mathcal{A}^L \), \( \Pi^{-1}(T_j) \) will be defined to a randomly sampled value, and thus \( H(1||R_j||V_j||c_j) = \text{trunc}(\Pi^{-1}(T_j)) \) exactly fits into the definition of range-oriented preimage resistance. Therefore, we could use an adversary \( \mathcal{A}_{pr} \) to emulate the interaction between \( \mathcal{A}^L \) and the eavesdropper security challenger, and if \text{PreimgH} happens then \( \mathcal{A}_{pr} \) turns out a preimage adversary against the hash function \( H \). Since \( \mathcal{A}_{pr} \)'s running time is also at most \( t' \), and there are at most \( q_d \) such decryption queries, we have \( \Pr[\text{PreimgH}] \leq \varepsilon_{pr} \) by the assumption that \( H \) is \( (q_d, t', \varepsilon_{pr}) \)-range-oriented preimage resistance. This further yields \( \Pr[\text{CollH} \lor \text{PreimgH}] \leq \varepsilon_{cr} + \varepsilon_{pr} \).

We now prove \( \Pr[\text{BadDec} \lor \text{BadChall} \mid \neg(\text{CollH} \lor \text{PreH})] = 0 \) to complete the proof. Assume that \text{BadDec} occurs, and let \((N_i, A_i, (V_i, c_i, T_i))\) be the decryption query with the smallest index that passes the integrity checking. We distinguish several cases:

(i) Case 1: upon this decryption query, it holds \( T_i \notin \Pi^{-1} \). Then \( \Pi^{-1}(T_i) \) will be defined random and \( H(1||R_i||V_i||c_i) = \text{trunc}(\Pi^{-1}(T_i)) \) implies the occurrence of \text{PreimgH}.

(ii) Case 2: upon this decryption query it holds \( T_i \in \Pi^{-1} \), and \( \Pi^{-1}(T_i) \) was defined due to an earlier backward call to \( \pi^{-1} \). Then it’s the same as Case 1.

(iii) Case 3: contrary to Case 1 & 2, and \( \Pi^{-1}(T_i) = 0 || Z_i \) for some \( Z_i \in B \). Then the entry \( \Pi(0||Z_i) \) was defined due to the “first” leak free call \( E'_1(0||.) \). Then \( H(1||R_i||V_i||c_i) = \text{trunc}(\Pi^{-1}(T_i)) \) means during the game there exists \( N', A' \) such that \( H(0||N'||A') = Z_i = H(1||R_i||V_i||c_i) \), indicating \text{CollH}.

(iv) Case 4: contrary to Case 1 & 2, and \( \Pi^{-1}(T_i) = 1 || Z_i \) for some \( Z_i \in B \). Then the entry \( \Pi(1||Z_i) \) was defined in an earlier encryption query, due to the “second” leak free call \( E'_1(1||.) \). Assume that this encryption query was \((N_j, A_j, M_j)\), and the corresponding response was \( C_j = (V_j, c_j, T_j) \). Clearly it has to be \( T_j = T_i \).

Since we are considering non-trivial decryption queries, this means \((N_j, A_j, V_j, c_j) \neq (N_i, A_i, V_i, c_i) \). Now, if \( V_j||c_j \neq V_i||c_i \) then it holds \( R_j||V_j||c_j \neq R_i||V_i||c_i \); otherwise, either \( N_j \neq N_i \) or \( A_j \neq A_i \) would imply \( R_j \neq R_i \) by \( \neg \text{CollH} \), which by \( \neg \text{CollH} \) further implies \( Z_i = H(1||R_j||V_j||c_j) \neq H(1||R_i||V_i||c_i) \).

Therefore, \( \Pr[\text{BadDec} \mid \neg(\text{CollH} \lor \text{PreH})] = 0 \).

Conditioned on that \text{BadDec} does not occur, entries of the form \( \Pi(0||R) \) may be defined due to two events:

- First, the “first” leak free call \( E^*(0||R) \) that occurs during an encryption query (not possible in decryption queries by \( \neg \text{BadDec} \));
- Second, the “second” backward leak free call \( (E^*)^{-1} \) during a decryption query.

If \text{BadChall} due to the first type of entries, then there necessarily exists an encryption query \((N, A, M)\) such that \( H(0||N||A) = H(0||N_c||A_c) \). According to the requirements of the CCAmL2 security game, it has to be \((N, A) \neq (N_c, A_c) \); therefore, \( H(0||N||A) = H(0||N_c||A_c) \) would contradict \( \neg \text{CollH} \). If \text{BadChall} due to the second type of entries, then it means \( H(0||N_c||A_c) = \text{trunc}(\Pi^{-1}(T)) \) for \( T \) specified in some decryption query. By the assumption on this type of entries, \( \text{trunc}(\Pi^{-1}(T)) \) is defined to random, and thus this case contradicts \( \neg \text{PreimgH} \). Thus the claim.
By all the above, define  
\[ \text{Bad} = \text{BadDec} \lor \text{BadChall}, \]
then we have  
\[ \Pr[(A^L_G^2, i) \Rightarrow 1] - \Pr[(A^L_G^2, i-1) \Rightarrow 1] \]
\[ \leq \Pr[(A^L_G^2, i) \Rightarrow 1 \land \text{Bad}] - \Pr[(A^L_G^2, i-1) \Rightarrow 1 \land \text{Bad}] + \epsilon_{\text{FEMALE-eav}}(\ell_i). \]

This means  
\[ |\Pr[(A^L_G^2) \Rightarrow 1] - \Pr[(A^L_G^1) \Rightarrow 1]| \leq \Pr[(A^L_G^{2,q_m}) \Rightarrow 1] - \Pr[(A^L_G^{2,0}) \Rightarrow 1] \]
\[ \leq \sum_{i=1}^{q_m} \left( \Pr[(A^L_G^{2,i}) \Rightarrow 1] - \Pr[(A^L_G^{2,i-1}) \Rightarrow 1] \right) \]
\[ \leq \epsilon_{\text{cr}} + \epsilon_{\text{pr}} + \sum_{i=1}^{q_m} \epsilon_{\text{FEMALE-eav}}(\ell_i). \]

These complete the proof.

**Influence of Empty Message.** It’s meaningless to use empty message for \((m)\text{CCAmL2}\) challenge message, as otherwise it’s not possible to pick two distinct challenges. Therefore, in the \((m)\text{CCAmL2}\) game, they can only appear in decryptions and non-challenge encryptions. Our proof has covered both cases (without specific treatment): for such decryptions, it can be seen the arguments around the bad events (in particular, \text{BadDec}) remains valid; for such non-challenge encryptions, the adversary \(A^L_{\tilde{G}}\) remains able to simulate the corresponding actions.

**F.3 Proof for \text{CIML2}**

Since the main body only contains an informal theorem we give the formal one as follows.

**Theorem 9.** Let \(H : \{0,1\}^* \rightarrow B\) be a \((0,t',\epsilon_{\text{cr}})\)-collision resistant and \((q_d+1, t', \epsilon_{\text{pr}})\)-range-oriented preimage resistant hash function. And let \(E : M \times M \rightarrow M\) be a \((2q_e + 2q_d + 2, t', \epsilon_{E^*})\)-SPRP with \(E^*\) being its strongly protected leak free implementation. Then \text{FEMALE} provides \((q_e, q_d, t, \epsilon_{\text{CIML2}})\)-ciphertext integrity with coin misuse and unbounded leakage on encryption and decryption as long as \(t \leq t' - (q_e + q_d + 1)t_{1-\text{pass}}\), where \(t_{1-\text{pass}}\) is the maximum running time of \text{FEMALE} upon a single (encryption or decryption) query, and

\[ \epsilon_{\text{CIML2}} \leq \epsilon_{E^*} + \epsilon_{\text{cr}} + \epsilon_{\text{pr}}. \]

The claim can be established by some intermediate results obtained during the proof of Theorem 4. To see this, let’s revisit these results:

- First, define \(G_0\) as the real \text{CIML2} security game between the adversary and \text{FEMALE}. Then replacing \(E_1^*\) by a random permutation \(\pi\), we obtain a game \(G_1\), with a gap of \(\epsilon_1^E\);
- Second, as long as neither of the two bad events \text{CollH} and \text{PreimgH} happens during the execution of \(G_1\), every non-trivial decryption query would yield ⊥ as the answer. And we have \(\Pr[\text{CollH} \lor \text{PreimgH}] \leq \epsilon_{\text{cr}} + \epsilon_{\text{pr}}\).

Note that the above steps do not rely on additional leakage assumptions. Therefore, the claim holds in the unbounded leakage model. And as remarked in the previous subsection, our proof has covered empty messages.

**F.4 Proof for MR**

Also we give the formal theorem as follows.
Theorem 10. Let $H: \{0,1\}^* \to \mathcal{B}$ be a $(0,t',\varepsilon_{cr})$-collision resistant and $(q_d+1,t',\varepsilon_{pr})$-range-oriented preimage resistant hash function. Let $E: \mathcal{M} \times \mathcal{M} \to \mathcal{M}$ be a $(2q_e+2q_d,t',\varepsilon_E)$-SPRP. Then the FEMALE scheme is $(q_e,q_d,t,\varepsilon_{MR})$-MR as long as $t \leq t' - (q_e + q_d)(t_{1-pass} + (4\ell + 4)t_q)$, where $t_{1-pass}$ is the maximum running time of FEMALE upon a single (encryption or decryption) query, $t_s$ is the time needed for randomly sampling a value from $\mathcal{M}$, and:

$$
\varepsilon_{MR} \leq \varepsilon_E + \varepsilon_{cr} + \varepsilon_{pr} + (2\ell + 4)q_\varepsilon E + \frac{2(\ell + 1)q_\varepsilon + (q_e + q_d)^2 + q_e^2}{2^{n+1}}.
$$

Let $A$ be a $(q_e, q_d, t, \varepsilon_{MR})$-bounded adversary against the misuse resistance of FEMALE, and assume $A$ never makes “redundant” queries, i.e.

- $A$ never repeats queries, and
- $A$ never decrypts the ciphertext produced by previous encryption queries.

Furthermore, we write the encryption queries as

$$(N_1, A_1, M_1), (N_2, A_2, M_2), \ldots, (N_{q_e}, A_{q_e}, M_{q_e});$$

and we assume the length of $M_i$ is $\ell_i \leq \ell$. We stress that $N_1 \parallel A_1, \ldots, N_{q_e} \parallel A_{q_e}$ are not necessarily distinct since we are in misuse setting.

We will use a sequence of hybrid games, beginning with the real game, named $G_0$, where $A$ interacts with $\operatorname{Enc}_k$ and $\operatorname{Dec}_k$, and ending with random-and-invalid game, named $G_{q_e+1}$, where $A$ interacts with $\perp$ and $\perp$. Then, using the adversary $A$ we show that any transition can be reduced to an efficient adversary against one of the assumptions. For convenience, we name $E_i$ the event that $A$ outputs 1 at the end of $G_i$.

We start from $G_0 = A^{\operatorname{Enc}_k, \operatorname{Dec}_k}$. The first two steps are similar to those appeared in the proof of Theorem 4:

(i) first replace all the occurrences of $E_k^*$ by a random permutation $\pi$ to obtain the first intermediate game $G_{0,1}$. And we introduce two abort conditions $\operatorname{CollH}$ and $\operatorname{PreimH}$ into $G_{0,1}$: the former is fulfilled if there exists two calls to $H(x)$ and $H(x')$ such that $x \neq x'$ yet $H(x) = H(x')$, while the latter is fulfilled if there exists a hash-call $H(x)$ and an fresh inverse call $\pi^{-1}(T)$ such that $H(x) = \operatorname{trunc}(\pi^{-1}(T))$. It can be seen $\Pr[G_{0,1} \text{ aborts}] \leq \varepsilon_{cr} + \varepsilon_{pr}$, otherwise a corresponding adversary running in time at most $t' \leq t + (q_e + q_d)t_{1-pass}$ can be built against $H$;

(ii) it holds $|\Pr[E_{0,1}] - \Pr[G_{0,1} \text{ does not abort}]| \leq \varepsilon_E$, otherwise from $A$ we can build an SPRP adversary $D_{SPRP}$ against $E_k$, and $D_{SPRP}$ makes at most $2(q_e + q_d)$ queries and runs in time at most $t'$.

We next define $\ell_0 = 1$, and consider the second intermediate game $G_{0,\ell_0}$, which is the same as $G_{0,1}$ except that all decryption queries are simply answered by $\perp$. In a similar vein to the proof of Theorem 4, we know that if neither of the two games abort, then they would have no difference. Therefore, $\Pr[E_{0,1} \mid G_{0,1} \text{ does not abort}] = \Pr[E_{0,\ell_0} \mid G_{0,\ell_0} \text{ does not abort}]$.

We then consider the encryption queries in turn, and use a sequence of games $G_{1,1}, G_{1,1}, G_{1,e_1}, G_{1,e_1+1}, G_{1,e_1+2} = G^*_1, G^*_1, \ldots, G^*_{1,e_1}, G_{2,1}, G_{2,2}, \ldots$

$$
G_{q-e,1} = G^*_{q-e,1,e_{q-e}+1}, \ldots, G_{q-e,1,e_{q-e}+1, e_{q-e}+2} = G^*_q, G^*_q, \ldots, G^*_q e_{q-e},
$$

to show that the ciphertexts are indistinguishable from uniform. In detail, for $i = 1, \ldots, q_e$, we start from $G^*_{i-1, \ell_{i-1}} = G_{i-1}$, and transit to $G^*_{i, \ell_i}$. In this game, the tuple $(N_i, A_i)$ would give rise to $R_i = H(0 \parallel N_i \parallel A_i)$ and $s_0^{(i)} = \pi(0 \parallel R_i)$. We distinguish two cases:

Non-reuse Case: $(N_i, A_i)$ never appeared before. Then conditioned on $\neg \operatorname{CollH}$, $R_i$ never appeared before, and thus $s_0^{(i)}$ must be distinct from all the already appeared intermediate values $s_0^{(1)}, \ldots, s_0^{(i-1)}$ since $\pi$ is a permutation. In the first phase of the transition, for $j = 0, \ldots, \ell_i + 1$, we consider the game $G_{j-1}$, and replace all the subsequently appearing calls of the form $E_{s_0^{(i)}}(x)$ by calls to a new random permutation $P_{i,j}(x)$. This
yields $G_{i,j}$. It holds $|\Pr[E_{i,j}] - \Pr[E_{i,j-1}]| \leq \varepsilon_E$, otherwise a PRP adversary $B_E$ against $E$ could be built by emulating the common part of $G_{i,0}$ and $G_{i,-1}$. In detail, $B_E$ interacts with $A$, and simulates the previously appeared random permutations $P_{1,0}, P_{1,1}, \ldots, P_{j-1,0}$ by lazy sampling, till the intermediate values $s_j^{(i)}$ and $d_{j-1}^{(i)}$ being computed. Then, $B_E$ supplies $p_A$ and $d_{j-1}^{(i)}$ to its challenge oracle $O$ (which is either $E_k$ for a secret key $k$ or the random permutation $P_{j,j}$), and uses the obtained $s_{j+1}^{(i)} \leftarrow O(p_A)$ to continue emulating the process of real FEMALE encryption. Moreover, to ensure consistency, for queries received in future with $(N_i, A_i)$ as the involved nonce, $B_E$ uses $P_{0,0}, \ldots, P_{j-1,0}, O$ for the first $j+1$ calls to $E$. To emulate these, $B_E$ makes at most $2\varepsilon_E$ queries to $O$, and the running time of $B_E$ is clearly at most $t'$. Thus $B_E$’s advantage does not exceed $\varepsilon_E$.

Then, consider the game $G_{i,\ell+1}$, and let $w^{(i)} = P_{i,0}(p_B)$. We replace all the subsequently appearing calls of the form $E_{w^{(i)}(x)}$ by calls to a new random permutation $P_{i,\ell+2}(x)$, to obtain the game $G_{i,\ell+2}$. Similarly to the above, $|\Pr[E_{i,\ell+2}] - \Pr[E_{i,\ell+1}]| \leq \varepsilon_E$. We further replace all the subsequently appearing calls of the form $E_{w^{(i)}(x)}$ by calls to a new random permutation $P_{i,\ell+3}(x)$, to obtain the game $G_{i,\ell+3}$, with $|\Pr[E_{i,\ell+3}] - \Pr[E_{i,\ell+2}]| \leq \varepsilon_E$. By this, the newly derived initial key $k_1^{(i)} = P_{i,\ell+3}(V^{(i)})$ for the one-time encryption is uniform.

Then in the second phase, for $j = 1, \ldots, \ell$, we replace all the subsequently appearing calls to $E_{s^{(i)}_{\ell,j}}$ by corresponding calls to a new random permutation $P'_{j,j}$, and this yields $G_{s^{(i)}_{\ell,j}}$. The gap between each pair of games $(G_{s^{(i)}_{\ell,j}}, G_{s^{(i)}_{\ell,j}})$ is at most $\varepsilon_E$. We also have $|\Pr[E_{s^{(i)}_{\ell,j}}] - \Pr[E_{s^{(i)}_{\ell,j-1}}]| \leq \varepsilon_E$.

It can be seen that in the last game $G_{s^{(i)}_{\ell,\ell}}$, the intermediate value $V^{(i)}$ as well as the ciphertext blocks $c_1^{(i)}, \ldots, c_{\ell}^{(i)}$ resulted from the encryption query $(N_i, A_i, M_i)$ are random. The next paragraph would discuss the nonce-reuse case.

**Reuse Case:** $(N_i, A_i)$ has appeared before. Assume that $\alpha$ is the smallest index such that $(N_{\alpha}, A_{\alpha}) = (N_i, A_i)$. We further distinguish two cases:

- **Case 1:** the length $\ell_{\alpha}$ does not exceed the maximum length of the messages already processed with the pair $(N_i, A_i)$. Then it can be seen that in the game $G_{s^{(i)}_{\ell,\ell}}$, when processing the encryption query $(N_i, A_i, M_i)$, all the calls to $E$ during the first pass have been replaced by calls to random permutations $P_{\alpha,0}, \ldots, P_{\alpha,\ell_{\alpha}}$. Therefore, we simply let $G_{s^{(i)}_{\ell,\ell}} = G_{s^{(i)}_{\ell,\ell}}$.

- **Case 2:** $\ell_{\alpha}$ is the longest message among the messages encrypted with the pair $(N_i, A_i)$. Assume that the calls $E_{s^{(i)}_{\alpha,0}}, \ldots, E_{s^{(i)}_{\alpha,j}}$ have been replaced by random permutations $P_{\alpha,0}, \ldots, P_{\alpha,\beta}$ (note that $s_{\beta}^{(i)} = U^{(j)}$ for some $j < i$), and let $s_{\beta+1}^{(i)} = P_{\alpha,\beta}(p_A)$. Define $G_{i,\beta} = G_{s^{(i)}_{\beta}}$. Then, for $j = \beta + 1, \ldots, \ell_{\alpha} + 2$, we replace all the subsequently appearing calls of the form $E_{s_{\alpha,j}}^{(i)}(x)$ (taking $s_{\ell_{\alpha}+2}^{(i)} = U^{(j)}$) by calls to a new random permutation $P_{\alpha,j}(x)$. This yields $G_{i,j+1}$. For convenience we also write $P_{\alpha,j}$ for the permutation $P_{i,j}$. These steps eventually yield the game $G_{s^{(i)}_{\ell,\ell}} = G_{s^{(i)}_{\ell,\ell}}$.

Unlike the non-reuse case, we introduce three abort conditions into the game $G_{s^{(i)}_{\ell,\ell}}$. In detail, $G_{s^{(i)}_{\ell,\ell}}$ aborts, if at least one of the following occurs during its execution:

- the event CollD$_i$, which happens if there exists $j < i$ such that $N_j \parallel A_j = N_i \parallel A_i$ and $\ell_j = \ell_i$ for the $j$-th encryption query $(N_j, A_j, M_j)$, and there exists an index $\gamma$ such that $m_1^{(i)} \parallel \cdots \parallel m_\gamma^{(i)} \neq m_1^{(j)} \parallel \cdots \parallel m_\gamma^{(j)}$, yet $d_{\gamma}^{(i)} = d_{\gamma}^{(j)}$.

- the event CollW$_i$, which happens if there exists $j < i$ such that $N_j \parallel A_j = N_i \parallel A_i$ for $(N_j, A_j, M_j)$, and $W^{(i)} = W^{(j)}$.

To analyze the events, we consider every index $j < i$. If $N_j \parallel A_j \neq N_i \parallel A_i$, then neither CollD$_i$ nor CollW$_i$ could happen with respect to $j$. If $N_j \parallel A_j = N_i \parallel A_i$ and $\ell_j \neq \ell_i$, then CollD$_i$ could happen. In this case, assume that $\alpha$ is the smallest index such that $(N_{\alpha}, A_{\alpha}) = (N_i, A_i)$. Then by our convention, we have

$$W^{(i)} = P_{\alpha,\ell_{\alpha}} (d_{\ell_{\alpha}}^{(i)}) \quad \text{and} \quad W^{(j)} = P_{\alpha,\ell_{\alpha}+1} (d_{\ell_{\alpha}+1}^{(j)}).$$

10 Similarly to the proof of Lemma 3, when $j = 0$ we take $d_{-1}^{(j)} = p_B$ and $w^{(i)}$ as the corresponding output to ease the language.
with $P_{\alpha, \ell_i+1}$ and $P_{\alpha, \ell_j+1}$ being two independent random permutations. Moreover, if abortion does not happen, then conditioned on the transcripts of queries and answers obtained so far, the values $P_{\alpha, \ell_i+1}(d^{(i)}_{\ell_i})$ and $P_{\alpha, \ell_j+1}(d^{(j)}_{\ell_j})$ remain uniform, since

- the ciphertexts in the transcript are clearly independent of these two values, and
- the values $V^{(1)}, \ldots, V^{(i-1)}$ in the transcripts are outputs of the permutations $P_{1, \ell_i+2}, \ldots, P_{i-1, \ell_i+1}$, thus also independent of these two values.

Therefore, it holds

$$\Pr[W^{(i)} = W^{(j)}] \leq \frac{1}{2^n}.$$

When $N_j \parallel A_j = N_i \parallel A_i$ and $\ell_i = \ell_j$, then both CollD$_i$ and CollW$_i$ could happen. In this case, it necessarily be $M_i \neq M_j$ since A does not repeat queries. Wlog assume that $\gamma$ is the smallest index such that $m^{(j)}_{\gamma} \neq m^{(i)}_{\gamma}$. Then it can be seen $d^{(j)}_{\gamma-1} = d^{(i)}_{\gamma-1}$. Moreover, $N_j \parallel A_j = N_i \parallel A_i$ implies $s^{(i)}_0 = s^{(j)}_0, \ldots, s^{(i)}_\gamma = s^{(j)}_\gamma$, and thus the first $\gamma$ random permutations used for processing the two queries are the same ones. Assume that $\alpha$ is the smallest index such that $(N_\alpha, A_\alpha) = (N_i, A_i)$. Then by our convention, these $\gamma$ random permutations are $P_{\alpha,0}, \ldots, P_{\alpha,\gamma}$. Then it can be seen $d^{(j)}_{\gamma} \neq d^{(i)}_{\gamma}$, since $m^{(j)}_{\gamma} \neq m^{(i)}_{\gamma}$ implies

$$P_{\alpha,\gamma}(d^{(j)}_{\gamma-1}) \oplus m^{(j)}_{\gamma} \neq P_{\alpha,\gamma}(d^{(i)}_{\gamma-1}) \oplus m^{(i)}_{\gamma}.$$ 

As discussed, conditioned on the transcripts of queries and answers obtained so far, the values $P_{\alpha,\gamma+1}(d^{(j)}_{\gamma})$ and $P_{\alpha,\gamma+1}(d^{(i)}_{\gamma})$ remain uniform. By these,

$$\Pr[d^{(j)}_{\gamma+1} = d^{(i)}_{\gamma+1}] = \Pr[P_{\alpha,\gamma+1}(d^{(j)}_{\gamma}) \oplus m^{(j)}_{\gamma+1} = P_{\alpha,\gamma+1}(d^{(i)}_{\gamma}) \oplus m^{(i)}_{\gamma+1}] \leq \frac{1}{2^n}.$$ 

Following the same line, we obtain $\Pr[d^{(j)}_{\gamma+2} = d^{(i)}_{\gamma+2}] \leq \frac{1}{2^n}$, $\ldots$, $\Pr[d^{(j)}_{\ell_j} = d^{(i)}_{\ell_j}] \leq \frac{1}{2^n}$, and $\Pr[W^{(j)} = W^{(i)}] \leq \frac{1}{2^n}$. Therefore, for such an index $j$, the probability that either CollD$_i$ or CollW$_i$ happens is at most $\frac{\ell+1}{2^n}$. 

By the above, in any case, the following holds:

$$\Pr[\text{CollD}_i \lor \text{CollW}_i \mid \neg \text{CollH}] \leq \frac{\ell+1}{2^n}.$$ 

If $G^*_{\alpha,0}$ does not abort, then the produced intermediate value $W^{(i)}$ is distinct from all the previous value $W^{(j)}$ for $N_j \parallel A_j = N_i \parallel A_i$. This means the further generated $V^{(i)}$ are also distinct from these $V^{(j)}$, and further $k^{(i)}_1 \neq k^{(j)}_1$. On the other hand, for the previous queries with $N_j \parallel A_j \neq N_i \parallel A_i$, conditioned on $\neg \text{CollH}$ we have $R_j \neq R_i$, and thus the subsequent processes utilize completed independent random permutations. This means for such index $j$ the produced one-time encryption initial key $k^{(j)}_1$ is independent from $k^{(i)}_1$. In all, the produced initial key $k^{(i)}_1$ is a new random value independent from all the appeared keys $k^{(1)}, \ldots, k^{(i-1)}$. Therefore, we start from $G^*_{\alpha,0}$ and make $\ell_j$ hops that are similar to those described in the non-abort case. These hops yield $G^*_{\alpha,\ell_j}$ at the end, with $\lvert \Pr[E^{(\ast)}_{\alpha,\ell_j}] - \Pr[E^{(\ast)}_{0,0}] \rvert \leq \ell \varepsilon + \frac{\ell+1}{2^n}$. 

Finally, in the game $G^*_{q_e,\ell_{q_e}}$, it can be seen that every intermediate value is derived via a call to a corresponding random permutation. And it can be seen

$$\Pr[G^*_{q_e,\ell_{q_e}} \text{ aborts}] \leq \sum_{i=1}^{q_e} \Pr[\text{CollD}_i \lor \text{CollW}_i \lor \text{CollH} \lor \text{PreimgH}] \leq \frac{(\ell+1)q_e}{2^n} + \varepsilon_{cr} + \varepsilon_{pr},$$

and

$$\Pr[E^{(\ast)}_{q_e,\ell_{q_e}} \mid G^*_{q_e,\ell_{q_e}} \text{ does not abort}] - \Pr[E^{(\ast)}_{0,\ell_0}] \leq (2\ell + 4)q_e \varepsilon_E.$$ 

The latter follows from the fact that during the transitions, the number of keys used for calling E is at most $(2\ell + 4)q_e$. 

45
We then consider $\Pr[E_{q_e+1}] = \Pr[A^\perp \Rightarrow 1]$, and derive an upper bound for $\Pr[E_{q_e+1}] - Pr[E^*_1, t_q_e | G^*_q, t_q_e]$ does not abort]. First, conditioned on that CollD never happened for any $i$, for any two intermediate values $W(i)$ and $W(j)$ we have:

- if their nonce values are the same, i.e. $N_i \parallel A_i = N_j \parallel A_j$, then $W(i) \neq W(j)$. Therefore, the two further derived values $V(i) = P_i, t_{i+2}(W(i))$ and $V(j) = P_i, t_{i+2}(W(j))$ are two random and independent values;
- if $N_i \parallel A_i \neq N_j \parallel A_j$, then $W(i)$ and $W(j)$ are random and independent, since they are the outputs of two independent random permutations. This also means the further derived $V(i)$ and $V(j)$ are random and independent.

Thus the $q_e$ outputs $V^{(1)}, \ldots, V^{(q_e)}$ are random and independent. Yet, some of them may be the outputs of the same random permutation (in the extreme case, i.e. all the nonce values are identical, all of them are the outputs of the same random permutation), and will thus be distinct. By this, the statistical distance between $V^{(1)}, \ldots, V^{(q_e)}$ and uniform is at most $q_e^2 / 2^{n+1}$.

Following the same line, the further derived $q_e$ initial keys $k^{(1)}_1, \ldots, k^{(q_e)}_1$ for the one-time encryption are also uniform and independent. This indicates the $\sum_{i=1}^{q_e} \ell_i$ blocks in the subsequent $q_e$ key streams are given by $\sum_{i=1}^{q_e} \ell_i$ independent random permutations, i.e. $P_i, 1, \ldots, P_i, \ell_i$ for $i = 1, \ldots, q_e$, and thus uniform and independent. Thus for the $q_e$ ciphertexts $(c^{(1)}_1, \ldots, c^{(q_e)}_1), \ldots, (c^{(1)}_{q_e}, \ldots, c^{(q_e)}_{q_e})$ the distribution is also uniform.

Finally, we argue that during each encryption query $(N_i, A_i, M_i)$, when the computation proceeds to the tag generation phase $h_i \leftarrow H(1||R_i||V_i||c_i)$ and $T_i \leftarrow \pi(1||h_i)$, the random permutation query $\pi(1||h_i)$ is fresh, i.e. it never happened before, and no previous inverse query resulted in $h_i$ (in this case, the subsequently generated tag $T_i$ is not random either). To this end, we exclude two possibilities:

- If the entry $\pi(1||h_i)$ is non-fresh due to a previous inverse query $\pi^{-1}(T), then $h_i = \text{trunc}(\pi^{-1}(T))$ contradicts $\neg$PreimgH.
- Otherwise, the query $\Pi(1||Z_i)$ was necessarily made in an earlier encryption query, due to its “second” leak free call $E^*(1||\cdot)$. Assume that this encryption query is $(N_j, A_j, M_j)$. This means $H(1||R_i||V_i||c_i) = H(1||R_j||V_j||c_j)$. If $R_i \neq R_j$, or $c_i \neq c_j$, then this clearly contradicts $\neg$CollH. Otherwise, if $N_i \parallel A_i = N_j \parallel A_j$ then $H(0||N_i \parallel A_i) = R_i = R_j = H(0||N_j \parallel A_j)$ also contradicts $\neg$CollH. If $N_i \parallel A_i = N_j \parallel A_j$ and $c_i \neq c_j$ then $(N_i \parallel A_i || M_i) = (N_j \parallel A_j || M_j)$, and it contradicts the assumption that $A$ never repeats queries;

- there exists an earlier decryption query $(N_j, A_j, (c_j, T_j))$ such that $h_i = \text{trunc}(\pi^{-1}(T_j))$. However, this would contradict $\neg$PreimgH.

By these, the $q_e$ tags $T_1, \ldots, T_{q_e}$ are distinct and random values, and deviate from the at most $q_d$ tags specified in the decryption queries. Therefore, the statistical distance between the distribution of $T_1, \ldots, T_{q_e}$ and uniform is at most $(q_e + q_d)/2^{n+1}$.

Gathering the above, we have

$$\Pr[E_{q_e+1}] - Pr[E^*_1, t_q_e | G^*_q, t_q_e]$ does not abort] \leq \frac{(q_e + q_d)^2 + q_e^2}{2^{n+1}}.$$ 

To conclude, we summarize the gaps as follows:

(i) $|\Pr[E_0] - \Pr[E_{0,1} | G_{0,1}$ does not abort] | \leq \varepsilon_E$;
(ii) $\Pr[E_{0,1} | G_{0,1}$ does not abort] = $\Pr[E^*_1, t_q_e | G^*_q, t_q_e$ does not abort];
(iii) $\Pr[E^*_q, t_q_e | G^*_q, t_q_e$ does not abort] - $\Pr[E_{0,0} | G_{0,0}$ does not abort] is upper-bounded by $(2\ell + 4)q_e \varepsilon_E$;
(iv) $\Pr[E_{q_e+1}] - Pr[E^*_1, t_q_e | G^*_q, t_q_e$ does not abort] \leq \frac{(q_e + q_d)^2 + q_e^2}{2^{n+1}};
(v) $\Pr[G^*_q, t_q_e$ aborts] \leq \frac{(\ell+1)q_e}{2^{n+1}} + \varepsilon_{cr} + \varepsilon_{pr}$.

Thus in total, $|\Pr[E_0] - \Pr[E_{q_e+1}]|$ is bounded by

$$\varepsilon_{E} + \varepsilon_{cr} + \varepsilon_{pr} + (2\ell + 4)q_e \varepsilon_E + \frac{2(\ell + 1)q_e + (q_e + q_d)^2 + q_e^2}{2^{n+1}}$$

as claimed.
Empty Messages. can be handled by the proof in this subsection without specific treatment. In detail, for the empty messages appearing in encryption queries the case resembles subsection F.2. When empty messages appear in the encryption queries, it can handle as well: for example, in the non-reuse case, with $\ell_i = 0$ the calls to $E_{s_0}$ and $E_{s_1}$ would be rightfully replaced by random permutations.

G CCAmL2 security of AEDT

![Fig. 13: The AEDT AEAD scheme.]

We didn’t formally describe AEDT in the main body: this gap is filled in Fig. 13.

For its (m)CCAmL2 analysis, the leakage function $L = (L_{\text{enc}}, L_{\text{dec}})$ of AEDT is defined similarly to the one of FEMALE, i.e., including all the “bounded” leakages generated by the underlying actions.

**Theorem 11.** Let $H : \{0, 1\}^* \rightarrow B$ be a $(0, t', \varepsilon_{cr})$-collision resistant and $(q_d, t', \varepsilon_{pr})$-range-oriented preimage resistant hash function. And let $E : M \times M \rightarrow M$ be a $(2q_e + 2q_d + 2t', \varepsilon_E)$-SPRP with two implementations: a strongly protected instantiation $E'$ is leak free, and a plain implementation $E$ have leakage function $L_E$ that is $(q_s, t_S, q_t, t, \varepsilon_{(p,2)-rsm})$ $(p, 2)$-R-simulatable. Then the AEDT implementation with leakage function $L = (L_{\text{enc}}, L_{\text{dec}})$ defined before is $(q_e, q_d, p - 1, q_m, q_l, t, \varepsilon_{mCCAmL2})$ mCCAmL2-secure, where

$$\varepsilon_{mCCAmL2} \leq 2\varepsilon_E + \varepsilon_{cr} + \varepsilon_{pr} + \sum_{i=1}^{q_m} \varepsilon_{AEDT-cav}(\ell_i),$$

and $\varepsilon_{AEDT-cav}(\ell_i)$ is the upper bound on the eavesdropper advantage of $(q_t, t')$-bounded adversaries on AEDT with $\ell_i$ block messages:

$$\varepsilon_{AEDT-cav}(\ell_i) \leq 4\ell_i(\varepsilon_E + \varepsilon_{(2,2)-rsm}) + \ell_i \cdot \varepsilon_{s-block},$$

and $\ell_i$ is the number of blocks in the $i$-th challenge messages and $\varepsilon_{s-block}$ as is defined in equation (1). Here $t' = t + (q_e + q_d + q_m)(t_S + t_{-pass}), t_{-pass}$ is the maximum running time of AEDT upon a single (encryption or decryption) query, and $t_S$ is the time needed for randomly sampling a value from $M$.

The proof follows the same line as sections F.1 and F.2. We first formally describe the two algorithms RESM and iESM, which denote the processes of using Real/Idealized AEDT to encrypt a Single Message respectively. They are described in Fig. 14 and 15 respectively.

**Lemma 7 (Indistinguishability of LRESM and LIESM).** Let $E : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}^n$ be a $(2, t, \varepsilon_E)$-PRP, whose implementation has a leakage function $L_E$ having $(q_S, t_S, q_t, t, \varepsilon_{(2,2)-rsm})$ $(2, 2)$-R-simulatable leakages, and let $S^L$ be an appropriate $(q_S, t_S)$-bounded leakage simulator. Then, for every $\ell$-block message $m$, every $p_A, p_B$, and every $(q_l - 2q_e - q'^* + t - 2t_r - t^*)$-bounded distinguisher $D^L$, the following holds:

$$|\Pr[D^L(m, LRESM_{k_0}(m)) \Rightarrow 1] - \Pr[D^L(m, LIESM_{k_0}(m)) \Rightarrow 1]| \leq (\varepsilon_E + \varepsilon_{(2,2)-rsm}).$$
We define \( E \) and \( q \). Here \( q \).

For every pair of \( \ell \)-block message relies on the security of \( \text{IESm} \).

**Lemma 8.** For every pair of \( \ell \)-block messages \( m^0 \) and \( m^1 \) and \( (q_r,t_r) \)-bounded adversary \( A^L \), there exists an \( (q_r + 2q_r,t + 2t_r) \)-bounded adversary \( A^U \) such that

\[
| \Pr[ A^L(\text{IESm}_{k_0}(m_0)) \Rightarrow 1] - \Pr[ A^L(\text{IESm}_{k_0}(m_1)) \Rightarrow 1] |
\]

\[
\leq \sum_{i=1}^{\ell} | \Pr[ A^U(\text{LEnc}_{k_{i-1}}^+(m_i^0)) \Rightarrow 1] - \Pr[ A^U(\text{LEnc}_{k_{i-1}}^+(m_i^1)) \Rightarrow 1] |
\]

where \( k_0, \ldots, k_{\ell-1} \) are chosen uniformly at random, the \( p_B \) value used in the scheme \( \text{LEnc}_{k_{i-1}}^+(m_i^1) \) is \( p_B \), and \( m_i^0 \) and \( m_i^1 \) are the \( i \)-th block of \( m^0 \) and \( m^1 \), respectively. Here \( q_r = \ell(2q_S + 1) \) and \( t_r = \ell(2t_S + 2t_S + t_\Theta) \), where \( t_E, t_S \), and \( t_\Theta \) are as assumed in Lemma 3.
Proof. We start by building a sequence of $\ell + 1$ messages $m_{h,0}, \ldots, m_{h,\ell}$ starting from $m^0$ and modifying its blocks one by one until obtaining $m^1$. That is, $m_{h,i} := m^0_0 || \ldots || m^0_{i-1} || m^1_{i-1} || \ldots || m^1_{\ell}$. Following the same line as the proof of Lemma 3, it can be shown

$$| \Pr[\mathcal{A}(LIESM_k_0(m_{h,i-1})) \Rightarrow 1] - \Pr[\mathcal{A}(LIESM_k_0(m_{h,i})) \Rightarrow 1] | \leq | \Pr[\mathcal{A}(LISEnc_{k_\ell-i}^+ (m^0_{\ell-i+1})) \Rightarrow 1] - \Pr[\mathcal{A}(LISEnc_{k_\ell-i}^+ (m^1_{\ell-i+1})) \Rightarrow 1] |.$$ 

Taking a summation yields the main claim.

Lemmas 2, 7, and 8 cinch the eavesdropper security on AEDT.

Lemma 9. Let $E : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ be a $(2,t,\varepsilon_E)$-PRP, whose implementation has a leakage function $L_E$ having $(q_S, t_S, q_t, t, \varepsilon(2,2), \text{rsim})$ $(2,2)$-R-simulatable leakages, and let $S^L$ be an appropriate $(q_S, t_S)$-bounded leakage simulator. Then, for every pair of $\ell$-block messages $m^0$ and $m^1$ and $(q_r-2q_r-q^*, t-2t_r-t^*)$-bounded adversary $\mathcal{A}^L$, it holds

$$| \Pr[\mathcal{A}(LRESM_k_0(m^0)) \Rightarrow 1] - \Pr[\mathcal{A}(LRESM_k_0(m^1)) \Rightarrow 1] | \leq 4\ell(\varepsilon_E + \varepsilon(2,2), \text{rsim}) + \ell \cdot \varepsilon_{s\text{-block}},$$

where $q_r, t_r$ are as defined in Lemma 7, and $q^*, t^*$ are as defined in Lemma 2.

Proof. In a similar vein to Lemma 5.

And then Theorem 11 (the CCAmL2 security of AEDT) could be derived. Briefly speaking, we note that the designs of AEDT and FEMALE share the following in common:

- First, their authentication components are the same;
- Second, their invalid decryption queries only leak some outputs of $(E_k^*)^{-1}$, which are indistinguishable from meaningless random values.

Therefore, the proof just follows the same line as that of Theorem 4.

On Empty Message. At the end of this section, we again consider the case the scheme is used on an empty message $M = \bot$. In that case, it can be seen AEDT collapses to a hash-then-MAC function, i.e., $C = E_k^*(1||H(1||R))$, where $R = H(0||N||A)$. Clearly, this provides authentication for $A$.

On the other hand, although it appears wasting to hash the input twice, it may not be wise to drop the second call for this special case: on one hand, this would require the scheme to incorporate a sub-mechanism to handle this case, resulting in an increased complexity; on the other hand, the existing CiML2 security proof for AEDT may not hold for the “reduced” authentication function.