Leakage-Resilient Authenticated Encryption with Misuse in the Leveled Leakage Setting: Definitions, Separation Results, and Constructions

Chun Guo, Olivier Pereira, Thomas Peters, François-Xavier Standaert.
UCLouvain, ICTEAM – Crypto Group, B-1348 Louvain-la-Neuve, Belgium.

Abstract. We propose definitions and constructions of authenticated encryption (AE) schemes that offer security guarantees even in the presence of side-channel leakages and nonce misuse. This is part of an important ongoing effort to make AE as robust as possible, while preserving appealing efficiency properties. In order to achieve this efficiency, we aim at modes of operation 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 computation can be performed by cheap and weakly protected blocks.
Our 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 FEMALE, a new mode of operation that satisfies our security definitions and supports efficient leveled implementations, and AEDT, another efficient mode of operation that offers the strongest form of misuse resistance that can be achieved in the presence of leakages, while not being fully misuse resistant in the black-box setting.

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.

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 a 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 authenticated encryption [35], which goes even one step further, by requiring ciphertexts to be indistinguishable of random strings. An encryption scheme satisfying this notion of misuse resistance is extremely appealing, as it is going as far as possible in protecting users from their own mistakes or from devices offering poor sources of randomness.

A second line of efforts then came, aiming at also protecting from weaknesses that implementers could introduce in the scheme, by creating observable behaviors that are not part of the AE scheme specification.

One type of implementation weakness comes from the decryption of invalid ciphertexts [10,3,21,28,4]. 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 [8,5,9]. 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 implementations and measurement devices that are at hand. The leakage of all secrets makes cryptography impossible. But the full protection against leakages at the implementation level brings us back to a situation in which we must completely trust the implementer to not make any mistake and, even in that case, this will come at high cost in terms of extra

computation time, energy, or circuit area: strong protections increase the usual "code size × cycle count" metric by 2 or 3 orders of magnitude compared to a non-protected implementation [6]. As a result, various types of limitations on leakages have been proposed – see Fuller and Hamlin [17] for a review and a comparison. For instance, leakages may be required to be limited in size, either in total life-time of the device [2], or at each round of computation [14], or required to be simulatable [37].

Leakages can also be considered to be uniform in the device – we call this a uniform implementation (or an implementation in the uniform leakage setting), or different levels of security can be required in different parts of the device: some components must be well protected, while a weak protection would be sufficient for others – we call this a leveled implementation (or an implementation in the leveled leakage setting) [32]. For example, in the context of AE, Barwell et al. [5] design leakage-resilient AE modes that require a uniformly protected implementation. This leads to strong feasibility results, but also to a high expected implementation cost: they suggest implementing the PRF that processes each message block using a pairing-based leakage-resilient PRF. In the area of leveled implementations, Berti et al. [8,9] propose AE modes in which each message encryption requires running for a small constant number of times a well-protected PRF implementation (which could be the pairing-based PRF mentioned above, or a block cipher implementation taking advantage of countermeasures like masking and shuffling [27]), then running a weakly protected block cipher implementation a number of times that is linear in the number of message blocks.

When messages are long, the cost of an encryption is expected to be close to the one of an unprotected implementation. As a result, leveled implementations offer an appealing setting for the design of leakage-resilient schemes, as they offer a good matching with practical engineering constraints, both in terms of efficiency and design strategy, while taking as much as possible from the benefits that a specific mode of operation can offer compared to strategies in which black-box security aspects and implementation issues are considered separately.

1.2 Contributions

Our main contributions are as follows. We define:

1. Security for authenticated encryption in the presence of nonce misuse and leakages (AEML).
2. New block cipher modes of operation for AEML that satisfy our definitions under relatively mild assumptions.

Security definition. Our definition of AEML security is a combination of three requirements: 1. The AE scheme must be misuse resistant (MR) in the black-box setting (without leakage), in the usual sense of Rogaway and Shrimpton [35]. 2. The AE scheme must offer CIML2 security, which is a form of ciphertext integrity in the presence of nonce misuse and leakage introduced by Berti et al. [8,9]. 3. The AE scheme must offer CCAML2 security, which is an extension of CCA security in the presence of misuse and leakage that we propose here.

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 leakages, we unfortunately cannot just extend the RS notion in any natural way, as it would lead to an overly strong requirement. The RS definition of MR requires that ciphertexts look random as soon as they are produced from a fresh pair of nonce and message. This, in effect, requires that the message is processed in full before the first bit of ciphertext is released. However, as observed by Berti et al. [8,9], this is extremely/overly demanding as soon as leakages happen. For example, in an implementation all the message blocks cannot be expected to be processed in parallel (especially for large messages), which means that leakages will happen during the processing of some of the blocks before others. If encryption queries are made in such a way that those first processed blocks are identical (but other blocks differ), the leakages about these first blocks will be identical, which is something that an adversary will be able to observe, unless strong randomized leakage protections are implemented for each processed block. This prompts the exploration of the best possible form of MR that could be achieved by a leveled implementation.

To this purpose, we turn back to the older formulation of authenticated encryption as a combination of IND-CPA and INT-CTXT security [7], a combination that is known to imply CCA security in the black-box setting. An important benefit of this style of formulation is that it removes the requirement of pseudorandom ciphertexts introduced by RS, a requirement that is known to be problematic since the early works on leakage-resilience by Micali and Reyzin on pseudorandom generators [29].

The extension of INT-CTXT to the setting of misuse and leakages has been recently proposed as the CIML2 notion [9]. (The same notion excluding decryption leakages has been proposed beforehand in [8]).

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 LMCPA notion [37]. However, this leads to a notion that is much weaker than one would expect, and arguably too weak. In particular, the implication towards a leaking variant of CCA security does not hold anymore. In a similar way, starting from the alternate definition of AE as INT-PTXT and IND-CCA security [24], we show that leaking variants of INT-PTXT and IND-CCA security do not imply CIML2 security.

So, the usual implications that hold in the black-box setting do not hold anymore in the leveled leaking setting and, as a result, we introduce the notion of CCAML2 security, which we propose to use in combination with CIML2 (and MR) to define AEML security. CCAML2 security offers strong confidentiality properties, even in the presence of leakages: all encryption and decryption oracles leak, including during the "challenge" query that the adversary can use to win the game. It also aims at the strongest possible form of misuse resistance in the presence of leakage: as long as the nonce used in the test query is fresh, confidentiality must hold. We will elaborate on CCAML2 and its motivations in Section 3.

New modes of operation. Our second contribution is the definition of two new modes of operation. The first mode, FEMALE (for Feedback-based Encryption with Misuse, Authentication and LEakage), offers AEML security. The second mode, AEDT, is not AEML secure, but still CIML2 and CCAML2 (hence, it is not MR). This scheme is proposed for situations in which full misuse is not a concern and side-channel attackers are present anyway (i.e., the mode will never be used without leakage), making the black-box MR property irrelevant.

FEMALE is a two-pass encryption scheme offering traditional AE security which is compatible with a leveled implementation: independently of the length of the message and associated data, a strongly protected (leak-free) block cipher (BC) must be called only twice. Apart from that, a weakly protected BC must be called $4\ell + 5$ times for a message of ℓ blocks. The MR security of FEMALE holds in the standard model. The CIML2 security holds in the unbounded leakage model [9], which lets weakly protected component 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 [37] (which we discuss later in this section). More precisely, 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. The single block security cannot be derived from the simulatable leakage assumption: the leakage of a few bits of the internal state of a device may not contradict simulatability, while the leakage of one single bit of a secret message is enough to break confidentiality [32]. Still, 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. The latter motivates our approach of reducing the security of our schemes for multiple long messages to the security of a single message block, which 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 this single message block remains an open problem.

AEDT is a simple variant of the EDT scheme of Berti et al. [9] which was designed to support a leveled implementation. It is already known to be CIML2 secure, and we show that AEDT also offers CCAML2 security. Giving up on black-box MR security leads to efficiency improvements: AEDT still requires two calls of leak-free BC, but only $2\ell - 1$ calls of a weakly protected BC, and 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, as in traditional MR AE modes, 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 online 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.

1.3 Related Works

Recently, Barwell et al. [5] introduced notions of leakage-resilient authenticated encryption, and proposed modes of operation satisfying their definitions. We will refer to this work as BMOS, by the initial of its authors.

Our work includes two main differences with BMOS. First, we aim at stronger security requirements. In particular, the BMOS definition captures leakage-resilient AE security as follows (we just focus on encryption queries for simplicity): they first follow the RS 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, no distinction can be made based on the content of leakages. On the positive side, they can show that their definition is compatible with strong composition results. However, according to this definition, an implementation that leaks plaintexts in full during encryption may be considered as a perfectly secure implementation of an AE scheme: this is a direct consequence of the separation between the black-box real or random oracles and the leaking oracle. On the contrary, our definition would hold such an implementation as completely insecure, which we think to be a reasonable conclusion.

The second difference is on the modes of operation, which target a uniform implementation: all components in the BMOS constructions are required to offer the same strong level of protection against side-channel attacks. This is expected to lead to a considerably more expensive implementation and, in particular, the proposed implementation strategy requires the evaluation of pairings for each message block.

Berti et al. [8,9] also investigate AE in the presence of nonce misuse and leakages, but focus on authentication, proposing the CIML2 definition that we are using here. Regarding confidentiality, they only us a leakage-resilient version of CPA security (without considering the case with both leakages and misuse).

Our AEML definition shares some features with the notion of misuse resilience introduced by Ashur, Dunkelman and Luykx [4]. Misuse resilience is introduced as a weakening of misuse resistance, that tolerates sub-optimal security degradation when nonces are reused, but requires that this security degradation does not "spill" on other uses of the AE scheme with fresh nonces. In a similar spirit, our CCAML2 security definition focuses on the confidentiality of messages that are encrypted with a fresh pair of nonce and associated data. 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 precisely our motivation for departing from traditional misuse resistance, as it is unlikely that full misuse resistance can hold in the presence of side-channel leakages on challenge ciphertexts [8,9].

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 leakage 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. The latter allows us to design schemes that combine strong (long-term) security guarantees against key leakages, and the best possible guarantees against plaintext leakages obtained "on-the-fly".

1.4 Leakage Assumptions

Fuller and Hamlin recently surveyed leakage assumptions [17] (excluding idealized assumptions such as used in [38]). Among the weaker assumptions that have been used for proving symmetric constructions, one can choose between indistinguishability-based notions, typically used by Dziembowski and Pietrzak [14] and follow up works [33,12,15], and the simulatability assumption introduced in [37] that we also exploit here. As later argued in [26], none of them is perfect: the first one are hard to validate empirically while it remains an open problem to design efficient instances of simulators. In this respect, we insist that the part of our results related to the definition of AEML is independent of this choice of assumptions (it only relates to our choice to work in a leveled setting and to provide the leakage of the challenge ciphertexts). Only the proofs of our two constructions require the simulatability assumption and we leave as an open problem to analyze (tweaks of) these constructions using an indistinguishability-based notion.

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 $S^* = \cup_{i=1}^{\infty} S^i$ so that $s \in S^*$ if and only if it exists a non negative integer ℓ such that $s \in S^\ell$. This notation is convenient for instance for variable block-length (authenticated) encryption with message space $\mathcal{M}^*$, where messages of $\mathcal{M}$ 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 ω oracles, can make at most q_i queries to its i-th oracle, and can perform computation bounded by running time t. For algorithms that have no oracle to access, we simply call them t-bounded. In this paper, we use subscripts to make a clear distinction between the number of queries to different oracles: the number of queries to the (authenticated) encryption oracle, decryption oracle, and leakage oracle L are denoted by q_e, q_d, and q_l respectively. E.g., 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 Ancryption with Associated Data (AEAD) scheme respectively, and makes q_l additional queries to the leakage oracle L.

Leaking algorithm. Let Algo be an efficient algorithm. A leaking version of Algo is denoted LAlgo. It runs both Algo and a *leakage function* $\mathsf{L}_{\mathsf{algo}}$ which captures the additional information given by an implementation of Algo during its execution. LAlgo simply returns the outputs of both Algo and $\mathsf{L}_{\mathsf{algo}}$ which all take the same input.

2.2 Security Definitions

Standard definitions of collision-resistant hash functions, range-oriented preimage-resistance, pseudorandom permutations, strong tweakable pseudorandom permutations and nonce-based AEAD are given in Appendix A. We next recall the definition of Misuse-Resistance (MR) that we use, as formalized in [35].

Definition 1 (MR). *A nonce-based authenticated encryption scheme with associated data* $\mathsf{AEAD} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* $(q_e, q_d, t, \varepsilon)$ *misuse resistant for a security parameter* n *if, for all* (q_e, q_d, t)*-bounded adversaries* $\mathcal{A}$*,*

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

where $\$(N, A, M)$ *outputs and associates a fresh random ciphertext* $C \leftarrow \mathcal{C}$ *to fresh inputs*[1]*, and the associated* C *otherwise, and* $\perp(N, A, C)$ *outputs* $\perp$ *except if* C *was associated to* (N, A, M) *for some message* M*, in which case it returns* M*.*

Ciphertext integrity with misuse and leakage (in encryption), denoted CIML, was introduced in [8] as a strong integrity guarantee of authenticated encryption in the leaking setting. This notion was enhanced in [9] into the so-called $\mathsf{CIML2}$ notion to further capture leakage in decryption. To formalize the leakage depending on an implementation, AEAD is associated to both an encryption leakage function $\mathsf{L}_{\mathsf{enc}}$ and a decryption leakage function $\mathsf{L}_{\mathsf{dec}}$, leading to the following definition:

Definition 2 ($\mathsf{CIML2}$). *An authenticated encryption* $\mathsf{AEAD} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with leakage function pair* $\mathsf{L} = (\mathsf{L}_{\mathsf{enc}}, \mathsf{L}_{\mathsf{dec}})$ *provides* $(q_e, q_d, q_l, t, \varepsilon)$*-ciphertext integrity with (nonce) misuse and leakage (on encryption and decryption) for security parameter* n *if, for all* (q_e, q_d, q_l, t)*-bounded adversaries* $\mathcal{A}^{\mathsf{L}}$*, we have:*

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

where the security game $\mathsf{PrivK}^{\mathsf{CIML2}}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$ *is defined in Table 1 (left part) when* $\mathcal{A}^{\mathsf{L}}$ *makes at most* q_e *leaking encryption queries,* q_d *leaking decryption queries and* q_l *leakage evaluation queries.*[2]

To ease comparison, Table 1 also contains a new $\mathsf{PrivK}^{\mathsf{PIML2}}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}(1^n)$ game (right part) which is the plaintext-integrity counterpart of the ciphertext-integrity game (left part). This will lead to the notion of plaintext integrity with misuse and leakage (in encryption and decryption) given in Section 3.1.

[1] This slight extension allows AEAD with polynomial size $\mathcal{C}$ to be misuse resistant.

[2] The meaning of $\mathcal{A}^{\mathsf{L}}$ is given in Subsection 2.3, here it indicates that the adversary may query L with at most q_l *chosen keys*. This is not reminded in the security games.

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

Table 1. The CIML2 and PIML2 security games.

2.3 The Leveled Leakage Setting

The leveled leakage implementation setting considers different types of implementations for the components (e.g., block ciphers) used in a mode of operation.

On the one hand, it relies on a limited number of highly protected or, in effect, leak-free components. As previously discussed [32], these components are expected to be protected using strong protections against side-channel attacks: they could be a block cipher masked at very high security orders [18,23] for instance.

On the other hand, the rest of the components will continuously leak a certain amount of information to the adversary every time they are used. Conceptually, we would like to avoid any unnecessary restrictions on the leakage function L. In particular, it is an open problem to determine the complexity of such a function, and whether it can be computed efficiently. So, in order to abstract the complexity of the leakage function, we model it as a *leaking oracle* L, and we allow the adversary $\mathcal{A}^{\mathsf{L}}$ to make q_l leakage queries to L. Note that these q_l leakage queries are quite different from the leaking encryption and decryption queries to the AEAD schemes: when querying L, $\mathcal{A}$ must select the key himself for the computations, while the encryption and decryption oracles provide $\mathcal{A}$ with leakages about a key k that $\mathcal{A}$ is not expected to know. Those L queries actually correspond to an offline training phase that $\mathcal{A}$ can perform as part of his attack of the circuit, a practice that is common in (profiled) side-channel attacks [11].

Our analyses put two types of requirements on the leaking parts of our constructions, depending on whether we are aiming at confidentiality or integrity properties. With respect to integrity guarantees, we prove security in the unbounded leakage model [8]: it assumes that, when queried, oracles return, in addition to the usual output values, a function L^* yielding all keys and random coins generated or used during the computation of the oracle's answer. Regarding confidentiality guarantees, we reduce the security of multiple blocks to the security of a single message block under an assumption of leakage simulatability that will be introduced in Section 4.3.

3 Authenticated Encryption against Misuse and Leakage

To define the security of authenticated encryption in the presence of misuse and leakage 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 leakage. This motivates our definition to be at least as secure as this strongest combination.

3.1 Misuse and Leakage Variants of Black-box Notions

We adapt the IND-CPA and the IND-CCA confidentiality notions as well as the INT-PTXT and INT-CTXT integrity notions of nonce-based authenticated encryption in the setting of nonce-misuse and leakage.

Confidentiality. Contrary to existing confidentiality notions in a leaking setting [5], our definition includes leakage during encryption and decryption even on the challenge ciphertext. We first focus on security against chosen-ciphertext attack with misuse and leakage, denoted CCAML2. Then, we derive the weaker notion of security against chosen-plaintext attack with misuse and leakage, denoted CPAML2, by removing some of the adversary's abilities. The natural extensions of these definitions supporting multiple challenges are deferred to Appendix B where we show the equivalence between both chosen-ciphertext notions.

Chosen-ciphertext security with misuse and leakage. To capture the CCAML2 security we define the game $\mathsf{PrivK}^{\mathsf{CCAML2},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$ detailed in Fig 1. This game takes as parameters an adversary $\mathcal{A}$, a nonce-based authenticated encryption AEAD and a (possibly probabilistic) leakage function pair $\mathsf{L} = (\mathsf{L}_{\mathsf{enc}}, \mathsf{L}_{\mathsf{dec}})$ resulting from the implementation of the scheme. During $\mathsf{PrivK}^{\mathsf{CCAML2},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$, the adversary $\mathcal{A}$ has to guess the bit b of which depend the challenge ciphertext C^b and the encryption leakage $\mathsf{leak}^b_{\mathsf{enc}}$, computes as a leaking encryption of (N_{ch}, A_{ch}, M^b), where the nonce N_{ch}, the associated data A_{ch} and the messages M^0, M^1 are chosen by $\mathcal{A}$ under certain conditions. All along this game $\mathcal{A}$ is also granted unbounded and adaptive access to three types of oracles: LEnc, a leaking encryption oracle; LDec, a leaking decryption oracle; and $\mathsf{L}_{\mathsf{decch}}$, a challenge decryption leakage oracle.

Overall, this definition follows the general pattern of CCA security.

In terms of misuse resistance, it lets the adversary pick the nonces. However, and contrary to black-box security definitions, the adversary is forbidden to reuse an old nonce N_{ch} in its challenge query. This is consistent with the reasons outlined in the introduction and for example the fact that it is unrealistic to expect that leakages will only depend on a full message, while any concrete implementation will not be able to process all the blocks of a long message in parallel. The freshness requirement on N_{ch} makes it possible that, when the encryption algorithm starts processing messages, it is already in an internal state that differs from the one of old queries, hence preventing the direct identification of the leakages. The latter is the only (seemingly unavoidable) restriction that we impose in our definition.

In terms of leakage-resilience, we observe that all encryption and decryption oracles leak (hence the "2" of CCAML2 for the two leaking oracles), including during the challenge query. We are going one step further with the $\mathsf{L}_{\mathsf{decch}}$ oracle, which offers the leakage 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 (imagine that this challenge ciphertext is a firmware update intended to a device controlled by $\mathcal{A}$). We let the adversary query the $\mathsf{L}_{\mathsf{decch}}$ oracle multiple times, as leakages can be non-deterministic (e.g., contain measurement noise), and $\mathcal{A}$ may get some benefits from the observation of leakages from multiple decryptions of the same plaintext.

$\mathsf{PrivK}^{\mathsf{CCAML2},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}(1^n)$ is the output of the following experiment:

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

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

(1) $\mathsf{LEnc}(N, A, M)$ computes $C \leftarrow \mathsf{Enc}_k(N, A, M)$ and $\mathsf{leak}_{\mathsf{e}} \leftarrow \mathsf{L}_{\mathsf{enc}}(k, N, A, M)$, updates $\mathcal{E} \leftarrow \mathcal{E} \cup \{N\}$ and finally returns $(C, \mathsf{leak}_{\mathsf{e}})$;

(2) $\mathsf{LDec}(N, A, C)$ computes $M \leftarrow \mathsf{Dec}_k(N, A, C)$ and $\mathsf{leak}_{\mathsf{d}} \leftarrow \mathsf{L}_{\mathsf{dec}}(k, N, A, C)$ and returns $(M, \mathsf{leak}_{\mathsf{d}})$; (We stress that $M = \perp$ may occur.)

Challenge query: on a single occasion $\mathcal{A}^{\mathsf{L}}$ 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 $C^b \leftarrow \mathsf{LEnc}_k(N_{ch}, A_{ch}, M^b)$ and $\mathsf{leak}^b_{\mathsf{e}} \leftarrow \mathsf{L}_{\mathsf{enc}}(k, N_{ch}, A_{ch}, M^b)$ and return $(C^b, \mathsf{leak}^b_{\mathsf{e}})$;

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

(3) $\mathsf{LEnc}(N, A, M)$ returns $\perp$ if $N = N_{ch}$, otherwise computes $C \leftarrow \mathsf{Enc}_k(N, A, M)$ and $\mathsf{leak}_{\mathsf{e}} \leftarrow \mathsf{L}_{\mathsf{enc}}(k, N, A, M)$, and finally returns $(C, \mathsf{leak}_{\mathsf{e}})$;

(4) $\mathsf{LDec}(N, A, C)$ returns $\perp$ if $(N, A, C) = (N_{ch}, A_{ch}, C^b)$, otherwise computes $M \leftarrow \mathsf{Dec}_k(N, A, C)$ and $\mathsf{leak}_{\mathsf{d}} \leftarrow \mathsf{L}_{\mathsf{dec}}(k, N, A, C)$ and returns $(M, \mathsf{leak}_{\mathsf{d}})$;

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

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

Fig. 1: The $\mathsf{PrivK}^{\mathsf{CCAML2},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}(1^n)$ game.

Definition 3 (CCAML2). *A nonce-based authenticated encryption with associated data* $\mathsf{AEAD} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with leakage function pair* $\mathsf{L} = (\mathsf{L}_{\mathsf{enc}}, \mathsf{L}_{\mathsf{dec}})$ *is* $(q_e, q_d, q_c, q_l, t, \varepsilon)$-CCAML2 *secure for a security parameter* n *if, for every* (q_e, q_d, q_c, q_l, t)*-bounded adversary* $\mathcal{A}^{\mathsf{L}}$*, we have*

$$\left|\Pr\left[\mathsf{PrivK}^{\mathsf{CCAML2},0}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}(1^n) \Rightarrow 1\right] - \Pr\left[\mathsf{PrivK}^{\mathsf{CCAML2},1}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}(1^n) \Rightarrow 1\right]\right| \leq \varepsilon,$$

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

We refer to CCAML2* as the security notion defined as CCAML2 except that we drop the challenge decryption leakage oracle $\mathsf{L}_{\mathsf{decch}}$ from the game in Fig. 1.

Chosen-plaintext security with misuse and leakage. Similarly, CPAML2 is defined from a game $\mathsf{PrivK}^{\mathsf{CPAML2},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$. This game is exactly as $\mathsf{PrivK}^{\mathsf{CCAML2},b}_{\mathcal{A}^{\mathsf{L}},\mathsf{AEAD}}$ except that we remove $\mathcal{A}$'s access to the leaking decryption oracle LDec in Figure 1 (Items 2,4). On the other hand, $\mathcal{A}$ is still able to get challenge decryption leakage $\mathsf{leak}^b_{\mathsf{d}}$.

Definition 4 (CPAML2). *A nonce-based authenticated encryption with associated data* $\mathsf{AEAD} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with leakage function pair* $\mathsf{L} = (\mathsf{L}_{\mathsf{enc}}, \mathsf{L}_{\mathsf{dec}})$ *is* $(q_e, q_c, q_l, t, \varepsilon)$-CPAML2 *secure for a security parameter* n *if, for every* (q_e, q_c, q_l, t)*-bounded adversary* $\mathcal{A}$*, we have*

$$\left|\Pr\left[\mathsf{PrivK}^{\mathsf{CPAML2},0}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}(1^n) \Rightarrow 1\right] - \Pr\left[\mathsf{PrivK}^{\mathsf{CPAML2},1}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}(1^n) \Rightarrow 1\right]\right| \leq \varepsilon,$$

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

Integrity. We adopt the natural and strong extensions of INT-CTXT and INT-PTXT to misuse and leakage in encryption and decryption: CIML2 and PIML2.

Ciphertext integrity with misuse and leakage. The CIML2 notion is defined in Definition 2 based on the security game $\mathsf{PrivK}^{\mathsf{CIML2}}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$ of Table 1 (left part). We refer to the [8,9] for the rationale supporting this definition.

Plaintext integrity with misuse and leakage. By analogy, we define the PIML2 security except that not only M is authenticated but also the associated data A.

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

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

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

3.2 Main Security Definition

By definition, we require that a secure authenticated encryption with misuse and leakage satisfies the strongest achievable guarantee presented in the paper: an AEML scheme is expected to offer CCAML2 and CIML2 security, together with being a MR AEAD scheme in the black-box setting. This definition departs from the traditional ones in the black-box setting, based on the combination of CPA and INT-CTXT security [7] 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.

Definition 6 (AEML). *An authenticated encryption scheme with security against nonce misuse and leakages,* AEML, *is an authenticated encryption scheme* $\mathsf{AEAD} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with a leakage function pair* $\mathsf{L} = (\mathsf{L}_{\mathsf{enc}}, \mathsf{L}_{\mathsf{dec}})$ *satisfying the following assertions: (i)* AEAD *is misuse resistant; (ii)* AEAD *is* CCAML2 *secure with leakage* L*; (iii)* AEAD *is* CIML2 *secure with leakage* L*.*

In all of our schemes to come, we will actually show that CIML2 security holds in the unbounded leakage setting, in which we have a function $\mathsf{L}^*_{\mathsf{enc}}$ (resp., $\mathsf{L}^*_{\mathsf{dec}}$) that reveals all the *ephemeral* values computed during encryption (resp., decryption). This is a very liberal leakage assumption which just assumes that there exists an efficient function ϕ such that $\mathsf{L} = \phi \circ \mathsf{L}^*$ even if this function is hard to describe. Therefore, all the information given by L is given by L^*.

3.3 Separation Results

We now explain why the strong security notions of MR, CCAML2 and CIML2 are needed to define AEML, while one could be tempted to make a definition as a combination of weaker notions, which may be easier to prove. Unfortunately, there is no such equivalence as 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.

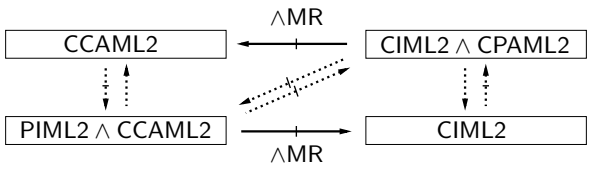

Fig. 2: Relations among security notions in the leveled leakage setting. An arrow denotes an implication while a barred arrow denotes a separation. Dotted (barred) arrows are trivial implied by other relations.

In contrast to the black-box setting, these relations also show that cryptographers do not have to choose between the different ways to achieve AEML since, for instance, $\mathsf{CCAML2} \wedge \mathsf{PIML2} \not\Rightarrow \mathsf{CPAML2} \wedge \mathsf{CIML2}$.

MR ∧ CPAML2 ∧ CIML2 ⇏ CCAML2*. It is not surprising that MR does not imply CCAML2 since the leakage function L is simply absent from black-box notion. Contrarily, L appears both in CPAML2 and CIML2. This claim thus says that leakage in decryption may not alter integrity but may alter confidentiality. To reflect this intrinsic separation of the leakage setting we show that the implication does not even hold for $\mathsf{CCAML2}^*$ where the challenge decryption leakage oracle $\mathsf{L}_{\mathsf{decch}}$ is unavailable. While the latter leakage is well motivated in the context of side-channel attacks [9], is is also quite specific to such attacks. So ignoring it in the separation makes our result stronger and more general.

Theorem 1. *Assuming that there exists an authenticated encryption which satisfies misuse resistance, chosen-plaintext security with misuse and leakage, and ciphertext integrity with misuse and unbounded leakage, then there exists an authenticated encryption with the same security properties but which fails to achieve the chosen-ciphertext security with misuse and leakage even without challenge decryption leakage.*

The proof is is Appendix C. The leakages "harmful" to $\mathsf{CCAML2}^*$ only appear during invalid decryption queries, rather than the challenges in CPAML2 game. So the implication does not hold either when starting from the *multiple challenge* variant of CPAML2 (defined in Appendix B) and the proof idea remains the same.

MR ∧ CCAML2 ∧ PIML2 ⇏ CIML2. As for the previous assertion, being MR does not say anything about leakage, so not being CIML2 is obviously compatible. The most interesting part comes from CCAML2 and PIML2 which include leakage. This claim exploits the fact that leakage on repeated queries may degrade ciphertext integrity but neither confidentiality nor plaintext integrity.

Theorem 2. *Assuming that there exists an authenticated encryption which satisfies misuse resistance, chosen-ciphertext security with misuse and leakage, and plaintext integrity with misuse and unbounded leakage, then*

there exists an authenticated encryption which satisfies misuse resistance, chosen-ciphertext security with misuse and leakage, and plaintext integrity with misuse and unbounded leakage but which fails to achieve the ciphertext integrity with misuse and leakage, so possibly even not with unbounded leakage.

The proof is in Appendix D. It proceeds by building a $\neg$CIML2 scheme AEAD$'$ from an MR $\wedge$ CCAML2 $\wedge$ PIML2 scheme AEAD. An interesting feature is that this counterexample AEAD$'$ preserves the tidyness of AEAD. This deviates from Bellare and Namprempre's well-known approach for establishing INT-PTXT $\not\Rightarrow$ INT-CTXT, which did utilize non-tidy counterexamples [7]. It is possible in our case due to the presence of leakages.

Further Security Definitions. In appendix B we extend our confidentiality notion to support multiple challenge ciphertexts. In both the single and multiple challenge settings, we also define several other variants of CCAML2 and CIML2 security and explore their relations in Appendix E. We finally discuss relations with the Eavesdropper Security with Decryption Leakage introduced in [9] (which is implied by AEML) in Appendix F.

4 FEMALE: an AEML-Secure Realization

We design FEMALE, an AEML scheme that can be implemented in the leveled leakage setting. FEMALE is named after *Feedback-based Encryption with Misuse, Authentication and LEakage* as it starts processing the message blocks using the (ciphertext) feedback mode of operations during their encryption. It makes only two calls to a leak-free tweakable block cipher. Relying on the leveled leakage setting, all the other computations can be less protected and then much more efficient. The intuition behind the design of FEMALE is given in Section 4.1 followed by the full specifications of the scheme in Section 4.2. To prove the CCAML2 security, we rely on an extension of the leakage simulatability model [37]. This model and related discussions are provided in Section 4.3, and the security analysis is in Section 4.4.

4.1 Intuition

FEMALE is an improvement of the following blueprint named 3LF. The 3LF blueprint has three parts which all use the secret key: (i) *Ephemeral IV generation:* on input (N, A, M), derives a pseudorandom value V; (ii) *IV-based encryption:* on input (V, M) computes an encryption c of M with initialized vector V; (iii) *Authentication:* compute a tag T on (N, A, V, c). The ciphertext is $C = (V, c, T)$. On (N, A, C), the 3LF decryption first checks (iii) before extracting M from (ii) in the obvious way. We observe that, on the one hand, even if V looks independent of M it guarantees that nonce misuse will not affect the pseudorandomness of ciphertexts in the MR game as long as the messages change, on the other hand, the CIML2 security can be achieved exactly as [9] from a strong (tweakable) PRP at the end: the validity of T can be verified by evaluating its pre-image, which avoids leaking extra valid tags. Satisfying the CCAML2 notion deeply depends on the leakage filtering from a 3LF implementation of these three parts.

While some 3LF implementations could be AEML-secure if all the computations involving the secret key were leak-free, leading to a total of at least three leak-free calls (hence the name 3LF), it is not straightforward to identify sufficient conditions in the leveled leakage setting to come up with a generic composition with a single secret key. Anyway, FEMALE meets the desired AEML security with the following advantages. First, it processes the key only twice and then only requires two leak-free calls. Second, it processes the message blocks only once. From a leakage point of view, we stress that the less the sensitive data are manipulated, the less information would leak about them. To achieve this, FEMALE slightly modifies the 3LF blueprint into: (i) *Ephemeral key-IV generation:* on input (N, A, M), derives a pair (U, V) where the session key U is pseudorandom on (N, A) and V is pseudorandom on (N, A, M) so that this process also pre-encrypts M into d; (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 (N, A, V, c), computes a pseudorandom tag T. The ciphertext $C = (V, c, T)$ does not include d. To encrypt d into c at step (ii) FEMALE no more uses the (long-term) secret key. To decrypt (N, A, C), FEMALE first checks (iii) before deriving U from (N, A) as in step (i) in order to reverse the process at step (ii) to extract d. Eventually, (N, A, d) allows retrieving M at step (i).

4.2 Specification

Given (N, A, M) with an ℓ-block message M, the *ephemeral key-IV generation* first computes the hash $R = \mathsf{H}(N, A)$ and sequentially derives a random key stream $(s_0, s_1, \ldots, s_\ell, s_{\ell+1}, w, U)$. Then it pre-encrypts $M\|0$ as $d\|W$ in a CFB mode except that each block are processed with different keys, from s_1 to $s_{\ell+1}$. As long as W remains secret this value can be seen as a hash of (N, A, M) so that it is safe to set $V = \mathsf{E}_w(W)$. Without leakage, V is a pseudorandom IV uses to encrypt d in the next step and before authenticate all the computations. From that point, (black-box) misuse resistance easily follows. With leakage, as long as (N, A) is fresh the challenge ciphertext will get the desired indistinguishability notion thanks to our careful re-keying schedule. The scheme is depicted in Fig. 3 and the full description is in Fig. 4.

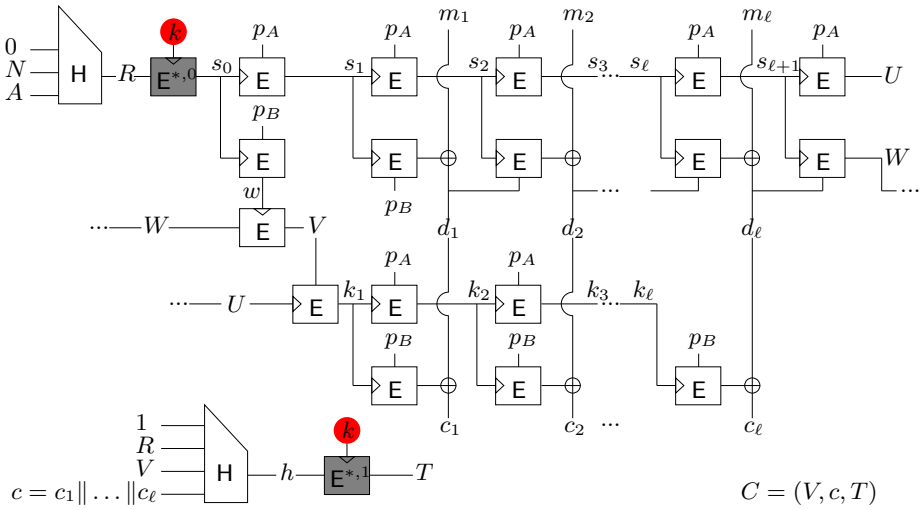

Fig. 3: (Top) The initialization and encryption parts of FEMALE, with the notations to be used in its specification and security analysis. (Bottom) The authentication part of FEMALE. The final output is $C = (V, c, T)$.

Description of FEMALE:

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

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

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

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

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

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

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

Fig. 4: The FEMALE AEAD scheme.

4.3 Recyclable Leakage Simulatability

The leakage-resilience of FEMALE relies on the assumption that leakages satisfy (p, q)-recyclable-simulatability defined below, and based on the (p, q)-rsim-game in Table 2. This assumption is an extension of the q-

simulatability notion [37]: (p, q)-recyclable-simulatability is simply q-simulatability, in which each of the q leakages can be obtained p times.

Game (p,q)-$\mathsf{rsim}(\mathcal{A}, \mathsf{E}, \mathsf{L}, \mathcal{S}, b)$.		
The challenger selects two random keys $k, k^* \xleftarrow{\$} \mathcal{K}$. The output of the game is a bit b' computed by $\mathcal{A}^{\mathsf{L}}$ based on the challenger responses to a total of at most q adversarial queries of the following type, each repeated at most p times:		
Query	Response if $b = 0$	Response if $b = 1$
$\mathsf{Enc}(x)$	$\mathsf{E}_k(x), \mathsf{L}(k, x)$	$\mathsf{E}_k(x), \mathcal{S}^{\mathsf{L}}(k^*, x, \mathsf{E}_k(x))$
and one query of the following type, repeated at most p times:		
Query	Response if $b = 0$	Response if $b = 1$
$\mathsf{Gen}(k_{pre}, x)$	$\mathcal{S}^{\mathsf{L}}(k_{pre}, x, k)$	$\mathcal{S}^{\mathsf{L}}(k_{pre}, x, k^*)$

Table 2. The (p, q)-rsim-game.

Definition 7 ((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 $(q_{\mathcal{S}}, t_{\mathcal{S}}, q_l, t, \varepsilon_{(p,q)\text{-}rsim})$ (p, q)-recyclable-simulatability, if there exists a $(q_{\mathcal{S}}, t_{\mathcal{S}})$-bounded simulator $\mathcal{S}^{\mathsf{L}}$ such that, for every (q_l, t)-bounded adversary $\mathcal{A}^{\mathsf{L}}$ (making at most q_l queries to L and running in time t), we have*

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

Throughout the remaining, we would simply call such leakages *R-simulatable*.

The Necessity of the Recyclable Assumption completely stems from the fact that our CCAML2 game offers the challenge decryption leakage oracle $\mathsf{L}_{\mathsf{decch}}$, which in cooperation with the challenge encryption leakage may result in the same set of leakage traces being generated *more than once*. For example, consider using FEMALE to encrypt a single block challenge m^b, and assume that the corresponding intermediate values are s_0, s_1, s_2, y_1, U, V, k_1, and z_1. Then the calls $\mathsf{E}_{s_0}(p_A)$, $\mathsf{E}_{s_1}(p_A)$, $\mathsf{E}_{s_1}(p_B)$, $\mathsf{E}_{s_2}(p_A)$, $\mathsf{E}_U(V)$, and $\mathsf{E}_{k_1}(p_B)$ are made during both the encryption and the decryption. Therefore, by querying $\mathsf{L}_{\mathsf{decch}}$, $\mathcal{A}$ could obtain the corresponding leakage traces many times. Consequently, merely assuming 2-simulatability would not suffice. Concretely, recyclable simulatability is admittedly a stronger assumption than q-simulatability, since the typical q one considers in leakage-resilient constructions is 2 and the p values allowed by decryption leakages are polynomial. In this respect, the following remarks are important:

First, in case only a weakened CCAML2* security is required, in which only challenge encryption leakages are generated, the original q-simulatability is sufficient. Overall, what our proofs demonstrate is that security with only challenge encryption leakages for multiple 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 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.

Second, as usual in discussions of confidentiality where challenges with leakages are made available to the adversary, our expectation is not that the adversarial advantages will be negligible as in the classical (black box) setting. By contrast, we aim for (i) the ability to discriminate constructions where the confidentiality is trivially minimum (e.g., the plaintext is leaked in full) and constructions where some leakage is available but still bounded in some sense, and (ii) definitions that allows the specification of useful guidelines for the design of leveled implementations. The latter is exactly where the introduction of recyclable simulatability comes in handy: we argue that for implementers, the design of R-simulatable gadgets will fall between the design of a leak-free component and the design of a q-simulatable one as follows. As already mentioned, leak-free components will typically be implemented with very high-order masking to resist DPA (and thus be very expensive).[3] On the other hand, q-simulatability is expected to be achievable with weakly protected implementations and essentially requires resistance against SPA with noisy measurements [37,32].[4] R-simulatability would rather require resistance against SPA with noise-free measurements and could for example be efficiently obtained with very low latency hardware (e.g., a parallel and unrolled AES implementation in one or two clock cycles [25], limiting the number of noise-free samples that the adversary can collect).

[3] Informally, DPA is a side-channel attack taking advantage of the leakage of multiple (different) inputs.
[4] Informally, SPAs are side-channel attacks taking advantage of the leakage of a single input.

4.4 Security Analysis

We focus on CCAML2 security first. To make it formal, we define the leakage function $\mathsf{L} = (\mathsf{L}_{\mathsf{enc}}, \mathsf{L}_{\mathsf{dec}})$ of FEMALE as:

- $\mathsf{L}_{\mathsf{enc}}$ consists of the follows that are generated during the encryption:
  - the leakages $\mathsf{L}_{\mathsf{E}}(s, x)$ generated by all the internal calls to $\mathsf{E}_s(x)$, and
  - the leakages $\mathsf{L}_{\oplus}(a, b)$ 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).
- $\mathsf{L}_{\mathsf{dec}}$ 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. As mentioned in introduction, if a single XOR of the message leaks enough to win the CCAML2 game, then nothing can be done, and the natural conclusion would be that an implementation offering stronger protections is needed. Otherwise, using this block in FEMALE offers the best possible confidentiality guarantees.

Description of LRSE scheme: (tool for the proof and for establishing confidentiality conclusion)
RSGen(1^n) picks $k_{ch} \xleftarrow{\$} \{0,1\}^n$, $\mathcal{M}, \mathcal{C} = \{0,1\}^n$ $(p_A, p_B \in \{0,1\}^n)$
RSEnc$_{k_{ch}}(m)$ returns (k_{up}, c), where $c = y_{ch} \oplus m$, $y_{ch} = \mathsf{E}_{k_{ch}}(p_B)$, and $k_{up} = \mathsf{E}_{k_{ch}}(p_A)$. (The term "up" is short for "update".)
RSDec$_{k_{ch}}(c)$ proceeds in the natural way.

The leakage $\mathsf{L}_{\mathsf{RSE}} = (\mathsf{L}_{\mathsf{rsenc}}, \mathsf{L}_{\mathsf{rsdec}}, k_{pre})$ resulting from the LRSE implementation is defined as $\mathsf{L}_{\mathsf{rsenc}}(k_{ch}, m) = (\mathsf{L}_{\mathsf{E}}(k_{ch}, p_A), \mathsf{L}_{\mathsf{E}}(k_{ch}, p_B), \mathsf{L}_{\oplus}(y_{ch}, m), \mathcal{S}^{\mathsf{L}}(k_{pre}, p_A, k_{ch}))$, $\mathsf{L}_{\mathsf{rsdec}}(k_{ch}, c) = (\mathsf{L}_{\mathsf{E}}(k_{ch}, p_A), \mathsf{L}_{\mathsf{E}}(k_{ch}, p_B), \mathsf{L}_{\oplus}(y_{ch}, c), \mathcal{S}^{\mathsf{L}}(k_{pre}, p_A, k_{ch}))$ for a fixed random $k_{pre} \xleftarrow{\$} \{0,1\}^n$. As usual we denote $\mathsf{LRSEnc}_{k_{ch}}(m) = (\mathsf{RSEnc}_{k_{ch}}(m), \mathsf{L}_{\mathsf{rsenc}}(k_{ch}, m))$.

Fig. 5: Basic unit: the single-block encryption scheme LRSE.

In detail, following Pereira et al.'s approach [32], we consider a Leaking Real Single-block Encryption scheme LRSE defined in Fig. 5 as the basic unit of FEMALE. The LRSE scheme is introduced to determine the CCAML2 security bound of FEMALE. Due to the re-keying process of FEMALE we only need LRSE to be "secure enough" against eavesdropper adversaries in the presence of both encryption and decryption leakages (the last requirement being necessary to bound the information the adversary will get from the $\mathsf{L}_{\mathsf{decch}}$ oracle). Since for each generated key k_{ch} LRSE will be used to encrypt a single message m composed of a single block, we assume that given a security parameter n, LRSE is $(p, q_l, t, \varepsilon_{s\text{-}block})$ secure in the following sense: for any (q_l, t)-bounded eavesdropper adversary $\mathcal{A}^{\mathsf{L}_{\mathsf{RSE}}}$ choosing $m^0, m^1 \in \{0,1\}^n$, it holds

$$\left| \Pr[\mathcal{A}^{\mathsf{L}_{\mathsf{RSE}}}(\mathsf{LRSEnc}^{+}_{k_{ch}}(m^0)) \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{L}_{\mathsf{RSE}}}(\mathsf{LRSEnc}^{+}_{k_{ch}}(m^1)) \Rightarrow 1] \right| \leq \varepsilon_{s\text{-}block}, \tag{1}$$

where $\mathsf{LRSEnc}^{+}_{k_{ch}}(m^b) = (\mathsf{LRSEnc}_{k_{ch}}(m^b), [\mathsf{L}_{\mathsf{rsdec}}(k_{ch}, c^b)]^{p-1}, k_{pre})$ for $(c^b, k_{up}) = \mathsf{RSEnc}_{k_{ch}}(m^b)$. The reason why the adversary also gets the auxiliary outputs k_{pre} and k_{up} is for composability purpose which will be apparent in the proof.

Based on the assumption (1), the CCAML2 security bound could be established below. We remark that: (i) First, the arguments p_A, p_B can be chosen by the adversary $\mathcal{A}$, but they have to be distinct; (ii) Second, as discussed before, it might be the case that, as related to an indistinguishability notion with leakage, $\varepsilon_{s\text{-}block}$ is non negligible.

Theorem 3. *Let $H : \{0,1\}^{\kappa} \times \{0,1\}^* \to \{0,1\}^n$ be a $(0, t', \varepsilon_{cr})$-collision resistant and $(q_d, t', \varepsilon_{pr})$-range-oriented preimage resistant hash function, $E^* : \{0,1\}^n \times \{0,1\} \times \{0,1\}^n \to \{0,1\}^n$ be a $(2q_e + 2q_d + 2, t', \varepsilon_{E^*})$-strong tweakable pseudorandom permutation, and $E : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a $(2, t, \varepsilon_E)$-PRP, whose implementation leakage function L_E has $(q_{\mathcal{S}}, t_{\mathcal{S}}, q_l, t, \varepsilon_{(p,2)\text{-}rsim})$ $(p, 2)$-R-simulatable leakages. Then the* FEMALE *implementation with leakage function $\mathsf{L} = (\mathsf{L}_{\mathsf{enc}}, \mathsf{L}_{\mathsf{dec}})$ defined before is $(q_e, q_d, p-1, q_l, t, \varepsilon_{\mathsf{CCAML2}})$* CCAML2*-secure, where*

$$\varepsilon_{\mathsf{CCAML2}} \leq 2\varepsilon_{E^*} + \frac{q_e}{2^n} + \varepsilon_{cr} + \varepsilon_{pr} + \varepsilon_{\mathsf{FEMALE}\text{-}eav}(\ell),$$

ℓ is the number of blocks in the challenge message, and $\varepsilon_{\mathsf{FEMALE}\text{-}eav}(\ell)$ is the upper bound on the eavesdropper advantage of (q_l, t')-bounded adversaries on FEMALE *on messages with ℓ blocks—concretely,*

$$\varepsilon_{\mathsf{FEMALE}\text{-}eav}(\ell) \leq (6\ell+8)(\varepsilon_{E} + \varepsilon_{(p,2)\text{-}rsim}) + \ell \cdot \varepsilon_{s\text{-}block} + \frac{6\ell+4}{2^n}.$$

Here $t' = t + (q_e + q_d + 1)(t_{\$} + t_{1-pass})$, t_{1-pass} is the maximum running time of FEMALE *upon a single (encryption or decryption) query, and $t_{\$}$ is the time needed for randomly sampling a value from $\{0,1\}^n$.*

Proof. See Appendix G. The proof is quite lengthy, which is the reason to present in another section. □

On the other hand, the CIML2 and MR security of FEMALE are as follows.

Theorem 4. *Let $H : \{0,1\}^n \times \{0,1\}^* \to \{0,1\}^n$ be a $(0, t', \varepsilon_{cr})$-collision resistant and $(q_d+1, t', \varepsilon_{pr})$-range-oriented preimage resistant hash function. Let $E^* : \{0,1\}^\kappa \times \{0,1\} \times \{0,1\}^n \to \{0,1\}^n$ be a $(2q_e+2q_d+2, t', \varepsilon_{E^*})$-strong tweakable pseudorandom permutation. Then* FEMALE *provides $(q_e, q_d, t, \varepsilon_{ciml2})$-ciphertext integrity with coin misuse and unbounded leakage on encryption and decryption as long as $t \leq t' - (q_e + q_d + 1)t_{1-pass}$, where t_{1-pass} is the maximum running time of* FEMALE *upon a single (encryption or decryption) query, and*

$$\varepsilon_{ciml2} \leq \varepsilon_{E^*} + \varepsilon_{cr} + \varepsilon_{pr}.$$

Proof. See Appendix G.3. □

Theorem 5. *Let $H : \{0,1\}^n \times \{0,1\}^* \to \{0,1\}^n$ be a $(0, t', \varepsilon_{cr})$-collision resistant and $(q_d, t', \varepsilon_{pr})$-range-oriented preimage resistant hash function. Let $E^* : \{0,1\}^\kappa \times \{0,1\} \times \{0,1\}^n \to \{0,1\}^n$ be a $(2q_e + 2q_d, t', \varepsilon_{E^*})$-strong tweakable pseudorandom permutation, and $E : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a $(q_e, t', \varepsilon_{E})$-pseudorandom permutation. Then the* FEMALE *scheme is $(q_e, q_d, t, \varepsilon_{mr})$-MR as long as $t \leq t' - (q_e + q_d)(t_{1-pass} + (4\ell+4)t_{\$})$, where t_{1-pass} is the maximum running time of* FEMALE *upon a single (encryption or decryption) query, $t_{\$}$ is the time needed for randomly sampling a value from $\{0,1\}^n$, and*

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

Proof. See Appendix G.4.

5 AEDT

We finally refer to Appendix H for the description and analysis of the (more efficient but not MR) AEDT scheme.

Acknowledgments. Thomas Peters is a postdoctoral researcher and François-Xavier Standaert is a senior research associate of the Belgian Fund for Scientific Research (F.R.S.-FNRS). This work has been funded in parts by the European Union through the ERC project SWORD (724725), the INNOVIRIS projects SCAUT and C-Cure, and the European Union and Walloon Region FEDER USERMedia project 501907-379156.

References

1. Albrecht, M.R., Paterson, K.G., Watson, G.J.: Plaintext recovery attacks against SSH. In: S&P. pp. 16–26. IEEE Computer Society (2009), http://dx.doi.org/10.1109/SP.2009.5
2. Alwen, J., Dodis, Y., Naor, M., Segev, G., Walfish, S., Wichs, D.: Public-key encryption in the bounded-retrieval model. In: Advances in Cryptology – EUROCRYPT 2010. pp. 113–134. Springer (2010)
3. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., Yasuda, K.: How to securely release unverified plaintext in authenticated encryption. In: Sarkar and Iwata [36], pp. 105–125, http://dx.doi.org/10.1007/978-3-662-45611-8_6
4. Ashur, T., Dunkelman, O., Luykx, A.: Boosting Authenticated Encryption Robustness with Minimal Modifications. In: Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III. pp. 3–33 (2017), https://doi.org/10.1007/978-3-319-63697-9_1
5. Barwell, G., Martin, D.P., Oswald, E., Stam, M.: Authenticated Encryption in the Face of Protocol and Side Channel Leakage. In: ASIACRYPT. Lecture Notes in Computer Science, vol. 10624, pp. 693–723. Springer (2017), https://doi.org/10.1007/978-3-319-70694-8_24

6. Belaïd, S., Grosso, V., Standaert, F.: Masking and Leakage-Resilient Primitives: One, the Other(s) or Both? IACR Cryptology ePrint Archive 2014, 53 (2014), http://eprint.iacr.org/2014/053
7. Bellare, M., Namprempre, C.: Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. J. Cryptology 21(4), 469–491 (2008), https://doi.org/10.1007/s00145-008-9026-x
8. Berti, F., Koeune, F., Pereira, O., Peters, T., Standaert, F.: Ciphertext Integrity with Misuse and Leakage: Definition and Efficient Constructions with Symmetric Primitives. To appear in the proceedings of ASIACCS 2018
9. Berti, F., Pereira, O., Peters, T., Standaert, F.: On Leakage-Resilient Authenticated Encryption with Decryption Leakages. IACR Trans. Symmetric Cryptol. 2017(3), 271–293 (2017), https://doi.org/10.13154/tosc.v2017.i3.271-293
10. Boldyreva, A., Degabriele, J.P., Paterson, K.G., Stam, M.: On symmetric encryption with distinguishable decryption failures. In: FSE 2013. LNCS, vol. 8424, pp. 367–390. Springer (2013)
11. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Jr., B.S.K., Koç, Ç.K., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers. Lecture Notes in Computer Science, vol. 2523, pp. 13–28. Springer (2002), https://doi.org/10.1007/3-540-36400-5_3
12. Dodis, Y., Pietrzak, K.: Leakage-resilient pseudorandom functions and side-channel attacks on feistel networks. In: Rabin, T. (ed.) Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6223, pp. 21–40. Springer (2010), https://doi.org/10.1007/978-3-642-14623-7_2
13. Duong, T., Rizzo, J.: Cryptography in the web: The case of cryptographic design flaws in ASP.NET. In: S&P. pp. 481–489. IEEE Computer Society (2011), http://dx.doi.org/10.1109/SP.2011.42
14. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: FOCS. pp. 293–302 (2008), http://dx.doi.org/10.1109/FOCS.2008.56
15. Faust, S., Pietrzak, K., Schipper, J.: Practical leakage-resilient symmetric cryptography. In: Prouff and Schaumont [34], pp. 213–232, https://doi.org/10.1007/978-3-642-33027-8_13
16. Fleischmann, E., Forler, C., Lucks, S.: Mcoe: A family of almost foolproof on-line authenticated encryption schemes. In: Fast Software Encryption - FSE 2012. Lecture Notes in Computer Science, vol. 7549, pp. 196–215. Springer (2012)
17. Fuller, B., Hamlin, A.: Unifying leakage classes: Simulatable leakage and pseudoentropy. In: ICITS. pp. 69–86 (2015), http://dx.doi.org/10.1007/978-3-319-17470-9_5
18. Goudarzi, D., Rivain, M.: How fast can higher-order masking be in software? In: Coron, J., Nielsen, J.B. (eds.) Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10210, pp. 567–597 (2017), https://doi.org/10.1007/978-3-319-56620-7_20
19. Halderman, J.A., Schoen, S.D., Heninger, N., Clarkson, W., Paul, W., Calandrino, J.A., Feldman, A.J., Appelbaum, J., Felten, E.W.: Lest we remember: cold-boot attacks on encryption keys. Commun. ACM 52(5), 91–98 (2009), http://doi.acm.org/10.1145/1506409.1506429
20. Halevi, S., Lin, H.: After-the-fact leakage in public-key encryption. In: Ishai, Y. (ed.) Theory of Cryptography - 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6597, pp. 107–124. Springer (2011), https://doi.org/10.1007/978-3-642-19571-6_8
21. Hoang, V.T., Krovetz, T., Rogaway, P.: Robust authenticated-encryption AEZ and the problem that it solves. In: EUROCRYPT. LNCS, vol. 9056, pp. 15–44. Springer (2015)
22. Hoang, V.T., Reyhanitabar, R., Rogaway, P., Vizár, D.: Online authenticated-encryption and its nonce-reuse misuse-resistance. In: Advances in Cryptology - CRYPTO 2015. Lecture Notes in Computer Science, vol. 9215, pp. 493–517. Springer (2015)
23. Journault, A., Standaert, F.: Very high order masking: Efficient implementation and security evaluation. In: Fischer, W., Homma, N. (eds.) Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10529, pp. 623–643. Springer (2017), https://doi.org/10.1007/978-3-319-66787-4_30
24. Katz, J., Yung, M.: Unforgeable encryption and chosen ciphertext secure modes of operation. In: FSE. pp. 284–299 (2000), http://dx.doi.org/10.1007/3-540-44706-7_20
25. Kerckhof, S., Durvaux, F., Hocquet, C., Bol, D., Standaert, F.: Towards green cryptography: A comparison of lightweight ciphers from the energy viewpoint. In: Prouff and Schaumont [34], pp. 390–407, https://doi.org/10.1007/978-3-642-33027-8_23
26. Longo, J., Martin, D.P., Oswald, E., Page, D., Stam, M., Tunstall, M.: Simulatable Leakage: Analysis, Pitfalls, and New Constructions. In: Sarkar and Iwata [36], pp. 223–242, https://doi.org/10.1007/978-3-662-45611-8_12
27. Mangard, S., Oswald, E., Popp, T.: Power analysis attacks - revealing the secrets of smart cards. Springer (2007)
28. Martin, D.P., Oswald, E., Stam, M., Wójcik, M.: A Leakage Resilient MAC. In: Groth, J. (ed.) Cryptography and Coding - 15th IMA International Conference, IMACC 2015, Oxford, UK, December 15-17, 2015. Proceedings. Lecture Notes in Computer Science, vol. 9496, pp. 295–310. Springer (2015), https://doi.org/10.1007/978-3-319-27239-9_18
29. Micali, S., Reyzin, L.: Physically observable cryptography (extended abstract). In: TCC. pp. 278–296 (2004), http://dx.doi.org/10.1007/978-3-540-24638-1_16
30. Namprempre, C., Rogaway, P., Shrimpton, T.: Reconsidering Generic Composition. In: Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings. pp. 257–274 (2014), https://doi.org/10.1007/978-3-642-55220-5_15
31. Paterson, K.G., AlFardan, N.J.: Plaintext-recovery attacks against datagram TLS. In: NDSS (2012), http://www.internetsociety.org/plain-text-recovery-attacks-against-datagram-tls
32. Pereira, O., Standaert, F., Vivek, S.: Leakage-Resilient Authentication and Encryption from Symmetric Cryptographic Primitives. In: Ray, I., Li, N., Kruegel, C. (eds.) CCS 2015. pp. 96–108. ACM (2015), http://doi.acm.org/10.1145/2810103.2813626

33. Pietrzak, K.: A leakage-resilient mode of operation. In: Joux, A. (ed.) Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5479, pp. 462–482. Springer (2009), https://doi.org/10.1007/978-3-642-01001-9_27
34. Prouff, E., Schaumont, P. (eds.): Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings, Lecture Notes in Computer Science, vol. 7428. Springer (2012), https://doi.org/10.1007/978-3-642-33027-8
35. Rogaway, P., Shrimpton, T.: A Provable-Security Treatment of the Key-Wrap Problem. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 373–390. Springer (2006), https://doi.org/10.1007/11761679_23
36. Sarkar, P., Iwata, T. (eds.): ASIACRYPT, LNCS, vol. 8873. Springer (2014), http://dx.doi.org/10.1007/978-3-662-45611-8
37. Standaert, F., Pereira, O., Yu, Y.: Leakage-resilient symmetric cryptography under empirically verifiable assumptions. In: CRYPTO. pp. 335–352 (2013), http://dx.doi.org/10.1007/978-3-642-40041-4_19
38. Yu, Y., Standaert, F., Pereira, O., Yung, M.: Practical Leakage-Resilient Pseudorandom Generators. In: Al-Shaer, E., Keromytis, A.D., Shmatikov, V. (eds.) Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010. pp. 141–151. ACM (2010), http://doi.acm.org/10.1145/1866307.1866324

A Definitions of Primitives

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

Definition 8 (Collision-Resistant Hash Function). *A (t, ε_{cr})-collision resistant hash function $\mathsf{H} : \mathcal{S} \times \mathcal{M} \to \mathcal{B}$ for security parameter n is a function that is such that, for every t-bounded adversary $\mathcal{A}$, the probability that $\mathcal{A}(1^n, s)$ outputs a pair of distinct messages $(m_0, m_1) \in \mathcal{M}^2$ such that $\mathsf{H}_s(m_0) = \mathsf{H}_s(m_1)$ is bounded by ε_{cr}, where $s \xleftarrow{\$} \mathcal{S}$ is selected uniformly at random.*

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

Definition 9 (Range-Oriented Preimage-Resistant Hash Function). *A (t, ε_{pr})-range-oriented preimage resistant hash function $\mathsf{H} : \mathcal{S} \times \mathcal{M} \to \mathcal{B}$ for security parameter n is a function that is such that, for every adversary $\mathcal{A}$ running in time t, the probability that $\mathcal{A}(1^n, s, y)$ outputs a message $m \in \mathcal{M}$ such that $\mathsf{H}_s(m) = y$ is bounded by ε_{pr}, where $s \xleftarrow{\$} \mathcal{S}$, $y \xleftarrow{\$} \mathcal{B}$ 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 10 (Pseudorandom Permutation). *A function $\mathsf{E} : \mathcal{K} \times \mathcal{M} \to \mathcal{M}$ is a $(q, t, \varepsilon_{\mathsf{E}})$-pseudorandom permutation (PRP) for a security parameter n if, for all (q, t)-bounded adversaries $\mathcal{A}$, we have*

$$\left|\Pr\left[k \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\mathsf{E}_k}(1^n) \Rightarrow 1\right] - \Pr\left[P \xleftarrow{\$} \mathcal{P} : \mathcal{A}^{P}(1^n) \Rightarrow 1\right]\right| \leq \varepsilon_{\mathsf{F}},$$

where $\mathcal{P}$ denotes the set of all permutations on $\mathcal{M}$.

The tweakable pseudorandom permutation notion is as follows.

Definition 11 (Strong Tweakable Pseudorandom Permutation). *A function $\mathsf{E} : \mathcal{K} \times \mathcal{TW} \times \mathcal{M} \to \mathcal{M}$ is a $(q, t, \varepsilon_{\mathsf{E}})$-strong tweakable pseudorandom permutation (STPRP) for a security parameter n if, for all (q, t)-bounded adversaries $\mathcal{A}$, we have:*

$$\left|\Pr\left[k \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\mathsf{E}_k, \mathsf{E}_k^{-1}}(1^n) \Rightarrow 1\right] - \Pr\left[P \xleftarrow{\$} \mathcal{TP} : \mathcal{A}^{P, P^{-1}}(1^n) \Rightarrow 1\right]\right| \leq \varepsilon_{\mathsf{E}},$$

where $\mathcal{TP}$ denotes the set of all tweakable permutations on $\mathcal{M}$ and with tweak space $\mathcal{TW}$ so that for any tweakable permutation P, and for any tweak tw, $P^{tw} = P(tw, \cdot)$ and $P^{tw,-1} = P^{-1}(tw, \cdot)$ are the inverse of each other.

We will focus on authenticated encryption with the following formalism.

Definition 12 (Nonce-Based AEAD). *A* nonce-based authenticated encryption scheme with associated data *is a tuple* $\mathsf{AEAD} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *such that, for any security parameter* n*, and keys in* $\mathcal{K}$ *generated from* $\mathsf{Gen}(1^n)$*:*

- $\mathsf{Enc} : \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{M} \to \mathcal{C}$ *maps a key selected from* $\mathcal{K}$*, a nonce value from* $\mathcal{N}$*, some blocks of associated data selected from* $\mathcal{AD}$*, and a message from* $\mathcal{M}$ *to a ciphertext in* $\mathcal{C}$*.*
- $\mathsf{Dec} : \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{C} \to \mathcal{M} \cup \{\perp\}$ *maps a key from* $\mathcal{K}$*, a nonce from* $\mathcal{N}$*, some associated data from* $\mathcal{AD}$*, and a ciphertext from* $\mathcal{C}$ *to a message in* $\mathcal{M}$ *that is the decryption of that ciphertext, 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* $k \leftarrow \mathsf{Gen}(1^n)$*,* $\mathsf{Enc}_k(N, A, M) := \mathsf{Enc}(k, N, A, M)$ *and* $\mathsf{Dec}_k(N, A, M) := \mathsf{Dec}(k, N, A, M)$ *are deterministic functions whose implementations may be probabilistic.*

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

B Multi-Challenge Setting

We revisit the confidentiality against chosen-ciphertext and chosen-plaintext attacks with misuse and leakage in the multi-challenge setting. While we have the equivalence of the chosen-ciphertext notions in the single-challenge and the multi-challenge setting it is not the case with our chosen-plaintext notions.

B.1 Multi-challenge chosen-ciphertext case

A natural extension of the CCAML2 experiment of Figure 1 to the multi-challenge setting is given by the following mCCAML2 experiment.

$\mathsf{PrivK}^{\mathsf{mCCAML2},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}(1^n)$ is the output of the following experiment:

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

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

Leaking decryption queries: $\mathcal{A}^{\mathsf{L}}$ gets adaptive access to $\mathsf{LDec}(\cdot,\cdot,\cdot)$,
$\mathsf{LDec}(N, A, C)$ outputs $\perp$ if $(N, A, C) \in \mathcal{E}_{ch}$, else computes $M \leftarrow \mathsf{Dec}_k(N, A, C)$ and $\mathsf{leak}_{\mathsf{d}} \leftarrow \mathsf{L}_{\mathsf{dec}}(k, N, A, C)$ and returns $(M, \mathsf{leak}_{\mathsf{d}})$; (Where $M = \perp$ may occur.)

Challenge queries: on possibly many occasions $\mathcal{A}^{\mathsf{L}}$ submits $(N_{ch}, A_{ch}, M^0, M^1)$,
If M^0 and M^1 have different (block) length or $N_{ch} \in \mathcal{E}$ or $(N_{ch}, *, *) \in \mathcal{E}$, returns $\perp$; Else computes $C^b \leftarrow \mathsf{LEnc}_k(N_{ch}, A_{ch}, M^b)$ and $\mathsf{leak}^b_{\mathsf{e}} \leftarrow \mathsf{L}_{\mathsf{enc}}(k, N_{ch}, A_{ch}, M^b)$, updates $\mathcal{E}_{ch} \leftarrow \mathcal{E}_{ch} \cup \{(N_{ch}, A_{ch}, C^b)\}$ and finally returns $(C^b, \mathsf{leak}^b_{\mathsf{e}})$;

Decryption challenge leakage queries: $\mathcal{A}^{\mathsf{L}}$ gets adaptive access to $\mathsf{L}_{\mathsf{decch}}(\cdot)$,
$\mathsf{L}_{\mathsf{decch}}(i)$ takes the i-th challenge ciphertext $(N_{ch}, A_{ch}, C^b) \in \mathcal{E}_{ch}$ and outputs a leakage trace $\mathsf{leak}^b_{\mathsf{d}} \leftarrow \mathsf{L}_{\mathsf{dec}}(k, N_{ch}, A_{ch}, C^b)$;

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

Fig. 6: The $\mathsf{PrivK}^{\mathsf{mCCAML2},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}(1^n)$ game.

Definition 13 (mCCAML2). *A nonce-based authenticated encryption with associated data* $\mathsf{AEAD} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with leakage function pair* $\mathsf{L} = (\mathsf{L}_{\mathsf{enc}}, \mathsf{L}_{\mathsf{dec}})$ *is* $(q_e, q_d, q_c, q_m, q_l, t, \varepsilon)$*-*mCCAML2 *secure for a security parameter* n *if, for every* $(q_e, q_d, q_c, q_m, q_l, t)$*-bounded adversary* $\mathcal{A}^{\mathsf{L}}$*, we have*

$$\left|\Pr\left[\mathsf{PrivK}^{\mathsf{mCCAML2},0}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}(1^n) \Rightarrow 1\right] - \Pr\left[\mathsf{PrivK}^{\mathsf{mCCAML2},1}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}(1^n) \Rightarrow 1\right]\right| \leq \varepsilon,$$

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

Equivalent notions. If the adversary $\mathcal{A}$ in the multi-challenge experiment above is restricted to make at most $q_m \leq 1$ challenge query then the mCCAML2 security and the CCAML2 security collide. Therefore, mCCAML2 security trivially implies CCAML2 security. The next theorem states that the both notions are in fact equivalent.

Theorem 6. *The* CCAML2 *security is equivalent to the* mCCAML2 *security. Formally, for any* $(q_e, q_d, q_c, q_m, q_l, t)$*-bounded adversary* $\mathcal{A}^{\mathsf{L}}$ *against the* mCCAML2 *security of* AEAD *for a security parameter* n*, if* AEAD *is* $(q_e + q_m - 1, q_d + q_c, q_c, q_l, t, \varepsilon_{\mathsf{CCAML2}})$*-*CCAML2 *secure for a security parameter* n*,*

$$\left|\Pr\left[\mathsf{PrivK}^{\mathsf{mCCAML2},0}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}(1^n) \Rightarrow 1\right] - \Pr\left[\mathsf{PrivK}^{\mathsf{mCCAML2},1}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}(1^n) \Rightarrow 1\right]\right| \leq q_m \times \varepsilon_{\mathsf{CCAML2}}.$$

Proof. Given an adversary $\mathcal{A}^{\mathsf{L}}$ against the mCCAML2 security of AEAD making q_m leaking challenge queries, we build q_m adversaries $\mathcal{A}'_i$ against the CCAML2 security of AEAD, for $i = 1$ to q_m. To simulate the mCCAML2 game in front of $\mathcal{A}^{\mathsf{L}}$, each $\mathcal{A}'_i$ has to emulate the responses to the different types of queries in $\mathsf{PrivK}^{\mathsf{mCCAML2},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}(1^n)$. Therefore, $\mathcal{A}'_i$ proceeds as follows with $\mathcal{E}'_i = \emptyset$:

Leaking encryption queries: when $\mathcal{A}^{\mathsf{L}}$ queries $\mathsf{LEnc}(N, A, M)$,
If $(N, *, *) \in \mathcal{E}'_i$, $\mathcal{A}'_i$ returns $\perp$, else $\mathcal{A}'_i$ queries $\mathsf{LEnc}(N, A, M)$ and gets back either $\perp$ or $(C, \mathsf{leak_e})$ which it sends to $\mathcal{A}^{\mathsf{L}}$.

Leaking decryption queries: when $\mathcal{A}^{\mathsf{L}}$ queries $\mathsf{LDec}(N, A, C)$,
If $(N, A, C) \in \mathcal{E}'_i$, $\mathcal{A}'_i$ returns $\perp$, else $\mathcal{A}'_i$ queries $\mathsf{LDec}(N, A, C)$ and gets back either $\perp$ or $(M, \mathsf{leak_d})$, where $M = \perp$ may occur, and sends it to $\mathcal{A}^{\mathsf{L}}$.

Challenge queries: for $j = 1$ to q_m, $\mathcal{A}^{\mathsf{L}}$ queries $(N^j_{ch}, A^j_{ch}, M^0_j, M^1_j)$,
If $(N^j_{ch}, *, *) \in \mathcal{E}'_i$, $\mathcal{A}'_i$ returns $\perp$, else
- $j < i$, $\mathcal{A}'_i$ queries its leaking encryption oracle $\mathsf{LEnc}(N_{ch}, A_{ch}, M^1)$ and gets either $\perp$ or $(C^1_j, \mathsf{leak}^1_{\mathsf{e},j})$ which it sends to $\mathcal{A}^{\mathsf{L}}$. $\mathcal{A}'_i$ updates $\mathcal{E}'_i = \mathcal{E}'_i \cup (N^j_{ch}, A^j_{ch}, C^1_j)$;
- $j = i$, $\mathcal{A}'_i$ queries its leaking *challenge* oracle on $(N^i_{ch}, A^i_{ch}, M^0_i, M^1_i)$ and gets either $\perp$ or $(C^b, \mathsf{leak}^b_{\mathsf{e}})$ which it sends to $\mathcal{A}^{\mathsf{L}}$;
- $j > i$, $\mathcal{A}'_i$ queries its leaking encryption oracle $\mathsf{LEnc}(N_{ch}, A_{ch}, M^0)$ and gets either $\perp$ or $(C^0_j, \mathsf{leak}^0_{\mathsf{e},j})$ which it sends to $\mathcal{A}^{\mathsf{L}}$. $\mathcal{A}'_i$ updates $\mathcal{E}'_i = \mathcal{E}'_i \cup (N^j_{ch}, A^j_{ch}, C^0_j)$.

Decryption challenge leakage queries: when $\mathcal{A}^{\mathsf{L}}$ queries $\mathsf{L}_{\mathsf{decch}}(j)$, for $j = 1$ to q_m,
- $j \neq i$, if $\mathcal{A}'_i$ sent $\perp$ to the j-th challenge query made by $\mathcal{A}^{\mathsf{L}}$ it sends $\perp$, else it queries its leaking decryption oracle $(N^j_{ch}, A^j_{ch}, C^{[j<i]}_j)$ and gets $\mathsf{leak}^{[j<i]}_{\mathsf{d},j}$ which it sends to $\mathcal{A}^{\mathsf{L}}$;
- $j = i$, $\mathcal{A}'_i$ queries its oracle $\mathsf{L}_{\mathsf{decch}}$ and receives either $\perp$ or $\mathsf{leak}^b_{\mathsf{d}}$ which it sends to $\mathcal{A}^{\mathsf{L}}$.

Finalization: $\mathcal{A}'_i$ outputs whatever is the $\mathcal{A}^{\mathsf{L}}$'s output bit b'.

The list $\mathcal{E}'_i = \{(N^j_{ch}, A^j_{ch}, C^{[j<i]}_j)\}^{q_m}_{j=1, j\neq i}$ maintained by $\mathcal{A}'_i$ serves to identify forbidden query attempts made by $\mathcal{A}^{\mathsf{L}}$ that would not be deemed as such in the CCAML2 experiment played by $\mathcal{A}'_i$. 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 $\mathsf{L}_{\mathsf{decch}}$ from the game in Figure 6, we define the security notion of mCCAML2*. It is easy to see that CCAML2* security is equivalent to mCCAML2* security as well.

B.2 Multi-challenge chosen-plaintext case

A natural extension of the CPAML2 experiment to the multi-challenge setting is obtained by removing the leaking decryption oracle of the above mCCAML2 experiment. The resulting experiment is called the mCPAML2 experiment, where the 2 highlights the access to the decryption challenge leakage oracle.

Definition 14 (mCPAML2). *A nonce-based authenticated encryption with associated data* $\mathsf{AEAD} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with leakage function pair* $\mathsf{L} = (\mathsf{L}_{\mathsf{enc}}, \mathsf{L}_{\mathsf{dec}})$ *is* $(q_e, q_c, q_m, q_l, t, \varepsilon)$-mCPAML2 *secure for a security parameter* n *if, for every* (q_e, q_c, q_m, q_l, t)*-bounded adversary* $\mathcal{A}^{\mathsf{L}}$*, we have*

$$\left|\Pr\left[\mathsf{PrivK}^{\mathsf{mCPAML2},0}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}(1^n) \Rightarrow 1\right] - \Pr\left[\mathsf{PrivK}^{\mathsf{mCPAML2},1}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}(1^n) \Rightarrow 1\right]\right| \leq \varepsilon,$$

when the adversary $\mathcal{A}^{\mathsf{L}}$ *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 mCPAML2 security obviously implies CPAML2 security, the converse is false.

Theorem 7. *Assuming there exists a* CPAML2 *secure authenticated encryption then there exists* CPAML2 *secure authenticated encryption which is not* mCPAML2 *secure.*

Proof. Let $\mathsf{AEAD} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a CPAML2 secure authenticated encryption with leakage function pair $\mathsf{L} = (\mathsf{L}_{\mathsf{enc}}, \mathsf{L}_{\mathsf{dec}})$. Then, we build the following authenticated encryption $\mathsf{AEAD}' = (\mathsf{Gen}', \mathsf{Enc}, \mathsf{Dec})$ with leakage function pairs $\mathsf{L}' = (\mathsf{L}_{\mathsf{enc}}, \mathsf{L}'_{\mathsf{dec}})$, where N_0, N_1 below are the outputs of a publicly samplable distribution parametrized by n:

$\mathsf{Gen}'(1^n)$**:** runs $k \leftarrow \mathsf{Gen}(1^n)$, $k_0 \leftarrow \mathsf{Gen}(1^n)$ and computes $k_1 = k \oplus k_0$. The secret key is defined as (k, k_0, k_1).
$\mathsf{L}'_{\mathsf{dec}}((k, k_0, k_1), N, A, C)$ outputs $\mathsf{L}_{\mathsf{dec}}(k, N, A, C)$ 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 CPAML2 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 CPAML2 security still holds from AEAD because we can simulated the additional leakage by picking $k' \leftarrow \mathsf{Gen}(1^n)$ and set $k_0 = k'$ or $k_1 = k'$ on-the-fly when needed. However, an mCPAML2 adversary simply has to make two challenge queries involving N_0 and N_1 and then to call the decryption challenge leakage oracle $\mathsf{L}_{\mathsf{decch}}$ 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. □

By removing the $\mathsf{L}_{\mathsf{decch}}$ oracle as well we find back the equivalence of $\mathsf{CPAML} := \mathsf{CPAML2}^*$, for the single challenge notion, with $\mathsf{mCPAML} := \mathsf{mCPAML2}^*$, for the multi challenge notion.

C Proof of Theorem 1

Let $\mathsf{AEAD} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with leakage function $\mathsf{L} = (\mathsf{L}_{\mathsf{enc}}, \mathsf{L}_{\mathsf{dec}})$ be MR, CPAML2 with respect to L and CIML2 with respect to L^* as such authenticated encryption exists by assumption. Then we build $\mathsf{AEAD}' = (\mathsf{Gen}', \mathsf{Enc}, \mathsf{Dec})$ with leakage $\mathsf{L}' = (\mathsf{L}'_{\mathsf{enc}}, \mathsf{L}'_{\mathsf{dec}})$ such that, for a fixed message $M^\dagger \in \mathcal{M}$:

$\mathsf{Gen}'(1^n)$**:** returns $k \leftarrow \mathsf{Gen}(1^n)$ and $k' \leftarrow \mathsf{Gen}(1^n)$;
$\mathsf{L}'_{\mathsf{enc}}((k, k'), N, A, M)$**:** outputs $(\mathsf{leak}_{\mathsf{e}}, C', \mathsf{leak}_{\mathsf{e}'})$ where $\mathsf{leak}_{\mathsf{e}} = \mathsf{L}_{\mathsf{enc}}(k, N, A, M)$ (comes from the computation of $C \leftarrow \mathsf{Enc}_k(N, A, M)$), the ciphertext $C' = \mathsf{Enc}_{k'}(N, A, M)$ and consequently $\mathsf{leak}_{\mathsf{e}'} = \mathsf{L}_{\mathsf{enc}}(k', N, A, M)$;
$\mathsf{L}'_{\mathsf{dec}}((k, k'), N, A, C)$**:** outputs $\mathsf{leak}_{\mathsf{d}} = \mathsf{L}_{\mathsf{dec}}(k, N, A, C)$ if $M \neq \perp$ (which comes from the computation of $M \leftarrow \mathsf{Enc}_k(N, A, C)$) and outputs $(\mathsf{leak}_{\mathsf{d}}, C^\dagger, \mathsf{leak}^\dagger_{\mathsf{e}'})$ otherwise, where $\mathsf{leak}_{\mathsf{d}} = \mathsf{L}_{\mathsf{dec}}(k, N, A, C)$, $C^\dagger \leftarrow \mathsf{Enc}_{k'}(N, A, M^\dagger)$ and consequently also $\mathsf{leak}^\dagger_{\mathsf{e}'} = \mathsf{L}_{\mathsf{enc}}(k', N, A, M^\dagger)$.

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

CPAML2. In the $\mathsf{PrivK}^{\mathsf{CPAML2},b}_{\mathcal{A}',\mathsf{AEAD}',\mathsf{L}'}(1^n)$ game, the adversary $\mathcal{A}'$ does not have access to $\mathsf{L}'_{\mathsf{dec}}$ except from the challenge decryption leakage through $\mathsf{L}'_{\mathsf{decch}}$. But since the challenge ciphertext is valid, $\mathsf{L}'_{\mathsf{decch}} = \mathsf{L}_{\mathsf{decch}}$ which returns $\mathsf{L}_{\mathsf{dec}}(k, N_{\mathsf{ch}}, A_{\mathsf{ch}}, C^b)$. Consequently, an adversary $\mathcal{A}$ in $\mathsf{PrivK}^{\mathsf{CPAML2},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$ can easily simulate the view of $\mathcal{A}'$

simply by picking $k' \leftarrow \mathsf{Gen}(1^n)$ transmitting all the queries to its own oracles and just add the encryption leakage $(C', \mathsf{leak}_{\mathsf{e}'})$ if necessary.

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

$\neg\mathsf{CCAML2}^*$. We build a distinguisher $\mathcal{A}'$ against AEAD'. In the security game $\mathsf{PrivK}^{\mathsf{CCAML2},b}_{\mathcal{A}',\mathsf{AEAD}',\mathsf{L}'}(1^n)$, the adversary queries leaking decryption of $(N_{\mathsf{ch}}, A_{\mathsf{ch}}, C)$ for any chosen $N_{\mathsf{ch}}, A_{\mathsf{ch}}$ and C. If the ciphertext is valid, it receives some $M \neq \perp$ and it sets $(M^0, C^0) = (M, C)$. If not, it receives $(\perp, (\mathsf{leak}_{\mathsf{e}}, C^\dagger, \mathsf{leak}^\dagger_{\mathsf{e}'}))$ from $\mathsf{LDec}_{k,k'}(N_{\mathsf{ch}}, A_{\mathsf{ch}}, C)$ and sets $(M^0, C^0) = (M^\dagger, C^\dagger)$. In the challenge phase, $\mathcal{A}'$ sends $(N_{\mathsf{ch}}, A_{\mathsf{ch}}, M^0, M^1)$ for any distinct M^1 than M^0. Since the pair $(N_{\mathsf{ch}}, A_{\mathsf{ch}})$ has never been queried for (leaking) encryption, $\mathcal{A}'$ does not receives $\perp$. In the answer $\mathsf{LEnc}_k(N_{\mathsf{ch}}, A_{\mathsf{ch}}, M^b)$, $\mathcal{A}'$ gets C^b. If C^b equals the known C^0, $\mathcal{A}'$ outputs 0, otherwise it outputs 1. Obviously the distinction holds with probability 1.

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

D Proof of Theorem 2

Let $\mathsf{AEAD} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with leakage function $\mathsf{L} = (\mathsf{L}_{\mathsf{enc}}, \mathsf{L}_{\mathsf{dec}})$ be MR, $\mathsf{CCAML2}$ with respect to L and $\mathsf{PIML2}$ with respect to L^* as such authenticated encryption exists by assumption. Then we build $\mathsf{AEAD}' = (\mathsf{Gen}', \mathsf{Enc}, \mathsf{Dec})$ with leakage $\mathsf{L}' = (\mathsf{L}'_{\mathsf{enc}}, \mathsf{L}_{\mathsf{dec}})$ 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:

$\mathsf{Gen}'(1^n)$**:** generates $k \leftarrow \mathsf{Gen}(1^n)$. Then, it selects distinct nonces $N^\dagger, N^\circ \in \mathcal{N}$, distinct $A^\dagger, A^\circ \in \mathcal{AD}$ and distinct messages $M^\dagger, M^\circ \in \mathcal{M}$. It computes the ciphertext $C^\dagger \leftarrow \mathsf{Enc}_k(N^\dagger, A^\dagger, M^\dagger)$ and splits it into four random shares: it samples three $|C^\dagger|$-bit strings R_0, R_1, S_0 and sets $S_1 = C^\dagger \oplus R_0 \oplus R_1 \oplus S_0$. It outputs (k, sh) where $sh = (R_0, R_1, S_0, S_1)$.

$\mathsf{L}'_{\mathsf{enc}}((k, sh), N, A, M)$**:** outputs $\mathsf{leak}_{\mathsf{e}} = \mathsf{L}_{\mathsf{enc}}(k, N, A, M)$ (which comes from the computation of the ciphertext $C \leftarrow \mathsf{Enc}_k(N, A, C)$) as well as the additional value B but only in four cases:

- Case 1.1: $(N, A) = (N^\dagger, A^\circ), M \neq M^\circ$: $B = R_0$;
- Case 1.2: $(N, A) = (N^\dagger, A^\circ), M = M^\circ$: $B = R_1$;
- Case 2.1: $(N, A) = (N^\circ, A^\dagger), M \neq M^\dagger$: $B = S_0$;
- Case 2.2: $(N, A) = (N^\circ, A^\dagger), M = M^\dagger$: $B = S_1$.

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:

$\mathsf{CCAML2}$. We prove that if there is a $\mathsf{CCAML2}$ adversary $\mathcal{A}'$ against AEAD', then there is an adversary $\mathcal{A}$ which uses $\mathcal{A}'$ to break the $\mathsf{CCAML2}$ security of AEAD. In detail, once $\mathsf{PrivK}^{\mathsf{CCAML2},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}(1^n)$ is setup, $\mathcal{A}(1^n)$ publicly sample $N^\dagger, A^\dagger, M^\dagger, N^\circ, A^\circ, M^\circ$ and sends to $\mathcal{A}'$ whatever is specified by AEAD'. Even if the ciphertext $C^\dagger = \mathsf{Enc}_k(N^\dagger, A^\dagger, M^\dagger)$ has never been computed its length is fully determined by n and $(N^\dagger, A^\dagger, M^\dagger)$. Therefore, $\mathcal{A}$ picks random $|C^\dagger|$-bit strings R_1, S_0, S_1. Then it runs $\mathcal{A}'$ and simulates $\mathsf{PrivK}^{\mathsf{CCAML2},b}_{\mathcal{A}',\mathsf{AEAD}',\mathsf{L}'}$ using its own interaction $\mathsf{PrivK}^{\mathsf{CCAML2},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$. For each query from $\mathcal{A}'$, the actions of $\mathcal{A}$ 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) $\mathcal{A}$ queries a leaking encryption on $(N^\dagger, A^\dagger, M^\dagger)$ to its own oracle and gets back $C^\dagger$ and then computes $R_0 = C^\dagger \oplus R_1 \oplus S_0 \oplus S_1$;

(2) $\mathcal{A}$ queries its own leaking encryption oracle on (N, A, M) and gets back some $(C, \mathsf{leak}_{\mathsf{e}})$ or possibly $\perp$ (for forbidden queries);

(3) $\mathcal{A}$ checks whether it should append an additional leakage B to $(C, \mathsf{leak}_{\mathsf{e}})$ depending on (N, A, M) falls into one of the four cases described above.

$\mathcal{A}$ respectively returns $(C, \mathsf{leak}_{\mathsf{e}})$ or $\perp$ or even $(C, \mathsf{leak}_{\mathsf{e}}, B)$ to $\mathcal{A}'$ according to the above situations.

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

Leaking challenge query: on input $(N_{\mathsf{ch}}, A_{\mathsf{ch}}, M^0, M^1)$, $\mathcal{A}$ sends it to its leaking challenge oracle and gets the tuple $(C^b, \mathsf{leak}_{\mathsf{e}}^b)$ or possibly $\perp$. If not $\perp$,

(1) if $(N_{\mathsf{ch}}, A_{\mathsf{ch}}) = (N^\dagger, A^\circ)$, $\mathcal{A}$ returns $(C^b, \mathsf{leak}_{\mathsf{e}}^b, R_1)$. In other words, the additional leakage is always R_1 regardless of the messages M^0, M^1;

(2) if $(N_{\mathsf{ch}}, A_{\mathsf{ch}}) = (N^\circ, A^\dagger)$, $\mathcal{A}$ returns $(C^b, \mathsf{leak}_{\mathsf{e}}^b, S_0)$ regardless of M^0, M^1;

(3) else, $\mathcal{A}$ just returns $(C^b, \mathsf{leak}_{\mathsf{e}}^b)$.

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

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

Now we explain why $\mathcal{A}$ properly emulates the $\mathsf{CCAML2}$ game in front of $\mathcal{A}'$. First, as long as $C^\dagger$ is returned to $\mathcal{A}$ in a leaking encryption queries $\mathcal{A}'$ can not make a leaking challenge query with $N_{ch} = N^\dagger$ and then case (i) in the challenge phase will not occur since $\mathcal{A}$ will receive $\perp$ and will send it to $\mathcal{A}'$ 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° 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 $\mathsf{PrivK}^{\mathsf{CCAML2},b}_{\mathcal{A}',\mathsf{AEAD}',\mathsf{L}'}$. 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 $\mathcal{A}$ will receive $\perp$ and send it to $\mathcal{A}'$ if $\mathcal{A}'$ 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 $A_{ch} = A^\circ$ the additional leakage $B^b = R_1$ will be given in the encryption leakage of the challenge ciphertext. Here it is easy to see, since $\mathcal{A}'$ will receive a single share among those specified in Case 1.1 and Case 1.2 of $\mathsf{L}'_{\mathsf{enc}}$, that this distribution is exactly as the one expected from AEAD'. Four, if $N_{ch} = N^\circ$ appears for the first time in the challenge phase, a similar argument shows that $\mathcal{A}'$ will only get a single share among those specified in Case 1.1 and Case 1.2 of $\mathsf{L}'_{\mathsf{enc}}$ and only if $A_{ch} = A^\dagger$ independently of the choice M^0, M^1. Once again, this is exactly the right distribution where $B^b = S_0$ 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^b included) can be perfectly emulated by $\mathcal{A}$ so that we have

$$\Pr\left[\mathsf{PrivK}^{\mathsf{CCAML2},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}(1^n) \Rightarrow 1\right] = \Pr\left[\mathsf{PrivK}^{\mathsf{CCAML2},b}_{\mathcal{A}',\mathsf{AEAD}',\mathsf{L}'}(1^n) \Rightarrow 1\right].$$

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

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

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

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

$\neg\mathsf{CIML2}$. We mount a ciphertext forgery attack against AEAD'. Let $\mathcal{A}'$ be an adversary which given $N^\dagger, A^\dagger, M^\dagger, N^\circ, A^\circ, M^\circ$ sequentially queries a leaking encryption on $(N^\dagger, A^\circ, M^\dagger)$, $(N^\dagger, A^\circ, M^\circ)$, $(N^\circ, A^\dagger, M^\circ)$, and $(N^\circ, A^\dagger, M^\dagger)$

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, $\mathcal{A}'$ output $C^\dagger = R_0 \oplus R_1 \oplus S_0 \oplus S_1$ which is a valid encryption of $(N^\dagger, A^\dagger, M^\dagger)$ which has never been received as an answer to some leaking encryption query. □

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

E Completing the Definitions' ZOO

To give a full picture of the different security flavors of authenticated encryption with misuse and leakage, we list all the security definitions that can be derived from our confidentiality and integrity notions. The study of all their relations shows that, apart from the obvious implications between the different flavors of confidentiality (resp., integrity), all the notions are separated from each other.

In the remaining, we first concentrate on the single challenge notions in E.1 and E.2, then extend the discussion to the multi-challenge setting in E.3.

E.1 Security Definition List: Single Challenge Setting

The CCAML2 security game $\mathsf{PrivK}^{\mathsf{CCAML2},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$ is defined in Figure 1, Section 3.1. By dropping some accesses to the distinct oracles of this game we naturally derive other confidentiality notions. For instance this is how we defined CPAML2 by removing items (2) and (4) from the security game. By doing similar modifications we can find many different integrity notions from the CIML2 security game $\mathsf{PrivK}^{\mathsf{CIML2}}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$ defined in Table 1, Section 2.2.

Prefix-Suffix Definitions. In all the notions derived from CCAML2 and CIML2 we only focus on those capturing leakage. Therefore all the definitions below keep the "L" in their notation. This leads to considering 16 different notions denoted as "pre-suf" with prefix $\mathsf{pre} \in \{\mathsf{CCA}, \mathsf{CPA}, \mathsf{CI}, \mathsf{PI}\}$ and suffix $\mathsf{suf} \in \{\mathsf{ML2}, \mathsf{ML}, \mathsf{L2}, \mathsf{L}\}$. We recall that when there is no "M" in suf the security game is *nonce-respecting*, which only restricts *leaking encryption* queries.

Zoo of Confidentiality. For $\mathsf{pre} \in \{\mathsf{CCA}, \mathsf{CPA}\}$ we obtain the following 8 notions, by starting from CCAML2 and by removing one security layer at a time:

$$\mathsf{CCAML2} \rightarrow \mathsf{CCAML}, \mathsf{CCAL2}, \mathsf{CPAML2} \rightarrow \mathsf{CPAML}, \mathsf{CPAL2}, \mathsf{CCAL} \rightarrow \mathsf{CPAL}$$

Definition 15. *A nonce-based authenticated encryption with associated data* $\mathsf{AEAD} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with leakage* $\mathsf{L} = (\mathsf{L}_{\mathsf{enc}}, \mathsf{L}_{\mathsf{dec}})$ *is* $(q_{\mathsf{pre\text{-}suf}}, q_l, t, \varepsilon)$*-pre-suf secure for a security parameter* n *if, for every* $(q_{\mathsf{pre\text{-}suf}}, q_l, t)$*-bounded adversary* $\mathcal{A}^{\mathsf{L}}$*, we have* $\mathsf{pre} \in \{\mathsf{CCA}, \mathsf{CPA}\}$ *and*

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

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

(i) $\mathsf{PrivK}^{\mathsf{CCAML2},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$: $q_{\mathsf{CCAML2}} = (q_e, q_d, q_c)$, the CCAML2 game, see Figure 1.
(ii) $\mathsf{PrivK}^{\mathsf{CCAML},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$: $q_{\mathsf{CCAML}} = (q_e, q_d)$ and the CCAML security game "removes 2" from the CCAML2 game meaning that $\mathsf{L}_{\mathsf{dec}}$ is removed from all the oracles. In other words items (2),(4) become black-box and (5) disappears with $\mathsf{L}_{\mathsf{decch}}$.
(iii) $\mathsf{PrivK}^{\mathsf{CCAL2},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$: $q_{\mathsf{CCAL2}} = (q_e, q_d, q_c)$ and the CCAL2 security game "removes M" from the CCAML2 game, i.e., a nonce-respecting version of CCAML2.
(iv) $\mathsf{PrivK}^{\mathsf{CPAML2},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$: $q_{\mathsf{CPAML2}} = (q_e, q_c)$ and no decryption oracle access is given in Figure 1: items (2) and (4) are removed but not item (5), hence the 2.

(v) $\mathsf{PrivK}^{\mathsf{CPAML},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$: $q_{\mathsf{CPAML}} = (q_e)$ and the CPAML game only keeps items (1) and (3) from the CCAML2 game, i.e., like the CPAML2 game without $\mathsf{L}_{\mathsf{decch}}$.
(vi) $\mathsf{PrivK}^{\mathsf{CPAL2},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$: $q_{\mathsf{CPAL2}} = (q_e, q_c)$, the CPAL2 game is a nonce-respecting version of the CPAML2. $\mathsf{L}_{\mathsf{decch}}$ is still available in item (5).
(vii) $\mathsf{PrivK}^{\mathsf{CCAL},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$: $q_{\mathsf{CCAL}} = (q_e, q_d)$ and the CCAL is a nonce-respecting version of CCAML, i.e., black-box decryption and nonce-respecting leaking encryption.
(viii) $\mathsf{PrivK}^{\mathsf{CPAL},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$: $q_{\mathsf{CPAL}} = (q_e)$ and the CPAL game only keeps nonce-respecting leaking encryption.

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

$$\mathsf{CIML2} \rightarrow \mathsf{CIML}, \mathsf{CIL2}, \mathsf{PIML2} \rightarrow \mathsf{PIML}, \mathsf{PIL2}, \mathsf{CIL} \rightarrow \mathsf{PIL}$$

Definition 16. *A nonce-based authenticated encryption with associated data* $\mathsf{AEAD} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with leakage* $\mathsf{L} = (\mathsf{L}_{\mathsf{enc}}, \mathsf{L}_{\mathsf{dec}})$ *is* $(q_e, d_d, q_l, t, \varepsilon)$*-pre-suf secure for a security parameter* n *if, for every* (q_e, q_d, q_l, t)*-bounded adversary* $\mathcal{A}^{\mathsf{L}}$*, we have* $\mathsf{pre} \in \{\mathsf{CI}, \mathsf{PI}\}$ *and*

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

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

(i) $\mathsf{PrivK}^{\mathsf{CIML2}}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$: the CIML2 game, see Table 1.
(ii) $\mathsf{PrivK}^{\mathsf{CIML}}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$: the CIML game removes $\mathsf{L}_{\mathsf{dec}}$, i.e., decryption is black-box.
(iii) $\mathsf{PrivK}^{\mathsf{CIL2}}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$: the CIL2 game is a nonce-respecting version of CIML2.
(iv) $\mathsf{PrivK}^{\mathsf{PIML2}}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$: in the PIML2 game the winning condition changed (Table 1).
(v) $\mathsf{PrivK}^{\mathsf{PIML}}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$: the PIML game removes $\mathsf{L}_{\mathsf{dec}}$ from PIML2.
(vi) $\mathsf{PrivK}^{\mathsf{PIL2}}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$: the PIL2 game is a nonce respecting version of PIML2.
(vii) $\mathsf{PrivK}^{\mathsf{CIL}}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$: the CIL game is a nonce-respecting version of CIML2 free of $\mathsf{L}_{\mathsf{dec}}$.
(viii) $\mathsf{PrivK}^{\mathsf{PIL}}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$: the PIL game is a nonce-respecting version of PIML2 free of $\mathsf{L}_{\mathsf{dec}}$.

Equivalence with Existing Notions. Among the above 16 notions 3 of them are equivalent to pre-existing ones: somehow CPAL appeared in [32] under the name of LMCPA, CIML was introduced in [8] and CIML2 was introduced in [9].

CPAL is our weakest confidentiality notion. In [32], LMCPA, after *leakage-resilient, multiple-block chosen-plaintext security*, was defined for IV-based encryption schemes, and was actually an artifact of their security analysis approach rather than a general security definition. However, by tweaking their security game, the notion could be used as the general definition for IND-CPA security with leakage corresponding to our CPAL notion.

The notions of CIML2 and CIML that we define in this paper consist of a rewriting of the definitions introduced in [8]. We simply adapt the definition to a general and coherent formalism.

E.2 Relations Within The Zoo: Single Challenge Setting

We picture all the 16 notions with their natural implications in Fig. 7.

Theorem 8 (Long diagonals (informal)). *There exist authenticated encryptions* $\mathsf{AEAD} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *showing that*

$$\begin{array}{ccc} \mathsf{CCAML} \not\Rightarrow \mathsf{CPAL2} & \mathsf{CCAL2} \not\Rightarrow \mathsf{CPAML} & \mathsf{CPAML2} \not\Rightarrow \mathsf{CCAL} \\ \mathsf{CIML} \not\Rightarrow \mathsf{PIL2} & \mathsf{CIL2} \not\Rightarrow \mathsf{PIML} & \mathsf{PIML2} \not\Rightarrow \mathsf{CIL} \end{array}$$

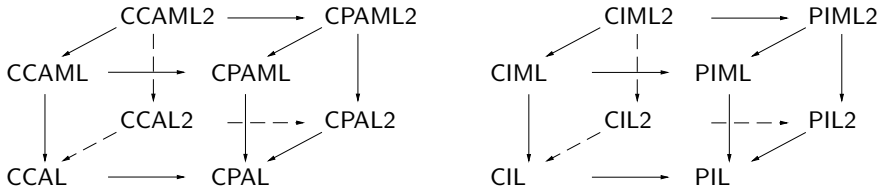

Fig. 7: Cubes for all the single-challenge security notions with different combinations of C/P (Ciphertext/Plaintext), M (Misuse), and 1 or 2 (the number of leaking oracles). (Top) the cube for the confidentiality notions. (Bottom) the cube for the integrity notions. Each arrow indicates an implication (to make it clearer, we picture some arrows in dashed form). As will be seen, these implications are all strict.

As a corollary all the arrows of Figure 7 are strict. To prove the theorem we only have to show 2 of the 3 assertions for confidentiality (resp., integrity).

Proof (Sketch). We first show that a security notion X *without* decryption leakages cannot imply the corresponding notion $\mathsf{X2}$ *with* decryption leakages. This would in particular establish six separations: $\mathsf{CCAML} \not\Rightarrow \mathsf{CPAL2}$, $\mathsf{CPAML} \not\Rightarrow \mathsf{CCAL2}$, $\mathsf{CCAL} \not\Rightarrow \mathsf{CPAML2}$, $\mathsf{CIML} \not\Rightarrow \mathsf{PIL2}$, $\mathsf{PIML} \not\Rightarrow \mathsf{CIL2}$, and $\mathsf{CIL} \not\Rightarrow \mathsf{PIML2}$. For this, assume that AEAD is a X secure scheme with master-key K. We define a new scheme AEAD^*, which is the same as AEAD except that its leakages for decryption queries explicitly include the master-key K. In this way, AEAD^* is clearly not $\mathsf{X2}$ secure (as the key is leaked). But it remains X secure, since this enhancement of decryption leakage *cannot* be observed in the X security game.

We then show that a security notion X *without* supporting nonce-misuse resistance/resilience cannot imply the corresponding notion XM *with* misuse resilience. This would establish four separations: $\mathsf{CPAL2} \not\Rightarrow \mathsf{CCAML}$, $\mathsf{CCAL2} \not\Rightarrow \mathsf{CPAML}$, $\mathsf{PIL2} \not\Rightarrow \mathsf{CIML}$, and $\mathsf{CIL2} \not\Rightarrow \mathsf{PIML}$. For this, assume that $\mathsf{AEAD} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with leakage $\mathsf{L} = (\mathsf{L}_{\mathsf{enc}}, \mathsf{L}_{\mathsf{dec}})$ is a X secure scheme. We define a new scheme $\mathsf{AEAD}^* = (\mathsf{Gen}', \mathsf{Enc}, \mathsf{Dec})$ with leakage $\mathsf{L} = (\mathsf{L}'_{\mathsf{enc}}, \mathsf{L}'_{\mathsf{dec}})$ as follows:

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

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

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

We finally consider the two remaining ones $\mathsf{CPAML2} \not\Rightarrow \mathsf{CCAL}$ and $\mathsf{PIML2} \not\Rightarrow \mathsf{CIL}$: the proofs follow the standard idea of showing CCA$\not\Rightarrow$CPA and $\mathsf{INT\text{-}PTXT} \not\Rightarrow \mathsf{INT\text{-}CTXT}$. In detail, consider $\mathsf{CPAML2} \not\Rightarrow \mathsf{CCAL}$ first, and assume that $\mathsf{AEAD} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is $\mathsf{CPAML2}$ secure. We define a new scheme $\mathsf{AEAD}^* = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}')$ as follows:

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

This very artificial scheme "gives up" by appending its key to the decrypted message upon any decryption query. Therefore, it cannot be CCA secure under any reasonable definition. Thus $\mathsf{CPAML2} \not\Rightarrow \mathsf{CCAL}$.

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

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

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

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

Remark. By revisiting the proof for $\mathsf{MR} \wedge \mathsf{CCAML2} \wedge \mathsf{PIML2} \not\Rightarrow \mathsf{CIML2}$ in subsection 3.3, it can be seen that the exhibited CIML2 adversary *only* rely on the leaking encryption. This means it also breaks the CIML security. Therefore, we already know that $\mathsf{MR} \wedge \mathsf{CCAML2} \wedge \mathsf{PIML2} \not\Rightarrow \mathsf{CIML}$.

E.3 Extending to the Multi-challenge Setting

Please recall the definition of $\mathsf{PrivK}^{\mathsf{mCCAML2},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$ from B. Similarly to E.1, by dropping some accesses to the distinct oracles of this game we derive the other confidentiality notions. Formally,

Definition 17. *A nonce-based authenticated encryption with associated data* $\mathsf{AEAD} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with leakage* $\mathsf{L} = (\mathsf{L}_{\mathsf{enc}}, \mathsf{L}_{\mathsf{dec}})$ *is* $(q_{\mathsf{mpre\text{-}suf}}, q_l, t, \varepsilon)$*-pre-suf secure for a security parameter* n *if, for every* $(q_{\mathsf{mpre\text{-}suf}}, q_l, t)$*-bounded adversary* $\mathcal{A}^{\mathsf{L}}$*, we have* $\mathsf{pre} \in \{\mathsf{CCA}, \mathsf{CPA}\}$ *and*

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

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

(i) $\mathsf{PrivK}^{\mathsf{mCCAML2},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$: $q_{\mathsf{mCCAML2}} = (q_e, q_d, q_c, q_m)$, the mCCAML2 game, see Figure 1.
(ii) $\mathsf{PrivK}^{\mathsf{mCCAML},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$: $q_{\mathsf{mCCAML}} = (q_e, q_d, q_m)$ and the mCCAML security game "removes 2" from the mCCAML2 game meaning that $\mathsf{L}_{\mathsf{dec}}$ is removed from all the oracles.
(iii) $\mathsf{PrivK}^{\mathsf{mCCAL2},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$: $q_{\mathsf{mCCAL2}} = (q_e, q_d, q_c, q_m)$ and the mCCAL2 security game "removes M" from the mCCAML2 game, i.e., a nonce-respecting version of mCCAML2.
(iv) $\mathsf{PrivK}^{\mathsf{mCPAML2},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$: $q_{\mathsf{mCPAML2}} = (q_e, q_c, q_m)$ and no decryption oracle access is given.
(v) $\mathsf{PrivK}^{\mathsf{mCPAML},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$: $q_{\mathsf{mCPAML}} = (q_e, q_m)$. This game is like the mCPAML2 game without $\mathsf{L}_{\mathsf{decch}}$.
(vi) $\mathsf{PrivK}^{\mathsf{mCPAL2},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$: $q_{\mathsf{mCPAL2}} = (q_e, q_c, q_m)$, the mCPAL2 game is a nonce-respecting version of the mCPAML2.
(vii) $\mathsf{PrivK}^{\mathsf{mCCAL},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$: $q_{\mathsf{mCCAL}} = (q_e, q_d, q_m)$ and the mCCAL is a nonce-respecting version of mCCAML.
(viii) $\mathsf{PrivK}^{\mathsf{mCPAL},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$: $q_{\mathsf{mCPAL}} = (q_e, q_m)$ and the mCPAL game only keeps nonce-respecting leaking encryption.

Relations Among Multi-challenge Notions. Obviously, if $\mathsf{mpre} - \mathsf{suf} \Leftrightarrow \mathsf{pre} - \mathsf{suf}$ for every combination of $\mathsf{pre} \in \{\mathsf{CCA}, \mathsf{CPA}\}$ and suf, then the relations proved in E.2 trivially extends to the multi-challenge setting. However, this is not always the case: for CPA notions with (challenge) decryption leakages the single-challenge notion does not imply the corresponding multi-challenge notion, see Theorem 7 for $\mathsf{CPAML2} \not\Rightarrow \mathsf{mCPAML2}$. In a similar vein, it can be seen $\mathsf{CPAL2} \not\Rightarrow \mathsf{mCPAL2}$.

We thus reconsider the relations around mCPAML2 and mCPAL2. The equivalence relations shown in does establish

$$\mathsf{mCCAL2} \not\Leftrightarrow \mathsf{mCPAML}.$$

This means we need to reconsider the two claims

$$\mathsf{mCCAML} \not\Rightarrow \mathsf{mCPAL2}, \text{ and } \mathsf{mCPAML2} \not\Rightarrow \mathsf{mCCAL}.$$

However, their proofs have been included in the proof of Theorem 8 as well: recall that

- We first showed that a notion X *without* decryption leakages cannot imply the corresponding notion X2 *with* decryption leakages. This is independent of the number of challenges, thus showing $\mathsf{mCPAML} \not\Rightarrow \mathsf{mCCAL2}$ and $\mathsf{mCCAL} \not\Rightarrow \mathsf{mCPAML2}$ as well.
- We then showed that a notion X *without* supporting nonce-misuse resistance/resilience cannot imply the corresponding notion XM *with* misuse resilience. So $\mathsf{mCCAL2} \not\Rightarrow \mathsf{mCPAML}$.
- We finally showed IND-CPA variants cannot imply IND-CCA variants. This covers $\mathsf{mCPAML2} \not\Rightarrow \mathsf{mCCAL}$.

So the cube in Fig. 7 (left) holds for the multi-challenge setting as well.

F EavDL and AEML

The notion EavDL was introduced by Berti et al. [8]. 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.

F.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 mEavDL, which is based on the experiment in Fig. 8.

$\mathsf{PrivK}^{\mathsf{mEavDL}(2),b}_{\mathcal{A}^{\mathsf{L}},\mathsf{AEAD}}(1^n)$ is the output of the following experiment.

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

Challenge queries: on possibly many occasions $\mathcal{A}^{\mathsf{L}}$ submits $(N_{ch}, A_{ch}, M^0, M^1)$,
If M^0 and M^1 have different (block) length or $N_{ch} \in \mathcal{E}$, returns $\perp$; Else updates $\mathcal{E}_{ch} \leftarrow \mathcal{E}_{ch} \cup \{N_{ch}\}$ and finally
- $\mathsf{PrivK}^{\mathsf{mEavDL},b}_{\mathcal{A}^{\mathsf{L}},\mathsf{AEAD}}$: computes and returns $C^b \leftarrow \mathsf{Enc}_k(N_{ch}, A_{ch}, M^b)$;
- $\mathsf{PrivK}^{\mathsf{mEavDL2},b}_{\mathcal{A}^{\mathsf{L}},\mathsf{AEAD}}$: computes and returns $(C^b, \mathsf{leak}^b_{\mathsf{e}}) \leftarrow \mathsf{LEnc}_k(N_{ch}, A_{ch}, M^b)$.

Decryption leakage queries: $\mathcal{A}^{\mathsf{L}}$ gets adaptive access to $\mathsf{L}_{\mathsf{Dec}}(\cdot,\cdot,\cdot)$,
$\mathsf{L}_{\mathsf{Dec}}(N, A, C)$ outputs $\mathsf{leak}_{\mathsf{d}} \leftarrow \mathsf{L}_{\mathsf{dec}}(k, N, A, C)$.

Finalization: $\mathcal{A}^{\mathsf{L}}$ outputs a guess bit b'. If $b = b'$, return 1, else return 0.

Fig. 8: The $\mathsf{PrivK}^{\mathsf{mEavDL},b}_{\mathcal{A}^{\mathsf{L}},\mathsf{AEAD}}$ and $\mathsf{PrivK}^{\mathsf{mEavDL2},b}_{\mathcal{A}^{\mathsf{L}},\mathsf{AEAD}}$ games.

Note that the challenge nonce has to be respecting for both mEavDL and mEavDL2. If we restrict the number of challenges to 1 in mEavDL then we recover the original EavDL notion of Berti et al. [8]

Definition 18 (mEavDL: Eavesdropper Security with Decryption Leakage). *An authenticated encryption* $\mathsf{AEAD} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with decryption leakage function* $\mathsf{L}_{\mathsf{dec}}$ *provides* $(q_m, q_d, t, \varepsilon)$*-indistinguishability of ciphertexts against eavesdropping with differential leakage attacks for a security parameter* n*, 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* $\mathsf{L}_{\mathsf{dec}}$*, and runs in time* t*,*

$$\Pr\left[\mathsf{PrivK}^{\mathsf{mEavDL},b}_{\mathcal{A}^{\mathsf{L}},\mathsf{AEAD}}(1^n) \Rightarrow 1\right] \leq \frac{1}{2} + \varepsilon$$

for the $\mathsf{PrivK}^{\mathsf{mEavDL},b}_{\mathcal{A}^{\mathsf{L}},\mathsf{AEAD}}$ *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 19 (mEavDL2: Eavesdropper Security with Encryption & Decryption Leakage). *An authenticated encryption* $\mathsf{AEAD} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with leakage function pair* $\mathsf{L} = (\mathsf{L}_{\mathsf{enc}}, \mathsf{L}_{\mathsf{dec}})$ *provides* $(q_m, q_d, t, \varepsilon)$*-indistinguishability of ciphertexts against eavesdropping with differential leakage attacks in encryption and decryption for a security parameter* n*, 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* $\mathsf{L}_{\mathsf{dec}}$*, and runs in time* t*,*

$$\Pr\left[\mathsf{PrivK}^{\mathsf{mEavDL},b}_{\mathcal{A}^{\mathsf{L}},\mathsf{AEAD}}(1^n) \Rightarrow 1\right] \leq \frac{1}{2} + \varepsilon$$

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

mCCAML2 ⇒ mEavDL2. Our definition of mCCAML2 almost explicitly incorporates the elements of mEavDL2, and thus mCCAML2 ⇒ mEavDL2. This means CCAML2 captures all pre-existing confidentiality notions.

F.2 mCCAML2* $\not\Rightarrow$ 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 ruins out the possibility of EavDL, since in the EavDL security game, the challenge plaintext M^b has to be hidden from $\mathcal{A}$.

More clearly, let $\mathsf{AEAD} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with leakage function $\mathsf{L} = (\mathsf{L}_{\mathsf{enc}}, \mathsf{L}_{\mathsf{dec}})$ be mCCAML2* secure with respect to L. Then we build $\mathsf{AEAD}' = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with leakage $\mathsf{L}' = (\mathsf{L}_{\mathsf{enc}}, \mathsf{L}'_{\mathsf{dec}})$ such that:

$\mathsf{L}'_{\mathsf{dec}}(k, N, A, C)$**:** outputs $\mathsf{leak}_{\mathsf{d}} = \mathsf{L}_{\mathsf{dec}}(k, N, A, C)$ if $\mathsf{Dec}_k(N, A, C) = \perp$ and outputs $(\mathsf{leak}_{\mathsf{d}}, M)$ otherwise, where $\mathsf{leak}_{\mathsf{d}} = \mathsf{L}_{\mathsf{dec}}(k, N, A, C)$, $M = \mathsf{Dec}_k(N, A, C)$.

It's not hard to see AEAD^* remains mCCAML2* secure, since the additional decryption leakage (the correct message) essentially contains no new information. However, during the EavDL security game, the challenge plaintext M^b is directly given by this leakage, allowing to precisely determine the value of b.

F.3 MR $\wedge$ mCPAML2 $\wedge$ CIML2 $\wedge$ mEavDL2 $\not\Rightarrow$ CCAML2*

In subsection 3.3 we have proved MR $\wedge$ CPAML2 $\wedge$ CIML2 $\not\Rightarrow$ CCAML2*. In this subsection we prove an even stronger claim of MR $\wedge$ mCPAML2 $\wedge$ CIML2 $\wedge$ mEavDL2 $\not\Rightarrow$ CCAML2*. The idea stems from the following observations:

- in the MR game, no leakage traces are given;
- in the mCPAML2 game, decryption leakage is only available for challenge ciphertexts, and thus "nonce respecting";
- in the CIML2 game, additional information unrelated to the key k but only to messages M are irrelevant;
- in the mEavDL2 game, assuming CIML2 holds, the decryption leakage oracle will essentially be "nonce respecting";
- on the other hand, in the CCAML2* 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 $\mathsf{AEAD} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be MR, mCPAML2 $\wedge$ mEavDL2 with respect to leakage function $\mathsf{L} = (\mathsf{L}_{\mathsf{enc}}, \mathsf{L}_{\mathsf{dec}})$ and CIML2 with respect to L^*. Then we build $\mathsf{AEAD}' = (\mathsf{Gen}', \mathsf{Enc}, \mathsf{Dec})$ with leakage function $\mathsf{L}' = (\mathsf{L}'_{\mathsf{enc}}, \mathsf{L}'_{\mathsf{dec}})$ as follows, where $N^\dagger, N^\circ, M^\dagger$ below are the outputs of a publicly samplable distribution parametrized by n:

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

$\mathsf{L}'_{\mathsf{enc}}((k, sh), N, A, M)$**:** outputs $\mathsf{leak}_{\mathsf{e}} = \mathsf{L}_{\mathsf{enc}}(k, N, A, M)$ 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: $N = N^\dagger$ and $M \neq M^\dagger$, then $B = t_0 \oplus t_1$.

$\mathsf{L}'_{\mathsf{dec}}((k, sh), N, A, C)$**:** outputs $\mathsf{leak}_{\mathsf{d}} = \mathsf{L}_{\mathsf{dec}}(k, N, A, C)$ as well as the additional value B if $\mathsf{Dec}_k(N, A, C) = M \neq \perp$ 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:

MR. Both scheme are the same from a black-box perspective.

CIML2. Given a CIML2 adversary $\mathcal{A}'$ against AEAD', it's not hard to see a CIML2 adversary $\mathcal{A}$ could use the oracles of AEAD and internally sampled bits t_0 and t_1 to perfectly simulate the oracles of AEAD' in front of $\mathcal{A}'$. By this, AEAD' is CIML2 as long as AEAD is CIML2.

mCPAML2. we show that if there is a mCPAML2 adversary $\mathcal{A}'$ against AEAD', then there is an adversary $\mathcal{A}$ which uses $\mathcal{A}'$ to break the mCPAML2 security of AEAD. In detail, once $\mathsf{PrivK}^{\mathsf{CCAML2},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}(1^n)$ is setup, $\mathcal{A}(1^n)$ publicly samples $N^\dagger, N^\circ, M^\dagger$ and sends to $\mathcal{A}'$ whatever is specified by AEAD'. $\mathcal{A}$ also picks two random bits t_0, t_1. Then it runs $\mathcal{A}'$ and simulates $\mathsf{PrivK}^{\mathsf{mCPAML2},b}_{\mathcal{A}',\mathsf{AEAD}',\mathsf{L}'}$ using its own interaction $\mathsf{PrivK}^{\mathsf{mCPAML2},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}$. For each query from $\mathcal{A}'$, the actions of $\mathcal{A}$ are as follows:

Leaking (non-challenge) encryption queries: On input (N, A, M),

(i) $\mathcal{A}$ queries its own leaking encryption oracle on (N, A, M) and gets back some $(C, \mathsf{leak_e})$ or possibly $\perp$;
(ii) If not $\perp$, $\mathcal{A}$ checks whether it should append an additional leakage B to $(C, \mathsf{leak_e})$ in the case (N, A, M) falls into one of the two cases described above.

$\mathcal{A}$ respectively returns $(C, \mathsf{leak_e})$ or $\perp$ or even $(C, \mathsf{leak_e}, B)$ to $\mathcal{A}'$ according to the above situations.

Leaking challenge query: on input $(N_{\mathsf{ch}}, A_{\mathsf{ch}}, M^0, M^1)$, $\mathcal{A}$ sends it to its leaking challenge oracle and gets the tuple $(C^b, \mathsf{leak}_{\mathsf{e}}^b)$ or possibly $\perp$. In the latter case it returns $\perp$ as well, otherwise

(i) if $N_{\mathsf{ch}} = N^\dagger$, regardless of what is encrypted by the challenge oracle, $\mathcal{A}$ uniformly samples a new random bit s^* and returns $(C^b, \mathsf{leak}_{\mathsf{e}}^b, s^*)$;
(ii) else, $\mathcal{A}$ just returns $(C^b, \mathsf{leak}_{\mathsf{e}}^b)$.

Challenge decryption leakage: on input i, $\mathcal{A}$ sends i to its challenge decryption leakage oracle and gets $\mathsf{leak}_{\mathsf{d}}^b$. Then,

(i) if the i-th challenge ciphertext contains $N_{\mathsf{ch}} = N^\circ$, $\mathcal{A}$ returns $(\mathsf{leak}_{\mathsf{d}}^b, t_0)$;
(ii) else, $\mathcal{A}$ just returns $\mathsf{leak}_{\mathsf{d}}^b$.

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

Now we explain why $\mathcal{A}$ properly emulates the mCPAML2 game in front of $\mathcal{A}'$. As long as $N^\dagger$ is never involved in a challenge query, the additional decryption leakage associated to the nonce N° is independent of the involved message. Therefore, the simulation is of no deviation.

We next concentrate on the case $N^\dagger$ is first involved in a challenge query. Since $N^\dagger$ can only appear once there it means that $\mathcal{A}$ will never sees either $t_0 \oplus t_1 \oplus 1$ or $t_0 \oplus t_1$. By the definition of mCPAML2 we know there is at most one challenge query under the nonce N°. 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 $\mathsf{PrivK}^{\mathsf{mCPAML2},b}_{\mathcal{A},\mathsf{AEAD}',\mathsf{L}}(1^n)$ and the game simulated by $\mathcal{A}$ are the same, and thus

$$\Pr\left[\mathsf{PrivK}^{\mathsf{mCPAML2},b}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}(1^n) \Rightarrow 1\right] = \Pr\left[\mathsf{PrivK}^{\mathsf{mCPAML2},b}_{\mathcal{A}',\mathsf{AEAD}',\mathsf{L}'}(1^n) \Rightarrow 1\right].$$

mEavDL2. The proof just follows the same line as the proof for 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 $\Pr[\mathsf{PrivK}^{\mathsf{CIML2}}_{\mathcal{A},\mathsf{AEAD},\mathsf{L}}(1^n) \Rightarrow 1]$, which is small since AEAD is CIML2 secure. Now, assuming that abort does not occur we are in the same situation than in the mCPAL argument above.

$\neg$CCAML2*. Let $\mathcal{A}'$ be an adversary which given $N^\dagger, N^\circ, M^\dagger$ sequentially makes the following queries for $M \neq M^\dagger$:

(i) $(N^\circ, A, M^\dagger)$ to the leaking encryption oracle and get $C^\dagger$;
(ii) (N°, A, M) to the leaking encryption oracle and get C;
(iii) $(N^\circ, A, C^\dagger)$ to the leaking decryption oracle and get t_0;
(iv) (N°, A, C) to the leaking decryption oracle and get t_1.

$\mathcal{A}'$ finally submits $(N^\dagger, A, M^\dagger, M)$ 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 MR $\wedge$ mCPAML2 $\wedge$ CIML2 $\wedge$ mEavDL2 $\not\Rightarrow$ CCAML2*.

G FEMALE Security Proofs

We first analyze the CCA security of FEMALE in sections G.1 and G.2, and then prove CIML2 and MR in sections G.3 and G.4 resp.

G.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 B. An approach is to first prove the CCAML2 bound claimed in Theorem 3 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 G.1 and G.2 devote to prove the following theorem.

Theorem 9. *Let* $H : \{0,1\}^\kappa \times \{0,1\}^* \to \{0,1\}^n$ *be a* $(0, t', \varepsilon_{cr})$*-collision resistant and* $(q_d, t', \varepsilon_{pr})$*-range-oriented preimage resistant hash function,* $E^* : \{0,1\}^n \times \{0,1\} \times \{0,1\}^n \to \{0,1\}^n$ *be a* $(2q_e + 2q_d + 2q_m, t', \varepsilon_{E^*})$*-strong tweakable pseudorandom permutation, and* $E : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ *be a* $(2, t, \varepsilon_E)$*-PRP, whose implementation leakage function* L_E *has* $(q_S, t_S, q_l, t, \varepsilon_{(p,2)\text{-}rsim})$ $(p, 2)$*-R-simulatable leakages. Then the* FEMALE *implementation with leakage function* $\mathsf{L} = (\mathsf{L}_{\mathsf{enc}}, \mathsf{L}_{\mathsf{dec}})$ *defined before is* $(q_e, q_d, p-1, q_m, q_l, t, \varepsilon_{\mathsf{mCCAML2}})$ mCCAML2*-secure, where*

$$\varepsilon_{\mathsf{mCCAML2}} \leq 2\varepsilon_{E^*} + \frac{q_e + q_m - 1}{2^n} + \varepsilon_{cr} + \varepsilon_{pr} + \textstyle\sum_{i=1}^{q_m} \varepsilon_{\mathsf{FEMALE}\text{-}eav}(\ell_i),$$

and $\varepsilon_{\mathsf{FEMALE}\text{-}eav}(\ell_i)$ *is as defined in Theorem 3 but where* ℓ_i *corresponds to the block-length of the* i*-th challenge messages. Here* $t' = t + (q_e + q_d + q_m)(t_\$ + t_{1-pass})$*,* t_{1-pass} *is the maximum running time of* FEMALE *upon a single (encryption or decryption) query, and* $t_\$$ *is the time needed for randomly sampling a value from* $\{0,1\}^n$*.*

As mentioned, setting $q_m = 1$ recovers Theorem 3. It will be apparent in the proof that FEMALE actually satisfies an even stronger notion of CCAML2. Since the encryption starts with $R \leftarrow \mathsf{H}(0\|N\|A)$, 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 "indistinguishablity-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. 5, 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 $(\mathsf{FEMALE}(M), \mathsf{L}_{\mathsf{FEMALE}(M)})$ and $(\$, \mathcal{S}_{\mathsf{FEMALE}}(M))$, for a single message M (Lemma 3). In other words, $(\mathsf{FEMALE}(M), \mathsf{L}_{\mathsf{FEMALE}(M)})$, the process of using FEMALE to encrypt a single message, is indistinguishable from an idealized process that produces random outputs \$ and simulated leakages $\mathcal{S}_{\mathsf{FEMALE}}(M)$. 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 $\mathsf{FEMALE}(M)$, since the leakages $\mathsf{L}_{\mathsf{FEMALE}(M)}$ or its indistinguishable counterpart $\mathcal{S}_{\mathsf{FEMALE}}(M)$ may leak M completely. To remedy this, we focus on the idealized process $(\$, \mathcal{S}_{\mathsf{FEMALE}}(M))$, and show how to relate its eavesdropper security to the eavesdropper security of applying LISE—the single-block encryption scheme—to independently encrypt $|M|$ blocks. Since we've established the indistinguishability of LISE and LRSE, the eavesdropper security of $(\mathsf{FEMALE}(M), \mathsf{L}_{\mathsf{FEMALE}(M)})$ can be established via the following chain:

$$\begin{aligned}
&\text{eavesdropper security of LRSE (our assumption, see (1))}\\
\Rightarrow&\text{eavesdropper security of LISE (using indistinguishability of LRSE and LISE)}\\
\Rightarrow&\text{eavesdropper security of the idealized process } (\$, \mathcal{S}_{\mathsf{FEMALE}}(M))\\
\Rightarrow&\text{eavesdropper security of } (\mathsf{FEMALE}(M), \mathsf{L}_{\mathsf{FEMALE}(M)}).
\end{aligned}$$

Eventually, based on the eavesdropper security of $(\mathsf{FEMALE}(M), \mathsf{L}_{\mathsf{FEMALE}(M)})$, we establish the mCCAML2 security (this step is in subsection G.1). Roughly, the proof relies on the following features of FEMALE:

(i) Every invalid decryption query only leaks a pseudorandom value, i.e. $\mathsf{E}_k^{*,1}(T)$ for some T. So the encryption can be seen as independent from these values;

(ii) 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 $(\mathsf{FEMALE}(M), \mathsf{L}_{\mathsf{FEMALE}(M)})$ with a lot of offline computations (i.e. all the other encryptions are turned into offline computations).

Indistinguishability of Real-Leaking World and Ideal-Simulating World. Standaert et al. proved that based on the pseudorandom security of E and the simulatability of the leakage, (roughly) the "real-leaking world" $(\mathsf{E}_k(p), \mathsf{L}(k,p))$ is indistinguishable from the "ideal-simulating" world $(\$, \mathcal{S}^{\mathsf{L}}(k,p,\$))$ [37]. 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 $[\mathsf{leak}_1, \ldots, \mathsf{leak}_\ell]^p$ for the vector of ℓp leakages, which consist of ℓ (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 $\mathsf{L}(x)$ 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 $\mathsf{L}(x)$.

Lemma 1. *Let $\mathsf{E}\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a $(2,t,\varepsilon_{\mathsf{E}})$-PRP, whose implementation has a leakage function L_{E} having $(q_{\mathcal{S}}, t_{\mathcal{S}}, q_l, t, \varepsilon_{(p,2)\text{-}rsim})$ $(p,2)$-R-simulatable leakages, and let $\mathcal{S}^{\mathsf{L}}$ be an appropriate $(q_{\mathcal{S}}, t_{\mathcal{S}})$-bounded leakage simulator. Then, for every $k_{pre}, p_A, p_B, z \in \{0,1\}^n$ and every $(q_l - q^*, t - t^*)$-bounded distinguisher $\mathcal{D}^{\mathsf{L}}$, the following holds:*

$$\Big|\Pr[k_{ch} \xleftarrow{\$} \{0,1\}^n : \mathcal{D}^{\mathsf{L}}(\mathsf{E}_{k_{ch}}(p_A), \mathsf{E}_{k_{ch}}(p_B), [\mathsf{L}_{\mathsf{E}}(k_{ch}, p_A), \mathsf{L}_{\mathsf{E}}(k_{ch}, p_B), \mathcal{S}^{\mathsf{L}}(k_{pre}, z, k_{ch})]^p) \Rightarrow 1]$$
$$- \Pr[k_{ch}, c_A, c_B \xleftarrow{\$} \{0,1\}^n, c_A \neq c_B \text{ iff. } p_A \neq p_B :$$
$$\mathcal{D}^{\mathsf{L}}(c_A, c_B, [\mathcal{S}^{\mathsf{L}}(k_{ch}, p_A, c_A), \mathcal{S}^{\mathsf{L}}(k_{ch}, p_B, c_B), \mathcal{S}^{\mathsf{L}}(k_{pre}, z, k_{ch})]^p) \Rightarrow 1]\Big| \leq \varepsilon_{\mathsf{E}} + \varepsilon_{(p,2)\text{-}rsim}.$$

Here $q^* = 3p \cdot q_{\mathcal{S}}$, while $t^* = Max\{t_r, t_{sim}\}$, in which t_r is equal to $3p \cdot t_{\mathcal{S}}$ augmented with the time needed to make 2 oracle queries to the PRP challenger and select a uniformly random key in $\{0,1\}^n$, and t_{sim} is the time needed to relay the content of $2p$ Enc and p Gen queries from and to a $(p,2)$-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 $\mathsf{E}_{k_{ch}}(p_A)$ and $\mathsf{E}_{k_{ch}}(p_B)$ 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 $\mathcal{S}^{\mathsf{L}}$. The resulted algorithm is defined in Fig. 9.

Description of LISE: (tool for the proof)
ISGen(1^n) picks $k_{ch} \xleftarrow{\$} \{0,1\}^n$, $\mathcal{M}, \mathcal{C} = \{0,1\}^n$ $(p_A, p_B \in \{0,1\}^n)$
ISEnc$_{k_{ch}}(m)$ returns (k_{up}, c), where $c = y_{ch} \oplus m$, and $k_{up}, y_{ch} \xleftarrow{\$} \{0,1\}^n$, $k_{up} \neq y_{ch}$ as long as $p_B \neq p_A$ (and $k_{up} = y_{ch}$ otherwise).
ISDec$_{k_{ch}}(c)$ proceeds in the natural way.

The leakage $\mathsf{L}_{\mathsf{ISE}} = (\mathsf{L}_{\mathsf{isenc}}, \mathsf{L}_{\mathsf{isdec}}, k_{pre})$ resulting from the LISE implementation is defined as $\mathsf{L}_{\mathsf{isenc}}(k_{ch}, m) = (\mathcal{S}^{\mathsf{L}}(k_{ch}, p_A, k_{up}), \mathcal{S}^{\mathsf{L}}(k_{ch}, p_B, y_{ch}), \mathsf{L}_{\oplus}(y_{ch}, m), \mathcal{S}^{\mathsf{L}}(k_{pre}, p_A, k_{ch}))$, $\mathsf{L}_{\mathsf{isdec}}(k_{ch}, c) = (\mathcal{S}^{\mathsf{L}}(k_{ch}, p_A, k_{up}), \mathcal{S}^{\mathsf{L}}(k_{ch}, p_B, y_{ch}), \mathsf{L}_{\oplus}(y_{ch}, c), \mathcal{S}^{\mathsf{L}}(k_{pre}, p_A, k_{ch}))$ for a fixed random $k_{pre} \xleftarrow{\$} \{0,1\}^n$.

Fig. 9: The ideal single-block encryption scheme ISEnc.

We also define $\mathsf{LISEnc}^+_{k_{ch}}(m) = (\mathsf{LISEnc}_{k_{ch}}(m), [\mathsf{L}_{\mathsf{isdec}}(k_{ch}, c)]^{p-1}, k_{pre})$ for $(c, k_{up}) = \mathsf{ISEnc}_{k_{ch}}(m)$. Similarly to Pereira et al. [32], our ISEnc scheme is indistinguishable from its real version RSEnc.

Lemma 2. *Let $\mathsf{E}\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a $(2,t,\varepsilon_{\mathsf{E}})$-PRP, whose implementation has a leakage function L_{E} having $(q_{\mathcal{S}}, t_{\mathcal{S}}, q_l, t, \varepsilon_{(p,2)\text{-}rsim})$ $(p,2)$-R-simulatable leakages, and let $\mathcal{S}^{\mathsf{L}}$ be an appropriate $(q_{\mathcal{S}}, t_{\mathcal{S}})$-bounded*

leakage simulator. Then, for every $p_A, p_B \in \{0,1\}^n$, $p_A \neq p_B$, and every $(q_l - q^*, t - t^*)$-bounded distinguisher $\mathcal{D}^{\mathsf{L}}$, the following holds:

$$|\Pr[\mathcal{D}^{\mathsf{L}_{\mathsf{RSE}}}(m, \mathsf{LRSEnc}^{+}_{k_{ch}}(m)) \Rightarrow 1] - \Pr[\mathcal{D}^{\mathsf{L}_{\mathsf{ISE}}}(m, \mathsf{LISEnc}^{+}_{k_{ch}}(m)) \Rightarrow 1]| \leq \varepsilon_{\mathsf{E}} + \varepsilon_{(p,2)\text{-}rsim}.$$

Here $q^* = 3p \cdot q_{\mathcal{S}} + p$, while $t^* = Max\{t_r, t_{sim}\}$, in which t_r is equal to $3p \cdot t_{\mathcal{S}} + 2t_{\oplus}$ augmented with the time needed to make 2 oracle queries to the PRP challenger and select a uniformly random key in $\{0,1\}^n$, $t_{\oplus}$ is the time needed to evaluate the $\oplus$ action on an n-bit input, and t_{sim} is the time needed to relay the content of four Enc and two Gen queries from and to a $(p,2)$-rsim challenger.

Proof. The proof just follows the same line as Lemma 1. Note that to generate the leakage $\mathsf{L}_{\oplus}(y_{ch}, c)$ $p-1$ 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 $p_A = p_B$. However, as we remarked before, when the input p_B equals p_A, neither RSEnc not 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 ISEnc we do not enforce the constraint $p_B \neq p_A$, since the case of $p_B = p_A$ would be used in some of our arguments below (of course, these arguments do not rely on the eavesdropper security of ISEnc in the case of $p_B = p_A$).

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. 10, while LIFSM is in Fig. 11.

Description of RFSM:

- Gen picks $s_0 \xleftarrow{\$} \{0,1\}^n$
- $\mathsf{RFSM}_{k_0}(m_1, \ldots, m_\ell)$ proceeds in four steps:
  - (i) Initializes an empty list leak for the leakage;
  - (ii) Computes $s_1 \leftarrow \mathsf{E}_{s_0}(p_A)$ and $w \leftarrow \mathsf{E}_{s_0}(p_B)$, and adds $[\mathsf{L}_{\mathsf{E}}(s_0, p_A)]^p$ and $\mathsf{L}_{\mathsf{E}}(s_0, p_B)$ to the list leak;
  - (iii) For $i = 1, \ldots, \ell$, computes $s_{i+1} \leftarrow \mathsf{E}_{s_i}(p_A)$, $y_i \leftarrow \mathsf{E}_{s_i}(d_{i-1})$, and $d_i \leftarrow y_i \oplus m_i$, and adds $\mathsf{L}_{\mathsf{E}}(s_i, p_A)$, $\mathsf{L}_{\mathsf{E}}(s_i, d_{i-1})$, $\mathsf{L}_{\oplus}(y_i, m_i)$, and $[\mathsf{L}_{\mathsf{E}}(s_i, p_A), \mathsf{L}_{\mathsf{E}}(s_i, d_{i-1}), \mathsf{L}_{\oplus}(y_i, d_i)]^{p-1}$ to the list leak;
  - (iv) Computes $U \leftarrow \mathsf{E}_{s_{\ell+1}}(p_A)$, $W \leftarrow \mathsf{E}_{s_{\ell+1}}(d_\ell)$, $V \leftarrow \mathsf{E}_w(W)$, and $k_1 \leftarrow \mathsf{E}_U(V)$; and adds $\mathsf{L}_{\mathsf{E}}(s_{\ell+1}, p_A)$, $\mathsf{L}_{\mathsf{E}}(s_{\ell+1}, d_\ell)$, $\mathsf{L}_{\mathsf{E}}(w, W)$, $\mathsf{L}_{\mathsf{E}}(U, V)$, and $[\mathsf{L}_{\mathsf{E}}(s_{\ell+1}, p_A), \mathsf{L}_{\mathsf{E}}(U, V)]^{p-1}$ to the list leak;
  - (v) for $i = 1, \ldots, \ell$, computes $k_{i+1} \leftarrow \mathsf{E}_{k_i}(p_A)$, $z_i \leftarrow \mathsf{E}_{k_i}(p_B)$, and $c_i \leftarrow z_i \oplus d_i$, and adds $\mathsf{L}_{\mathsf{E}}(k_i, p_A)$, $\mathsf{L}_{\mathsf{E}}(k_i, p_B)$, $\mathsf{L}_{\oplus}(z_i, d_i)$, and $[\mathsf{L}_{\mathsf{E}}(k_i, p_A), \mathsf{L}_{\mathsf{E}}(k_i, p_B), \mathsf{L}_{\oplus}(z_i, c_i)]^{p-1}$ to the list leak.

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

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

Fig. 10: The RFSM scheme and the involved leakages.

Description of IFSM:

- $\mathsf{IFSM}_{k_0}(m_1, \ldots, m_\ell)$ proceeds in four steps:
  - (i) Initializes an empty list leak for the leakage;
  - (ii) Samples $s_1 \xleftarrow{\$} \{0,1\}^n$, and adds $[\mathcal{S}^{\mathsf{L}}(s_0, p_A, s_1)]^p$ to the list leak;
  - (iii) For $i = 1, \ldots, \ell$, samples $s_{i+1} \xleftarrow{\$} \{0,1\}^n$ and $y_i \xleftarrow{\$} \{0,1\}^n$ 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 $\mathcal{S}^{\mathsf{L}}(s_i, p_A, s_{i+1})$, $\mathcal{S}^{\mathsf{L}}(s_i, d_{i-1}, y_i)$, $\mathsf{L}_{\oplus}(y_i, m_i)$, and $[\mathcal{S}^{\mathsf{L}}(s_i, p_A, s_{i+1}), \mathcal{S}^{\mathsf{L}}(s_i, d_{i-1}, y_i), \mathsf{L}_{\oplus}(y_i, d_i)]^{p-1}$ to the list leak;
  - (iv) Samples $U \xleftarrow{\$} \{0,1\}^n$, $W \xleftarrow{\$} \{0,1\}^n$, $U \neq W$ iff. $d_\ell \neq p_A$; $w \xleftarrow{\$} \{0,1\}^n$, $w \neq s_1$; $V \xleftarrow{\$} \{0,1\}^n$, and $k_1 \xleftarrow{\$} \{0,1\}^n$; and adds $\mathcal{S}^{\mathsf{L}}(s_{\ell+1}, p_A, U)$, $\mathcal{S}^{\mathsf{L}}(s_{\ell+1}, d_\ell, W)$, $\mathcal{S}^{\mathsf{L}}(s_0, p_B, w)$, $\mathcal{S}^{\mathsf{L}}(w, W, V)$, $\mathcal{S}^{\mathsf{L}}(U, V, k_1)$, $[\mathcal{S}^{\mathsf{L}}(s_{\ell+1}, p_A, U), \mathcal{S}^{\mathsf{L}}(U, V, k_1)]^{p-1}$ to the list leak;
  - (v) For $i = 1, \ldots, \ell$, samples $k_{i+1} \xleftarrow{\$} \{0,1\}^n$, $z_i \xleftarrow{\$} \{0,1\}^n$, and $c_i \leftarrow z_i \oplus d_i$, and adds $\mathcal{S}^{\mathsf{L}}(k_i, p_A, k_{i+1})$, $\mathcal{S}^{\mathsf{L}}(k_i, p_B, z_i)$, $\mathsf{L}_{\oplus}(z_i, d_i)$, and $[\mathcal{S}^{\mathsf{L}}(k_i, p_A, k_{i+1}), \mathcal{S}^{\mathsf{L}}(k_i, p_B, z_i), \mathsf{L}_{\oplus}(z_i, c_i)]^{p-1}$ to the list leak.

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

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

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

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

Lemma 3. *Let $\mathsf{E}\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a $(2,t,\varepsilon_{\mathsf{E}})$-PRP, whose implementation has a leakage function L_{E} having $(q_{\mathcal{S}}, t_{\mathcal{S}}, q_l, t, \varepsilon_{(p,2)\text{-}rsim})$ $(p,2)$-R-simulatable leakages, and let $\mathcal{S}^{\mathsf{L}}$ be an appropriate $(q_{\mathcal{S}}, t_{\mathcal{S}})$-bounded leakage simulator. Then, for every ℓ-block message m, every $p_A \neq p_B$, and every $(q_l - p\cdot q_r - q^*, t - p\cdot t_r - t^*)$-bounded distinguisher $\mathcal{D}^{\mathsf{L}}$ (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:*

$$\left|\Pr[\mathcal{D}^{\mathsf{L}}(m, \mathsf{LRFSM}_{s_0}(m)) \Rightarrow 1] - \Pr[\mathcal{D}^{\mathsf{L}}(m, \mathsf{LIFSM}_{s_0}(m)) \Rightarrow 1]\right| \leq 2(\ell+2)(\varepsilon_{\mathsf{E}} + \varepsilon_{(p,2)\text{-}rsim}) + \frac{2\ell+2}{2^n}.$$

Here $q_r = (4\ell+5)(q_{\mathcal{S}}+1)+2\ell$, q^* and t^* are as defined in Lemma 1, and $t_r = (4\ell+5)(t_{\mathsf{E}} + t_{\$} + t_{\mathcal{S}}) + 2\ell\cdot t_{\oplus}$, where t_{E} is the time needed for evaluating E once, $t_{\$}$ is the time needed for randomly sampling a value from $\{0,1\}^n$, and $t_{\oplus}$ is the time needed for evaluating $\oplus$ once.

Proof. We define $\mathsf{G}_{0,1}$ as the security game in which $\mathcal{D}^{\mathsf{L}}$ receives $\mathsf{LRFSM}_{s_0}(m)$ as the input, and G^*_{ℓ} as the game in which $\mathcal{D}^{\mathsf{L}}$ receives $\mathsf{LIFSM}_{s_0}(m)$ as the input.

We show that $\mathsf{G}_{0,1}$ could be transited to G^*_{ℓ} via a sequence of games

$$\mathsf{G}_0, \mathsf{G}_1, \ldots, \mathsf{G}_{\ell}, \mathsf{G}_{\ell+1}, \mathsf{G}_{\ell+2}, \mathsf{G}_{\ell+3} = \mathsf{G}^*_0, \mathsf{G}^*_1, \ldots, \mathsf{G}^*_{\ell-1}.$$

The first half of the sequence $\mathsf{G}_0, \ldots, \mathsf{G}_{\ell}, \mathsf{G}_{\ell+1}, \mathsf{G}_{\ell+2}$, and $\mathsf{G}_{\ell+3}$ "idealizes" the *Ephemeral key-IV generation* phase of the encryption. In detail, we first consider $\mathsf{G}_{0,1}$, and replace the two intermediate values $\mathsf{E}_{s_0}(p_A)$ and $\mathsf{E}_{s_0}(p_B)$ by two distinct random values s_1 and w. We also replace the leakages $[\mathsf{L}_{\mathsf{E}}(s_0, p_A)]^p$ and $\mathsf{L}_{\mathsf{E}}(s_0, p_B)$ with $[\mathcal{S}^{\mathsf{L}}(s_0, p_A, s_1)]^p$ and $\mathcal{S}^{\mathsf{L}}(s_0, p_B, w)$. This yields the game G_0.

We next derive an upper bound for $|\Pr[(\mathcal{D}^{\mathsf{L}})^{\mathsf{G}_0} \Rightarrow 1] - \Pr[(\mathcal{D}^{\mathsf{L}})^{\mathsf{G}_{0,1}} \Rightarrow 1]|$. For this, we assume a $(q_l - p\cdot q_r - q^*, t - p\cdot t_r - t^*)$-bounded distinguisher $\mathcal{D}^{\mathsf{L}}$ against G_0 and $\mathsf{G}_{0,1}$, and we build a distinguisher $\mathcal{D}^{\mathsf{L}'}$ against the real-leaking-world and the ideal-simulation-world. Assume that $\mathcal{D}^{\mathsf{L}'}$ receives the tuple

$$(c_A, c_B, [\mathsf{leak}_1, \mathsf{leak}_2, \mathcal{S}^{\mathsf{L}}(\cdot, p_A, k_{ch})]^p)$$

as inputs, with $c_A \neq c_B$. $\mathcal{D}^{\mathsf{L}'}$ proceeds in two steps:

(1) $\mathcal{D}^{\mathsf{L}'}$ first uses $[\mathsf{leak}_1]^p$ and leak_2 as the leakages of the first iteration;
(2) $\mathcal{D}^{\mathsf{L}'}$ 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 $\mathcal{D}^{\mathsf{L}}$, and outputs whatever $\mathcal{D}^{\mathsf{L}}$ outputs.

It can be seen depending on whether the inputs to $\mathcal{D}^{\mathsf{L}'}$ follow real-leaking or ideal-simulating distribution, $\mathcal{D}^{\mathsf{L}}$ is eventually interacting with $\mathsf{G}_{0,1}$ or G_0. We show that to perform the additional operations, $\mathcal{D}^{\mathsf{L}'}$ 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 2ℓ 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 $\mathcal{S}$, which costs $q_{\mathcal{S}}$ queries to L and $(t_{\$} + t_{\mathcal{S}})$ running time;
- each xor operation costs 1 query to L and $t_{\oplus}$ running time.

Therefore, to emulate the "hybrid" encryption process once, $\mathcal{D}^{\mathsf{L}'}$ needs at most $(4\ell+5)(q_{\mathcal{S}}+1)+2\ell = q_r$ queries to L and $(4\ell+5)(t_{\mathsf{E}} + t_{\$} + t_{\mathcal{S}}) + 2\ell\cdot t_{\oplus} = t_r$ running time. To obtain the required decryption leakage traces, $\mathcal{D}^{\mathsf{L}'}$ has to additionally perform the "hybrid" *decryption* process for $p-1$ times, which contributes to $(p-1)q_r$ more queries and $(p-1)t_r$ more time. Therefore, as claimed, $\mathcal{D}^{\mathsf{L}'}$ 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

$$|\Pr[(\mathcal{D}^{\mathsf{L}})^{\mathsf{G}_0} \Rightarrow 1] - \Pr[(\mathcal{D}^{\mathsf{L}})^{\mathsf{G}_{0,1}} \Rightarrow 1]| \leq \varepsilon_{\mathsf{E}} + \varepsilon_{(p,2)\text{-}rsim}.$$

Then, for $1 \leq i \leq \ell$, to obtain G_i, we modify the game G_{i-1} by replacing the two intermediate values $\mathsf{E}_{s_i}(p_A)$ and $\mathsf{E}_{s_i}(d_{i-1})$ 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 $[\mathsf{L}_{\mathsf{E}}(s_i, p_A), \mathsf{L}_{\mathsf{E}}(s_i, d_{i-1})]^p$, $\mathsf{L}_{\oplus}(\mathsf{E}_{s_i}(d_{i-1}), m_i)$, and $[\mathsf{L}_{\oplus}(\mathsf{E}_{s_i}(d_{i-1}), d_i)]^{p-1}$ with $[\mathcal{S}^{\mathsf{L}}(s_i, p_A, s_{i+1}), \mathcal{S}^{\mathsf{L}}(s_i, d_{i-1}, y_i)]^p$, $\mathsf{L}_{\oplus}(y_i, m_i)$, and $[\mathsf{L}_{\oplus}(y_i, d_i)]^{p-1}$. To show the indistinguishability of G_i and G_{i-1}, assume that $\mathcal{D}^{\mathsf{L}'}$ receives $(c_A, c_B, [\mathsf{leak}_1, \mathsf{leak}_2, \mathcal{S}^{\mathsf{L}}(\cdot, p_A, k_{ch})]^p)$ as inputs, with $c_A \neq c_B$. $\mathcal{D}^{\mathsf{L}'}$ proceeds in five steps (to ease description, we define $d_{-1} = p_B$ and $y_0 = w$):

(1) $\mathcal{D}^{\mathsf{L}'}$ first uniformly samples s_0;
(2) For $j = 0, \ldots, i-2$, $\mathcal{D}^{\mathsf{L}'}$ uniformly samples random values s_{j+1}, y_j such that $s_{j+1} \neq y_j$ iff. $d_{j-1} \neq p_A$, simulates the traces $[\mathcal{S}^{\mathsf{L}}(s_j, p_A, s_j), \mathcal{S}^{\mathsf{L}}(s_j, d_{j-1}, y_j)]^p$,[5] computes $d_j \leftarrow y_j \oplus m_j^0$ and $m_j^0 \leftarrow y_j \oplus d_j$ and obtains the traces $\mathsf{L}_{\oplus}(y_j, m_j^0)$ and $[\mathsf{L}_{\oplus}(y_j, d_j)]^{p-1}$;[6]
(3) Then, if $d_{i-2} \neq p_A$, it uniformly samples y_{i-1}, computes $d_{i-1} \leftarrow y_{i-1} \oplus m_{i-1}$, $m_{i-1} \leftarrow y_{i-1} \oplus d_{i-1}$, and uses $[\mathcal{S}^{\mathsf{L}}(s_{i-1}, p_A, k_{ch}), \mathcal{S}^{\mathsf{L}}(s_{i-1}, d_{i-2}, y_{i-1})]^p$, $\mathsf{L}_{\oplus}(y_{i-1}, m_{i-1}), [\mathsf{L}_{\oplus}(y_{i-1}, d_{i-1})]^{p-1}$ as the traces of the i-th iteration in the first pass;[7]
(4) Sets $s_{i+1} \leftarrow c_A$ and $y_i \leftarrow c_B$, computes $d_i \leftarrow y_i \oplus m_i$, and uses $[\mathsf{leak}_1, \mathsf{leak}_2]^p$, $\mathsf{L}_{\oplus}(y_i, m_i), [\mathsf{L}_{\oplus}(y_i, d_i)]^{p-1}$ as the corresponding leakages;
(5) Takes s_{i+1} and d_i as the starting points and emulates the remaining part of the execution of LRFSM encryption. Eventually, $\mathcal{D}^{\mathsf{L}'}$ serves the obtained ciphertext $V \| c_1 \| \ldots \| c_\ell$ as well as the leakage traces to $\mathcal{D}^{\mathsf{L}}$, and outputs whatever $\mathcal{D}^{\mathsf{L}}$ outputs.

It can be seen that depending on the input tuple received by $\mathcal{D}^{\mathsf{L}'}$ is real-leaking or ideal-simulation, $\mathcal{D}^{\mathsf{L}}$ 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 Bad_i, we have

$$\Pr[(\mathcal{D}^{\mathsf{L}})^{\mathsf{G}_i} \Rightarrow 1] - \Pr[(\mathcal{D}^{\mathsf{L}})^{\mathsf{G}_{i-1}} \Rightarrow 1] \leq \Pr[\mathsf{Bad}_i] + \varepsilon_{\mathsf{E}} + \varepsilon_{(p,2)\text{-}rsim}.$$

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, $\Pr[\mathsf{Bad}_i] = \frac{2}{2^n}$; 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 $\Pr[\mathsf{Bad}_i] = \frac{1}{2^n}$.

The hop from G_ℓ to $\mathsf{G}_{\ell+1}$ is slightly different, as no message block would be xored with $\mathsf{E}_{s_{\ell+1}}(d_\ell)$. In detail, we replace the two intermediate values $\mathsf{E}_{s_{\ell+1}}(p_A)$ and $\mathsf{E}_{s_{\ell+1}}(d_\ell)$ by two random values U and W such that $W = U$ iff. $d_\ell = p_A$, and replace the leakage $[\mathsf{L}_{\mathsf{E}}(s_{\ell+1}, p_A)]^p$ and $\mathsf{L}_{\mathsf{E}}(s_{\ell+1}, d_\ell)$ by $[\mathcal{S}^{\mathsf{L}}(s_{\ell+1}, p_A, U)]^p$ and $\mathcal{S}^{\mathsf{L}}(s_{\ell+1}, d_\ell, W)$. By an analysis similar to the argument above, $\mathcal{D}^{\mathsf{L}'}$ could consistently simulate $\mathsf{G}_{i-1}/\mathsf{G}_i$ against $\mathcal{D}^{\mathsf{L}}$ unless $d_{\ell-1} = p_A$ or $y_\ell = k_{ch}$ (denoted $\mathsf{Bad}_{\ell+1}$). Therefore,

$$|\Pr[(\mathcal{D}^{\mathsf{L}})^{\mathsf{G}_{\ell+1}} \Rightarrow 1] - \Pr[(\mathcal{D}^{\mathsf{L}})^{\mathsf{G}_\ell} \Rightarrow 1]| \leq \Pr[\mathsf{Bad}_{\ell+1}] + \varepsilon_{\mathsf{E}} + \varepsilon_{(p,2)\text{-}rsim} \leq \frac{2}{2^n} + \varepsilon_{\mathsf{E}} + \varepsilon_{(p,2)\text{-}rsim}.$$

We then replace the intermediate value $\mathsf{E}_w(W)$ by a random V, and replace the leakage $\mathsf{L}_{\mathsf{E}}(w, W)$ by $\mathcal{S}^{\mathsf{L}}(w, W, V)$. This yields G_{l+2}. In a similar vein to the above, we have

$$|\Pr[\mathcal{D}^{\mathsf{G}_{\ell+2}} \Rightarrow 1] - \Pr[\mathcal{D}^{\mathsf{G}_{\ell+1}} \Rightarrow 1]| \leq \Pr[d_{-1} = p_A \vee w = k_{ch}] + \varepsilon_{\mathsf{E}} + \varepsilon_{(p,2)\text{-}rsim} \leq \frac{1}{2^n} + \varepsilon_{\mathsf{E}} + \varepsilon_{(p,2)\text{-}rsim}.$$

We then replace the intermediate value $\mathsf{E}_U(V)$ by a random k_1, and replace the leakage $[\mathsf{L}_{\mathsf{E}}(U, V)]^p$ by $[\mathcal{S}^{\mathsf{L}}(U, V, k_1)]^p$. This yields $\mathsf{G}_{l+3} = \mathsf{G}_0^*$. We have

$$|\Pr[\mathcal{D}^{\mathsf{G}_{\ell+3}} \Rightarrow 1] - \Pr[\mathcal{D}^{\mathsf{G}_{\ell+2}} \Rightarrow 1]| \leq \Pr[d_\ell = p_A] + \varepsilon_{\mathsf{E}} + \varepsilon_{(p,2)\text{-}rsim} \leq \frac{1}{2^n} + \varepsilon_{\mathsf{E}} + \varepsilon_{(p,2)\text{-}rsim}.$$

Therefore,

$$|\Pr[(\mathcal{D}^{\mathsf{L}})^{\mathsf{G}_{\ell+3}} \Rightarrow 1] - \Pr[(\mathcal{D}^{\mathsf{L}})^{\mathsf{G}_{0,1}} \Rightarrow 1]| \leq \frac{2\ell+2}{2^n} + (\ell+4)(\varepsilon_{\mathsf{E}} + \varepsilon_{(p,2)\text{-}rsim}).$$

[5] When $j = 0$ the second part of the leakages is $\mathcal{S}^{\mathsf{L}}(s_0, p_B, w)$ without repeating.
[6] These xor actions are omitted when $j = 0$.
[7] When $i = 1$, these xor operations are omitted. More clearly, $\mathcal{D}^{\mathsf{L}'}$ uniformly samples w, and uses $[\mathcal{S}^{\mathsf{L}}(s_0, p_A, k_{ch})]^p$ and $\mathcal{S}^{\mathsf{L}}(s_0, p_B, w)$ as the traces of the 1st iteration.

Then, for j from 1 to ℓ, 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 $\mathsf{E}_{k_j}(p_A)$ and $\mathsf{E}_{k_j}(p_B)$ 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 $[\mathsf{L}_{\mathsf{E}}(k_j, p_A), \mathsf{L}_{\mathsf{E}}(k_j, p_B)]^p$, $\mathsf{L}_{\oplus}(\mathsf{E}_{k_j}(p_B), d_j)$, and $[\mathsf{L}_{\oplus}(\mathsf{E}_{k_j}(p_B), c_j)]^{p-1}$ by $[\mathcal{S}^{\mathsf{L}}(k_j, p_A, k_{j+1}), \mathcal{S}^{\mathsf{L}}(k_j, p_B, z_j)]^p$, $\mathsf{L}_{\oplus}(z_j, d_j)$, and $[\mathsf{L}_{\oplus}(z_j, c_j)]^{p-1}$. We again have $|\Pr[\mathcal{D}^{\mathsf{G}^*_j} \Rightarrow 1] - \Pr[\mathcal{D}^{\mathsf{G}^*_{j-1}} \Rightarrow 1]| \leq \varepsilon_{\mathsf{E}} + \varepsilon_{(p,2)\text{-}rsim}$. By these, we eventually obtain the ideal game G^*_l.

Gathering all the above, we have

$$\begin{aligned}
&|\Pr[\mathcal{D}^{\mathsf{G}^*_\ell} \Rightarrow 1] - \Pr[\mathcal{D}^{\mathsf{G}_{0,1}} \Rightarrow 1]| \\
=&(|\Pr[\mathcal{D}^{\mathsf{G}^*_\ell} \Rightarrow 1] - \Pr[\mathcal{D}^{\mathsf{G}^*_0} \Rightarrow 1]|) + (|\Pr[\mathcal{D}^{\mathsf{G}_{\ell+3}} \Rightarrow 1] - \Pr[\mathcal{D}^{\mathsf{G}_{0,1}} \Rightarrow 1]|) \\
\leq&\ell(\varepsilon_{\mathsf{E}} + \varepsilon_{(p,2)\text{-}rsim}) + \frac{2\ell+2}{2^n} + (\ell+4)(\varepsilon_{\mathsf{E}} + \varepsilon_{(p,2)\text{-}rsim}) \\
\leq&2(\ell+2)(\varepsilon_{\mathsf{E}} + \varepsilon_{(p,2)\text{-}rsim}) + \frac{2\ell+2}{2^n}
\end{aligned}$$

as claimed. □

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

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

$$\begin{aligned}
&|\Pr[\mathcal{A}^{\mathsf{L}}(\mathsf{LIFSM}_{s_0}(m^0)) \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{L}}(\mathsf{LIFSM}_{s_0}(m^1)) \Rightarrow 1]| \\
\leq&\frac{2\ell}{2^n} + \sum_{i=1}^{\ell} |\Pr[\mathcal{A}^{\mathsf{L}'}(\mathsf{LISEnc}^+_{s_{i-1}}(m^0_i)) \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{L}'}(\mathsf{LISEnc}^+_{s_{i-1}}(m^1_i)) \Rightarrow 1]|,
\end{aligned}$$

where $s_0, \ldots, s_{\ell-1}$ are chosen uniformly at random, the p_B value of $\mathsf{LISEnc}^+_{s_{i-1}}(m^b_i)$ is d_{i-1}, $d_0 = p_B$, $d_1, \ldots, d_{\ell-1}$ are chosen uniformly at random and $d_1, \ldots, d_{\ell-1} \neq p_A$, and m^0_i and m^1_i are the i-th block of m^0 and m^1 respectively. Here $q_r = (4\ell+5)q_{\mathcal{S}} + 2\ell$ and $t_r = (4\ell+5)(t_{\$} + t_{\mathcal{S}}) + 2\ell \cdot t_{\oplus}$, where t_{E}, $t_{\$}$, and $t_{\oplus}$ 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, $m_{h,i} := m^0_1 \| \ldots \| m^0_{\ell-i} \| m^1_{\ell-i+1} \| \ldots \| m^1_\ell$.

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

$$\begin{aligned}
&|\Pr[\mathcal{A}^{\mathsf{L}}(\mathsf{LIFSM}_{s_0}(m_{h,i-1})) \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{L}}(\mathsf{LIFSM}_{s_0}(m_{h,i})) \Rightarrow 1]| \\
\leq&\frac{2}{2^n} + |\Pr[\mathcal{A}^{\mathsf{L}'}(\mathsf{LISEnc}^+_{s_{\ell-i}}(m^0_{\ell-i+1})) \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{L}'}(\mathsf{LISEnc}^+_{s_{\ell-i}}(m^1_{\ell-i+1})) \Rightarrow 1]|.
\end{aligned}$$

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.,

$$m_{h,0} = m^0_1 \| \ldots \| m^0_{\ell-1} \| m^0_\ell, \text{ and } m_{h,1} = m^0_1 \| \ldots \| m^0_{\ell-1} \| m^1_\ell,$$

as example. For this, assuming a (q_l, t)-bounded adversary $\mathcal{A}^{\mathsf{L}}$ against $\mathsf{LIFSM}_{s_0}(m_{h,0})$ and $\mathsf{LIFSM}_{s_0}(m_{h,1})$, we build a $(q_l + p \cdot q_r, t + p \cdot t_r)$-bounded adversary $\mathcal{A}^{\mathsf{L}'}$ against LISEnc. In detail, $\mathcal{A}^{\mathsf{L}'}$ proceeds in six steps:

(1) $\mathcal{A}^{\mathsf{L}'}$ uniformly samples s_0, s_1, w, such that $s_1 \neq w$, and obtains the simulated leakage traces $[\mathcal{S}^{\mathsf{L}}(s_0, p_A, s_1)]^p, \mathcal{S}^{\mathsf{L}}(s_0, p_B,$
(2) for $j = 1, \ldots, \ell-3$, $\mathcal{A}^{\mathsf{L}'}$ 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 $[\mathcal{S}^{\mathsf{L}}(s_j, p_A, s_j), \mathcal{S}^{\mathsf{L}}(s_j, d_{j-1}, y_j)]^p$, computes $d_j \leftarrow y_j \oplus m^0_j$ and obtains the traces $\mathsf{L}_{\oplus}(y_j, m^0_j)$ and $[\mathsf{L}_{\oplus}(y_j, d_j)]^{p-1}$;

(3) $\mathcal{A}^{\mathsf{L}'}$ samples $d_{\ell-1} \neq p_A$, and computes $y_{\ell-1} \leftarrow d_{\ell-1} \oplus m^0_{\ell-1}$. $\mathcal{A}^{\mathsf{L}'}$ then sets the p_B value of its eavesdropper security challenger of LISEnc to $d_{\ell-1}$, and submits m^0_ℓ, and m^1_ℓ to the challenger to obtain the tuple

$$((k_{up}, c^b_{ch}), ([\mathsf{leak}_1, \mathsf{leak}_2]^p, \mathsf{L}_\oplus(y_{ch}, m^b_{ch}), [\mathsf{L}_\oplus(y_{ch}, c^b_{ch})]^{p-1}, [\mathcal{S}^{\mathsf{L}}(k_{pre}, p_A, k_{ch})]^p, k_{pre})$$

as outputs (which is produced by either $\mathsf{LISEnc}^+_{k_{ch}}(m^0_\ell)$ or $\mathsf{LISEnc}^+_{k_{ch}}(m^1_\ell)$).

(4) if $d_{\ell-3} = p_A$ then $\mathcal{A}^{\mathsf{L}'}$ sets $y_{\ell-2} \leftarrow k_{pre}$, otherwise samples $y_{\ell-2}$ such that $y_{\ell-2} \neq k_{pre}$. It then computes $d_{\ell-2} \leftarrow y_{\ell-2} \oplus m^0_{\ell-2}$. At this stage, $\mathcal{A}^{\mathsf{L}'}$ 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, $\mathcal{A}^{\mathsf{L}'}$ uses the leakage traces $[\mathcal{S}^{\mathsf{L}}(s_{\ell-2}, p_A, k_{pre}), \mathcal{S}^{\mathsf{L}}(s_{\ell-2}, d_{\ell-3}, y_{\ell-2})]^p$, $\mathsf{L}_\oplus(y_{\ell-2}, m^0_{\ell-2})$, $[\mathsf{L}_\oplus(y_{\ell-2}, d_{\ell-2})]^{p-1}$ $[\mathcal{S}^{\mathsf{L}}(s_{\ell-2}, p_A, k_{pre}), \mathcal{S}^{\mathsf{L}}(s_{\ell-2}, d_{\ell-3}, y_{\ell-2})]^p$, $\mathsf{L}_\oplus(y_{\ell-2}, m^0_{\ell-2})$, $[\mathsf{L}_\oplus(y_{\ell-2}, d_{\ell-2})]^{p-1}$ as the leakages of the $\ell-1$ th iteration, $[\mathcal{S}^{\mathsf{L}}(k_{pre}, p_A, k_{ch}), \mathcal{S}^{\mathsf{L}}(k_{pre}, d_{\ell-2}, y_{\ell-1})]^p$, $\mathsf{L}_\oplus(y_{\ell-1}, m^0_{\ell-1})$, $[\mathsf{L}_\oplus(y_{\ell-1}, d_{\ell-1})]^{p-1}$ as the leakages of the ℓ th iteration in the first pass;

(5) sets $s_{\ell+1} \leftarrow k_{up}$ and $d_\ell \leftarrow c^b_{ch}$, and uses $[\mathsf{leak}_1, \mathsf{leak}_2]^p$, $\mathsf{L}_\oplus(y_{ch}, m^b_{ch})$, and $[\mathsf{L}_\oplus(y_{ch}, c^b_{ch})]^{p-1}$ as the corresponding leakage traces (now the message absorbing phase of the first pass has been completed);

(6) Then, $\mathcal{A}^{\mathsf{L}'}$ takes $s_{\ell+1}$ and d_ℓ as the starting points, and emulates the remaining actions of LIFSM encrypting the message $m^0_1 \| \dots \| m^0_{\ell-1} \| m^b_{ch}$ to obtain $c^0_1 \| \dots \| c^0_{\ell-1} \| c^b_\ell$. Note that this requires $\mathcal{A}^{\mathsf{L}'}$ to uniformly sample z_ℓ and further compute $c^b_\ell \leftarrow z_\ell \oplus d_\ell = z_\ell \oplus c^b_{ch}$ and generates the traces $\mathsf{L}_\oplus(z_\ell, c^b_{ch})$ and $[\mathsf{L}_\oplus(z_\ell, c^b_\ell)]^{p-1}$ at the end of the second pass. Eventually, $\mathcal{A}^{\mathsf{L}'}$ serves the ciphertext $V \| c^0_1 \| \dots \| c^0_{\ell-1} \| c^b_\ell$ as well as all the generated simulated leakages to $\mathcal{A}^{\mathsf{L}}$, and outputs whatever $\mathcal{A}^{\mathsf{L}}$ outputs.

It can be seen that as long as $\mathcal{A}^{\mathsf{L}'}$ 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 $\mathcal{A}^{\mathsf{L}'}$ captures LISEnc encrypting m^0_ℓ or m^1_ℓ, the inputs to $\mathcal{A}^{\mathsf{L}}$ capture LIFSM encrypting $m^0_1 \| \dots \| m^0_{\ell-1} \| m^0_\ell$ or $m^0_1 \| \dots \| m^0_{\ell-1} \| m^1_\ell$. Moreover, $\mathcal{A}^{\mathsf{L}'}$ is indeed $(q_l + p \cdot q_r, t + p \cdot t_r)$-bounded if $\mathcal{A}^{\mathsf{L}}$ is (q_l, t)-bounded. These complete the proof. □

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

Lemma 5. *Let $E : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a $(2, t, \varepsilon_E)$-PRP, whose implementation has a leakage function L_E having $(q_{\mathcal{S}}, t_{\mathcal{S}}, q_l, t, \varepsilon_{(p,2)\text{-}rsim})$ $(p,2)$-R-simulatable leakages, and let $\mathcal{S}^{\mathsf{L}}$ be an appropriate $(q_{\mathcal{S}}, t_{\mathcal{S}})$-bounded leakage simulator. Then, for every pair of ℓ-block messages m^0 and m^1 and $(q_l - p \cdot q_r - q^*, t - p \cdot t_r - t^*)$-bounded adversary $\mathcal{A}^{\mathsf{L}}$, it holds*

$$\left| \Pr[\mathcal{A}^{\mathsf{L}}(\mathsf{LRFSM}_{s_0}(m^0)) \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{L}}(\mathsf{LRFSM}_{s_0}(m^1)) \Rightarrow 1] \right| \leq (6\ell + 8)(\varepsilon_E + \varepsilon_{(p,2)\text{-}rsim}) + \ell \cdot \varepsilon_{s\text{-}block} + \frac{6\ell + 4}{2^n},$$

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

Proof.

$$\begin{aligned}
&\left| \Pr[\mathcal{A}^{\mathsf{L}}(\mathsf{LRFSM}_{s_0}(m^0)) \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{L}}(\mathsf{LRFSM}_{s_0}(m^1)) \Rightarrow 1] \right| \\
\leq &\underbrace{\left| \Pr[\mathcal{A}^{\mathsf{L}}(\mathsf{LIFSM}_{s_0}(m^0)) \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{L}}(\mathsf{LIFSM}_{s_0}(m^1)) \Rightarrow 1] \right|}_{A} \\
&+ \underbrace{\sum_{b=0,1} \left| \Pr[\mathcal{A}^{\mathsf{L}}(\mathsf{LRFSM}_{s_0}(m^b)) \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{L}}(\mathsf{LIFSM}_{s_0}(m^b)) \Rightarrow 1] \right|}_{\leq 2\left((2\ell+4)(\varepsilon_{\mathsf{E}} + \varepsilon_{(p,2)\text{-}rsim}) + \frac{2\ell+2}{2^n} \right) \text{ (by Lemma 3)}}.
\end{aligned}$$

For the involved term A, by Lemma 4 there exists a $(q_l - q^*, t - t^*)$-bounded adversary $\mathcal{A}^{\mathsf{L}'}$ that satisfies

$$\begin{aligned} A &\le \frac{2\ell}{2^n} + \sum_{i=1}^{\ell} |\Pr[\mathcal{A}^{\mathsf{L}'}(\mathsf{LISEnc}^+_{s_{i-1}}(m_i^0)) \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{L}'}(\mathsf{LISEnc}^+_{s_{i-1}}(m_i^1)) \Rightarrow 1]| \\ &\le \underbrace{\sum_{i=1}^{\ell} |\Pr[\mathcal{A}^{\mathsf{L}'}(\mathsf{LRSEnc}^+_{s_{i-1}}(m_i^0)) \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{L}'}(\mathsf{LRSEnc}^+_{s_{i-1}}(m_i^1)) \Rightarrow 1]|}_{\le \ell \cdot \varepsilon_{s\text{-}block} \text{ (by (1))}} \\ &\quad + \frac{2\ell}{2^n} + 2\ell(\varepsilon_{\mathsf{E}} + \varepsilon_{(p,2)\text{-}rsim}) \text{ (by Lemma 2).} \end{aligned}$$

The claim thus follows. □

Theorem 3, the CCAML2 security of FEMALE, could then be derived from Lemma 5. The argument is in the next subsection.

G.2 Proof of Theorem 9

We introduce a notion of *trivial decryption queries*: if the adversary queries $\mathsf{Enc}_k(N, A, M) \to C$ and makes the decryption query $\mathsf{Dec}_k(N, A, C)$ 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 G_0 as the game $\mathsf{PrivK}^{\mathsf{CCAML2},0}_{\mathcal{A}^{\mathsf{L}},\mathsf{FEMALE}}$, and G^*_0 as the game $\mathsf{PrivK}^{\mathsf{CCAML2},1}_{\mathcal{A}^{\mathsf{L}},\mathsf{FEMALE}}$. We show that they are indistinguishable from two games G_1 and 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 G_1, which is obtained from G_0 by replacing all the occurrences of E^*_k and its inverse function by a truly tweakable random permutation $\widetilde{P}$ and its inverse. To upper bound $|\Pr[\mathsf{G}_1 \Rightarrow 1] - \Pr[\mathsf{G}_0 \Rightarrow 1]|$, we build a $(2q_e + 2q_d + 2q_m, t_{\mathcal{B}})$-bounded adversary $\mathcal{B}_{STPRP}$ against the STPRP security of E^*. The challenger $\mathcal{B}_{STPRP}$ picks all the necessary constants itself, and emulates the encryption and decryption oracles to interact with $\mathcal{A}^{\mathsf{L}}$ as follows. On each query made by $\mathcal{A}^{\mathsf{L}}$, $\mathcal{B}_{STPRP}$ emulates all the actions described by FEMALE, except for the computations involving the tweakable permutation E^*, which is replaced by calls to its own tweakable permutation oracle $\mathcal{O}$ (which is either E^*_k or $\widetilde{P}$). When $\mathcal{A}^{\mathsf{L}}$ outputs its guess bit, $\mathcal{B}_{STPRP}$ returns that bit as its own guess. It's clear that depending on whether $\mathcal{B}_{STPRP}$ is interacting with E^*_k or $\widetilde{P}$, $\mathcal{A}^{\mathsf{L}}$ is playing G_0 or G_1. Therefore, any difference between $\Pr[\mathsf{G}_1 \Rightarrow 1]$ and $\Pr[\mathsf{G}_0 \Rightarrow 1]$ leads to the same difference in $\mathcal{B}_{STPRP}$ distinguishing E^*_k from $\widetilde{P}$. During the emulation, the simulated FEMALE scheme receives $q_e + q_m$ encryption queries and q_d decryption queries. Therefore, $\mathcal{B}_{STPRP}$ is a $(2q_e + 2q_d + 2q_m, t')$-bounded adversary against E^*, making at most $2q_e + 2q_d + 2q_m$ queries to $\mathcal{O}$ and running in time at most $t + (q_e + q_d + q_m)t_{1-pass} \le t'$. Thus $|\Pr[\mathsf{G}_1 \Rightarrow 1] - \Pr[\mathsf{G}_0 \Rightarrow 1]| \le \varepsilon_{\mathsf{E}^*}$ follows from the assumption that E^* is a $(2q_e + 2q_d + 2q_m, t', \varepsilon_{\mathsf{E}^*})$-secure TPRP.

Similarly, we replace E^* by $\widetilde{P}$ to turn G^*_0 into G^*_1, which also introduces a gap of $\varepsilon_{\mathsf{E}^*}$.

We then prove

$$\left|\Pr[(\mathcal{A}^{\mathsf{L}})^{\mathsf{G}_1} \Rightarrow 1] - \Pr[(\mathcal{A}^{\mathsf{L}})^{\mathsf{G}^*_1} \Rightarrow 1]\right| \le \frac{q_e + q_m - 1}{2^n} + \varepsilon_{cr} + \varepsilon_{pr} + \textstyle\sum_{i=1}^{q_m} \varepsilon_{\mathsf{FEMALE}\text{-}eav}(\ell_i),$$

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

$$(Nc_1, Ac_1, Mc_1^0, Mc_1^1), \ldots, (Nc_{q_m}, Ac_{q_m}, Mc_{q_m}^0, Mc_{q_m}^1).$$

Then, we note that in G_1, the q_m messages being encrypted by the challenge encryption oracle are $Mc_1^0, \ldots, Mc_{q_m}^0$, while those encrypted in G^*_1 are $Mc_1^1, \ldots, Mc_{q_m}^1$. We use q_m hops to replace $Mc_1^0, \ldots, Mc_{q_m}^0$ by $Mc_1^1, \ldots, Mc_{q_m}^1$

in turn, to show that G_1 can be transited to G_1^*. For convenience, we define $\mathsf{G}_{2,0} = \mathsf{G}_1$, and define a sequence of games

$$\mathsf{G}_{2,1}, \mathsf{G}_{2,2}, \ldots, \mathsf{G}_{2,q_m},$$

such that in the i-th system $\mathsf{G}_{2,i}$, the first i messages processed by the challenge encryption oracle are $Mc_1^0, \ldots, Mc_i^0$, while the remaining $q_m - i$ messages being processed are $Mc_{i+1}^1, \ldots, Mc_{q_m}^1$. It can be seen actually $\mathsf{G}_{2,q_m} = \mathsf{G}_1^*$.

We then show that for $i = 1, \ldots, q_m$, $\mathsf{G}_{2,i-1}$ and $\mathsf{G}_{2,i}$ are indistinguishable for $\mathcal{A}^{\mathsf{L}}$. For this, from $\mathcal{A}^{\mathsf{L}}$ we build an adversary $\mathcal{A}^{\mathsf{L}'}$, such that if $\mathcal{A}^{\mathsf{L}}$ distinguishes $\mathsf{G}_{2,i-1}$ and $\mathsf{G}_{2,i}$ then $\mathcal{A}^{\mathsf{L}'}$ breaks the eavesdropper security of LRFSM. In detail, $\mathcal{A}^{\mathsf{L}'}$ keeps two pairs of tables (P_0, P_0^{-1}) and (P_1, P_1^{-1}) to simulate the tweakable random permutation $\widetilde{P}$ (via lazy sampling). And it runs $\mathcal{A}^{\mathsf{L}}$, and reacts as follows:

- Upon an encryption query (N_i, A_i, M_i) from $\mathcal{A}^{\mathsf{L}}$, $\mathcal{A}^{\mathsf{L}'}$ first computes $R_i \leftarrow \mathsf{H}(0\|N_i\|A_i)$. Then:
  - if $R_i \notin P_0$, $\mathcal{A}^{\mathsf{L}'}$ samples $s_0^{(i)} \xleftarrow{\$} \{0,1\}^n \backslash P_0^{-1}$, sets $P_0(R_i) \leftarrow s_0^{(i)}$ and $P_0^{-1}(s_0^{(i)}) \leftarrow R_i$, and then runs $\mathsf{LRFSM}_{s_0^{(i)}}(M_i)$ to get the ciphertext (V_i, c_i) and leakages. $\mathcal{A}^{\mathsf{L}'}$ then computes $h_i \leftarrow \mathsf{H}(1\|R_i\|V\|c_i)$ and $T_i \leftarrow P_1(h_i)$ (if $h_i \notin P_1$ then $\mathcal{A}^{\mathsf{L}'}$ defines $P_1(h_i)$ as a value newly sampled from $\{0,1\}^n \backslash P_1^{-1}$). Finally, $\mathcal{A}^{\mathsf{L}'}$ returns the outputs (V_i, c_i, T_i) and the leakages to $\mathcal{A}^{\mathsf{L}}$;
  - if $R_i \in P_0$, $\mathcal{A}^{\mathsf{L}'}$ simply runs $\mathsf{LRFSM}_{P_0(R_i)}(M_i)$, performs the tag generation action $h_i \leftarrow \mathsf{H}(1\|R_i\|V_i\|c_i)$ and $T_i \leftarrow P_1(h_i)$ on the obtained V_i and c_i, and returns (V_i, c_i, T_i) and the leakages to $\mathcal{A}^{\mathsf{L}}$.
- Upon a trivial decryption query (N_j, A_j, C_j) from $\mathcal{A}^{\mathsf{L}}$ (cf. the beginning of this subsection for the meaning of "trivial"), $\mathcal{A}^{\mathsf{L}'}$ computes $R_j \leftarrow \mathsf{H}(0\|N_j\|A_j)$. Since (N_j, A_j, C_j) is trivial, $R_j \in P_0$ necessarily holds. Therefore, $\mathcal{A}^{\mathsf{L}}$ parses $C_j = (V_j, c_j, T_j)$, runs $\mathsf{LRFSM.Dec}(P_0(R_j), c_j)$, and relays the outputs to $\mathcal{A}^{\mathsf{L}}$.
- Upon a non-trivial decryption query (N_j, A_j, C_j) from $\mathcal{A}^{\mathsf{L}}$, $\mathcal{A}^{\mathsf{L}'}$ parses $C_j = (V_j, c_j, T_j)$, and computes $R_j \leftarrow \mathsf{H}(0\|N_j\|A_j)$ and $h_j \leftarrow \mathsf{H}(1\|R_j\|V_j\|c_j)$. Then,
  - if $T_j \notin P_1^{-1}$, $\mathcal{A}^{\mathsf{L}'}$ samples $h_j^* \xleftarrow{\$} \{0,1\}^n \backslash P_1$, and sets $P_1(h_j^*) \leftarrow T_j$ and $P_1^{-1}(T_j) \leftarrow h_j^*$;
  - if $T_j \in P_1^{-1}$, $\mathcal{A}^{\mathsf{L}'}$ simply sets $h_j^* \leftarrow P_1^{-1}(T_j)$.

  Now $\mathcal{A}^{\mathsf{L}'}$ aborts if $h_j = h_j^*$ (this type of abortion is defined as BadDec), and returns $(\perp, h_j^*)$ to $\mathcal{A}^{\mathsf{L}}$.
- Upon $\mathcal{A}^{\mathsf{L}}$ submitting the j-th challenge tuple $(Nc_j, Ac_j, Mc_j^0, Mc_j^1)$, $\mathcal{A}^{\mathsf{L}'}$ computes $Rc_j \leftarrow \mathsf{H}(0\|Nc_j\|Ac_j)$. Then, if $Rc_j \in P_0$, $\mathcal{A}^{\mathsf{L}'}$ aborts (this type of abortion is defined as $\mathsf{BadChall}^+$); otherwise, its action depends on j:
  - When $j < i$, it simply encrypts Mc_j^0 and returns. In detail, $\mathcal{A}^{\mathsf{L}'}$ samples $sc_0^{(j)} \xleftarrow{\$} \{0,1\}^n \backslash P_0^{-1}$, sets $P_0(Rc_j) \leftarrow sc_0^{(j)}$ and $P_0^{-1}(sc_0^{(j)}) \leftarrow Rc_j$, and then runs $\mathsf{LRFSM}_{P_0(Rc_j)}(Mc_j^0)$, performs the tag generation action $hc_j \leftarrow \mathsf{H}(1\|Rc_j\|Vc_j\|cc_j)$ and $Tc_j \leftarrow P_1(hc_j)$ on the obtained Vc_j and cc_j, and returns (Vc_j, cc_j, Tc_j) and the leakages to $\mathcal{A}^{\mathsf{L}}$.
  - When $j = i$, it relays Mc_j^0 and Mc_j^1 to its eavesdropper security challenger to obtain (Vc_j^b, cc_j^b) and leakages leak_{enc} and $[\mathsf{leak}_{dec}]^{p-1}$, and then computes $Tc_j \leftarrow P_1(\mathsf{H}(1\|Rc_j\|Vc_j^b\|cc_j^b))$ (possibly defines a new entry in P_1) and returns $C_{ch}^b = (Vc_j^b, cc_j^b, Tc_j)$ to $\mathcal{A}^{\mathsf{L}}$. Note that this means the relation $P_0(Rc_i) = s_0^{ch}$ is implicitly fixed, where s_0^{ch} is the secret key generated inside the eavesdropper security challenger;
  - When $j > i$, it simply encrypts Mc_j^1 and returns. The details are similar to the described case $j < i$.
- Upon $\mathcal{A}^{\mathsf{L}}$ making the λ-th query to $\mathsf{L}_{\mathsf{decch}}(j)$ $(1 \le \lambda \le p-1)$,
  - When $j \neq i$, $\mathcal{A}^{\mathsf{L}'}$ performs a corresponding decryption process (decrypting (Nc_j, Ac_j, Cc_j^0) when $j < i$, and (Nc_j, Ac_j, Cc_j^1) when $j > i$) and returns the obtained leakages to $\mathcal{A}^{\mathsf{L}}$;
  - When $j = i$, $\mathcal{A}^{\mathsf{L}'}$ simply returns the λ-th trace in the vector $[\mathsf{leak}_{dec}]^{p-1}$ as the answer.

Moreover, whenever new entries are added to P_0, $\mathcal{A}^{\mathsf{L}'}$ aborts if $Rc_j \in P_0$ (so that the implicit relation $P_0(Rc_i) = s_0^{ch}$ never causes inconsistency).

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

(i) BadDec never occurs, and
(ii) $\mathsf{BadChall}^+$ never occurs, and

(iii) $\mathsf{BadChall}^-$ never occurs: it never holds $s_0^{ch} \in P_0^{-1}$. Recall that entries in P_0^{-1} are the initial session keys generated by $\mathcal{A}^{\mathsf{L}'}$ itself. This is crucial because the relation $P_0(Rc_i) = s_0^{ch}$ is implicitly fixed (as mentioned), and thus $s_0^{ch} \in P_0^{-1}$ would cause inconsistency.

On the other hand, besides running $\mathcal{A}^{\mathsf{L}}$, $\mathcal{A}^{\mathsf{L}'}$ samples at most $2(q_e + q_d + q_m)$ random values (to emulate $\widetilde{P}$) and internally processes $q_e + q_d + q_m$ queries. Therefore, the running time of $\mathcal{A}^{\mathsf{L}'}$ is at most $t_{\mathcal{A}^{\mathsf{L}'}} \leq t + (q_e + q_d + q_m)(2t_{\$} + t_{1-pass})$ (while $\mathcal{A}^{\mathsf{L}'}$ makes q_l queries to L, which is the same as $\mathcal{A}^{\mathsf{L}}$).

We need to bound the probabilities of the defined events to complete the proof. They are as follows.

Lemma 6. $\Pr[\mathit{BadDec} \vee \mathit{BadChall}^+] \leq \varepsilon_{cr} + \varepsilon_{pr}$.

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

- CollH: during the interaction between $\mathcal{A}^{\mathsf{L}'}$ and the eavesdropper security challenger, there appears two calls to $\mathsf{H}(x)$ and $\mathsf{H}(x')$ such that $x \neq x'$ while $\mathsf{H}(x) = \mathsf{H}(x')$;
- $\mathsf{PreimgH}$: during the interaction between $\mathcal{A}^{\mathsf{L}'}$ and the eavesdropper security challenger, there appears a decryption query $(N_j, A_j, (V_j, c_j, T_j))$ such that $T_j \notin P_1^{-1}$ before this query is made, yet it holds $\mathsf{H}(1\|R_j\|V_j\|c_j) = P_1^{-1}(T_j)$ after $P_1^{-1}(T_j)$ is defined later $(R_j = \mathsf{H}(0\|N_j\|A_j))$.

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

On the other hand, if $\mathsf{PreimgH}$ happens, then from $\mathcal{A}^{\mathsf{L}'}$ we are able to build a preimage adversary against H. To this end, note that by the definition of $\mathsf{PreimgH}$, if it happens with respect to a decryption query $(N_j, A_j, (V_j, c_j, T_j))$, then $T_j \notin P_1^{-1}$ before this query is made. According to the description of $\mathcal{A}^{\mathsf{L}'}$, $P_1^{-1}(T_j)$ will be defined to a randomly sampled value, and thus $\mathsf{H}(1\|R_j\|V_j\|c_j) = P_1^{-1}(T_j)$ exactly fits into the definition of range-oriented preimage resistance. Therefore, we could use an adversary $\mathcal{B}_{pr}$ to emulate the interaction between $\mathcal{A}^{\mathsf{L}'}$ and the eavesdropper security challenger, and if $\mathsf{PreimgH}$ happens then $\mathcal{B}_{pr}$ turns out a preimage adversary against the hash function $\mathsf{H}(1\|\cdot)$. Since $\mathcal{B}_{pr}$'s running time is also at most t', and there are at most q_d such decryption queries, we have $\Pr[\mathsf{PreimgH}] \leq \varepsilon_{pr}$ by the assumption that H is $(q_d, t', \varepsilon_{pr})$-range-oriented preimage resistance. This further yields $\Pr[\mathsf{CollH} \vee \mathsf{PreimgH}] \leq \varepsilon_{cr} + \varepsilon_{pr}$.

We now prove $\Pr[\mathsf{BadDec} \vee \mathsf{BadChall}^+ \mid \neg(\mathsf{CollH} \vee \mathsf{PreH})] = 0$ to complete the proof. Assume that 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: T_i appears in a response to some previous encryption query (N_j, A_j, M_j) where j is the smallest index satisfying this property. Assume that the corresponding response was $C_j = (V_j, c_j, T_j)$. Then we further distinguish two subcases:
  - Case 1.a: $(N_j, A_j, V_j, c_j) = (N_i, A_i, V_i, c_i)$. This means $R_j = R_i$. Then it has to be $T_j \neq T_i$, as otherwise $(N_i, A_i, (V_i, c_i, T_i))$ is trivial. Then since we have $\mathsf{H}(1\|R_j\|V_j\|c_j) = P_1^{-1}(T_j)$ and P_1 is a permutation, we cannot further have $\mathsf{H}(1\|R_i\|V_i\|c_i) = P_1^{-1}(T_i)$;
  - Case 1.b: $(N_j, A_j, V_j, c_j) \neq (N_i, A_i, V_i, c_i)$. 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 $\neg\mathsf{CollH}$, which by $\neg\mathsf{CollH}$ further implies $\mathsf{H}(1\|R_j\|V_j\|c_j) = P_1^{-1}(T_j) \neq \mathsf{H}(1\|R_i\|V_i\|c_i)$.

(ii) Case 2: T_i does not appear in any response to earlier encryption query. Then the entry $P_1^{-1}(T_i)$ was necessarily defined to a randomly sampled value. Therefore, $\mathsf{H}(1\|R_i\|V_i\|c_i) = P_1^{-1}(T_i)$ implies the occurrence of $\mathsf{PreimgH}$.

Finally, conditioned on that BadDec does not occur, entries can only be added to P_0 upon $\mathcal{A}^{\mathsf{L}}$ making encryption queries. Therefore, if $\mathsf{BadChall}^+$ happens, then there necessarily exists an encryption query (N, A, M) such that $\mathsf{H}(0\|N\|A) = \mathsf{H}(0\|Nc_i\|Ac_i)$. According to the requirements of the $\mathsf{CCAML2}$ security game, it has to be $(N, A) \neq (Nc_i, Ac_i)$; therefore, $\mathsf{H}(0\|N\|A) = \mathsf{H}(0\|Nc_i\|Ac_i)$ would contradict either $\neg\mathsf{BadDec}$ or $\neg\mathsf{CollH}$. Thus the claim. □

Lemma 7. $\Pr[\mathit{BadChall}^{-} \mid \neg \mathit{BadDec}] \leq \frac{q_e+q_m-1}{2^n}$.

Proof. The bound follows from the following observations:

(i) conditioned on $\neg\mathsf{BadDec}$, $\mathcal{A}^{\mathsf{L}'}$ only samples new initial session keys upon $\mathcal{A}^{\mathsf{L}}$ makes new encryption queries, and
(ii) except for the initial session key for the i-th challenge encryption, all the other initial session keys are randomly sampled by $\mathcal{A}^{\mathsf{L}'}$.

Thus we have $\Pr[\mathsf{BadChall}^{-} \mid \neg\mathsf{BadDec}] \leq \frac{q_e+q_m-1}{2^n}$. □

By all the above, define

$$\mathsf{Bad} = \mathsf{BadDec} \vee \mathsf{BadChall}^{+} \vee \mathsf{BadChall}^{-},$$

then we have

$$\Pr[(\mathcal{A}^{\mathsf{L}})^{\mathsf{G}_{2,i}} \Rightarrow 1] - \Pr[(\mathcal{A}^{\mathsf{L}})^{\mathsf{G}_{2,i-1}} \Rightarrow 1] \leq \Pr[(\mathcal{A}^{\mathsf{L}})^{\mathsf{G}_{2,i}} \Rightarrow 1 \wedge \mathsf{Bad}] - \Pr[(\mathcal{A}^{\mathsf{L}})^{\mathsf{G}_{2,i-1}} \Rightarrow 1 \wedge \mathsf{Bad}] + \varepsilon_{\mathsf{FEMALE}\text{-}eav}(\ell_i).$$

This means

$$\begin{aligned}
|\Pr[(\mathcal{A}^{\mathsf{L}})^{\mathsf{G}_1^*} \Rightarrow 1] - \Pr[(\mathcal{A}^{\mathsf{L}})^{\mathsf{G}_1} \Rightarrow 1]| &\leq \Pr[(\mathcal{A}^{\mathsf{L}})^{\mathsf{G}_{2,q_m}} \Rightarrow 1] - \Pr[(\mathcal{A}^{\mathsf{L}})^{\mathsf{G}_{2,0}} \Rightarrow 1] \\
&\leq \sum_{i=1}^{q_m} \left(\Pr[(\mathcal{A}^{\mathsf{L}})^{\mathsf{G}_{2,i}} \Rightarrow 1] - \Pr[(\mathcal{A}^{\mathsf{L}})^{\mathsf{G}_{2,i-1}} \Rightarrow 1] \right) \\
&\leq \frac{q_e + q_m - 1}{2^n} + \varepsilon_{cr} + \varepsilon_{pr} + \textstyle\sum_{i=1}^{q_m} \varepsilon_{\mathsf{FEMALE}\text{-}eav}(\ell_i).
\end{aligned}$$

These complete the proof.

Influence of Empty Message. It's meaningless to use empty message for the CCAML2 challenge message, as otherwise it's not possible to pick two distinct challenges. Therefore, in the 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, BadDec) remains valid; for such non-challenge encryptions, the adversary $\mathcal{A}^{\mathsf{L}'}$ remains able to simulate the corresponding actions.

G.3 Proof of Theorem 4

The claim can be established by some intermediate results obtained during the proof of Theorem 3. To see this, let's revisit these results:

- First, define G_0 as the real CIML2 security game between the adversary and FEMALE. Then replacing E_k^* by a tweakable random permutation $\widetilde{P}$, we obtain a game G_1, with a gap of $\varepsilon_{\mathsf{E}^*}$;
- Second, as long as neither of the two bad events CollH and PreimgH happens during the execution of G_1, every non-trivial decryption query would yield $\perp$ as the answer. And we have $\Pr[\mathsf{CollH} \vee \mathsf{PreimgH}] \leq \varepsilon_{cr} + \varepsilon_{pr}$.

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.

G.4 Proof of Theorem 5

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

- $\mathcal{A}$ never repeats queries, and
- $\mathcal{A}$ 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 $N_1\|A_1, \ldots, N_{q_e}\|A_{q_e}$ 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 $\mathcal{A}$ interacts with Enc_k and Dec_k, and ending with random-and-invalid game, named G_{q_e+1}, where $\mathcal{A}$ interacts with \$ and $\perp$. Then, using the adversary $\mathcal{A}$ 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 $\mathcal{A}$ outputs 1 at the end of G_i.

We start from $\mathsf{G}_0 = \mathcal{A}^{\mathsf{Enc}_k, \mathsf{Dec}_k}$. The first two steps are similar to those appeared in the proof of Theorem 3:

(i) we first replace all the occurrences of E_k^* by a tweakable random permutation $\widetilde{P}$ to obtain the first intermediate game $\mathsf{G}_{0,1}$. And we introduce two abort conditions CollH and $\mathsf{PreimgH}$ into $\mathsf{G}_{0,1}$: the former is fulfilled if there exists two calls to $\mathsf{H}(x)$ and $\mathsf{H}(x')$ such that $x \neq x'$ yet $\mathsf{H}(x) = \mathsf{H}(x')$, while the latter is fulfilled if there exists a decryption query (N_j, A_j, C_j) with $C_j = (V_j, c_j, T_j)$ such that T_j never appeared in answers to earlier encryption queries and $\mathsf{H}(1\|R_j\|V_j\|c_j) = (\widetilde{P}^1)^{-1}(T_j)$. It can be seen $\Pr[\mathsf{G}_{0,1} \text{ aborts}] \leq \varepsilon_{cr} + \varepsilon_{pr}$, 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 $|\Pr[E_0] - \Pr[E_{0,1} \mid \mathsf{G}_{0,1} \text{ does not abort}]| \leq \varepsilon_{\mathsf{E}^*}$, otherwise from $\mathcal{A}$ we can build an STPRP adversary $\mathcal{B}_1$ against E_k^*, and $\mathcal{B}_1$ 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,ℓ_0}, which is the same as $\mathsf{G}_{0,1}$ except that all decryption queries are simply answered by $\perp$. In a similar vein to the proof of Theorem 3, we know that if neither of the two games abort, then they would have no difference. Therefore, $\Pr[E_{0,1} \mid \mathsf{G}_{0,1} \text{ does not abort}] = \Pr[E^*_{0,\ell_0} \mid \mathsf{G}^*_{0,\ell_0} \text{ does not abort}]$.

We then consider the encryption queries in turn, and use a sequence of games

$$\begin{aligned}
&\mathsf{G}_{1,-1} = \mathsf{G}^*_{0,\ell_0}, \mathsf{G}_{1,1}, \ldots, \mathsf{G}_{1,\ell_1}, \mathsf{G}_{1,\ell_1+1}, \mathsf{G}_{1,\ell_1+2} = \mathsf{G}^*_{1,0}, \mathsf{G}^*_{1,1}, \ldots, \mathsf{G}^*_{1,\ell_1}, \mathsf{G}_{2,-1},\\
&\quad\ldots\\
&\mathsf{G}_{q_e,-1} = \mathsf{G}^*_{q_e-1,\ell_{q_e-1}}, \ldots, \mathsf{G}_{q_e,\ell_{q_e}}, \mathsf{G}_{q_e,\ell_{q_e}+1}, \mathsf{G}_{q_e,\ell_{q_e}+2} = \mathsf{G}^*_{q_e,0}, \mathsf{G}^*_{q_e,1}, \ldots, \mathsf{G}^*_{q_e,\ell_{q_e}},
\end{aligned}$$

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

Non-reuse Case: (N_i, A_i) never appeared before. Then conditioned on $\neg\mathsf{CollH}$, 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 $\widetilde{P}^0$ is a permutation. In the first phase of the transition, for $j = 0, \ldots, \ell_i + 1$, we consider the game $\mathsf{G}_{i,j-1}$, and replace all the subsequently appearing calls of the form $\mathsf{E}_{s_j^{(i)}}(x)$ by calls to a new random permutation $P_{i,j}(x)$. This yields $\mathsf{G}_{i,j}$. It holds $|\Pr[E_{i,j}] - \Pr[E_{i,j-1}]| \leq \varepsilon_{\mathsf{E}}$, otherwise a PRP adversary $\mathcal{B}_{\mathsf{E}}$ against E could be built by emulating the common part of $\mathsf{G}_{i,0}$ and $\mathsf{G}_{i,-1}$. In detail, $\mathcal{B}_{\mathsf{E}}$ interacts with $\mathcal{A}$, and simulates the previously appeared random permutations $P_{1,0}, P_{1,1}, \ldots, P_{i,0}, \ldots, P_{i,j-1}$ by lazy sampling, till the intermediate values $s_j^{(i)}$ and $d_{j-1}^{(i)}$ being computed.[8] Then, $\mathcal{B}_{\mathsf{E}}$ supplies p_A and $d_{j-1}^{(i)}$ to its challenge oracle $\mathcal{O}$ (which is either E_k for a secret key k or the random permutation $P_{i,j}$), and uses the obtained $s_{j+1}^{(i)} \leftarrow \mathcal{O}(p_A)$ 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, $\mathcal{B}_{\mathsf{E}}$ uses $P_{i,0}, \ldots, P_{i,j-1}, \mathcal{O}$ for the first $j+1$ calls to E. To emulate these, $\mathcal{B}_{\mathsf{E}}$ makes at most $2q_e$ queries to $\mathcal{O}$, and the running time of $\mathcal{B}_{\mathsf{E}}$ is clearly at most t'. Thus $\mathcal{B}_{\mathsf{E}}$'s advantage does not exceed ε_{E}.

Then, consider the game G_{i,ℓ_i+1}, and let $w^{(i)} = P_{i,0}(p_B)$. We replace all the subsequently appearing calls of the form $\mathsf{E}_{w^{(i)}}(x)$ by calls to a new random permutation $P_{i,\ell_i+2}(x)$, to obtain the game G_{i,ℓ_i+2}. Similarly

[8] Similarly to the proof of Lemma 3, when $j = 0$ we take $d_{-1}^{(i)} = p_B$ and $w^{(i)}$ as the corresponding output to ease the language.

to the above, $|\Pr[E_{i,\ell+2}] - \Pr[E_{i,\ell+1}]| \leq \varepsilon_{\mathsf{E}}$. We further replace all the subsequently appearing calls of the form $\mathsf{E}_{U^{(i)}}(x)$ by calls to a new random permutation $P_{i,\ell_i+3}(x)$, to obtain the game $\mathsf{G}_{i,\ell_i+3} = \mathsf{G}^*_{i,0}$, with $|\Pr[E_{i,\ell+3}] - \Pr[E_{i,\ell+2}]| \leq \varepsilon_{\mathsf{E}}$. By this, the newly derived initial key $k_1^{(i)} = P_{i,\ell_i+3}(V^{(i)})$ for the one-time encryption is uniform.

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

It can be seen that in the last game G^*_{i,ℓ_i}, 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 α is the smallest index such that $(N_\alpha, A_\alpha) = (N_i, A_i)$. We further distinguish two cases:

- Case 1: the length ℓ_i 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 $\mathsf{G}_{i,-1}$, 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 $P_{\alpha,0}, \ldots, P_{\alpha,\ell_i+3}$. Therefore, we simply let $\mathsf{G}^*_{i,0} = \mathsf{G}_{i,-1}$.
- Case 2: M_i is the longest message among the messages encrypted with the pair (N_i, A_i). Assume that the calls $\mathsf{E}_{s_0^{(\alpha)}}, \ldots, \mathsf{E}_{s_\beta^{(\alpha)}}$ have been replaced by random permutations $P_{\alpha,0}, \ldots, P_{\alpha,\beta}$ (note that $s_\beta^{(\alpha)} = U^{(j)}$ for some $j < i$), and let $s_{\beta+1}^{(i)} = P_{\alpha,\beta}(p_A)$. Define $\mathsf{G}_{i,\beta} = \mathsf{G}_{i,\beta}$. Then, for $j = \beta + 1, \ldots, \ell_i + 2$, we replace all the subsequently appearing calls of the form $\mathsf{E}_{s_j^{(i)}}(x)$ (taking $s_{\ell_i+2}^{(i)} = U^{(i)}$) by calls to a new random permutation $P_{i,j}(x)$. This yields $\mathsf{G}_{i,j+1}$. For convenience we also write $P_{\alpha,j}$ for the permutation $P_{i,j}$. These steps eventually yield the game $\mathsf{G}_{i,\ell_i+3} = \mathsf{G}^*_{i,0}$.

Unlike the non-reuse case, we introduce three abort conditions into the game $\mathsf{G}^*_{i,0}$. In detail, $\mathsf{G}^*_{i,0}$ 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 $N_j \| A_j = N_i \| A_i$ and $\ell_i = \ell_j$ for the j-th encryption query (N_j, A_j, M_j), and there exists an index γ such that $m_1^{(i)} \| \ldots \| m_\gamma^{(i)} \neq m_1^{(j)} \| \ldots \| m_\gamma^{(j)}$, yet $d_\gamma^{(i)} = d_\gamma^{(j)}$;
- the event CollW_i, which happens if there exists $j < i$ such that $N_j \| A_j = N_i \| A_i$ for (N_j, A_j, M_j), and $W^{(i)} = W^{(j)}$.

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

$$W^{(i)} = P_{\alpha,\ell_i+1}(d_{\ell_i}^{(i)}), \text{ and } W^{(j)} = P_{\alpha,\ell_j+1}(d_{\ell_j}^{(j)}),$$

with P_{α,ℓ_i+1} and P_{α,ℓ_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 $P_{\alpha,\ell_i+1}(d_{\ell_i}^{(i)})$ and $P_{\alpha,\ell_j+1}(d_{\ell_j}^{(j)})$ 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 $P_{1,\ell_1+2}, \ldots, P_{i-1,\ell_{i-1}+2}$, thus also independent of these two values.

Therefore, it holds

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

When $N_j\|A_j = N_i\|A_i$ 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 $\mathcal{A}$ does not repeat queries. Wlog assume that γ is the smallest index such that $m_\gamma^{(j)} \neq m_\gamma^{(i)}$. Then it can be seen $d_{\gamma-1}^{(j)} = d_{\gamma-1}^{(i)}$. Moreover, $N_j\|A_j = N_i\|A_i$ implies $s_0^{(i)} = s_0^{(j)}, \ldots, s_\gamma(i) = s_\gamma^{(j)}$, and thus the first γ random permutations used for processing the two queries are the same ones. Assume that α is the smallest index such that $(N_\alpha, A_\alpha) = (N_i, A_i)$. Then by our convention, these γ random permutations are $P_{\alpha,0}, \ldots, P_{\alpha,\gamma}$. Then it can be seen $d_\gamma^{(j)} \neq d_\gamma^{(i)}$, since $m_\gamma^{(j)} \neq m_\gamma^{(i)}$ implies

$$P_{\alpha,\gamma}(d_{\gamma-1}^{(j)}) \oplus m_\gamma^{(j)} \neq P_{\alpha,\gamma}(d_{\gamma-1}^{(i)}) \oplus m_\gamma^{(i)}.$$

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

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

Following the same line, we obtain $\Pr[d_{\gamma+2}^{(j)} = d_{\gamma+2}^{(i)}] \leq \frac{1}{2^n}$, …, $\Pr[d_{\ell_j}^{(j)} = d_{\ell_i}^{(i)}] \leq \frac{1}{2^n}$, 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 $\frac{\ell_i+1}{2^n} \leq \frac{\ell+1}{2^n}$.

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

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

If $\mathsf{G}_{i,0}^*$ does not abort, then the produced intermediate value $W^{(i)}$ is distinct from all the previous value $W^{(j)}$ for $N_j\|A_j = N_i\|A_i$. This means the further generated $V^{(i)}$ are also distinct from these $V^{(j)}$, and further $k_1^{(i)} \neq k_1^{(j)}$. On the other hand, for the previous queries with $N_j\|A_j \neq N_i\|A_i$, conditioned on $\neg\mathsf{CollH}$ 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 $k_1^{(j)}$ is independent from $k_1^{(i)}$. In all, the produced initial key $k_1^{(i)}$ is a new random value independent from all the appeared keys $k_1^{(1)}, \ldots, k_1^{(i-1)}$. Therefore, we start from $\mathsf{G}_{i,0}^*$ and make ℓ_i hops that are similar to those described in the non-reuse case. These hops yield G_{i,ℓ_i}^* at the end, with $|\Pr[E_{i,\ell_i}^*] - \Pr[E_{i,0}^*]| \leq \ell\varepsilon_{\mathsf{E}^*} + \frac{\ell+1}{2^n}$.

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

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

and

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

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

We then consider $\Pr[E_{q_e+1}] = \Pr[\mathcal{A}^{\$,\perp} \Rightarrow 1]$, and derive an upper bound for $\Pr[E_{q_e+1}] - \Pr[E_{q_e,\ell_{q_e}}^* \mid \mathsf{G}_{q_e,\ell_{q_e}}^* \text{ does not abort}]$. First, conditioned on that CollD_i 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. $N_i\|A_i = N_j\|A_j$, then $W^{(i)} \neq W^{(j)}$. Therefore, the two further derived values $V^{(i)} = P_{i,\ell_i+2}(W^{(i)})$ and $V^{(j)} = P_{i,\ell_j+2}(W^{(j)})$ are two random and independent values;
- if $N_i\|A_i \neq N_j\|A_j$, 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 $k_1^{(1)}, \ldots, k_1^{(q_e)}$ 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. $P'_{1,1}, \ldots, P'_{1,\ell_i}$ for $i = 1, \ldots, q_e$, and thus uniform and independent. Thus for the q_e ciphertexts $(c_1^{(1)}, \ldots, c_\ell^{(1)}), \ldots, (c_1^{(q_e)}, \ldots, c_\ell^{(q_e)})$ 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 $h_i \leftarrow \mathsf{H}(1\|R_i\|V_i\|c_i)$ and $T_i \leftarrow \widetilde{P}^1(h_i)$, the tweakable random permutation query $\widetilde{P}^1(h_i)$ 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:

- the query $\widetilde{P}^1(h_i)$ has been made during encrypting an earlier query (N_j, A_j, M_j). This means $\mathsf{H}(1\|R_i\|V_i\|c_i) = \mathsf{H}(1\|R_j\|V_j\|c_j)$. If $R_i \neq R_j$, or $c_i \neq c_j$, then this clearly contradicts $\neg\mathsf{CollH}$. Otherwise, if $N_i\|A_i \neq N_j\|A_j$ then $\mathsf{H}(0\|N_i\|A_i) = R_i = R_j = \mathsf{H}(0\|N_j\|A_j)$ also contradicts $\neg\mathsf{CollH}$. If $N_i\|A_i = N_j\|A_j$ and $c_i \neq c_j$ then $(N_i\|A_i\|M_i) = (N_j\|A_j\|M_j)$, and it contradicts the assumption that $\mathcal{A}$ never repeats queries;
- there exists an earlier decryption query $(N_j, A_j, (c_j, T_j))$ such that $h_i = (\widetilde{P}^1)^{-1}(T_j)$. However, this would contradict $\neg\mathsf{PreimgH}$.

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 $(q_e + q_d)^2/2^{n+1}$.

Gathering the above, we have

$$\Pr[E_{q_e+1}] - \Pr[E^*_{q_e,\ell_{q_e}} \mid \mathsf{G}^*_{q_e,\ell_{q_e}} \text{ does not abort}] \leq \frac{(q_e+q_d)^2 + q_e^2}{2^{n+1}}.$$

To conclude, we summarize the gaps as follows:

(i) $|\Pr[E_0] - \Pr[E_{0,1} \mid \mathsf{G}_{0,1} \text{ does not abort}]| \leq \varepsilon_{\mathsf{E}^*}$;
(ii) $\Pr[E_{0,1} \mid \mathsf{G}_{0,1} \text{ does not abort}] = \Pr[E^*_{0,\ell_0} \mid \mathsf{G}^*_{0,\ell_0} \text{ does not abort}]$;
(iii) $\Pr[E^*_{q_e,\ell_{q_e}} \mid \mathsf{G}^*_{q_e,\ell_{q_e}} \text{ does not abort}] - \Pr[E^*_{0,\ell_0} \mid \mathsf{G}^*_{0,\ell_0} \text{ does not abort}]$ is upper-bounded by $(2\ell+4)q_e\varepsilon_{\mathsf{E}}$;
(iv) $\Pr[E_{q_e+1}] - \Pr[E^*_{q_e,\ell_{q_e}} \mid \mathsf{G}^*_{q_e,\ell_{q_e}} \text{ does not abort}] \leq \frac{(q_e+q_d)^2+q_e^2}{2^{n+1}}$;
(v) $\Pr[\mathsf{G}^*_{q_e,\ell_{q_e}} \text{ aborts}] \leq \frac{(\ell+1)q_e}{2^n} + \varepsilon_{cr} + \varepsilon_{pr}$.

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

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

as claimed.

Empty Messages. can be handled by the proof in this subsetion without specific treatment. In detail, for the empty messages appearing in decryption queries the case resembles subsection G.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.

H AEDT and CCAML2 Security

In this section we give an overview of the AEDT scheme as well as our result. The scheme somewhat resembles a FEMALE variant with only the second pass: given (N, A, M) with an ℓ-block message M, AEDT computes the hash $R = \mathsf{H}(N, A)$, and then just uses $\mathsf{E}_k^{*,0}(R)$ as the initial key to "start" the one-time encryption (mentioned in the previous section). It is depicted in Fig. 12, and the full description is given in Fig. 13.

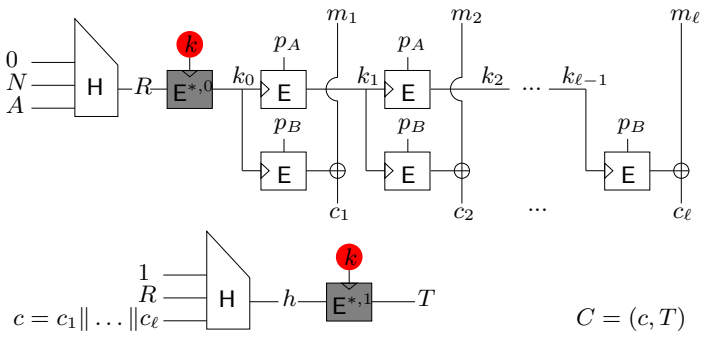

Fig. 12: (Top) The initialization and encryption parts of AEDT, with the notations to be used in its specification and security analysis. (Bottom) The authentication part of AEDT, with notations consistent with those in the top half. The final output is $C = (c, T)$.

Description of AEDT:

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

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

1. *One-time encryption:* First computes $k_0 \leftarrow \mathsf{E}_k^{*,0}(R)$, then, for $i = 1$ to $\ell - 1$, computes $k_i \leftarrow \mathsf{E}_{k_{i-1}}(p_A)$, $z_i \leftarrow \mathsf{E}_{k_{i-1}}(p_B)$, and $c_i \leftarrow z_i \oplus m_i$. Eventually, computes $z_\ell \leftarrow \mathsf{E}_{k_{\ell-1}}(p_B)$ and $c_\ell \leftarrow z_\ell \oplus m_\ell$.
2. *Authentication:* sets $c = c_1\| \ldots \|c_\ell$, and computes $T \leftarrow \mathsf{E}_k^{*,1}(\mathsf{H}(1\|R\|c))$.

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

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

1. *Integrity Checking:* computes $R = \mathsf{H}(0\|N\|A)$ and $h^* \leftarrow (\mathsf{E}_k^{*,1})^{-1}(T)$. Then, if $h^* = \mathsf{H}(1\|R\|c)$, it enters the next phase, and returns $\perp$ otherwise.
2. *One-time decryption:* first computes $s_0 \leftarrow \mathsf{E}_k^{*,0}(R)$, then, for $i = 1$ to $\ell - 1$, computes $k_i \leftarrow \mathsf{E}_{k_{i-1}}(p_A)$, $z_i \leftarrow \mathsf{E}_{k_{i-1}}(p_B)$, and $m_i \leftarrow z_i \oplus c_i$. Eventually, computes $z_\ell \leftarrow \mathsf{E}_{k_{\ell-1}}(p_B)$ and $m_\ell \leftarrow z_\ell \oplus c_\ell$.

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

Fig. 13: The AEDT AEAD scheme.

AEDT uses 2 leak-free blocks, just as FEMALE, but is twice more efficient in terms of number of weakly protected blocks to be evaluated. This may be a difference for long messages, in which the weakly protected blocks are the main computational cost. In terms of memory, and if the hash function used to compute the second hash, at the bottom of Fig. 13, proceeds block-by-block, AEDT is considerably more efficient for encryption than FEMALE: each ciphertext block c_i can be pushed immediately in the hash function and given as output. As a result, AEDT can be evaluated with constant memory, contrary to FEMALE that requires a memory at least the size of each encrypted message. The down-side of this is of course that AEDT cannot satisfy black-box MR, which is natural for a single-pass encryption scheme. But, for someone looking for an AEAD that will be used in the presence of leakages anyway (hence making black-box security mostly pointless), AEDT can be a very interesting choice.

Security of AEDT. The leakage function $\mathsf{L} = (\mathsf{L}_{\mathsf{enc}}, \mathsf{L}_{\mathsf{dec}})$ of AEDT is defined similarly to the one of FEMALE, i.e., including all the "bounded" leakages generated by the underlying actions.

In this setting, the CIML2 security of AEDT follows immediately from the one of the EDT mode defined in [9]. Its mCCAML2 security is as follows.

Theorem 10. *Let $H : \{0,1\}^\kappa \times \{0,1\}^* \to \{0,1\}^n$ be a $(0, t', \varepsilon_{cr})$-collision resistant and $(q_d, t', \varepsilon_{pr})$-range-oriented preimage resistant hash function, $E^* : \{0,1\}^n \times \{0,1\} \times \{0,1\}^n \to \{0,1\}^n$ be a $(2q_e + 2q_d + 2q_m, t', \varepsilon_{E^*})$-strong tweakable pseudorandom permutation, and $E : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a $(2, t, \varepsilon_E)$-PRP, whose implementation leakage function L_E has $(q_{\mathcal{S}}, t_{\mathcal{S}}, q_l, t, \varepsilon_{(p,2)\text{-}rsim})$ $(p,2)$-R-simulatable leakages. Then the* AEDT *implementation with leakage function $\mathsf{L} = (\mathsf{L}_{\mathsf{enc}}, \mathsf{L}_{\mathsf{dec}})$ defined before is $(q_e, q_d, p-1, q_m, q_l, t, \varepsilon_{\mathsf{mCCAML2}})$* mCCAML2*-secure, where*

$$\varepsilon_{\mathsf{mCCAML2}} \leq 2\varepsilon_{E^*} + \frac{q_e + q_m - 1}{2^n} + \varepsilon_{cr} + \varepsilon_{pr} + \sum_{i=1}^{q_m} \varepsilon_{\mathsf{AEDT}\text{-}eav}(\ell_i),$$

and $\varepsilon_{\mathsf{FEMALE}\text{-}eav}(\ell_i)$ *is the upper bound on the eavesdropper advantage of* (q_l, t')*-bounded adversaries on* AEDT *with* ℓ_i *block messages:*

$$\varepsilon_{\mathsf{AEDT}\text{-}eav}(\ell_i) \le 4\ell_i(\varepsilon_{\mathsf{E}} + \varepsilon_{(2,2)\text{-}rsim}) + \ell_i \cdot \varepsilon_{s\text{-}block},$$

and ℓ_i *is the number of blocks in the* i*-th challenge messages and* $\varepsilon_{s\text{-}block}$ *is as defined in subsection 4.4. Here* $t' = t + (q_e + q_d + q_m)(t_{\$} + t_{1-pass})$*,* t_{1-pass} *is the maximum running time of* AEDT *upon a single (encryption or decryption) query, and* $t_{\$}$ *is the time needed for randomly sampling a value from* $\{0,1\}^n$*.*

The remaining of this section devotes to prove Theorem 10. It follows the same line as subsection 4.4. 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.

Description of RESM:

- Gen picks $k_0 \xleftarrow{\$} \{0,1\}^n$
- $\mathsf{RESM}_{k_0}(m_1, \ldots, m_\ell)$ proceeds in four steps:
  (1) Initializes an empty list leak for the leakage;
  (2) for $i = 1, \ldots, \ell$, computes $k_i \leftarrow \mathsf{E}_{k_{i-1}}(p_A)$, $z_i \leftarrow \mathsf{E}_{k_{i-1}}(p_B)$, and $c_i \leftarrow z_i \oplus m_i$, and adds $[\mathsf{L}_{\mathsf{E}}(k_{i-1}, p_A), \mathsf{L}_{\mathsf{E}}(k_{i-1}, p_B)]^p$, $\mathsf{L}_{\oplus}(z_i, m_i)$, and $[\mathsf{L}_{\oplus}(z_i, c_i)]^{p-1}$ to the list leak.
  $\mathsf{RESM}_{k_0}(m_1, \ldots, m_\ell)$ eventually returns $(c_1, \ldots, c_\ell)$.

We define $\mathsf{LRESM}_{k_0}(m) = (\mathsf{RESM}_{k_0}(m), \mathsf{leak})$ for the list leak standing at the end of the above process.

Fig. 14: The RESM scheme.

Description of IESM:

- $\mathsf{IESM}_{k_0}(m_1, \ldots, m_\ell)$ proceeds in four steps:
  (1) Initializes an empty list leak for the leakage;
  (2) for $i = 1, \ldots, \ell$, samples $k_i \xleftarrow{\$} \{0,1\}^n$ and $z_i \xleftarrow{\$} \{0,1\}^n$ such that $k_i \neq z_i$, sets $c_i \leftarrow z_i \oplus m_i$, and adds $[\mathcal{S}^{\mathsf{L}}(k_{i-1}, p_A, k_i), \mathcal{S}^{\mathsf{L}}(k_{i-1}, p_B, z_i)]^p$, $\mathsf{L}_{\oplus}(z_i, m_i)$, and $[\mathsf{L}_{\oplus}(z_i, c_i)]^{p-1}$ to the list leak.
  $\mathsf{IESM}_{k_0}(m_1, \ldots, m_\ell)$ eventually returns $(c_1, \ldots, c_\ell)$.

We define $\mathsf{LIESM}_{k_0}(m) = (\mathsf{IESM}_{k_0}(m), \mathsf{leak})$ for the list leak standing at the end of the above process.

Fig. 15: The IESM scheme.

Lemma 8 (Indistinguishability of LRESM and LIESM). *Let* $E : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ *be a* $(2, t, \varepsilon_E)$*-PRP, whose implementation has a leakage function* L_E *having* $(q_{\mathcal{S}}, t_{\mathcal{S}}, q_l, t, \varepsilon_{(2,2)\text{-}rsim})$ $(2,2)$*-R-simulatable leakages, and let* $\mathcal{S}^{\mathsf{L}}$ *be an appropriate* $(q_{\mathcal{S}}, t_{\mathcal{S}})$*-bounded leakage simulator. Then, for every* ℓ*-block message* m*, every* p_A, p_B*, and every* $(q_l - 2q_r - q^*, t - 2t_r - t^*)$*-bounded distinguisher* $\mathcal{D}^{\mathsf{L}}$*, the following holds:*

$$|\Pr[\mathcal{D}^{\mathsf{L}}(m, \mathsf{LRESM}_{k_0}(m)) \Rightarrow 1] - \Pr[\mathcal{D}^{\mathsf{L}}(m, \mathsf{LIESM}_{k_0}(m)) \Rightarrow 1]| \le \ell(\varepsilon_E + \varepsilon_{(2,2)\text{-}rsim}).$$

Here $q_r = \ell(2q_{\mathcal{S}} + 3)$*,* q^* *and* t^* *are as defined in Lemma 2, and* $t_r = 2\ell(t_{\mathcal{S}} + t_{\$} + t_E) + \ell \cdot t_{\oplus}$*, where* t_E*,* $t_{\$}$*, and* $t_{\oplus}$ *are as assumed in Lemma 3.*

Proof. We define G_0 as the security game in which $\mathcal{A}^{\mathsf{L}}$ receives $\mathsf{LRFSM}_{s_0}(m)$ as the input, and G_ℓ as the game in which $\mathcal{A}^{\mathsf{L}}$ receives $\mathsf{LIFSM}_{s_0}(m)$ as the input. We show that G_0 could be transited to G_ℓ via a sequence of intermediate games

$$\mathsf{G}_1, \mathsf{G}_2, \ldots, \mathsf{G}_{\ell-1},$$

which resemble the games $\mathsf{G}_1^*, \ldots, \mathsf{G}_\ell^*$ appeared in the proof of Lemma 3. In detail, for j from 1 to ℓ, we consider the game G_{j-1}: we replace the two intermediate values $\mathsf{E}_{k_{j-1}}(p_A)$ and $\mathsf{E}_{k_{j-1}}(p_B)$ by two distinct random values k_j and z_j, and replace the leakages $[\mathsf{L}_{\mathsf{E}}(k_{j-1}, p_A), \mathsf{L}_{\mathsf{E}}(k_{j-1}, p_B)]^p$, $\mathsf{L}_{\oplus}(\mathsf{E}_{k_{j-1}}(p_B), m_j)$, and $[\mathsf{L}_{\oplus}(\mathsf{E}_{k_{j-1}}(p_B), c_j)]^{p-1}$ by $[\mathcal{S}^{\mathsf{L}}(k_{j-1}, p_A, k_j), \mathcal{S}^{\mathsf{L}}(k_{j-1}, p_B, z_j)]^p$, $\mathsf{L}_{\oplus}(z_j, m_j)$, and $[\mathsf{L}_{\oplus}(z_j, c_j)]^{p-1}$. This yields the game G_j. Clearly,

$$\Pr[\mathcal{D}^{\mathsf{G}_j} \Rightarrow 1] - \Pr[\mathcal{D}^{\mathsf{G}_{j-1}} \Rightarrow 1] \le \varepsilon_{\mathsf{E}} + \varepsilon_{(2,2)\text{-}rsim}.$$

Therefore, the ℓ transitions eventually yield

$$\left|\Pr[\mathcal{D}^{\mathsf{G}_\ell} \Rightarrow 1] - \Pr[\mathcal{D}^{\mathsf{G}_0} \Rightarrow 1]\right| \leq \ell(\varepsilon_{\mathsf{E}} + \varepsilon_{(2,2)\text{-}rsim})$$

as claimed. □

We then show the eavesdropper security of LIESM encrypting an ℓ-block message relies on the security of ISEnc.

Lemma 9. *For every pair of ℓ-block messages m^0 and m^1 and (q_l, t)-bounded adversary $\mathcal{A}^{\mathsf{L}}$, there exists a $(q_l + 2q_r, t + 2t_r)$-bounded adversary $\mathcal{A}^{\mathsf{L}'}$ such that*

$$\begin{aligned}&|\Pr[\mathcal{A}^{\mathsf{L}}(\mathsf{LIESM}_{k_0}(m^0)) \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{L}}(\mathsf{LIESM}_{k_0}(m^1)) \Rightarrow 1]| \\ \leq &\sum_{i=1}^{\ell} |\Pr[\mathcal{A}^{\mathsf{L}'}(\mathsf{LISEnc}^+_{k_{i-1}}(m_i^0)) \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{L}'}(\mathsf{LISEnc}^+_{k_{i-1}}(m_i^1)) \Rightarrow 1]|,\end{aligned}$$

where $k_0, \ldots, k_{\ell-1}$ are chosen uniformly at random, the p_B value used in the scheme $\mathsf{LISEnc}^+_{k_{i-1}}(m_i^1)$ is p_B, and m_i^0 and m_i^1 are the i-th block of m^0 and m^1 respectively. Here $q_r = \ell(2q_{\mathcal{S}} + 1)$ and $t_r = \ell(2t_{\mathcal{S}} + 2t_{\$} + t_{\oplus})$, where t_{E}, $t_{\$}$, and $t_{\oplus}$ 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, $m_{h,i} := m_1^0 \| \ldots \| m_{\ell-i}^0 \| m_{\ell-i+1}^1 \| \ldots \| m_\ell^1$. Following the same line as the proof of Lemma 3, it can be shown

$$\begin{aligned}&|\Pr[\mathcal{A}^{\mathsf{L}}(\mathsf{LIESM}_{k_0}(m_{h,i-1})) \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{L}}(\mathsf{LIESM}_{k_0}(m_{h,i})) \Rightarrow 1]| \\ \leq &|\Pr[\mathcal{A}^{\mathsf{L}'}(\mathsf{LISEnc}^+_{k_{\ell-i}}(m^0_{\ell-i+1})) \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{L}'}(\mathsf{LISEnc}^+_{k_{\ell-i}}(m^1_{\ell-i+1})) \Rightarrow 1]|.\end{aligned}$$

Taking a summation yields the main claim. □

Lemmas 2, 8, and 9 cinch the eavesdropper security on AEDT.

Lemma 10. *Let $E : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a $(2, t, \varepsilon_E)$-PRP, whose implementation has a leakage function L_E having $(q_{\mathcal{S}}, t_{\mathcal{S}}, q_l, t, \varepsilon_{(2,2)\text{-}rsim})$ $(2,2)$-R-simulatable leakages, and let $\mathcal{S}^{\mathsf{L}}$ be an appropriate $(q_{\mathcal{S}}, t_{\mathcal{S}})$-bounded leakage simulator. Then, for every pair of ℓ-block messages m^0 and m^1 and $(q_l - 2q_r - q^*, t - 2t_r - t^*)$-bounded adversary $\mathcal{A}^{\mathsf{L}}$, it holds*

$$|\Pr[\mathcal{A}^{\mathsf{L}}(\mathsf{LRESM}_{k_0}(m^0)) \Rightarrow 1] - \Pr[\mathcal{A}^{\mathsf{L}}(\mathsf{LRESM}_{k_0}(m^1)) \Rightarrow 1]| \leq 4\ell(\varepsilon_E + \varepsilon_{(2,2)\text{-}rsim}) + \ell \cdot \varepsilon_{s\text{-}block},$$

where q_r, t_r are as defined in Lemma 8, and q^*, t^* are as defined in Lemma 2.

Proof. In a similar vein to Lemma 5. □

And then Theorem 10 (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 $(\mathsf{E}^*_k)^{-1}$, which are indistinguishable from meaningless random values.

Therefore, the proof just follows the same line as that of Theorem 3.

On Empty Message. At the end of this section, we again consider the case the scheme is used on an empty message $M = \perp$. In that case, it can be seen AEDT collapses to a hash-then-MAC function, i.e.,

$$C = \mathsf{E}^{*,1}_k(\mathsf{H}(1\|R)),$$

where $R = \mathsf{H}(0\|N\|A)$. 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.