A Characterization of AE Robustness as Decryption Leakage Indistinguishability under Multiple Failure Conditions

Ganyuan Cao[a]

EPFL, Lausanne, Switzerland

Abstract. Robustness has emerged as an important criterion for authenticated encryption, alongside the requirements of confidentiality and integrity. We introduce a novel notion, denoted as IND-CCLA, to formalize the robustness of authenticated encryption from the perspective of decryption leakage. This notion is an augmentation of common notions defined for AEAD schemes by considering indistinguishability of potential leakage due to decryption failure, particularly in the presence of multiple failure conditions. With this notion, we study the disparity between a single-error decryption function and the actual leakage incurred during decryption. We introduce the concept of error unicity to require that only one error is disclosed, whether explicitly via decryption or implicitly via leakage, even there are multiple failure conditions. This aims to mitigate the security issue caused by disclosing multiple errors via leakage. We further extend this notion to IND-sf-CCLA to formalize the stateful security involving out-of-order ciphertext. Furthermore, we revisit the robustness of the Encode-then-Encrypt-then-MAC (EEM) paradigm, addressing concerns arising from the disclosure of multiple error messages. We then propose a modification to boost its robustness, thereby ensuring error unicity.

Keywords: AE Robustness · Decryption Leakage · IND-CCLA · Error Unicity · Security Proof

1 Introduction

1.1 Background and Motivation

Robustness of authenticated encryption has been defined in various ways. The most commonly accepted definition is with *robust authenticated encryption* (RAE), a term first introduced in [HKR15]. We follow the idea of RAE to say that an AEAD scheme is *robust* if confidentiality and authenticity are still guaranteed even if a nonce is inadvertently misused, or if part or all of the plaintext is leaked due to an authenticity-check failure (or other failures defined by the scheme). Additionally, an AEAD scheme qualifies as an RAE scheme when users can freely select any expansion factor τ to determine the ciphertext length relative to the plaintext, and the level of authenticity provided is contingent upon the chosen τ parameter.

The security of an RAE scheme is initially formalized in sense of a *pseudorandom injection* (PRI) in [HKR15]. Nevertheless, PRI formalizes the security in presence of decryption leakage in a very generalized way. The case where a scheme may involve multiple conditions for decryption failures has not been extensively explored with PRI. There are numerous attacks exploiting error messages, such as the notable padding oracle

E-mail: ganyuan.cao@epfl.ch (Ganyuan Cao)
[a]The author is jointly affiliated at EPFL and ETH Zürich

Date of this document: 2024-05-06.

attack introduced by Vaudenay [Vau02], which has been further developed to target SSL/TLS [CHVV03, PRS11], IPsec [DP07, DP10], and other systems. While adopting a unified error message in the decryption function for all error types appears promising as a mitigation, there may still be decryption leakage that grants adversaries an additional advantage. That is because decryption functions typically reveal plaintext only upon successful completion of all verification steps, whereas leakage may occur prior to their completion, potentially exposing partial plaintext and error messages. Additionally, these attacks are generally based on the errors about encoding, it comes to a natural question to ask:

What security do we want about errors related to encoding and decoding?

Furthermore, there has been limited exploration into the stateful security with PRI for out-of-order ciphertexts. Thus our goal is to introduce notions that view robustness of AE as an augmentation from conventional security notions defined for AEAD, such as IND-CCA3 proposed by Shrimpton in [Shr04], and the combination of semantic security (IND-CPA) plus ciphertext integrity (INT-CTXT) in [Rog02]. With our notions, we aim to study the disparity between a single-error decryption function and the actual leakage incurred during decryption including candidate plaintexts and error messages, while also allowing for an analysis that captures stateful security in the presence of out-of-order ciphertexts.

Simultaneously, many attempts towards enhancing the robustness of AEAD are to construct RAE schemes following an *enciphering-based* approach by using a VIL cipher, such as AEZ in [HKR15]. This characteristic poses challenges when attempting to boosting the robustness of generic compositions based on blockcipher modes of operation, such as Encode-then-Encrypt-then-MAC (EEM), which are more commonly used in practice. Thus our objective is to revisit the robustness of EEM as a generic composition and propose minimal modifications to it to boost its robustness by addressing the concerns arising from disclosure of multiple error messages.

1.2 Related Work

The security of a RAE scheme was initially formalized in sense of pseudorandom injection by the indistinguishability from a random injection $\pi_{N,A,\tau}$ such that the oracle returns M if there exists a plaintext M such that the ciphertext $C = \pi_{N,A,\tau}(M)$, and otherwise returns a plaintext which fails the authenticity check with the help of a decryption simulator. RAE formalizes the decryption leakage in a very generalized way where the cases involving multiple checks for errors are not studied. That may not be a problem for *enciphering-based* AE since they always decipher first and the authenticity check is on the deciphered string. The deciphered string is usually meaningless if the authenticity check fails. However, for other schemes, leakage may be meaningful if multiple checks are involved.

One key part of AE robustness is to ensure security even when part of the plaintext is accidentally leaked. There are several works that introduce notions to formalize the security under decryption leakage.

In [BDPS14], Boldyreva et al. studied the situation where an encryption scheme may output multiple errors and focused on the influence of error messages but not the actual leaked plaintext. They introduced the notion of IND-CVA which gives an IND-CPA adversary an additional oracle to tell if the queried ciphertext is valid or not. They also introduced the notion of *error invariance* (INV-ERR) which requires that no efficient adversary can generate more than one of the possible error messages. Notably, [BDPS14] also extended their study to consider the stateful security.

In [ABL+14], Andreeva et al. introduced the notion of *release-of-unverified-plaintext* (RUP) and associated it with the ciphertext integrity (INT) notion. Specifically, in

IND-RUP, the decryption algorithm always outputs a bitstring M. The adversary's goal is to make the validation function to accept a forged ciphertext given an encryption oracle and a decryption oracle.

In [BPS15], Barwell et al. introduced the notion of *subtle AE* (SAE), incorporating the idea of *error indistinguishability* (ERR-CCA) alongside IND-CPA and INT-CTXT. Specifically, ERR-CCA involves a leakage function that outputs the actual leakage. The security requirement is that an adversary should be unable to distinguish between the leakage under the same key and different keys. However, we believe that this notion may not perfectly capture the indistinguishability of leakage and there is a certain overlap with the integrity notion.

1.3 Our Contribution

We introduce a novel notion, denoted as IND-CCLA, to formalize the robustness of AE. The IND-CCLA notion extends conventional AE notions, such as IND-CCA3 [Shr04], by augmenting with a leakage simulator function inspired by subtle AE [BPS15]. This addition captures the leakage when decryption fails. However, we redefine the leakage simulator function, aiming to better separate it from the integrity notion and focus on the indistinguishability of the leakage itself.

In our notions, we consider the scenario where there may be multiple possible leaked plaintexts with multiple failure conditions. We require that an adversary should not be able to distinguish the leaked plaintext from a random bitstring of the minimum possible leakage length. On the top of this requirement, we introduce two sub-notions IND-CCLA1 and IND-CCLA2 about the disclosure of error messages. In IND-CCLA1, we follow [BDPS14] to require that the adversary should not be able to trigger an error except for a predefined error in the error space.

In IND-CCLA2, we impose a stronger security requirement that there should be only one error disclosed (whether implicitly by leakage or explicitly by decryption), even there may be multiple failure conditions. We call this property as *error unicity*. With this notion, we aim to align the actual leakage as closely as possible to the behavior of a single-error decryption function, thus to mitigate the security issue caused by disclosing errors via leakage. This notion concurrently ensures that if one of the checks fails to work properly (e.g. due to implementation flaws), the adversary should not be able to tell such a failure, which should be considered as an important perspective of "robustness". This notion also captures that if adversary's query passes one of the checks, then this does not grant the adversary with certainty that its strategy is effective in breaking the check thus to prevent such a strategy to be used in the subsequent queries.

We then extend this notion to a stronger version, IND-sf-CCLA, to formalize stateful security in scenarios involving out-of-order ciphertext delivery for stateful AE schemes. We follow our notion to analyze the stateful security of the Encode-then-Encipher (EtE) paradigm [BR00], which is the mainstream way to construct robust AE, by assuming the use of counter as nonce.

We revisit the robustness of the Encode-then-Encrypt-then-MAC (EEM) paradigm by considering the security issues from disclosing multiple error messages. We then present a minimal modification to boost its robustness to achieve error unicity. Our construction can be interpreted as a substitution of the MAC with a tweakable VIL cipher. At a high level, we encipher the segment of the message containing the encoding in the plaintext by additionally padding it with zeros for authenticity, and just encrypt the remaining segment with any symmetric encryption scheme as usual. This enables simultaneous verification for authenticity via checks for presence of correct number of zeros in the deciphered string and validation of correct encoding. Thus the ciphertext is authentic if the encoding is correct. This also allows us to obfuscate error messages thus achieve error unicity. We provide

a proof that our construction satisfies common robustness and security requirements according to our notion.

2 Preliminaries

2.1 Notation

We introduce the following notations that will be used throughout the paper. Let $\mathbb{N} = \{1, 2, \ldots\}$ denote the set of natural numbers. For each $n \in \mathbb{N}$, we define the set $[n] := \{1, \ldots, n\}$. Given a set S, we use the notation $S^{\geq n} := \bigcup_{i \geq n} S^i$ to denote the set of all non-empty sequences of length at least n over S, and we define $S^+ := S^{\geq 1}$. Let $x = (x_1, \cdots, x_\ell) \in S^+$ with $\ell \in \mathbb{N}$ be a sequence. We denote the length of x by $|x| := \ell$. For $y = (y_1, \ldots, y_{\ell'}) \in S'$ with $\ell' \in \mathbb{N}$, we define the concatenation of x and y as $x||y = (x_1, \ldots, x_\ell, y_1, \ldots, y_{\ell'})$. When $S = \{0, 1\}$, we refer to such sequences as bit strings. Let $i \in \{0, 1, \ldots\}$, we denote the ℓ-bit string representation of i as $[i]_\ell$. We let notation $S[a..b]$ represent the substring of S that includes indices ranging from a to b. We use ε to denote empty string where $|\varepsilon| = 0$.

We model a look-up table $\mathbf{T}$ that maps key bit strings of length k to value bit strings of length v as a function $\{0,1\}^k \to \{0,1\}^v \cup \{\perp\}$, where $\perp$ is a special value not belonging to $\{0,1\}^v$. To initialize $\mathbf{T}$ to an empty table, we use the notation $\mathbf{T} \leftarrow [\,]$. To assign a value V to a key K in $\mathbf{T}$, we use the notation $\mathbf{T}[K] \leftarrow V$. If a value has previously been assigned to K in $\mathbf{T}$, it will be overwritten by V. To read a value associated with a key K in $\mathbf{T}$ and assign it to V, we use the notation $V \leftarrow \mathbf{T}[K]$. If there is no value associated with K in $\mathbf{T}$, V will be assigned the special value $\perp$.

Let S be a finite set. We define the notation $x \leftarrow\!\$\ S$ to represent the selection of a value from the set S uniformly at random, which we then assign to the variable x. For an algorithm $\mathcal{A}$, we use the notation $y \leftarrow \mathcal{A}^{\mathrm{O}_1, \mathrm{O}_2, \cdots}$ to denote running $\mathcal{A}$ given access to oracles $\mathrm{O}_1, \mathrm{O}_2, \ldots$, and then assigning of the output of $\mathcal{A}$ to y.

2.2 Game-Based Proof

We follow the code-based game-playing framework of Bellare and Rogaway [BR06]. This framework utilizes a game G that consists of an *Initialization* procedure (Init), a *Finalization* procedure (Finalize), and a set of oracle procedures, number of which varies depending on the specific game. An adversary $\mathcal{A}$ interacts with the oracles, which return responses to the queries made by the adversary via return statements specified in the oracles' codes.

A game G is initiated with the Init procedure, followed by the adversary's interaction with the oracle. After a number of oracle queries, the adversary halts and outputs an *adversary output*. The procedure Finalize is then executed to generate a *game output*. If a finalization procedure is not explicitly defined, we consider the *adversary output* as the *game output*. We denote $\Pr[\mathcal{A}^{\textsc{Init}, \mathrm{O}_1, \mathrm{O}_2, \cdots} \Rightarrow b]$ as the probability that the adversary $\mathcal{A}$ outputs a value b after the Init procedure and queries to the oracle $\mathrm{O}_1, \mathrm{O}_2, \cdots$. We denote $\Pr[\mathrm{G}(\mathcal{A}) \Rightarrow b]$ as the probability that a game G outputs b when the adversary $\mathcal{A}$ plays game G. For simplicity, we define $\Pr[\mathrm{G}(\mathcal{A})] := \Pr[\mathrm{G}(\mathcal{A}) \Rightarrow 0]$.

For notion simplicity, we write $\Delta_{\mathcal{A}}(\mathrm{O}_L; \mathrm{O}_R) := \Pr[\mathcal{A}^{\mathrm{O}_L} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathrm{O}_R} \Rightarrow 0]$ to denote $\mathcal{A}$'s advantage in distinguishing between the oracles O_L and O_R. We use the notation $\$^{\mathrm{O}}$ to refer to an oracle that, on an input X, selects a value Y' uniformly at random from the space of all possible outputs with $|Y| = |Y'|$ where $\mathrm{O}(X) = Y$, and then returns Y'. We implicitly assume that $\$^{\mathrm{O}}$ effectively employs lazy sampling, meaning that whenever a repeated input X is queried, $\$^{\mathrm{O}}(X)$ always returns the same output, and otherwise samples a fresh uniform value.

2.3 Robust Authenticated Encryption (RAE)

We present the definition for RAE since our notions can be also applied to formalize the security of RAE schemes. We extend the nonce-based definition in [HKR15] to a stateful scheme to address potential states utilized during encryption and decryption. We present two sets of definitions for *stateful* RAE (sRAE) in Definition 1 and *nonce-based* RAE (nRAE) in Definition 2.

Definition 1 (Stateful RAE (sRAE))**.** A stateful robust authenticated encryption (sRAE) scheme is a tuple $\Pi = (\mathcal{E}, \mathcal{D})$ specifies two *stateful* algorithms

$$\mathcal{E} : \mathcal{K} \times \mathcal{AD} \times \mathbb{N} \times \mathcal{M} \times \mathcal{ST}_{\mathcal{E}} \to \mathcal{C} \times \mathcal{ST}_{\mathcal{E}}$$

and

$$\mathcal{D} : \mathcal{K} \times \mathcal{AD} \times \mathbb{N} \times \mathcal{C} \times \mathcal{ST}_{\mathcal{D}} \to \mathcal{M} \cup \{\perp\} \times \mathcal{ST}_{\mathcal{D}}$$

where $\mathcal{K} \subseteq \{0,1\}^*$ is the space of keys, $\mathcal{M} \subseteq \{0,1\}^*$ is the space of plaintexts, $\mathcal{C} \subseteq \{0,1\}^*$ is the space of ciphertexts, $\mathcal{AD} \subseteq \{0,1\}^*$ is the space of associated data, $\mathcal{ST}_{\mathcal{E}}$ is the space of encryption states, $\mathcal{ST}_{\mathcal{D}}$ is the space of decryption states. The encryption algorithm $\mathcal{E}$ takes a five-tuple $(K, A, \tau, M; \mathsf{st}_{\mathcal{E}}) \in \mathcal{K} \times \mathcal{AD} \times \mathbb{N} \times \mathcal{M} \times \mathcal{ST}_{\mathcal{E}}$, returns a ciphertext-state pair $(C; \mathsf{st}'_{\mathcal{E}}) \leftarrow \Pi.\mathcal{E}_K^{A,\tau;\mathsf{st}_{\mathcal{E}}}(M)$, such that $C \in \mathcal{C}$ and $|C| = |M| + \tau$. The decryption algorithm $\mathcal{D}$ takes a five-tuple $(K, A, \tau, C; \mathsf{st}_{\mathcal{D}}) \in \mathcal{K} \times \mathcal{AD} \times \mathbb{N} \times \mathcal{C} \times \mathcal{ST}_{\mathcal{D}}$, and returns a message-state pair $(M; \mathsf{st}'_{\mathcal{D}}) \leftarrow \Pi.\mathcal{D}_K^{A,\tau;\mathsf{st}_{\mathcal{D}}}(C)$ such that $M \in \mathcal{M} \cup \{\perp\}$. If there is no $M \in \mathcal{M}$ such that $C = \Pi.\mathcal{E}_K^{A,\tau;\mathsf{st}_{\mathcal{E}}}(M)$, then $\Pi.\mathcal{D}_K^{A,\tau;\mathsf{st}_{\mathcal{D}}}(C) = \perp$.

Definition 2 (Nonce-Based RAE (nRAE))**.** A nonce-based robust authenticated encryption (nRAE) scheme is a tuple $\Pi = (\mathcal{E}, \mathcal{D})$ specifies two algorithms

$$\mathcal{E} : \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathbb{N} \times \mathcal{M} \to \mathcal{C}$$

and

$$\mathcal{D} : \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathbb{N} \times \mathcal{C} \to \mathcal{M} \cup \{\perp\}$$

where $\mathcal{K} \subseteq \{0,1\}^*$ is the space of keys, $\mathcal{N} \subseteq \{0,1\}^*$ is the space of nonces, $\mathcal{M} \subseteq \{0,1\}^*$ is the space of plaintexts, $\mathcal{C} \subseteq \{0,1\}^*$ is the space of ciphertexts, $\mathcal{AD} \subseteq \{0,1\}^*$ is the space of associated data. The encryption algorithm $\mathcal{E}$ takes a five-tuple $(K, N, A, \tau, M) \in \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathbb{N} \times \mathcal{M}$, returns a ciphertext $C \leftarrow \Pi.\mathcal{E}_K^{N,A,\tau}(M)$ such that $C \in \mathcal{C}$ and $|C| = |M| + \tau$. The decryption algorithm $\mathcal{D}$ takes a five-tuple $(K, N, A, \tau, C) \in \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathbb{N} \times \mathcal{C}$, and returns a message $M \leftarrow \Pi.\mathcal{D}_K^{N,A,\tau}(C)$ such that $M \in \mathcal{M} \cup \{\perp\}$. If there is no $M \in \mathcal{M}$ such that $C = \Pi.\mathcal{E}_K^{N,A,\tau}(M)$, then $\Pi.\mathcal{D}_K^{N,A,\tau}(C) = \perp$.

3 Security Notions

We introduce the notion IND-CCLA to formalize the robustness of a nonce-based AE scheme, and the notion IND-sf-CCLA for a synchronous stateful AE scheme. Our notions can be seen as a natural extension from common notions used to formalize AE security including IND-CCA3 [Shr04] for nonce-based schemes and IND-sfCCA [BKN04] for stateful schemes. We include the expansion parameter τ defined for RAE scheme in our notion. For fixed-expansion schemes, the parameter τ can be discarded.

We consider two types of errors: *implicit error flags* and *explicit error message*. In Definition 1 and 2, the decryption function $\mathcal{D}$ is defined to yield only one error message, denoted as $\perp$. This *explicit error message* $\perp$, is deliberately disclosed to the adversary. To illustrate, envision a scenario where $\perp$ corresponds to the message `"decryption failed"`, visibly displayed as output on a screen.

With our notions, we capture the information that is *implicitly* disclosed to the adversary, including plaintext to be verified and implicit error flags for failure conditions. This information is not directly output to an adversary, yet the adversary might still find them through some side channels e.g. memory, cache etc. We let $v_i : \{0,1\}^* \to \{\mathsf{true}, \mathsf{false}\}$ for $i \geq 1$ be the predicates for the failure conditions defined by the scheme, and we let $\perp_i = v_i(\cdot)$ for $i \geq 1$ be the *implicit error flags*. In real world, each of $\perp_i$'s can be considered as a variable of boolean value e.g. $\perp_1 =$ `"cond1 = false"`. Even though the adversary only sees $\perp$ from the decryption function, it can still examine the values of $\perp_i$'s to gain more information.

We omit the discussion for errors which adversary trivially knows the result even without querying, for example, ciphertext is shorter than the minimum length supported by a scheme, or ciphertext is not a multiple of the block size etc. Such query does not grant the adversary with extra advantage in distinguishing or forging since the adversary trivially knows the result of such a query.

Leakage Simulator Function. Inspired by SAE notion introduced by Barwell et al. in [BPS15], we use a *leakage simulator function* $\mathcal{L}$ to capture those inadvertent leakage. We improve the definition to capture the leakage of multiple candidate plaintexts and multiple error flags. We define the leakage simulator function $\mathcal{L}$ for an AEAD scheme Π as in Definition 3. Here we present the definition with respect to a nonce-based scheme, the nonce space $\mathcal{N}$ can be replaced with decryption state space $\mathcal{ST}_{\mathcal{D}}$ for the definition for stateful schemes.

Definition 3. The leakage simulator function $\mathcal{L}$ for an AEAD scheme Π with key space $\mathcal{K}$, nonce space $\mathcal{N}$, associated data space $\mathcal{AD}$, and ciphertext space $\mathcal{C}$, is a function

$$\mathcal{L} : \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathbb{N} \times \mathcal{C} \to (\{0,1\}^{\ell} \times (\mathcal{S}_{\perp} \cup \{\perp\})) \cup \{\top\} \cup \{\sqcup\}$$

where $\mathcal{S}_{\perp} = \{\perp_i\}_{\cup_{i \geq 1}}$ is the space for implicit error flags, and $\perp \notin \mathcal{S}_{\perp}$ is the explicit error message output by $\Pi.\mathcal{D}$, such that the following conditions hold:

1. $\mathcal{L}_K^{N,A,\tau}(C) = \sqcup$ if there is no leaked plaintext *and* $|\mathcal{S}_{\perp}| = 1$. This holds *regardless of* whether $\mathcal{D}_K^{N,A,\tau}(C) = \perp$ or not for a queried ciphertext C.
2. $\mathcal{L}_K^{N,A,\tau}(C) = \top$ if C is a valid ciphertext *and* $\mathcal{L}_K^{N,A,\tau}(C) \neq \sqcup$.
3. $\mathcal{L}_K^{N,A,\tau}(C) = (M, \perp)$ if there is one leaked plaintext M with $|M| > 0$ *and* $|\mathcal{S}_{\perp}| = 1$.
4. $\mathcal{L}_K^{N,A,\tau}(C) = (M_i, \perp_i) \in \{0,1\}^{\ell} \times \mathcal{S}_{\perp}$ where $\ell \geq 0$ and $|\mathcal{S}_{\perp}| \geq 2$. We let M_i be the lastly obtained plaintext before $\perp_i$. If a plaintext M_i is not available before a specific error flag $\perp_i$, we set $M_i = \varepsilon$ as empty string. We assume the check for $\perp_i$ occurs before $\perp_j$ when $i < j$.
5. The correctness is defined by: if $\mathcal{D}_K^{N,A,\tau}(C) = \perp$, then $\mathcal{L}_K^{N,A,\tau}(C) \neq \top$, and if $\mathcal{L}_K^{N,A,\tau}(C) = \top$, then $\mathcal{D}_K^{N,A,\tau}(C) \neq \perp$.

Remark 1. In Definition 3, we let $\mathcal{S}_{\perp}$ represent the set of error flags used to verify the validity of a ciphertext. To better syntactically separate the explicit error message $\perp$ from error flags, we let $\perp \notin \mathcal{S}_{\perp}$. Note that when $|\mathcal{S}_{\perp}| = 1$, it yields the equivalence between the only error flag $\perp_1$ and the explicit error message $\perp$ since the adversary trivially knows that the error $\perp$ is caused by the failure indicated by $\perp_1$, which is the reason why we define $\mathcal{L}$ to output $(M, \perp)$ when $|\mathcal{S}_{\perp}| = 1$ in Bulletpoint 3.

Example 1. We show two examples of the outputs of the leakage simulator function as follows. Here we consider tag-based schemes and omit the expansion parameter τ.

1. Encrypt-then-MAC (EtM) [BN00]: The paradigm reveals no plaintext when decryption fails, since the tag is authenticated using the MAC scheme on the ciphertext, and the ciphertext remains undecrypted when authentication fails. It is easy to see that EtM has $|\mathcal{S}_\perp| = 1$, that is, due to authenticity-check failure. Thus $\mathcal{L}_K^{N,A}(C) = \sqcup$.

2. Encode-then-Encrypt-then-MAC (EEM) [BKN04]: In this paradigm, the plaintext is first encoded using, for instance, PKCS padding [Hou09]. Thus for a ciphertext C, we have $\mathcal{L}_K^{N,A}(C) \in \{(\varepsilon, \perp_1), (M, \perp_2)\}$, where $\perp_1$ indicates a failure on the authenticity check with tag, $\perp_2$ indicates an error in decoding, and $M \in \{0,1\}^\ell$ denotes the plaintext in incorrect format.

Error Merging. If multiple error flags are results of the predicates on the same leaked plaintext M, then the scheme can "merge" those error flags into one error flag without incurring extra leakage, for example, using an AND statement. Consequently, M fails multiple checks simultaneously if one of the checks fails, thereby circumventing the need for checks across multiple phases. We state the observation assuming without loss of generality that $v(\cdot) = \mathsf{true}$ leads to successful decryption. The observation also holds if we assume $v(\cdot) = \mathsf{true}$ leads to unsuccessful decryption by changing $\wedge$ to $\vee$.

Observation 1. *Let $\perp_i = v_i(M)$ and $\perp_j = v_j(M)$ with $i \neq j$, where v_i and v_j are condition predicates on plaintext, and M is the candidate plaintext to be verified. Then there is a merged error flag $\perp_{i,j} = v_{i,j}(M)$ where $v_{i,j}(M) = v_i(M) \wedge v_j(M)$. Notably, if the all the condition predicates are on the same candidate plaintext M, then there is an error flag $\perp' = v'(M)$ where $v'(M) = v_1(M) \wedge v_2(M) \ldots \wedge v_n(M)$ and $n = |\mathcal{S}_\perp|$.*

Note that we cannot merge a predicate on ciphertext with a predicate on plaintext without incurring plaintext leakage. It is trivial to see that we have to obtain the plaintext first, which introduces leakage.

3.1 IND-CCLA Security

We introduce a new notion *Indistinguishability under Chosen Ciphertext with Leakage Attack*, denoted as IND-CCLA, for (robust) AE, as illustrated in Figure 1. We introduce an addition oracle Leak which implements $\mathcal{L}_K$ to capture the information leaked during a decryption failure. This notion is defined for a nonce-based scheme.

Definition 4 (IND-CCLAx)**.**

$$\mathbf{Adv}_\Pi^{\text{IND-CCLA}x}(\mathcal{A}) := \Pr[\mathrm{G}_\Pi^{\text{IND-CCLA}x\text{-}0}(\mathcal{A})] - \Pr[\mathrm{G}_\Pi^{\text{IND-CCLA}x\text{-}1}(\mathcal{A})]$$

for $x \in \{1, 2\}$.

Observation on the Notion. We adopt the *real-or-ideal* oracle for Leak. In the ideal world, the oracle first checks if $\mathcal{L}_K(\cdot) = \sqcup$ to indicate no leakage at all, or $\mathcal{L}_K(C) = \top$ to indicate a valid ciphertext. In this case, the adversary should have 0 advantage in distinguishing by leakage and we return the same result. Otherwise, the oracle samples a bitstring M_λ uniformly at random of the length of the minimum plaintext leakage ℓ_λ^* defined by the scheme with respect to a tuple (N, A, τ) and a ciphertext length $|C|$ (ε if the length is 0). For example, if the output of $\mathcal{L}$ is in $\{(M_1, \perp_1), (M_2, \perp_2)\}$ with $|M_1| < |M_2|$. Then the minimum length $\ell_\lambda^* = |M_1|$. Otherwise if $\mathcal{L}$ output $(M, \perp)$, then $\ell_\lambda^* = |M|$.

Error Invariance and Unicity. Based on $x_\perp \in \{\perp_i, \perp\}$, we define two sub-notions about the disclosure of error message. We name them as IND-CCLA1 and IND-CCLA2 respectively. For notation simplicity, we use IND-CCLA to denote both IND-CCLA1 and IND-CCLA2 if a result applies to both notions.

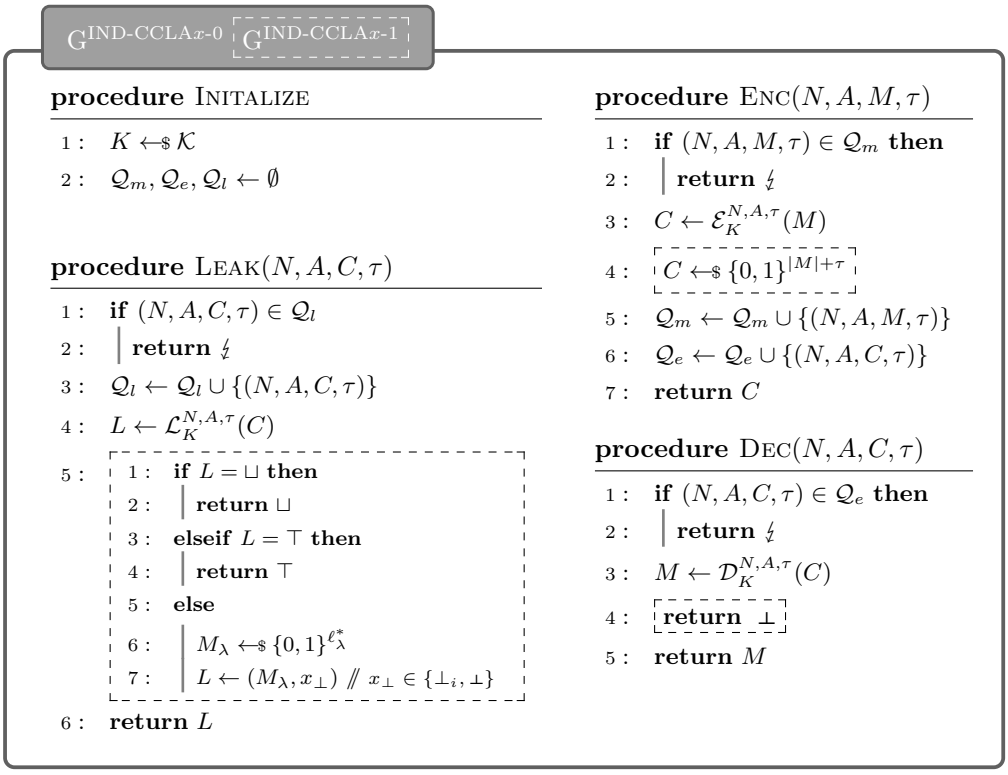

Figure 1: IND-CCLAx games for a nonce-based (robust) AE scheme Π. The dot-boxed parts are exclusive to $\mathrm{G}^{\text{IND-CCLA}x\text{-}1}$. We define ℓ^*_λ as the minimum achievable plaintext leakage length concerning a tuple (N, A, τ) and the ciphertext length $|C|$. We let $\ell^*_\lambda \geq 0$ if $x_\perp = \perp_i$ where $\perp_i \in \mathcal{S}_\perp$, and $\ell^*_\lambda > 0$ if $x_\perp = \perp$.

1. IND-CCLA1 (*Error Invariance*): The tuple $(M_\lambda, \perp_i)$ is returned in the ideal world for a $\perp_i \in \mathcal{S}_\perp$. Our goal with this sub-notion is to ensure that:

   - The adversary cannot distinguish between the leaked plaintext and a random bitstring of the minimum leakage length defined by the scheme.
   - The adversary cannot trigger an error flag except for $\perp_i$.

   This notion can be considered as a variant of the error invariance (INV-ERR) notion in [BDPS14] by also requiring the minimum disclosure of plaintext.

2. IND-CCLA2 (*Error Unicity*): The tuple $(M_\lambda, \perp)$ is returned in the ideal world. With this notion, in addition to ensuring that indistinguishability of the leaked plaintext, we require that the leakage function also discloses only one error just like decryption even there are multiple failure conditions. With this notion, we aim to align the actual leakage as closely as possible with the behavior of a single-error decryption function, thus to mitigate the security issue caused by disclosing multiple error flags via leakage. The intuition of the notion is to ensure that:

   - The adversary gains no meaningful plaintext unless all checks are successful.
   - If one of the checks fails to function properly (e.g. due to implementation flaws), the adversary remains unaware of such a failure.
   - If adversary's query passes one of the checks, then the adversary should not be certain that its strategy is effective in breaking the check.

At high level, IND-CCLA2 strictly requires that only one error flag is allowed even there may be multiple failure conditions, that is, $|\mathcal{S}_\perp| = 1$. On the other hand, IND-CCLA1 allows the existence of multiple error flags, which means $|\mathcal{S}_\perp| \geq 2$.

Observe that if there is not leakage at all i.e., the output of $\mathcal{L}$ is $\sqcup$, it trivially satisfies both IND-CCLA1 and IND-CCLA2 security. One may also notice that when $|\mathcal{S}_\perp| = 1$, then IND-CCLA1 is equivalent to IND-CCLA2 since there is no other error flag to be distinguished from the only error flag $\perp_1$. Thus to better separate between these two sub-notions, we define $\mathcal{L}$ to ouput $\perp$ when $|\mathcal{S}_\perp| = 1$ as discussed in Remark 1. Also, a scheme with $|\mathcal{S}_\perp| \geq 2$ cannot be IND-CCLA2 since almost for any query $\perp_1$ will be output to be distinguished from $\perp$.

However, to incentivize the development of single-error schemes, in Proposition 1, we show that IND-CCLA2 is strictly stronger than IND-CCLA1 when there are at least two implicit error flags i.e., $|\mathcal{S}_\perp| \geq 2$.

Proposition 1. IND-CCLA2 *implies* IND-CCLA1 *for a scheme that includes at least two implicit error flags i.e.,* $|\mathcal{S}_\perp| \geq 2$.

Proof (Sketch). (IND-CCLA2 $\rightarrow$ IND-CCLA1). Suppose we have an adversary $\mathcal{A}$ that breaks IND-CCLA1 security. Then $\mathcal{A}$'s query to LEAK yields $(M, \perp_j)$ with $|M| \geq |M_\lambda|$ or $\perp_j \neq \perp_i$ to be distinguished from $(M_\lambda, \perp_i)$ where M_λ is the random bitstring of the minimum leakage length. In all the cases, we can use $\mathcal{A}$ to distinguish from $(M_\lambda, \perp)$.

(IND-CCA1 $\not\rightarrow$ IND-CCLA2). Consider an AE scheme that is IND-CCLA1 secure with two error flags $\perp_1$ and $\perp_2$. It yields immediate distinguishing since $\perp_1$ will be output to be distinguished from $\perp$ for almost any query. □

EXTRACTION OF IND-EPL. We can then extract a notion particular for security under leakage from IND-CCLA. We name it as *Indistinguishability of Errors and Plaintext as Leakage*, denoted by IND-EPL. The adversary is granted access to the honest execution of encryption and decryption, allowing for an individual examination of the influence of the leakage. Similarly, we can define IND-EPL1 and IND-EPL2 based on $x_\perp \in \{\perp_i, \perp\}$ respectively. Here for simplicity, we slightly abuse the notations to let $\mathcal{E}_K$, $\mathcal{D}_K$ and $\mathcal{L}_K$ be ENC, DEC and LEAK in game $\mathrm{G}^{\text{IND-CCLA}x\text{-}0}$ in Figure 1 respectively, and we let $\$^{\mathcal{L}}$ be the LEAK oracle as in Figure $\mathrm{G}^{\text{IND-CCLA}x\text{-}1}$ in Figure 1. For notation simplicity, we use IND-EPL to denote both IND-EPL1 and IND-EPL2 if a result applies to both notions.

Definition 5 (IND-EPLx)**.**

$$\mathbf{Adv}_{\Pi}^{\text{IND-EPL}x}(\mathcal{A}) := \Delta_{\mathcal{A}}(\mathcal{E}_K, \mathcal{D}_K, \mathcal{L}_K; \mathcal{E}_K, \mathcal{D}_K, \$^{\mathcal{L}})$$

for key $K \leftarrow\!\!\$\, \mathcal{K}$ and $x \in \{1, 2\}$.

Corollary 1. IND-EPL2 *implies* IND-EPL1 *for a scheme that includes at least two implicit error flags i.e.,* $|\mathcal{S}_\perp| \geq 2$.

Proof. The proof follows a similar proof to Proposition 1. □

PROHIBITED & POINTLESS QUERIES. We specify the following generally prohibited queries to prevent trivial wins. These prohibitions are defined for both IND-CCLA and IND-EPL notions. We let the oracles return the invalid symbol $\lightning$ for those prohibited queries: 1, the adversary is not allowed to use the output of ENC to query DEC. 2, the adversary is not allowed to repeat a query to ENC or LEAK with the same tuple.

It is pointless to forward the query from ENC to LEAK since the oracle LEAK outputs $\top$ for a valid ciphertext in both real and ideal world.

Also, we do allow the adversary to repeat the nonce to capture the security when a nonce is possibly misused. Additionally, we stress that, we allow an adversary to query with variable stretch parameter, that is, the adversary can query with $\tau_1 \neq \tau_2$ in different

queries. Indeed, with small stretch, the adversary may trivially win the INT-CTXT game and IND-EPL game. However, this still captures the *best achievable* security with respect to a selected stretch parameter.

3.2 IND-sf-CCLA Security

We describe the game for IND-sf-CCLA notion for stateful (robust) AE as in Figure 2. This notion is specifically for the algorithm that uses synchronous states between the sender and the receiver. We make the extension from IND-sf-CCA notion introduce in [BKN04] by introducing the leakage oracle. We then define IND-sf-CCLA advantage in Definition 6.

$\mathrm{G}^{\text{IND-sf-CCLA}x\text{-0}}$ $\boxed{\mathrm{G}^{\text{IND-sf-CCLA}x\text{-1}}}$

procedure INITALIZE

1 : $K \leftarrow\!\!\$\ \mathcal{K}$
2 : $i \leftarrow 0$
3 : $j \leftarrow 0$
4 : $\mathsf{sync} \leftarrow 1$
5 : $\mathsf{rep} \leftarrow ()$
6 : $\mathbf{C} \leftarrow [\,]$
7 : $\mathsf{st}_{\mathcal{E}}, \mathsf{st}_{\mathcal{D}}, \mathsf{st}_{\mathcal{L}} \leftarrow \mathsf{Inital}$

procedure DEC(A, C, τ)

1 : $j \leftarrow j + 1$
2 : **if** $j > i \vee (A, C, \tau) \neq \mathbf{C}[j]$ **then**
3 : $\quad \mathsf{sync} \leftarrow 0$
4 : **if** $\mathsf{sync} = 1$ **then**
5 : $\quad$ **return** $\lightning$
6 : $\mathsf{st}_{\mathcal{L}} \leftarrow \mathsf{st}_{\mathcal{D}}$
7 : $M \leftarrow \mathcal{D}_K^{\mathsf{st}_{\mathcal{D}};A,\tau}(C)$
8 : $\mathsf{rep} \leftarrow ()$
9 : $\boxed{\textbf{return } \perp}$
10 : **return** M

procedure ENC(A, M, τ)

1 : $i \leftarrow i + 1$
2 : $C \leftarrow \mathcal{E}_K^{\mathsf{st}_{\mathcal{E}};A,\tau}(M)$
3 : $\boxed{C \leftarrow\!\!\$\ \{0,1\}^{|M|+\tau}}$
4 : $\mathbf{C}[i] \leftarrow (A, C, \tau)$
5 : **return** C

procedure LEAK(A, C, τ)

1 : **if** $\mathsf{rep} = (A, C, \tau)$ **then**
2 : $\quad$ **return** $\lightning$
3 : $\mathsf{rep} \leftarrow (A, C, \tau)$
4 : $L \leftarrow \mathcal{L}_K^{\mathsf{st}_{\mathcal{L}};A,\tau}(C)$
5 :
$\boxed{\begin{array}{ll} 1: & \textbf{if } L = \sqcup \textbf{ then} \\ 2: & \quad \textbf{return } \sqcup \\ 3: & \textbf{elseif } L = \top \textbf{ then} \\ 4: & \quad \textbf{return } \top \\ 5: & \textbf{else} \\ 6: & \quad M_\lambda \leftarrow\!\!\$\ \{0,1\}^{\ell^*_\lambda} \\ 7: & \quad L \leftarrow (M_\lambda, x_\perp) \ \ /\!/\ x_\perp \in \{\perp_i, \perp\} \end{array}}$
6 : **return** L

Figure 2: IND-sf-CCLAx games for a stateful (robust) AE scheme Π with synchronous states. The boxed parts are exclusively to game $\mathrm{G}^{\text{IND-sf-CCLA}x\text{-1}}$. We define ℓ^*_λ as the minimum achievable plaintext leakage length concerning a tuple (A, τ) and the ciphertext length $|C|$. We let $\ell^*_\lambda \geq 0$ if $x_\perp = \perp_i$ where $\perp_i \in \mathcal{S}_\perp$,and $\ell^*_\lambda > 0$ if $x_\perp = \perp$. We use $\mathsf{Initial}$ to denote the initial state. In Line 6 of DEC, we copy the decryption state $\mathsf{st}_{\mathcal{D}}$ to the leakage state $\mathsf{st}_{\mathcal{L}}$ for synchronization, and we still call $\mathcal{D}$ to update $\mathsf{st}_{\mathcal{D}}$ in ideal world and thus to update $\mathsf{st}_{\mathcal{L}}$ in case $\mathcal{L}_K^{\mathsf{st}_{\mathcal{L}};A,\tau}(C) = \top$. We use rep to ensure the adversary does not make prohibited queries.

Definition 6 (IND-sf-CCLAx)**.**

$$\mathbf{Adv}^{\text{IND-sf-CCLA}x}_{\Pi}(\mathcal{A}) := \Pr[\mathrm{G}^{\text{IND-sf-CCLA}x\text{-0}}_{\Pi}(\mathcal{A})] - \Pr[\mathrm{G}^{\text{IND-sf-CCLA}x\text{-1}}_{\Pi}(\mathcal{A})]$$

for $x \in \{1, 2\}$.

Proposition 2. IND-sf-CCLA2 *implies* IND-sf-CCLA1 *for a scheme that includes at least two implicit error flags i.e.,* $|\mathcal{S}_\perp| \geq 2$.

Proof. The proof follows a similar proof to Proposition 1. □

Extraction of IND-sf-EPL. We then similarly extract the IND-sf-EPL notion from that of IND-sf-CCLA. Again, for simplicity, we slightly abuse the notations to let $\mathcal{E}_K$, $\mathcal{D}_K$ and $\mathcal{L}_K$ denote the oracles Enc, Dec and Leak respectively in game $\mathrm{G}^{\text{IND-sf-CCLA}x\text{-}0}$ in Figure 2. Additionally, we use $\$^{\mathcal{L}}$ to denote the oracle Leak in game $\mathrm{G}^{\text{IND-sf-CCLA}x\text{-}1}$ in Figure 2. We define IND-sf-EPL as follows.

Definition 7 (IND-sf-EPLx)**.**

$$\mathbf{Adv}_{\Pi}^{\text{IND-sf-EPL}x}(\mathcal{A}) := \Delta_{\mathcal{A}}(\mathcal{E}_K, \mathcal{D}_K, \mathcal{L}_K; \mathcal{E}_K, \mathcal{D}_K, \$^{\mathcal{L}})$$

for key $K \leftarrow\!\!\$\, \mathcal{K}$ and $x \in \{1, 2\}$.

Corollary 2. IND-sf-EPL2 *implies* IND-sf-EPL1 *for a scheme that includes at least two implicit error flags i.e.,* $|\mathcal{S}_\perp| \geq 2$.

Proof. The proof follows a similar proof to Proposition 1. □

Prohibited & Pointless Queries. In addition to the queries to Dec with in-order ciphertext from Enc, we prohibit the adversary from making *consecutive* repeated queries to the Leak oracle. That is because two queries to Leak with the same tuple and the same state yield the same result and allow the adversary to trivially win the game. Thus we require that there must be one query to Dec between two successive queries to Leak.

This restriction also aligns with real-world scenarios since leakage can only occur if the decryption function is invoked. The underlying idea is that, following each update of states, even when queried with the same tuple, the leakage should be indistinguishable. Those prohibited queries are defined for both IND-sf-CCLA and IND-sf-EPL notions. We let the oracles return the invalid symbol $\lightning$ for those prohibited queries.

Similarly, it is pointless to query in-order ciphertext from Enc to Leak since the oracle yields $\top$ in both real and ideal world.

3.3 Separation and Relations

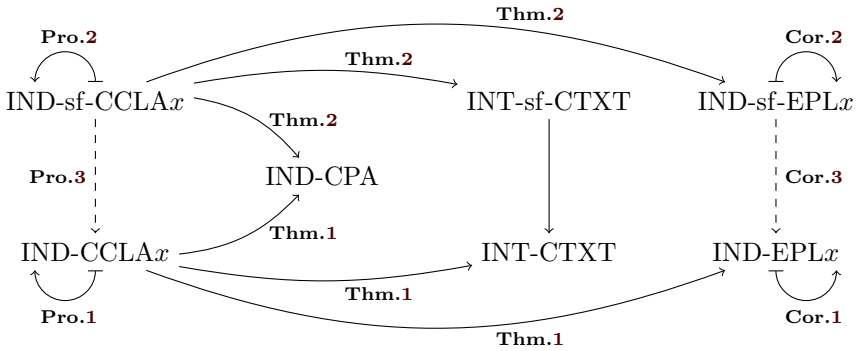

Figure 3: An illustration of implications between notions. We use $A \to B$ to denote that notion A implies notion B. We use $Ax \dashrightarrow Bx$ to denote that Ax implies Bx only when x is the same value for both Ax and Bx. We use $Ax \mapsto Ax$ to denote $A2$ imples $A1$.

Decomposition Theorems. We decompose IND-CCLA notion into IND-CPA plus INT-CTXT plus IND-EPL, which captures the security goals of confidentiality, authenticity, and security under decryption leakage respectively. Here we define IND-CPA as *real-or-random* security i.e., indistinguishability from random bits as defined in [AR02] and [RBBK01]. We follow the definition of INT-CTXT as in [BN00].

Theorem 1. *For $x \in \{1, 2\}$, for any* IND-CCLAx *adversary $\mathcal{A}$, there exist an* IND-CPA *adversary $\mathcal{A}_{cpa}$, an* INT-CTXT *adversary $\mathcal{A}_{int}$ and an* IND-EPLx *adversary $\mathcal{A}_{err}$ such that*

$$\mathbf{Adv}^{\text{IND-CCLA}x}_{\Pi}(\mathcal{A}) \leq \mathbf{Adv}^{\text{IND-CPA}}_{\Pi}(\mathcal{A}_{cpa}) + \mathbf{Adv}^{\text{INT-CTXT}}_{\Pi}(\mathcal{A}_{int}) + \mathbf{Adv}^{\text{IND-EPL}x}_{\Pi}(\mathcal{A}_{err}).$$

Proof. We rewrite the advantage as

$$\mathbf{Adv}^{\text{IND-CCLA}x}_{\Pi}(\mathcal{A}) = \Delta_{\mathcal{A}}(\mathcal{E}_K, \mathcal{D}_K, \mathcal{L}_K; \mathcal{E}_K, \mathcal{D}_K, \$^{\mathcal{L}}) + \Delta_{\mathcal{A}}(\mathcal{E}_K, \mathcal{D}_K, \$^{\mathcal{L}}; \$^{\mathcal{E}}, \perp, \$^{\mathcal{L}}).$$

By definition, we have that

$$\mathbf{Adv}^{\text{IND-EPL}x}_{\Pi}(\mathcal{A}_{err}) = \Delta_{\mathcal{A}_{err}}(\mathcal{E}_K, \mathcal{D}_K, \mathcal{L}_K; \mathcal{E}_K, \mathcal{D}_K, \$^{\mathcal{L}}).$$

for an IND-EPLx adversary $\mathcal{A}_{err}$.

Now given an adversary $\mathcal{A}_1$ with $\mathbf{Adv}_{\Pi}(\mathcal{A}_1) = \Delta_{\mathcal{A}_1}(\mathcal{E}_K, \mathcal{D}_K, \$^{\mathcal{L}}; \$^{\mathcal{E}}, \perp, \$^{\mathcal{L}})$, we can then construct an IND-CCA3 adversary $\mathcal{B}$ from $\mathcal{A}_1$. If $x = 1$, we simulate the oracle $\$^{\mathcal{L}}$ as follows. We let $\mathcal{A}_1$ simply return $\sqcup$ if no leakage is defined by the scheme. Otherwise, we first let $\mathcal{A}_1$ query $\mathcal{D}_K$ to first check if a ciphertext C is valid. If valid, $\mathcal{A}_1$ returns $\top$. Otherwise, we let $\mathcal{A}_1$ sample a bitstring M_λ of the minimum leakage length uniformly at random and return the tuple $(M_\lambda, \perp_i)$ as response to $\mathcal{A}$'s queries. Otherwise if $x = 2$, we instead let $\mathcal{A}_1$ return the tuple $(M_\lambda, \perp)$. We then have $\mathcal{B}$ return the same bit b returned by $\mathcal{A}_1$. We can then bound the advantage of $\mathcal{B}$ as

$$\Delta_{\mathcal{A}_1}(\mathcal{E}_K, \mathcal{D}_K, \$^{\mathcal{L}}; \$^{\mathcal{E}}, \perp, \$^{\mathcal{L}}) \leq \mathbf{Adv}^{\text{IND-CCA3}}_{\Pi}(\mathcal{B}).$$

Now following [Shr04, Theorem 2], we can further decompose the advantage as

$$\mathbf{Adv}^{\text{IND-CCLA}x}_{\Pi}(\mathcal{A}) \leq \mathbf{Adv}^{\text{IND-CPA}}_{\Pi}(\mathcal{A}_{cpa}) + \mathbf{Adv}^{\text{INT-CTXT}}_{\Pi}(\mathcal{A}_{int}) + \mathbf{Adv}^{\text{IND-EPL}x}_{\Pi}(\mathcal{A}_{err}).$$

□

Similarly, we can decompose IND-sf-CCLA notion into IND-CPA plus INT-sf-CTXT plus IND-sf-EPL. Here we replace the *left-or-right* encryption oracle with a *real-or-random* oracle in the definition of IND-sfCCA advantage in [BKN04] and we follow the definition of INT-sf-CTXT as in [BKN04].

Theorem 2. *For $x \in \{1, 2\}$, for any* IND-sf-CCLAx *adversary $\mathcal{A}$, there exist an* IND-CPA *adversary $\mathcal{A}_{cpa}$, an* INT-sf-CTXT *adversary $\mathcal{A}_{int}$ and an* IND-sf-EPLx *adversary $\mathcal{A}_{err}$ such that*

$$\mathbf{Adv}^{\text{IND-sf-CCLA}x}_{\Pi}(\mathcal{A}) \leq \mathbf{Adv}^{\text{IND-CPA}}_{\Pi}(\mathcal{A}_{cpa}) + \mathbf{Adv}^{\text{INT-sf-CTXT}}_{\Pi}(\mathcal{A}_{int}) + \mathbf{Adv}^{\text{IND-sf-EPL}x}_{\Pi}(\mathcal{A}_{err}).$$

Proof. The proof follows a similar proof of Theorem 1 by replacing IND-CCA3 with IND-sfCCA. □

Implication between Notions. The following set of relationships is inherently obvious. We present them here to provide completeness and we omit proofs since they are trivial.

Proposition 3. IND-sf-CCLAx *implies* IND-CCLAx *for* $x \in \{1, 2\}$.

Corollary 3. IND-sf-EPLx *implies* IND-EPLx *for* $x \in \{1, 2\}$.

Separation from AE Notions. In IND-EPL notions, we have the oracle return $\top$ both in the real world and the ideal world for a valid ciphertext. This removes the overlap with integrity notion. In Proposition 4, we separate IND-EPL from INT-CTXT and IND-CCA3 by showing that there is no implication between those notions.

Proposition 4. IND-EPL *does not imply* INT-CTXT *and* IND-CCA3 *does not imply* IND-EPL.

Proof (Sketch). (IND-EPL $\not\to$ INT-CTXT). We consider an Encrypt-then-MAC scheme where $C = M \oplus K_E$ and $T = C \oplus K_M$. The ciphertext returned is $C||T$. Note that there is no leaked plaintext since the tag is computed based on the ciphertext, and there is only one error message with tag checking. Thus both oracles will output $\sqcup$ meaning that IND-EPL advantage is 0. Nevertheless, an adversary can forge a valid ciphertext by querying the encryption oracle to obtain $C||T$ and returning $C \oplus 1^n||T \oplus 1^n$ as forgery.

(IND-CCA3 $\not\to$ IND-EPL2). We consider the Encrypt-then-MAC (EtM) paradigm in which the MAC scheme but a configuration in which the ciphertext is first decrypted before verifying the tag during the decryption. EtM is IND-CCA3 secure as established by combining the results from [BN00] and [Shr04]. We then have that $\mathcal{L}_K(C) = (M, \perp)$ where M is the plaintext. The adversary can replace the tag of a ciphertext from a previous encryption query to induce a decryption failure in the leakage oracle. Consequently, the adversary can break the IND-EPL2 security by comparing the plaintext used in that encryption query with the obtained leakage.

(IND-CCA3 $\not\to$ IND-EPL1). We consider Encode-then-Encrypt-then-MAC but with the "decryption first" configuration. Thus we have $\mathcal{L}_K(C) \in \{(M, \perp_1), (M, \perp_2)\}$, where $\perp_1$ indicates the authenticity failure, and $\perp_2$ indicates incorrect encoding. By also changing the tag for a valid ciphertext from a previous encryption query, the adversary can distinguish M from a random bitstring.

□

3.4 Comparison with Existing Notions

We make a brief comparison to differentiate our notion from the existing notions, specifically RAE security from [HKR15], the error invariance from [BDPS14], error simulatability from [BPS15], and plaintext awareness from [ABL+14].

RAE Security. In RAE security, the comparison is made with a random injection as a whole, whereas our notion focuses on the indistinguishability of the leakage itself. RAE formalize the leakage in a generalized way where a plaintext is always leaked in case of decryption failure, and the case involving multiple errors has not been studied.

Additionally, RAE claims to achieve the best-possible security with respect to a queried tuple (N, A, M, τ) (or (N, A, C, τ) respectively). Indeed, RAE also "make sense" when the stretch parameter τ is small. However, this fails to explicitly show what security goal, e.g. confidentiality, authenticity, and indistinguishability of leaked plaintext, can be achieved with respect to a particular stretch parameter. This should be conveyed through proofs since it gives the information on what stretch parameter should be chosen for security. In our notions, we decompose them to capture each security goal individually, which allows us to show what security can be achieved with a stretch parameter, as discussed in Remark 2.

Regarding the plaintext leakage, one particular requirement of RAE is to ensure that the leaked plaintext has length $|M| \neq |C| - \tau$ for a queried ciphertext C and an expansion parameter τ. Notably, our notion can also be adapted to capture that by additionally requiring ℓ^*_λ to be not equal to $|C| - \tau$.

Error Invariance. In [BDPS14], Boldyreva et al. investigated the case where a decryption scheme generates multiple error messages. The notion of error invariance (INV-ERR) dictates that the adversary should have negligible advantage in generating a ciphertext that triggers an error message other than a predefined one. Since the decryption scheme only outputs a single error message in our notion, we draw a parallel to our leakage simulator function, and consider the space of implict error flags $\mathcal{S}_\perp$ as the error space in their notion.

In IND-EPL1, we require that the adversary cannot induce an error flag $\perp_j$ such that $\perp_i \neq \perp_j$, where $\perp_i \in \mathcal{S}_\perp$ is the predefined error flag. This aligns with the idea of error invariance but we also takes the leaked plaintext into consideration. For IND-EPL2, we essentially require that $|\mathcal{S}_\perp| = 1$. This automatically satisfies error invariance.

Proposition 5. IND-EPL *implies* INV-ERR.

Proof (Sketch). Suppose that there is an INV-ERR adversary $\mathcal{A}$ against an AEAD scheme Π with error space $\mathcal{S}_\perp$. We first assume $|\mathcal{S}_\perp| \geq 2$. We can then use it to construct an IND-EPL1 adversary $\mathcal{B}$ as follows. For each of $\mathcal{A}$'s decryption query, we let $\mathcal{B}$ forward it to its oracle Leak. We let $\mathcal{B}$ response $\mathcal{A}$'s query with the error flag from Leak. Note that $\mathcal{A}$ eventually queries a ciphertext C yielding an error other than $\perp_i$. Then $\mathcal{B}$ can queries Leak with C to distinguish from $\perp_i$. Thus we have $\mathbf{Adv}^{\text{INV-ERR}}_{\Pi}(\mathcal{A}) \leq \mathbf{Adv}^{\text{IND-EPL1}}_{\Pi}(\mathcal{B})$.

Now if $|\mathcal{S}_\perp| = 1$, then $\mathbf{Adv}^{\text{INV-ERR}}_{\Pi}(\mathcal{A}) = 0$ since there is no $\perp_j \in \mathcal{S}_\perp$ to be distinguished from $\perp_i$. However, $\mathcal{B}$ may still distinguish by leaked plaintext M to break IND-EPL2. Thus we have $\mathbf{Adv}^{\text{INV-ERR}}_{\Pi}(\mathcal{A}) \leq \mathbf{Adv}^{\text{IND-EPL2}}_{\Pi}(\mathcal{B})$. □

Error Simulatability. In [BPS15], Barwell et al. introduced the concept of a leakage simulator function. They define the leakage function $\mathcal{L}$ as $\mathcal{L} : \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{C} \to \{\top\} \cup \Lambda$ where Λ represents the leakage space that accommodates various types of leakage including multiple errors, candidate plaintexts, arbitrary string, or the classical case where nothing is leaked. The security of error indistinguishability (ERR-CCA) is formalized by comparing oracles that implement $\mathcal{L}$ with the real key K and a different random key K'.

We observed that the definition is in a very generalized way, and the comparison does not adequately capture the impact of the leakage itself due to an overlap with the integrity notion. For a valid ciphertext C, $\mathcal{L}_K(C)$ yields $\top$ while $\mathcal{L}_{K'}(C)$ almost for sure outputs something other than $\top$ to be distinguished. This requires us to always consider an integrity adversary when bounding the advantage, which fails to capture the impact of the leakage itself. This is more apparent when there is no leakage. An adversary should have 0 advantage in distinguishing by leakage when there is no leakage but we have to consider an integrity adversary then. Thus to resolve this issue, we let the oracles in both real and ideal world to return $\top$ or $\sqcup$ when the leakage function yields such an output.

Plaintext Awareness. In [ABL+14], Andreeva et al. introduced *plaintext awareness* to capture the indistinguishability of the plaintext where the ciphertext is always decrypted and no check is not involved at all. Particularly, we consider the stronger version of PA2 security. In the original work, PA2 is defined by comparing the actual decryption function and a simulator for the decryption function. For our following example of EtE and the modification to EEM, we can essentially consider the indistinguishability of plaintext as a random bitstring. We then define it in Definition 8.

Definition 8 (PA2)**.** Let $\widetilde{\mathcal{D}}$ be the decryption function without authenticity check such

that $\widetilde{\mathcal{D}}$ always output a plaintext, then

$$\mathbf{Adv}^{\mathrm{PA2}}_{\Pi}(\mathcal{A}) := \Delta_{\mathcal{A}}(\mathcal{E}_K, \widetilde{\mathcal{D}}_K; \mathcal{E}_K, \$^{\widetilde{\mathcal{D}}})$$

for key $K \leftarrow\!\!\$\, \mathcal{K}$.

Notably, PA2 security consider the indistinguishability of the plaintext when all checks fail to work, whereas our IND-EPL2 notion stresses the indistinguishability (or rather, uniqueness) of the error message itself. As discussed in many studies including [Vau02, CHVV03, PRS11], the revelation of the error message alone can lead to significant attacks. In Proposition 6, we separate our IND-EPL2 notion from PA2 security. However, we acknowledge the significance of PA2 security in ensuring the AE robustness as it guarantees the confidentiality of the plaintext when all check mechanisms are circumvented. We conclude that an AE scheme should achieve both IND-EPL2 and PA2 to be considered robust.

Proposition 6. PA2 *security does not imply* IND-EPL2 *security, and* IND-EPL2 *security does not imply* PA2 *security.*

Proof. (IND-EPL2 $\not\Rightarrow$ PA2). We consider Encrypt-then-MAC paradigm as in Example 1. We know the IND-EPL2 advantage is 0 since there is no leaked plaintext and there is only one error. However, it is trivial to break PA2 security by changing the tag of a ciphertext obtained from a previous encryption query.

(PA2 $\not\Rightarrow$ IND-EPL2). We consider the Encode-then-Encipher paradigm with $|\mathcal{S}_\perp| = 2$. Thus the possible outputs of $\mathcal{L}$ are $(M, \perp_1)$ and $(M, \perp_2)$ where M is the deciphered string. Then it is PA2 secure if the cipher is secure as a tweakable pseudorandom permutation. However, it is not IND-EPL2 secure since for almost every query $\perp_1$ will be output to be distinguished from $\perp$. □

While this proof may seem abstract, there exist schemes that is PA2 secure yet not IND-EPL2 secure. An example is the Robust IV (RIV) proposed by Abed et al. in [AFL+16]. RIV attains PA2 security by binding the IV for decryption with the ciphertext and the tag, therefore altering any one of them yields a new plaintext. However, RIV follows a decryption-first approach, followed by the authenticity verification on the IV. Should an encoding check be required for the plaintext, this inevitably introduces a second error, rendering the scheme not IND-EPL2 secure.

4 Stateful Security of Encode-then-Encipher

Encode-then-Encipher (EtE), proposed by Bellare and Rogaway in [BR00], is the mainstream way to construct robust AE. In [HKR15], Hoang et al. proved that EtE with VIL cipher achieves the security as a pseudorandom injection (PRI). However, there has been limited studies about the stateful security of a robust AE scheme. In [BMM+15], Badertscher et al. showed the security of the Encode-then-Encipher from the view of composable security with the *Constructive Cryptography* (CC) framework proposed by Maurer in [Mau11], by constructing a *random injection channel* (RIC) from a uniform random injection (which ideally models a VIL cipher) and an insecure channel. The RIC models an *ideal world* in which a counter is used as nonce, and the adversary only has knowledge of the associated data and the message length. We then follow the idea of [BMM+15] by also using counter as nonce and prove the security and robustness of EtE from a more generalized game-based perspective with our notion.

4.1 EtE with Tweakable Cipher

Let $E : \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{M} \to \mathcal{C}$ be a variable-input-length cipher, we define a robust AE scheme $\Pi = (\mathcal{E}, \mathcal{D})$ using EtE as follows. Let $C = \Pi.\mathcal{E}_K^{N,A,\tau}(M) = E_{K,N,A}(M||0^\tau)$ and return C as ciphertext. Let $M' = E^{-1}_{K,N,A}(C)$, then if M' ends with λ zeros, $\Pi.\mathcal{D}_K^{N,A,\tau}(C)$ returns M' excluding ending τ zeros as plaintext. Otherwise, $\Pi.\mathcal{D}_K^{N,A,\tau}(C)$ return decryption failure symbol $\perp$.

Such cipher can be formulated as a *tweakable cipher* $\widetilde{E} : \mathcal{K} \times \mathcal{T} \times \mathcal{M} \to \mathcal{C}$ as described in [LRW02]. Here we set the tweak space $\mathcal{T} = \mathcal{N} \times \mathcal{AD}$. The security of a tweakable block cipher is defined as (strong) indistinguishability from *tweakable* random permutation ($\widetilde{(\pm)\mathrm{PRP}}$), which is a random permutation parameterized by tweak T. To adapt this notion to a VIL cipher, we introduce an additional length parameter. Let $\widetilde{\mathcal{P}}_\ell$ represent the set of all tweakable permutations on $\{0,1\}^\ell$. For each pair $(T, \ell) \in \mathcal{T} \times \mathbb{N}$, we define $\widetilde{\pi}_{T,\ell}(\cdot)$ as a tweakable permutation sampled independently and uniformly at random from $\widetilde{\mathcal{P}}_\ell$.

Lemma 1 (TRP/RND Switching Lemma)**.** *If each tweak T queried by an adversary $\mathcal{A}$ is distinct, then*

$$\Pr[\mathcal{A}^{\widetilde{\pi}_{T,\ell}(\cdot)} \Rightarrow 0] - \Pr[\mathcal{A}^{\$(\cdot)} \Rightarrow 0] = 0$$

for every $\ell \geq 0$, where $\widetilde{\pi}_{T,\ell} : \{0,1\}^\ell \to \{0,1\}^\ell$ is a tweakable random permutation and $\$$ is an oracle that samples a bitstring uniformly at random of length ℓ.

Proof. Consider that with an oracle that samples and outputs a random bitstring, the probability that a bitstring $L \in \{0,1\}^\ell$ is output to the adversary is $\frac{1}{2^\ell}$ at each query. On the other hand, in an oracle that implements random tweakable permutations, if the tweak does not repeat, it implies that a new tweakable random permutation is sampled for each query based on the tweak T. Thus the probability that M is mapped to the bitstring $L \in \{0,1\}^\ell$ is also $\frac{1}{2^\ell}$ at each query. Consequently, both oracles exhibit the same distribution, meaning that the adversary has 0 advantage in distinguishing between these two oracles. □

4.2 Proof of Security

Following [BMM+15], we also use counter as nonce when analyzing the stateful security. Here for simplicity we assume that $|\mathcal{N}| \geq q$ where q is the number of queries made by $\mathcal{A}$ and one can make the proof more rigorous by bounding the probability that a counter may repeat.

Theorem 3. *For any* IND-sf-CCLA2 *adversary $\mathcal{A}$ against the* EtE *construction* Π *making q_d decryption queries, there is a $\widetilde{\pm\mathrm{PRP}}$ adversary $\mathcal{A}_{stprp}$ against the tweakable* VIL *cipher $\widetilde{E}$ such that*

$$\mathbf{Adv}^{\text{IND-sf-CCLA2}}_{\Pi}(\mathcal{A}) \leq 3 \cdot \mathbf{Adv}^{\widetilde{\pm\mathrm{PRP}}}_{\widetilde{E}}(\mathcal{A}_{stprp}) + \frac{q_d}{2^\tau}$$

where τ is the minimum expansion parameter queried by $\mathcal{A}$.

Proof. The proof follows by combining Theorem 2 with Lemmas 2, 3 and 4. □

Lemma 2. *For any* IND-CPA *adversary $\mathcal{A}$ against the* EtE *construction* Π *making q encryption queries, there is a $\widetilde{\mathrm{PRP}}$ adversary $\mathcal{A}_{tprp}$ against the tweakable* VIL *cipher $\widetilde{E}$ such that*

$$\mathbf{Adv}^{\text{IND-CPA}}_{\Pi}(\mathcal{A}) = \mathbf{Adv}^{\widetilde{\mathrm{PRP}}}_{\widetilde{E}}(\mathcal{A}_{tprp}).$$

Proof (Sketch). We consider three games G_0 – G_2 where adversary's queries are answered with the tweakable VIL cipher $\widetilde{E}$, a tweakable random permutation $\widetilde{\pi}$, and a random bitstring of length $|M| + \tau$ respectively. We have that

$$\mathbf{Adv}_{\Pi}^{\text{IND-CPA}}(\mathcal{A}) = \sum_{i=0}^{1} \Pr[G_i(\mathcal{A})] - \Pr[G_{i+1}(\mathcal{A})].$$

Then we can bound $\Pr[G_0(\mathcal{A})] - \Pr[G_1(\mathcal{A})]$ by a $\widetilde{\text{PRP}}$ adversary $\mathcal{A}_{tprp}$. Following Lemma 1, we know that $\Pr[G_1(\mathcal{A})] - \Pr[G_2(\mathcal{A})] = 0$ since we assume counter does not repeat. □

Lemma 3. *For any* INT-sf-CTXT *adversary* $\mathcal{A}$ *against the* EtE *construction* Π *making* q *decryption queries, there is a* $\widetilde{\pm\text{PRP}}$ *adversary* $\mathcal{A}_{stprp}$ *against the tweakable* VIL *cipher* $\widetilde{E}$ *such that*

$$\mathbf{Adv}_{\Pi}^{\text{INT-sf-CTXT}}(\mathcal{A}) \leq \mathbf{Adv}_{\widetilde{E}}^{\widetilde{\pm\text{PRP}}}(\mathcal{A}_{stprp}) + \frac{q}{2^{\tau}}$$

where τ *is the minimum expansion parameter queried by* $\mathcal{A}$*.*

Proof (Sketch). We consider two games G_0 and G_1 where the adversary's encryption and decryption queries are answered with $\widetilde{E}$ and $\widetilde{E}^{-1}$, and $\widetilde{\pi}$ and $\widetilde{\pi}^{-1}$ respectively. We then have that

$$\mathbf{Adv}_{\Pi}^{\text{INT-sf-CTXT}}(\mathcal{A}) = \Pr[\mathcal{A} \text{ wins } G_0] - \Pr[\mathcal{A} \text{ wins } G_1] + \Pr[\mathcal{A} \text{ wins } G_1].$$

Similarly, we can bound $\Pr[\mathcal{A} \text{ wins } G_0] - \Pr[\mathcal{A} \text{ wins } G_1]$ by a $\widetilde{\pm\text{PRP}}$ adversary $\mathcal{A}_{stprp}$. Since we assume the counter does not repeat and we have a fresh permutation for each counter, the adversary wins G_1 when its query yields a bitstring ending with τ zeros, which is of probability at most $\frac{q}{2^{\tau}}$. □

We first define the leakage simulator function $\mathcal{L}$ for the EtE paradigm. Consider that during the decryption, $M' = \widetilde{E}^{-1}_{K;N,A}(C)$ is first deciphered. Depending if M' ends with τ zeros, $\mathcal{L}$ outputs either $\top$ or M'. Notably, there is only one error which is when M' does not end with τ zeros. Thus $\mathcal{L}_K^{N,A,\tau}(C) = (M', \perp)$ for an invalid ciphertext C.

Lemma 4. *For any* IND-sf-EPL2 *adversary* $\mathcal{A}$ *against the* EtE *construction* Π *making* q *leakage queries, there is a* $\widetilde{\pm\text{PRP}}$ *adversary* $\mathcal{A}_{stprp}$ *against the tweakable* VIL *cipher* $\widetilde{E}$ *such that*

$$\mathbf{Adv}_{\Pi}^{\text{IND-sf-EPL2}}(\mathcal{A}) = \mathbf{Adv}_{\widetilde{E}}^{\widetilde{\pm\text{PRP}}}(\mathcal{A}_{stprp}).$$

Proof (Sketch). We consider three games G_0 – G_2 for the proof. In G_0, $\mathcal{A}$'s queries are answered with $\widetilde{E}$ and $\widetilde{E}^{-1}$ respectively. In G_1, $\mathcal{A}$'s queries are answered with $\widetilde{\pi}$ and $\widetilde{\pi}^{-1}$ respectively. In game G_2, a bitstring M_λ is sampled uniformly at random of length $|M'|$ and the oracle LEAK returns $(M_\lambda, \perp)$ to $\mathcal{A}$. We still answer $\mathcal{A}$'s encryption and decryption query with $\widetilde{\pi}$ and $\widetilde{\pi}^{-1}$ respectively. We have that

$$\mathbf{Adv}_{\Pi}^{\text{IND-sf-EPL2}}(\mathcal{A}) = \sum_{i=0}^{1} \Pr[G_i(\mathcal{A})] - \Pr[G_{i+1}(\mathcal{A})].$$

Similarly, we bound $\Pr[G_0(\mathcal{A})] - \Pr[G_1(\mathcal{A})]$ by a $\widetilde{\pm\text{PRP}}$ adversary $\mathcal{A}_{stprp}$. Since we assume that the counter does not repeat, the tweak used to decipher a ciphertext C is always new for a query made by the adversary. Following Lemma 1, G_1 and G_2 are identical, thus $\mathcal{A}$ has 0 advantage in distinguishing between them. □

Remark 2. From Lemma 2 – 4, we can observe that the confidentiality and indistinguishability of leaked plaintext are independent of the stretch parameter τ, and it only affects the level of authenticity provided. This aligns with the results in previous works including [HII+22, Kha24]. Notably, while the adversary can query with the stretch parameter, in practice, it is typically predetermined by the communicating parties (but not an actual adversary). Hence, opting for a larger stretch parameter is a natural choice to ensure authenticity.

Authenticity from Existing Redundancy. One key feature of Encode-then-Encipher paradigm is that: when there exists redundancy in the plaintext, such redundancy can be exploited to establish or enhance authenticity. We define the *density* of message space $\mathcal{M}$ to measure how redundant the message space is as in Definition 9.

Definition 9 (δ-dense)**.** Let $v : \{0,1\}^\ell \to \{\mathsf{true}, \mathsf{false}\}$ be a predicate for $\ell \in \mathbb{N}$. We say $\mathcal{M} \subseteq \{0,1\}^\ell$ is δ-dense with respect to the predicate v if

$$\Pr[\ \forall M \in \mathcal{M} : v(M) = \mathsf{true}\ ] \leq \delta.$$

In that case, a valid forgery must pass two checks simultaneously, that is, satisfying the predicate and ending with τ zeros. Thus we obtain a new bound for the integrity as in Corollary 4.

Corollary 4. *Assuming the message space $\mathcal{M}$ is δ-dense, then for any* INT-sf-CTXT *adversary $\mathcal{A}$ against the* EtE *construction Π making q decryption queries, there is a $\widetilde{\pm\mathrm{PRP}}$ adversary $\mathcal{A}_{stprp}$ against the tweakable* VIL *cipher $\widetilde{E}$ such that*

$$\mathbf{Adv}^{\text{INT-sf-CTXT}}_{\Pi}(\mathcal{A}) \leq \mathbf{Adv}^{\widetilde{\pm\mathrm{PRP}}}_{\widetilde{E}}(\mathcal{A}_{stprp}) + \frac{\delta q}{2^\tau}.$$

Leakage with Multiple Errors. Suppose that $\mathcal{M}$ is δ-dense, the possible leakage tuples are $(M', \perp_1)$ and $(M', \perp_2)$. Fixing $\perp_1$ as the error flag to be distinguished, for IND-EPL1 security which requires that the adversary should not see the error flag $\perp_2$, we can obtain a bound as in Corollary 5.

Corollary 5. *Assuming the message space $\mathcal{M}$ is δ-dense, then for any* IND-sf-EPL1 *adversary $\mathcal{A}$ against the* EtE *construction Π making q leakage queries, there is a $\widetilde{\pm\mathrm{PRP}}$ adversary $\mathcal{A}_{stprp}$ against the tweakable* VIL *cipher $\widetilde{E}$ such that*

$$\mathbf{Adv}^{\text{IND-sf-EPL1}}_{\Pi}(\mathcal{A}) \leq \mathbf{Adv}^{\widetilde{\pm\mathrm{PRP}}}_{\widetilde{E}}(\mathcal{A}_{stprp}) + \frac{q}{2^\tau} \tag{1}$$

if $\perp_1$ implies M' does not ends with τ zeros, or

$$\mathbf{Adv}^{\text{IND-sf-EPL1}}_{\Pi}(\mathcal{A}) \leq \mathbf{Adv}^{\widetilde{\pm\mathrm{PRP}}}_{\widetilde{E}}(\mathcal{A}_{stprp}) + \delta \tag{2}$$

if $\perp_1$ implies $v(M') = \mathsf{false}$.

Leakage with Merged Error. Note that the condition predicates both concern the same leaked plaintext M'. Following Observation 1, we can merge them into one leakage tuple $(M', \perp)$. Note that if M' passes one of the checks, the leaked plaintext may exhibit certain property e.g., with τ ending zeros. Nevertheless, the probability that the oracle in ideal world generates such a bitstring is the same, yielding the adversary no more advantage than flipping a coin. Thus the adversary has no advantage in distinguishing only by leaked plaintext.

Nevertheless, the adversary can break IND-sf-EPL1 security by looking for $\perp_2$ to identify the real world when one of the checks passes (since $\perp_1$ is always output in the

ideal world). After merging two errors into one, $\perp$ will be output instead of $\perp_2$ thus the adversary can no longer distinguish by error flag. This allows to reduce the adversary's advantage by removing the term $\frac{q}{2^\tau}$ from Equation 1, and δ from Equation 2, to obtain the bound for IND-sf-EPL2 advantage as in Corollary 6.

Corollary 6. *Assuming the message space $\mathcal{M}$ is δ-dense, then for any* IND-sf-EPL2 *adversary $\mathcal{A}$ against the* EtE *construction* Π *with an merged error, there is a* $\widetilde{\pm\text{PRP}}$ *adversary $\mathcal{A}_{stprp}$ against the tweakable* VIL *cipher $\widetilde{E}$ such that*

$$\mathbf{Adv}^{\text{IND-sf-EPL2}}_{\Pi}(\mathcal{A}) = \mathbf{Adv}^{\widetilde{\pm\text{PRP}}}_{\widetilde{E}}(\mathcal{A}_{stprp}).$$

5 Modification for Robust EEM

Some Observations on EEM. We revisit the Encode-then-Encrypt-then-MAC as in Example 1. This also explains the intuition of keeping only one error in IND-CCLA2 as discussed in Paragraph 3.1.

We have that $\mathcal{L}_K(C) \in \{(\varepsilon, \perp_1), (M, \perp_2)\}$ for an invalid ciphertext C, where $\perp_1$ indicates the failure with check on tag, $\perp_2$ indicates the incorrect encoding, and M represents the bitstring that has not undergone the decoding process. Following the result in [BDPS14], it is easy to see that if the encryption scheme is IND-CPA and the MAC scheme is SUF-CMA, then EEM is IND-CCLA1 secure by setting $\perp_i = \perp_1$ in the notion definition.

Now suppose that the tag check fails to operate correctly (such that all ciphertexts being deemed valid), then there is an unavoidable leakage of plaintext (in incorrect encoding) since the encoding check has to be performed on plaintext. Thus the adversary sees $(M, \perp_2)$. Then the adversary immediately knows that the tag check is broken and the only error now is with the encoding. Then the adversary can continue with other attacks based on the error message and leaked plaintext.

Similarly, if the adversary manages to forge a tag, then the adversary is certain that its strategy is successful and can adopt that in the future forgery.

Proposition 7. EEM *is not* IND-CCLA2 *secure.*

Proof (Sketch). We first suppose the possible leakage tuples are $(\varepsilon, \perp_1)$ and $(M, \perp_2)$. Then it is trivially not IND-CCLA2 secure.

Now suppose the scheme forcefully combines $\perp_1$ and $\perp_2$ to produce a single error $\perp$, the leakage function's output becomes $(M, \perp)$. In this case, the adversary can simply exploit the ciphertext obtained from a previous encryption query and change the tag to an invalid one to distinguish M from a random bitstring. □

5.1 The Construction

We present the construction for an enhanced version of Encode-then-Encrypt-then-MAC as described in Figure 4. Following the discussion of authenticity from verifiable redundancy in [BR00], in this construction, we can take the encoding into consideration during the authenticity-check. Thus the authenticity is guaranteed if the encoding is correct. We assume that an encoding scheme $\mathsf{Encode} = (\textsc{Encode}, \textsc{Decode})$ is applied on certain part of the plaintext. As with the example of PKCS padding, we either fill the last block or create a new block full of paddings. Following Definition 9, we say Encode is a δ-dense encoding scheme if

$$\Pr[\ \forall M \in \mathcal{M} : \textsc{Decode}(M) \neq \mathsf{inval}\ ] \leq \delta$$

where $\textsc{Decode}(M) = \mathsf{inval}$ means that M does not follow the correct encoding.

$\mathsf{rEEM}[\Pi, \widetilde{E}, H_1, H_2].\mathcal{E}^{N,A,\tau}_{K_E,K_M,K_H}(M)$

1 : $M_L||M_R \leftarrow M$
2 : $N' \leftarrow H_{1K_H}(N, M_R, A, \tau)$
3 : $C_L \leftarrow \Pi.\mathcal{E}^{N'}_{K_E}(M_L)$
4 : $M^*_R \leftarrow \textsc{Encode}(M_R)$
5 : $M'_R \leftarrow M^*_R||0^\tau$
6 : $h \leftarrow H_{2K_H}(C_L, A, \tau)$
7 : $C_R \leftarrow \widetilde{E}_{K_M;h}(M'_R)$
8 : $C \leftarrow C_L||C_R$
9 : **return** C

$\mathsf{rEEM}[\Pi, \widetilde{E}, H_1, H_2].\mathcal{D}^{N,A,\tau}_{K_E,K_M,K_H}(C)$

1 : $C_L||C_R \leftarrow C$
2 : $h \leftarrow H_{2K_H}(C_L, A, \tau)$
3 : $M'_R \leftarrow \widetilde{E}^{-1}_{K_M;h}(C_R)$
4 : $\ell \leftarrow |M'_R|$
5 : $M^*_R \leftarrow M'_R[0..(\ell - \tau - 1)]$
6 : $\mathsf{pad} \leftarrow M'_R[(\ell - \tau)..(\ell - 1)]$
7 : **if** $\mathsf{pad} \neq 0^\tau \vee \textsc{Decode}(M^*_R) = \mathsf{inval}$ **then**
8 : **return** $\perp$
9 : $M_R \leftarrow \textsc{Decode}(M^*_R)$
10 : $N' \leftarrow H_{1K_H}(N, M_R, A, \tau)$
11 : $M_L \leftarrow \Pi.\mathcal{D}^{N'}_{K_E}(C_L)$
12 : **return** $M_L||M_R$

Figure 4: Robust EEM (rEEM) as a composition of nonce-based SE and Encode-then-Encipher paradigm. In line 7 of the decryption function, we let Decode return $\perp$ if M^*_R does not follow the correct encoding.

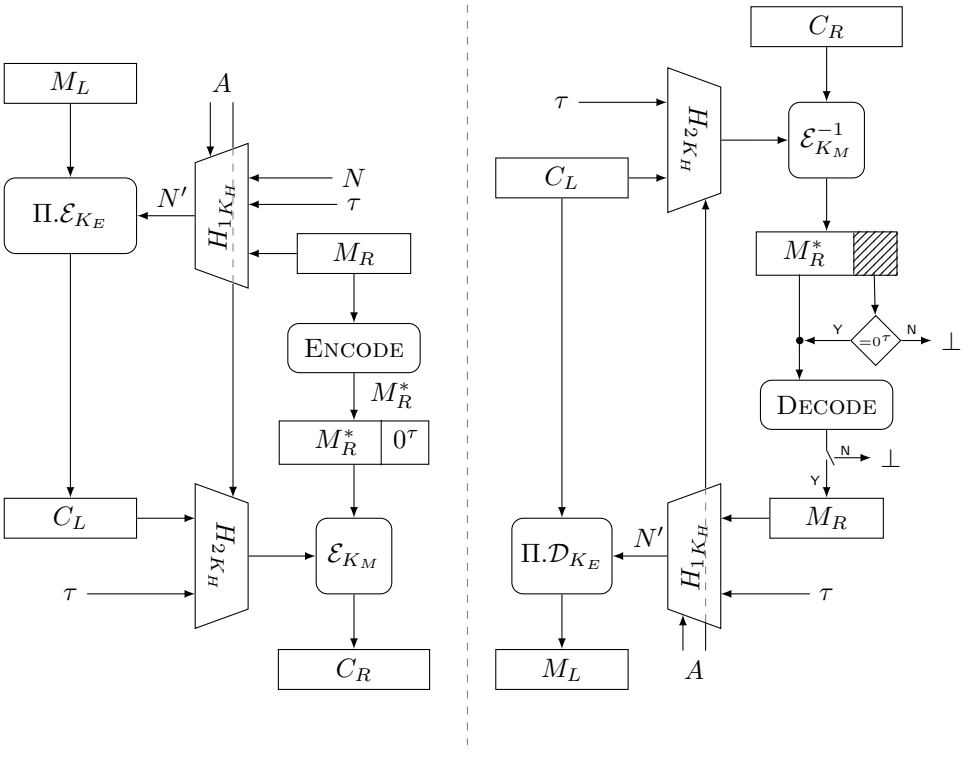

Figure 5: Graphic illustration for robust EEM. **Left**: Encryption function of rEEM. **Right**: Decryption function of rEEM.

During encryption, we first partition the plaintext into two segments M_L and M_R where M_R is to be encoded. We assume the length of M_L is a multiple of the block size. Thus we can use a symmetric encryption scheme to encrypt M_L to obtain the ciphertext

C_L. Then we encode M_R and pad the encoding bitstring with τ zeros after it, where τ is the stretch selected by the user. We let the resulted bitstring be M'_R. To connect the left and right part, we compute the hash of C_L, the associated data A, and the stretch τ. We use a tweakable VIL cipher with the hash as tweak to encipher M'_R to obtain the ciphertext C_R. The final ciphertext is the concatenation $C_L||C_R$.

During the decryption, we similarly partition the ciphertext C into C_L and C_R. We then reverse the encryption procedure to first decipher C_R, thereby yielding bitstring M'_R. We can then check whether M'_R ends with τ zeroes and the rest part of M'_R can be decoded correctly. If not, it indicates a authentication failure and the decryption process halts. Otherwise, the decryption process continues with the decryption of C_L, leading to the retrieval of the plaintext segment M_L. Finally, the concatenation of M_L and M_R forms the plaintext as output.

Comparison with Standard EEM. We let predicates v_1 and v_2 represent the failure conditions where

$$v_1(M'_R) = \text{“}M'_R[(\ell-\tau)..(\ell-1)] \neq 0^\tau\text{”}$$

and

$$v_2(M'_R) = \text{“}\textsc{Decode}(M'_R[0..(\ell-\tau-1)]) = \mathsf{inval}\text{”}.$$

Following Observation 1, this allows us to create a predicates $v'(M'_R) = v_1(M'_R) \vee v_2(M'_R)$ in Line 7 of Figure 4. Then $\mathcal{A}$'s query has to make $v'(\cdot) = v_1(\cdot) = v_2(\cdot) = \mathsf{false}$ to obtain a meaningful plaintext. Furthermore, without loss of generality, we suppose that v_1 fails to functions properly such that $v_1(M'_R) = \mathsf{false}$ for all M'_R. Then the adversary is not aware such a failure as long as v_2 function normally. On the other hand, suppose the adversary's query passes one check e.g. yielding $v_1(M'_R) = \mathsf{false}$. This does not guarantee the adversary that its strategy is effective in generating a bitstring like that, following the discussion on indistinguishability of M'_R from a random bitstring in Paragraph 4.2.

Nonce-Misuse Resistance. When instantiating our construction with a nonce-based encryption scheme, we follow the *Feistel structure* [Fei73] to provide nonce-misuse resistance. On the input of a tuple (N, M, A, τ), we compute the hash $H(N, M_R, A, \tau)$ and use the hash as the new nonce for encrypting M_L. The Feistel structure guarantees the uniqueness of the tuple (N, M, A, τ). Changing one of (N, M_R, A, τ) yields a new nonce. Otherwise, the adversary has to change M_L since query with a repeated tuple is not allowed, which yields a new C_L and thus a new C_R since the tweak for enciphering M'_R has changed.

Plaintext Awareness Security. We consider the indistinguishability of M_L and M_R separately. The probability that the adversary makes a valid forgery follows Lemma 6, and the probability that the adversary distinguish the deciphered string M'_R from a random bitstring follows Lemma 7.

For M_L, as long as M_R is not recovered by the adversary, M_L still remains confidential thanks to the Feistel structure. That is because the nonce for decrypting C_L is computed based on M_R with a universal hash function. If M^*_R does not satisfy the encoding, the part M_R that is used to computed the nonce cannot be extracted from M^*_R. Thus further decryption is not possible.

5.2 Security

Theorem 4. *Let* Encode *be a δ-dense encoding scheme, then for any* IND-CCLA2 *adversary $\mathcal{A}$ making q_d decryption queries and q_l leakage queries againsnt $\Psi = \mathsf{rEEM}[\Pi, \widetilde{E}, H_1, H_2]$, there exists an* IND-CPA *adversary $\mathcal{A}_{cpa}$ against Π, a $\widetilde{\pm\text{PRP}}$ adversary $\mathcal{A}_{stprp}$ against $\widetilde{E}$*

such that

$$\mathbf{Adv}^{\text{IND-CCLA2}}_{\Psi}(\mathcal{A}) \leq \mathbf{Adv}^{\text{IND-CPA}}_{\Pi}(\mathcal{A}_{cpa}) + 3 \cdot \mathbf{Adv}^{\widetilde{\pm\text{PRP}}}_{\widetilde{E}}(\mathcal{A}_{stprp}) + 3\epsilon + \frac{\delta q_d}{2^\tau} + \frac{q_l^2}{2^\tau}$$

where τ is the minimum stretch parameter queried by $\mathcal{A}$.

Proof. The proof follows by combining Theorem 1 with Lemmas 5, 6, and 7. □

Lemma 5. *For any* IND-CPA *adversary $\mathcal{A}$ against $\Psi = \mathsf{rEEM}[\Pi, \widetilde{E}, H_1, H_2]$ making q encryption queries, there exists an* IND-CPA *adversary $\mathcal{A}_{cpa}$ against Π, and a $\widetilde{\text{PRP}}$ adversary $\mathcal{A}_{tprp}$ against $\widetilde{E}$ such that*

$$\mathbf{Adv}^{\text{IND-CPA}}_{\Psi}(\mathcal{A}) \leq \mathbf{Adv}^{\text{IND-CPA}}_{\Pi}(\mathcal{A}_{cpa}) + \mathbf{Adv}^{\widetilde{\text{PRP}}}_{\widetilde{E}}(\mathcal{A}_{tprp}) + 2\epsilon.$$

Proof (Sketch). We need to show that the left and the right segments of the ciphertext are both indistinguishable from a random bitstring. For C_L, if a collision happens with H, then the nonce N' repeats yielding trivial distinguishing, which is bounded by probability ϵ. Otherwise, we can reduce it to the IND-CPA security of the encryption scheme Π.

For C_R, we follow a similar proof as in Lemma 2. Notably, a tweak repeats when the collision happens with H, which happens with probability at most ϵ. □

Lemma 6. *Let* Encode *be a δ-dense encoding scheme, then for any* INT-CTXT *adversary $\mathcal{A}$ against $\Psi = \mathsf{rEEM}[\Pi, \widetilde{E}, H_1, H_2]$ making q decryption queries, there exists an $\widetilde{\pm\text{PRP}}$ adversary $\mathcal{A}_{stprp}$ against $\widetilde{E}$ and such that*

$$\mathbf{Adv}^{\text{INT-CTXT}}_{\Psi}(\mathcal{A}) \leq \mathbf{Adv}^{\widetilde{\pm\text{PRP}}}_{\widetilde{E}}(\mathcal{A}_{stprp}) + \epsilon + \frac{\delta q}{2^\tau}$$

where τ is the minimum stretch parameter queried by $\mathcal{A}$.

Proof (Sketch). Observe that the authenticity of the construction only depends on C_R. We first consider two cases when a tweak used in decryption matches with a tweak used in a previous encryption query. Let $C = C_L||C_R$ be the result of that previous encryption query. In the first case, a collision happens with the hash function H, the the adversary can simply reuse C_R and change one of C_L, N and A to provoke a collision in H. This happens with probability at most ϵ. On the other hand, if $\mathcal{A}$ reuses (C_L, A, τ), then $\mathcal{A}$ has to query with C'_R different from C_R. In this case, C'_R has to be deciphered with a random permutation to a bitstring that can be decoded successfully and ends with τ zeros, which happens with probability at most $\frac{\delta q}{2^\tau}$.

Otherwise if each tweak used in decryption is distinct from that in previous encryption queries, we then have a fresh random permutation for every decryption query. Then $\mathcal{A}$ wins the game only when the ciphertext deciphers to a bitstring that ends with τ leading zeros and can be decoded successfully, which is of probability at most $\frac{\delta q}{2^\tau}$. □

The leakage simulator function is similar as in EtE construction but on C_R, that is, $\mathcal{L}_K^{N,A,\tau}(C) = \mathcal{L}_K^{N,A,\tau}(C_R) = (M'_R, \perp)$. If the authenticity fails, C_L remains undecrypted, which means no leaked plaintext then.

Lemma 7. *For any* IND-EPL2 *adversary $\mathcal{A}$ against $\Psi = \mathsf{rEEM}[\Pi, \widetilde{E}, H_1, H_2]$ making q leakage queries, there exists a $\widetilde{\pm\text{PRP}}$ adversary $\mathcal{A}_{stprp}$ against $\widetilde{E}$ such that*

$$\mathbf{Adv}^{\text{IND-EPL2}}_{\Psi}(\mathcal{A}) \leq \mathbf{Adv}^{\widetilde{\pm\text{PRP}}}_{\widetilde{E}}(\mathcal{A}_{stprp}) + \frac{q^2}{2^\tau}.$$

Proof (Sketch). Observe that the leakage only concerns M'_R. If each tweak in LEAK is distinct, then adversary has 0 advantage in distinguishing following Lemma 1. To repeat a tweak, the adversary can simply repeat the query with the same tuple (N, A, C_L, τ). In this case, the adversary may look for repeated output to distinguish the ideal world from the random world (since the real world implements a permutation). This happens when the oracle in ideal world samples the same bitstring, with probability at most $\frac{q^2-q}{2^{|M'_R|}} \leq \frac{q^2}{2^\tau}$. □

6 Conclusion and Future Work

We introduce a new notion IND-CCLA to formalize the robustness of AEAD scheme. Our notion can be seen as an extension from commonly accepted notions including IND-CCA3 by considering an additional oracle to capture the security when there is leakage due to decryption failure. We consider the situation where there are multiple failure conditions. We introduce the security requirement that the leaked plaintext is indistinguishable from a random bitstring of the minimum leakage. On top of this, we define two security notions regarding the error messages where one requires that the adversary should not be able trigger more errors than the first error. In the other one, we introduce the concept of error unicity by requiring the disclosure of a single error whether implicitly or explicitly even there are multiple checks for errors. This ensures that the adversary cannot distinguish a failure if one of the check fails to work in a scheme containing multiple checks. This also ensures that the adversary cannot be certain that its attack is effective even if the query passes one of the checks.

We further extend this notion to IND-sf-CCLA to capture the stateful security where out-of-order ciphertexts are queried by the adversary. We prove the stateful security of the Encode-then-Encipher (EtE) paradigm which is a mainstream way to construct robust AE by using counter for nonce.

We revisit the robustness of Encode-then-Encrypt-then-MAC (EEM) paradigm by examining the leakage when decryption fails. We propose a modification to the paradigm by replacing the MAC with Encode-then-Encipher on partial plaintext. This modification merges two checks into one with minimal plaintext leakage, which allows the scheme to reveal only one error message to achieve error unicity.

Our study evaluates the disparity between a single-error decryption function and the actual leakage incurred during the decryption. This shows that even though a scheme only discloses a single error in the decryption function, the actual leakage may still grant the adversary more advantage than decryption. Our modification to EEM serves a very preliminary example mitigation to achieve error unicity. Further work may be done including introducing a more rigorous notion than error unicity, and designing securer schemes to guarantee the security when multiple checks for errors are present.

References

[ABL+14] Elena Andreeva, Andrey Bogdanov, Atul Luykx, Bart Mennink, Nicky Mouha, and Kan Yasuda. How to securely release unverified plaintext in authenticated encryption. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 105–125. Springer, Heidelberg, December 2014. doi:10.1007/978-3-662-45611-8_6.

[AFL+16] Farzaneh Abed, Christian Forler, Eik List, Stefan Lucks, and Jakob Wenzel. RIV for robust authenticated encryption. In Thomas Peyrin, editor, *FSE 2016*, volume 9783 of *LNCS*, pages 23–42. Springer, Heidelberg, March 2016. doi:10.1007/978-3-662-52993-5_2.

[AR02] Martín Abadi and Phillip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *Journal of Cryptology*, 15(2):103–127, March 2002. doi:10.1007/s00145-001-0014-7.

[BDPS14] Alexandra Boldyreva, Jean Paul Degabriele, Kenneth G. Paterson, and Martijn Stam. On symmetric encryption with distinguishable decryption failures. In Shiho Moriai, editor, *FSE 2013*, volume 8424 of *LNCS*, pages 367–390. Springer, Heidelberg, March 2014. doi:10.1007/978-3-662-43933-3_19.

[BKN04] Mihir Bellare, Tadayoshi Kohno, and Chanathip Namprempre. Breaking and provably repairing the ssh authenticated encryption scheme: A case study of the encode-then-encrypt-and-mac paradigm. *ACM Transactions on Information and System Security (TISSEC)*, 7(2):206–241, 2004.

[BMM+15] Christian Badertscher, Christian Matt, Ueli Maurer, Phillip Rogaway, and Björn Tackmann. Robust authenticated encryption and the limits of symmetric cryptography. In Jens Groth, editor, *15th IMA International Conference on Cryptography and Coding*, volume 9496 of *LNCS*, pages 112–129. Springer, Heidelberg, December 2015. doi:10.1007/978-3-319-27239-9_7.

[BN00] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In Tatsuaki Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 531–545. Springer, Heidelberg, December 2000. doi:10.1007/3-540-44448-3_41.

[BPS15] Guy Barwell, Daniel Page, and Martijn Stam. Rogue decryption failures: Reconciling AE robustness notions. In Jens Groth, editor, *15th IMA International Conference on Cryptography and Coding*, volume 9496 of *LNCS*, pages 94–111. Springer, Heidelberg, December 2015. doi:10.1007/978-3-319-27239-9_6.

[BR00] Mihir Bellare and Phillip Rogaway. Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient cryptography. In Tatsuaki Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 317–330. Springer, Heidelberg, December 2000. doi:10.1007/3-540-44448-3_24.

[BR06] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Heidelberg, May / June 2006. doi:10.1007/11761679_25.

[CHVV03] Brice Canvel, Alain P. Hiltgen, Serge Vaudenay, and Martin Vuagnoux. Password interception in a SSL/TLS channel. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 583–599. Springer, Heidelberg, August 2003. doi:10.1007/978-3-540-45146-4_34.

[DP07] Jean Paul Degabriele and Kenneth G. Paterson. Attacking the IPsec standards in encryption-only configurations. In *2007 IEEE Symposium on Security and Privacy*, pages 335–349. IEEE Computer Society Press, May 2007. doi:10.1109/SP.2007.8.

[DP10] Jean Paul Degabriele and Kenneth G. Paterson. On the (in)security of IPsec in MAC-then-encrypt configurations. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM CCS 2010*, pages 493–504. ACM Press, October 2010. doi:10.1145/1866307.1866363.

[Fei73] Horst Feistel. Cryptography and computer privacy. *Scientific american*, 228(5):15–23, 1973.

[HII+22] Akinori Hosoyamada, Akiko Inoue, Ryoma Ito, Tetsu Iwata, Kazuhiko Mimematsu, Ferdinand Sibleyras, and Yosuke Todo. Cryptanalysis of Rocca and feasibility of its security claim. *IACR Trans. Symm. Cryptol.*, 2022(3):123–151, 2022. doi:10.46586/tosc.v2022.i3.123-151.

[HKR15] Viet Tung Hoang, Ted Krovetz, and Phillip Rogaway. Robust authenticated-encryption AEZ and the problem that it solves. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 15–44. Springer, Heidelberg, April 2015. doi:10.1007/978-3-662-46800-5_2.

[Hou09] S. Housley. Cryptographic Message Syntax (CMS). RFC 5652, IETF, September 2009. https://datatracker.ietf.org/doc/html/rfc5652#section-6.3.

[Kha24] Mustafa Khairallah. CCA security with short AEAD tags. *IACR Communications in Cryptology*, 1(1), 2024. doi:10.62056/aevua69p1.

[LRW02] Moses Liskov, Ronald L. Rivest, and David Wagner. Tweakable block ciphers. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 31–46. Springer, Heidelberg, August 2002. doi:10.1007/3-540-45708-9_3.

[Mau11] Ueli Maurer. Constructive cryptography – a new paradigm for security definitions and proofs. In S. Moedersheim and C. Palamidessi, editors, *Theory of Security and Applications (TOSCA 2011)*, volume 6993 of *Lecture Notes in Computer Science*, pages 33–56. Springer-Verlag, 4 2011.

[PRS11] Kenneth G. Paterson, Thomas Ristenpart, and Thomas Shrimpton. Tag size does matter: Attacks and proofs for the TLS record protocol. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 372–389. Springer, Heidelberg, December 2011. doi:10.1007/978-3-642-25385-0_20.

[RBBK01] Phillip Rogaway, Mihir Bellare, John Black, and Ted Krovetz. OCB: A block-cipher mode of operation for efficient authenticated encryption. In Michael K. Reiter and Pierangela Samarati, editors, *ACM CCS 2001*, pages 196–205. ACM Press, November 2001. doi:10.1145/501983.502011.

[Rog02] Phillip Rogaway. Authenticated-encryption with associated-data. In Vijayalakshmi Atluri, editor, *ACM CCS 2002*, pages 98–107. ACM Press, November 2002. doi:10.1145/586110.586125.

[Shr04] Tom Shrimpton. A characterization of authenticated-encryption as a form of chosen-ciphertext security. Cryptology ePrint Archive, Report 2004/272, 2004. https://eprint.iacr.org/2004/272.

[Vau02] Serge Vaudenay. Security flaws induced by CBC padding - applications to SSL, IPSEC, WTLS... In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 534–546. Springer, Heidelberg, April / May 2002. doi:10.1007/3-540-46035-7_35.

A Detailed Proofs

A.1 Proof of Lemma 2

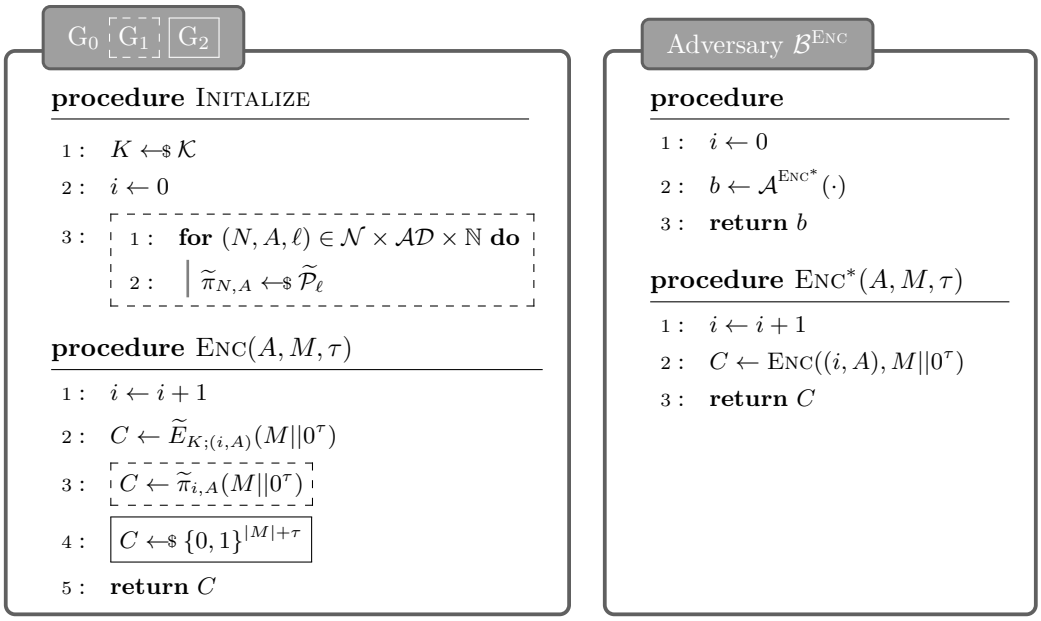

Figure 6: Left: Games G_0 – G_2 for proof of Lemma 2. Dot-boxed code is exclusive to G_1 and Frame-boxed code is exclusive to G_2. **Right**: $\widetilde{\text{PRP}}$ adversary $\mathcal{B}$ for proof for proof of Lemma 2.

Proof. We consider three games G_0 – G_2 as in Figure 6 and we use a counter i as nonce. In G_0, the encryption is done with the tweakable VIL cipher $\widetilde{E}$ and the oracle first appends τ zeros after M and returns $\widetilde{E}_{K;(i,A)}(M||0^\tau)$ as output. In G_1, the oracle samples a tweakable random permutation $\widetilde{\pi}$ and return $\widetilde{\pi}_{i,A}(M||0^\tau)$ as output. In G_2, the oracles sample a bitstring uniformly at random from $\{0,1\}^{|M|+\tau}$ and returns it as output. Then we have that

$$\mathbf{Adv}^{\text{IND-CPA}}_{\Pi}(\mathcal{A}) = \sum_{i=0}^{1} \Pr[G_i(\mathcal{A})] - \Pr[G_{i+1}(\mathcal{A})].$$

We can then construct a $\widetilde{\text{PRP}}$ adversary $\mathcal{B}$ from $\mathcal{A}$ as in Figure 6. We construct the simulated encryption oracle ENC* for $\mathcal{A}$ such that for each encryption query made by $\mathcal{A}$, we let $\mathcal{B}$ append τ zeros after it and forward it to $\mathcal{B}$'s oracle ENC, then $\mathcal{B}$ forwards the response from ENC to $\mathcal{A}$. We then let $\mathcal{B}$ return the same b that $\mathcal{A}$ returns. We then have that

$$\mathbf{Adv}^{\widetilde{\text{PRP}}}_{\widetilde{E}}(\mathcal{B}) = \Pr[G_0(\mathcal{A})] - \Pr[G_1(\mathcal{A})].$$

Since we use counter for the nonce and we assume that counter does not repeat, we know that the tweak never repeats, following Lemma 1, we have that

$$\Pr[G_1(\mathcal{A})] - \Pr[G_2(\mathcal{A})] = 0.$$

Finally, we have that

$$\mathbf{Adv}^{\text{IND-CPA}}_{\Pi}(\mathcal{A}) = \mathbf{Adv}^{\widetilde{\text{PRP}}}_{\widetilde{E}}(\mathcal{B}).$$

□

A.2 Proof of Lemma 3

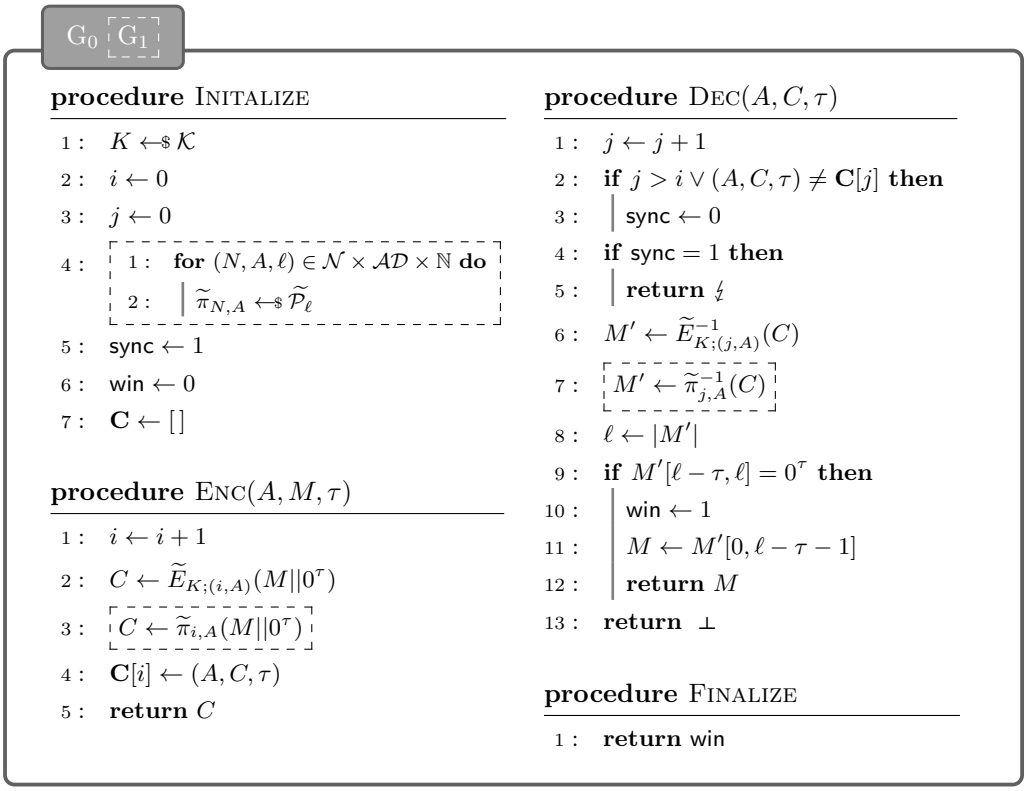

Figure 7: Games G_0 – G_1 for proof of Lemma 3. Dot-boxed code is exclusive to G_1.

Proof. We consider two games G_0 and G_1 as in Figure 7 for the proof. In G_0, $\mathcal{A}$'s queries are answered with $\widetilde{E}$ and $\widetilde{E}^{-1}$ respectively. In game G_1, the oracle samples a tweakable random permutation and answer $\mathcal{A}$'s query with $\widetilde{\pi}$ and $\widetilde{\pi}^{-1}$ respectively. We then have that

$$\begin{aligned}\mathbf{Adv}^{\text{INT-sf-CTXT}}_{\Pi}(\mathcal{A}) = &\Pr[G_0(\mathcal{A}) \Rightarrow 1] - \Pr[G_1(\mathcal{A}) \Rightarrow 1] \\ &+ \Pr[G_1(\mathcal{A}) \Rightarrow 1].\end{aligned}$$

Note that we can then construct a $\widetilde{\pm\text{PRP}}$ adversary $\mathcal{B}$ as described in Figure 8 against the tweakable VIL cipher $\widetilde{E}$ with $\mathcal{A}$ as subroutine. We define the simulated oracle Enc* for $\mathcal{A}$ such that for each $\mathcal{A}$'s encryption query, $\mathcal{B}$ first appends the τ zeros after the message then forwards it to its oracle Enc, and returns the result that $\mathcal{B}$ obtains from Enc to $\mathcal{A}$. Similarly, we define the simulated oracle Dec* for $\mathcal{A}$ such that for each $\mathcal{A}$'s decryption query, $\mathcal{B}$ returns $\lightning$ to $\mathcal{A}$ if it is in-order. Otherwise, $\mathcal{B}$ forwards the query to its oracle Dec. With the response, $\mathcal{B}$ checks if it ends with τ zeros and return $\perp$ or the plaintext accordingly. If $\mathcal{A}$ makes a valid forgery, then $\mathcal{B}$ returns 0, otherwise returns 1. We have that

$$\mathbf{Adv}^{\widetilde{\pm\text{PRP}}}_{\widetilde{E}}(\mathcal{B}) = \Pr[G_0(\mathcal{A}) \Rightarrow 1] - \Pr[G_1(\mathcal{A}) \Rightarrow 1].$$

Now we bound the probability that $\mathcal{A}$ wins in G_1. We consider two cases of $\mathcal{A}$'s queries. In first case, $\mathcal{A}$ queries a tuple (A, C, τ) that is the output of the oracle Enc. In this case, $\mathcal{A}$ has to make an out-of-order query, which means that the counter has been updated and a new random permutation will be used to decipher. Note that $\mathcal{A}$ wins if the deciphered

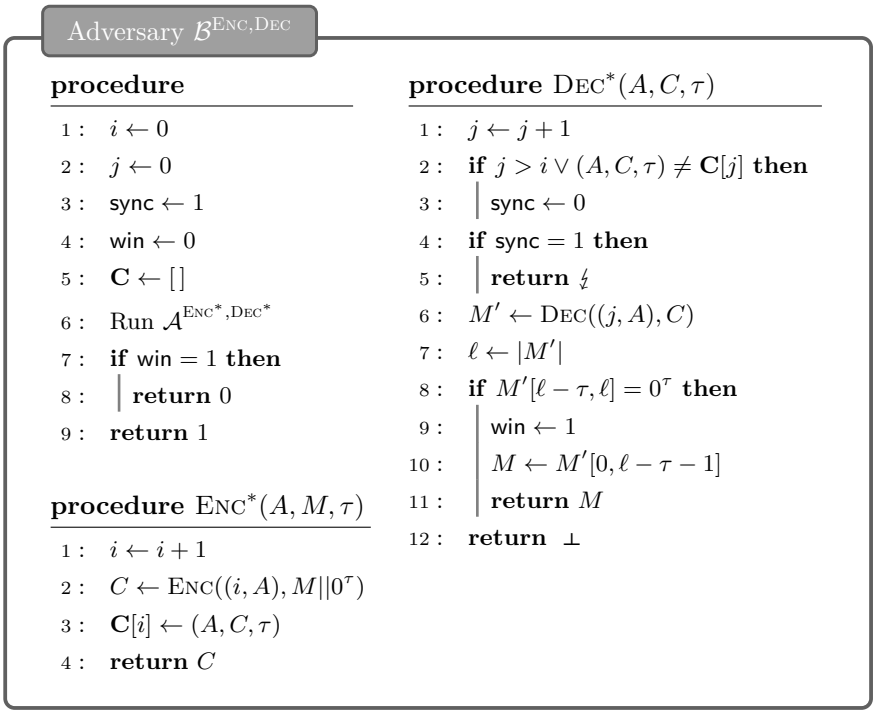

Figure 8: $\widetilde{\pm\text{PRP}}$ adversary $\mathcal{B}$ for proof of Lemma 3.

bitstring ends with τ zeros, which is of probability $\frac{q}{2^\tau}$. In the other case, (A, C, τ) has never been an output from ENC, then C is valid only if C deciphers to a bitstring with τ ending zeros with a random permutation, which is the same as the first case. Thus we have that

$$\Pr[\text{G}_1(\mathcal{A}) \Rightarrow 1] \leq \frac{q}{2^\tau}.$$

Finally, we have that

$$\mathbf{Adv}^{\text{INT-sf-CTXT}}_{\Pi}(\mathcal{A}) \leq \mathbf{Adv}^{\widetilde{\pm\text{PRP}}}_{\widetilde{E}}(\mathcal{B}) + \frac{q}{2^\tau}.$$

□

A.3 Proof of Lemma 4

Proof. We consider three games G_0 – G_2 as in Figure 9 for the proof. In G_0, $\mathcal{A}$'s queries are answered with $\widetilde{E}$ and $\widetilde{E}^{-1}$ respectively. In G_1, $\mathcal{A}$'s queries are answered with $\widetilde{\pi}$ and $\widetilde{\pi}^{-1}$ respectively. In game G_2, a bitstring M_λ is sampled uniformly at random of length $|M'|$ and the oracle LEAK returns $(M_\lambda, \perp)$ to $\mathcal{A}$. However, we still answer $\mathcal{A}$'s encryption and decryption query with $\widetilde{\pi}$ and $\widetilde{\pi}^{-1}$ respectively. We have that

$$\mathbf{Adv}^{\text{IND-sf-EPL2}}_{\Pi}(\mathcal{A}) = \sum_{i=0}^{1} \Pr[\text{G}_i(\mathcal{A})] - \Pr[\text{G}_{i+1}(\mathcal{A})].$$

Now we show that we can construct an $\widetilde{\pm\text{PRP}}$ adversary $\mathcal{B}$ as in Figure 10. For $\mathcal{A}$'s encryption and decryption queries, we can construct simulated oracles ENC^* and DEC^* as described in proofs of Lemma 2 and 3. For the leakage query, $\mathcal{B}$ forwards the ciphertext queried by $\mathcal{A}$ to its oracle DEC. Then $\mathcal{B}$ returns $\top$ if it is a valid ciphertext, and otherwise

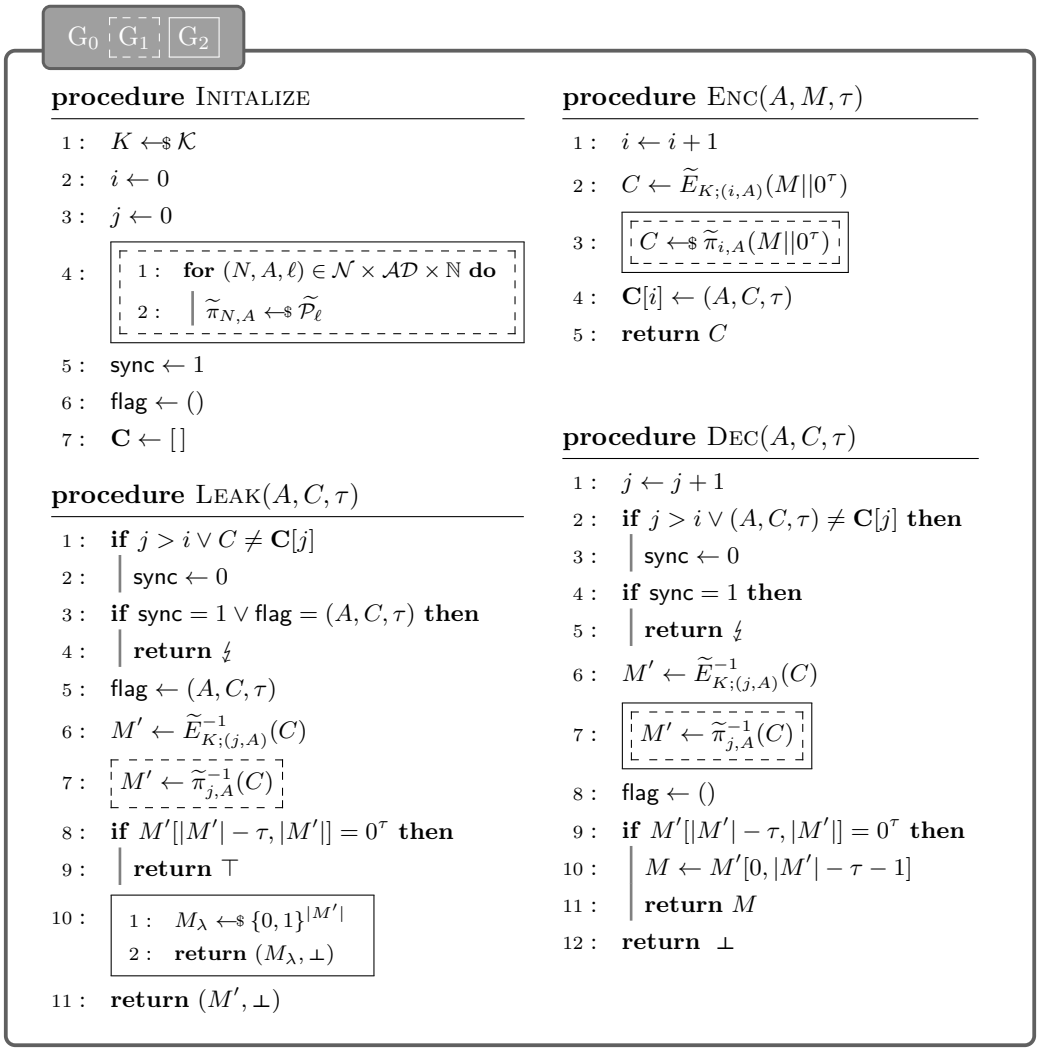

Figure 9: Game G_0 – G_2 for the proof of Lemma 4. Dot-boxed code is exclusive to G_1. Frame-boxed code is exclusive to G_2. Doubly-boxed code is for both G_1 and G_2.

returns $(M', \perp)$ to $\mathcal{A}$ where M' is the deciphered bitstring. We let $\mathcal{B}$ return the same bit b that $\mathcal{A}$ returns. We then have that

$$\mathbf{Adv}^{\widetilde{\pm\text{PRP}}}_{E}(\mathcal{B}) = \Pr[G_0(\mathcal{A})] - \Pr[G_1(\mathcal{A})].$$

Notably, the behaviors of G_1 and G_2 are identical. Since we assume the counter does not repeat, the tweak used in the oracle Leak to decipher a ciphertext C is always new for a query made by $\mathcal{A}$. Following Lemma 1, the adversary has 0 advantage in distinguishing between the two worlds. On the other hand, even if the adversary's query yields a valid ciphertext, both G_1 and G_2 output $\top$. Thus the adversary still has 0 advantage in distinguishing between G_1 and G_2. Thus we have

$$\Pr[G_1(\mathcal{A})] - \Pr[G_2(\mathcal{A})] = 0.$$

Finally, we have that

$$\mathbf{Adv}^{\text{IND-sf-EPL2}}_{\Pi}(\mathcal{A}) = \mathbf{Adv}^{\widetilde{\pm\text{PRP}}}_{\widetilde{E}}(\mathcal{B}).$$

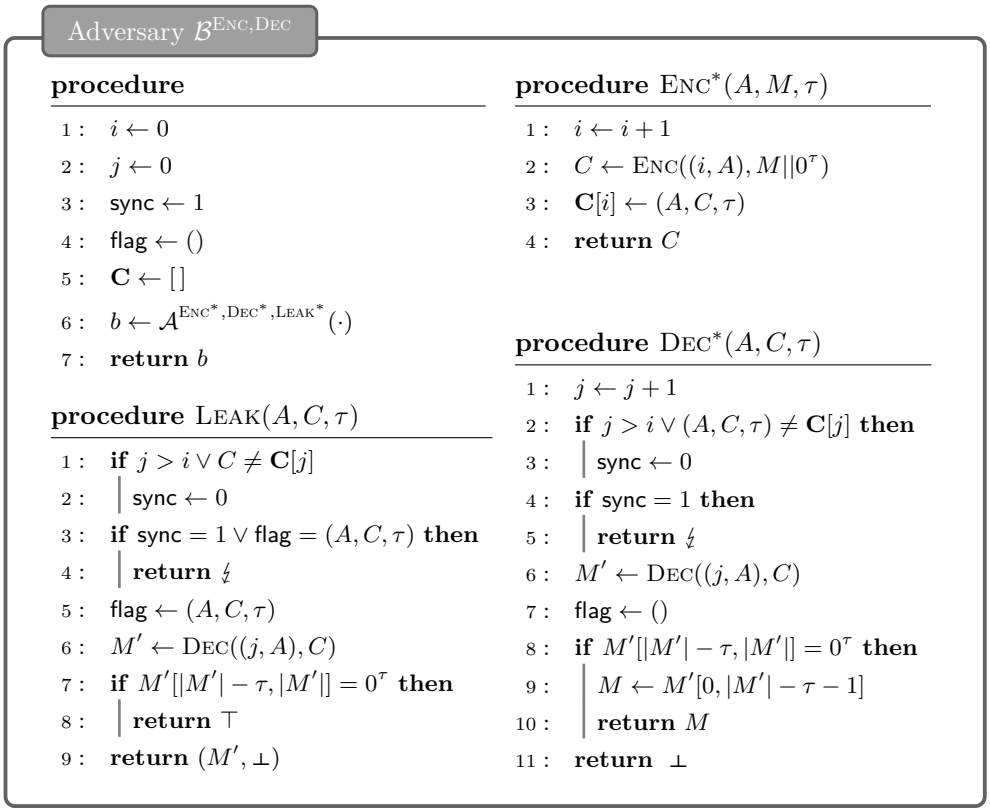

Figure 10: $\widetilde{\pm\text{PRP}}$ adversary $\mathcal{B}$ for the proof of Lemma 4.

□

A.4 Proof of Lemma 5

Proof. We need to show both the left and the right segments are indistinguishable from a random bitstring. We consider the games G_0 – G_3 as in Figure 11. We have that

$$\mathbf{Adv}^{\text{IND-CPA}}_{\Psi}(\mathcal{A}) = \sum_{i=0}^{2} \Pr[\text{G}_i(\mathcal{A})] - \Pr[\text{G}_{i+1}(\mathcal{A})]$$

In G_1, we replace C_L with a random bitstring of the same length. In G_2, we replace the tweakable cipher $\widetilde{E}$ with a tweakable random permutation $\widetilde{\pi}$. In G_3, we return a random bitstring of the length $|C_L||C_R|$.

Observe when the universal hash function H produces a repeated nonce, the adversary can change one of N, M_L and A and query again with M_R, which yields the same C_R as in a previous query. Otherwise, the indistinguishability of C_L depends on IND-CPA security of the encryption scheme Π. Then we can construct an IND-CPA adversary $\mathcal{B}_1$ as in Figure 11 against Π. By Union Bound, we have that

$$\Pr[\text{G}_0(\mathcal{A})] - \Pr[\text{G}_1(\mathcal{A})] \leq \mathbf{Adv}^{\text{IND-CPA}}_{\Pi}(\mathcal{B}_1) + \epsilon.$$

It leaves us to show that C_R is also indistinguishable from a random bitstring. We first construct a $\widetilde{\text{PRP}}$ adversary $\mathcal{B}_2$ as in Figure 11. We define the simulated oracle Enc^* for $\mathcal{A}$ such that for each query of $\mathcal{A}$, we let $\mathcal{B}$ sample a random bitstring C_R of length $|M_R|$,

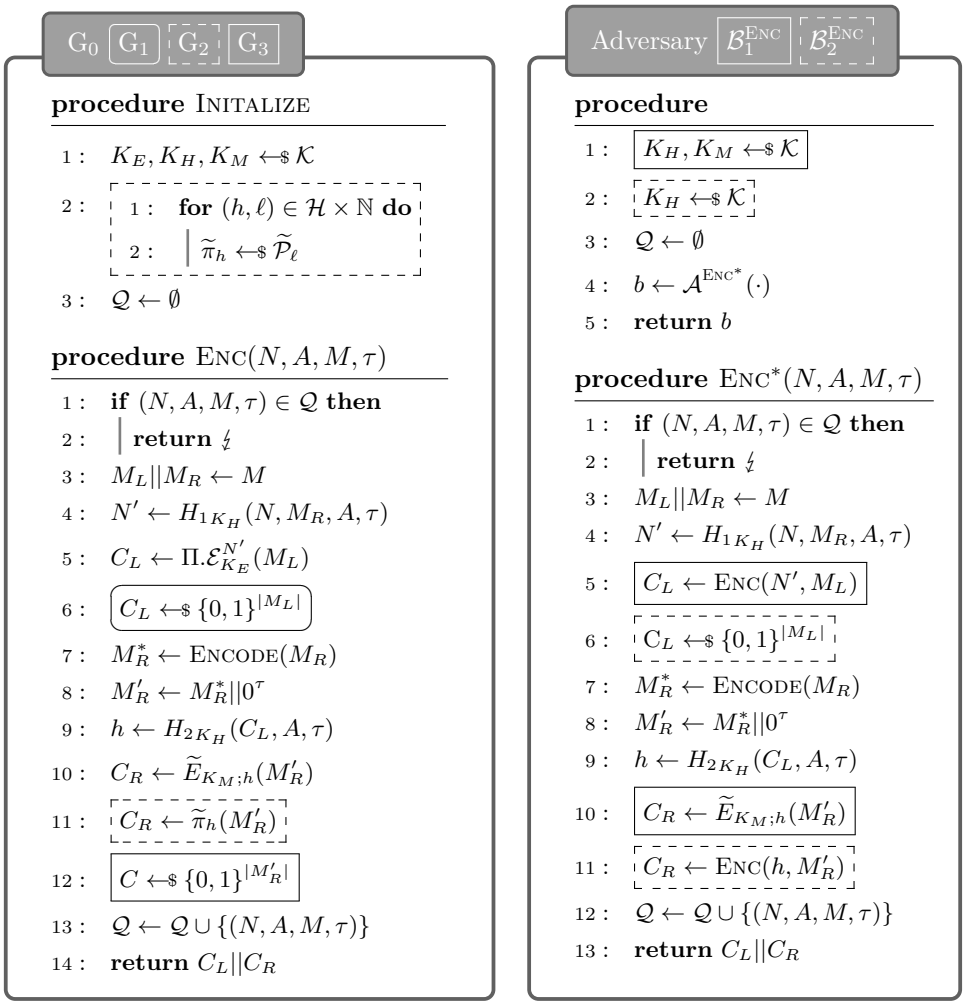

Figure 11: Left: Games G_0 – G_3 for proof of Lemma 5. Oval-boxed code is exclusive to G_1. Dot-boxed code is exclusive to G_2. Frame-boxed code is exclusive to G_3. **Right**: IND-CPA adversary $\mathcal{B}_1$ and $\widetilde{\text{PRP}}$ adversary $\mathcal{B}_2$ for proof of Lemma 5. Frame-boxed code is executed if $\mathcal{B}_1$ and dot-boxed code is executed if $\mathcal{B}_2$. Unboxed code is for both of the adversaries.

then $\mathcal{B}$ prepends τ zeros before M_L and queries its oracle Enc to get C_L. Then $\mathcal{B}$ returns $C_L||C_R$ to $\mathcal{A}$ as response. We let $\mathcal{B}$ returns the same bit b returned by $\mathcal{A}$. Thus we have that

$$\mathbf{Adv}^{\widetilde{\text{PRP}}}_{\widetilde{E}}(\mathcal{B}_2) = \Pr[G_1(\mathcal{A})] - \Pr[G_2(\mathcal{A})].$$

Now following Lemma 1, we know that the behaviors of G_2 and G_3 are identical unless a tweak repeats. Thus we provide a rough bound between G_2 and G_3 by ϵ where ϵ is the probability that the universal hash function H produces a repeated tweak h.

Finally, we have that

$$\mathbf{Adv}^{\text{IND-CPA}}_{\Psi}(\mathcal{A}) \leq \mathbf{Adv}^{\text{IND-CPA}}_{\Pi}(\mathcal{B}_1) + \mathbf{Adv}^{\widetilde{\text{PRP}}}_{\widetilde{E}}(\mathcal{B}_2) + 2\epsilon.$$

□

A.5 Proof of Lemma 6

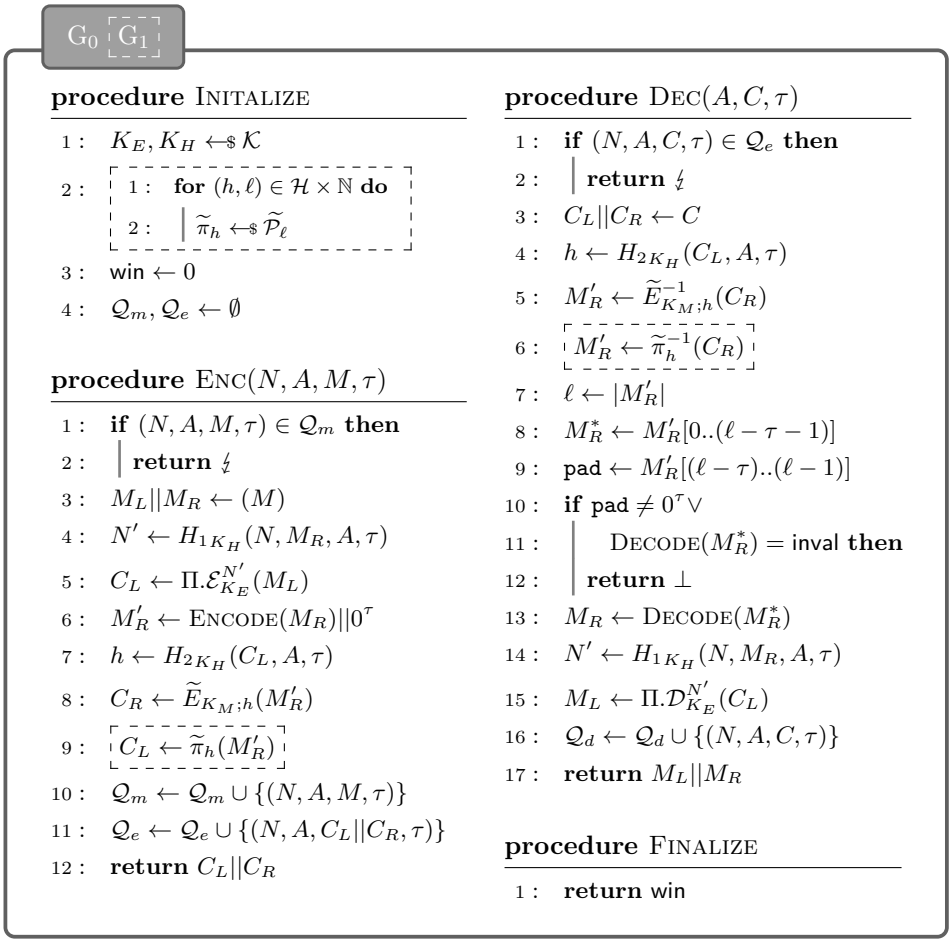

Figure 12: Games G_0 – G_1 for proof of Lemma 6. Dot-boxed code is exclusive to G_1.

Proof. Note that the authenticity of the scheme relies on C_R. We consider games G_0 – G_1 in Figure 12 where the tweakable cipher $\widetilde{E}$ is used in G_0 and a tweakable random permutation $\widetilde{\pi}$ is used in G_1. We have that

$$\begin{aligned}\mathbf{Adv}^{\text{INT-CTXT}}_{\Psi}(\mathcal{A}) = &\Pr[G_0(\mathcal{A}) \Rightarrow 1] - \Pr[G_1(\mathcal{A}) \Rightarrow 1] \\ &+ \Pr[G_1(\mathcal{A}) \Rightarrow 1].\end{aligned}$$

We can construct a $\widetilde{\pm\text{PRP}}$ adversary $\mathcal{B}$ as in Figure 13. We define the simulated encryption oracle ENC* for $\mathcal{A}$ such that for each encryption query of $\mathcal{A}$, we let $\mathcal{B}$ use the encryption scheme Π to first encrypt M_L. Then $\mathcal{B}$ pads τ zeros after M_R and queries its encryption oracle with the resulting string to get C_R. Then $\mathcal{B}$ returns $C_L||C_R$ as the ciphertext to $\mathcal{A}$. Similarly we define the simulated decryption oracle DEC* for $\mathcal{A}$ such that for $\mathcal{A}$'s decryption query, we let $\mathcal{B}$ first split the ciphertext into C_L and C_R. Then $\mathcal{B}$ queries its decryption oracle to obtain M'_R. Depending on if M'_R ends with τ zeros and can be decoded successfully, we let $\mathcal{B}$ returns $\perp$ or continues the decryption with Π. Finally, we let $\mathcal{B}$ returns 0 if $\mathcal{A}$ makes a valid forgery and let $\mathcal{B}$ returns 1 otherwise. We then have that

$$\mathbf{Adv}^{\widetilde{\pm\text{PRP}}}_{\widetilde{E}}(\mathcal{B}) = \Pr[G_0(\mathcal{A}) \Rightarrow 1] - \Pr[G_1(\mathcal{A}) \Rightarrow 1].$$

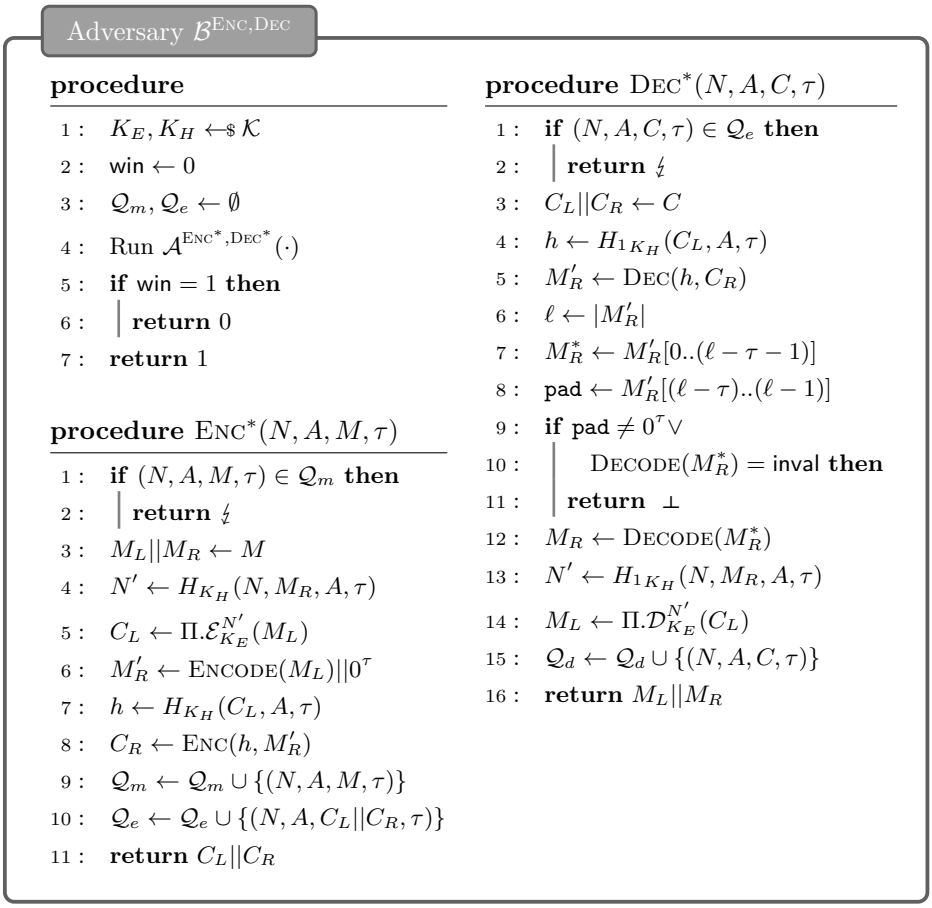

Figure 13: $\widetilde{\pm\text{PRP}}$ adversary $\mathcal{B}$ for proof of Lemma 6. Here ENC^* and DEC^* are simulated oracles for the INT-CTXT adversary $\mathcal{A}$.

We now consider when $\mathcal{A}$ wins in G_1. We first consider two cases when a tweak used in decryption matches with a tweak used in a previous encryption query. Let $C = C_L||C_R$ be the result of that previous encryption query. In the first case, a collision happens with the hash function H, the the adversary can simply reuse C_R and change one of C_L, N and A to provoke a collision in H, which is of probability at most ϵ. On the other hand, if $\mathcal{A}$ reuses (C_L, A, τ), then $\mathcal{A}$ has to query with C'_R different from C_R. In this case, C'_R has to be deciphered with a random permutation to a bitstring that ends with τ zeros and can be decoded successfully, which happens with probability less than $\frac{\delta q}{2^\tau}$.

Otherwise if each tweak used in decryption is distinct from that in encryption queries, we then have a fresh random permutation for every decryption query. Then $\mathcal{A}$ wins the game only when the ciphertext deciphers to a bitstring that ends with τ zeros and can be decoded successfully, which is of probability at most $\frac{\delta q}{2^\tau}$. Thus we have that

$$\Pr[\text{G}_1(\mathcal{A}) \Rightarrow 1] \leq \frac{q}{2^\tau} + \epsilon.$$

Finally, we have that

$$\mathbf{Adv}^{\text{INT-CTXT}}_{\Psi}(\mathcal{A}) \leq \mathbf{Adv}^{\widetilde{\pm\text{PRP}}}_{\widetilde{E}}(\mathcal{B}) + \frac{q}{2^\tau} + \epsilon.$$

□

A.6 Proof of Lemma 7

Proof. We consider three games G_0 – G_2 as in Figure 14. In G_0, the tweakable cipher $\widetilde{E}$ is used. In G_1, the tweakable random permutation $\widetilde{\pi}$ is used. In G_2, we sample a random bitstring M_λ of the length $|M'_L|$ and returns that along with $\perp$ as error message to the adversary. We have that

$$\mathbf{Adv}^{\text{IND-EPL2}}_{\Psi}(\mathcal{A}) = \sum_{i=0}^{1} \Pr[\mathrm{G}_i(\mathcal{A})] - \Pr[\mathrm{G}_{i+1}(\mathcal{A})].$$

Similarly, we can construct a $\widetilde{\pm\mathrm{PRP}}$ adversary against $\widetilde{\Pi}$ with $\mathcal{A}$ as subroutine as in Figure 15. For $\mathcal{A}$'s encryption and decryption queries, we follow the similar constructions of the simulated oracles $\textsc{Enc}^*$ and $\textsc{Dec}^*$ as in the proof of Lemma 6. For $\mathcal{A}$'s query to $\textsc{Leak}$, we let $\mathcal{B}$ query its oracle $\textsc{Dec}$ to first decipher C_R. Then if C_R deciphers to a bitstring M'_R that ends with τ zeros and can be decoded successfully, then $\mathcal{B}$ returns $\top$ to $\mathcal{A}$. Otherwise, $\mathcal{B}$ returns the tuple $(M'_R, \perp)$ to $\mathcal{A}$. We have that

$$\mathbf{Adv}^{\widetilde{\pm\mathrm{PRP}}}_{\widetilde{E}}(\mathcal{B}) = \Pr[\mathrm{G}_0(\mathcal{A})] - \Pr[\mathrm{G}_1(\mathcal{A})].$$

We now bound the probability $\Pr[\mathrm{G}_1(\mathcal{A})] - \Pr[\mathrm{G}_2(\mathcal{A})]$. Observe that the leakage only concerns M'_R. If each tweak used in $\textsc{Leak}$ is distinct, then adversary has 0 advantage in distinguishing between M'_R and M_λ following Lemma 1. To repeat a tweak, the adversary can simply repeat the query with the same tuple (N, A, C_L, τ). In this case, the oracle in the G_1 will never output a repeated bitstring since it is a permutation. Then the adversary may look for repeated outputs to distinguish G_2 from G_1 after several queries. This happens when the oracle in G_2 samples the same bitstring, which happens with probability at most $\frac{q^2-q}{2^{|M'_R|}} \leq \frac{q^2}{2^\tau}$. Thus we have that

$$\Pr[\mathrm{G}_1(\mathcal{A})] - \Pr[\mathrm{G}_2(\mathcal{A})] \leq \frac{q^2}{2^\tau}.$$

Finally, we have that

$$\mathbf{Adv}^{\text{IND-EPL2}}_{\Psi}(\mathcal{A}) \leq \mathbf{Adv}^{\widetilde{\pm\mathrm{PRP}}}_{\widetilde{E}}(\mathcal{B}) + \frac{q^2}{2^\tau}.$$

□

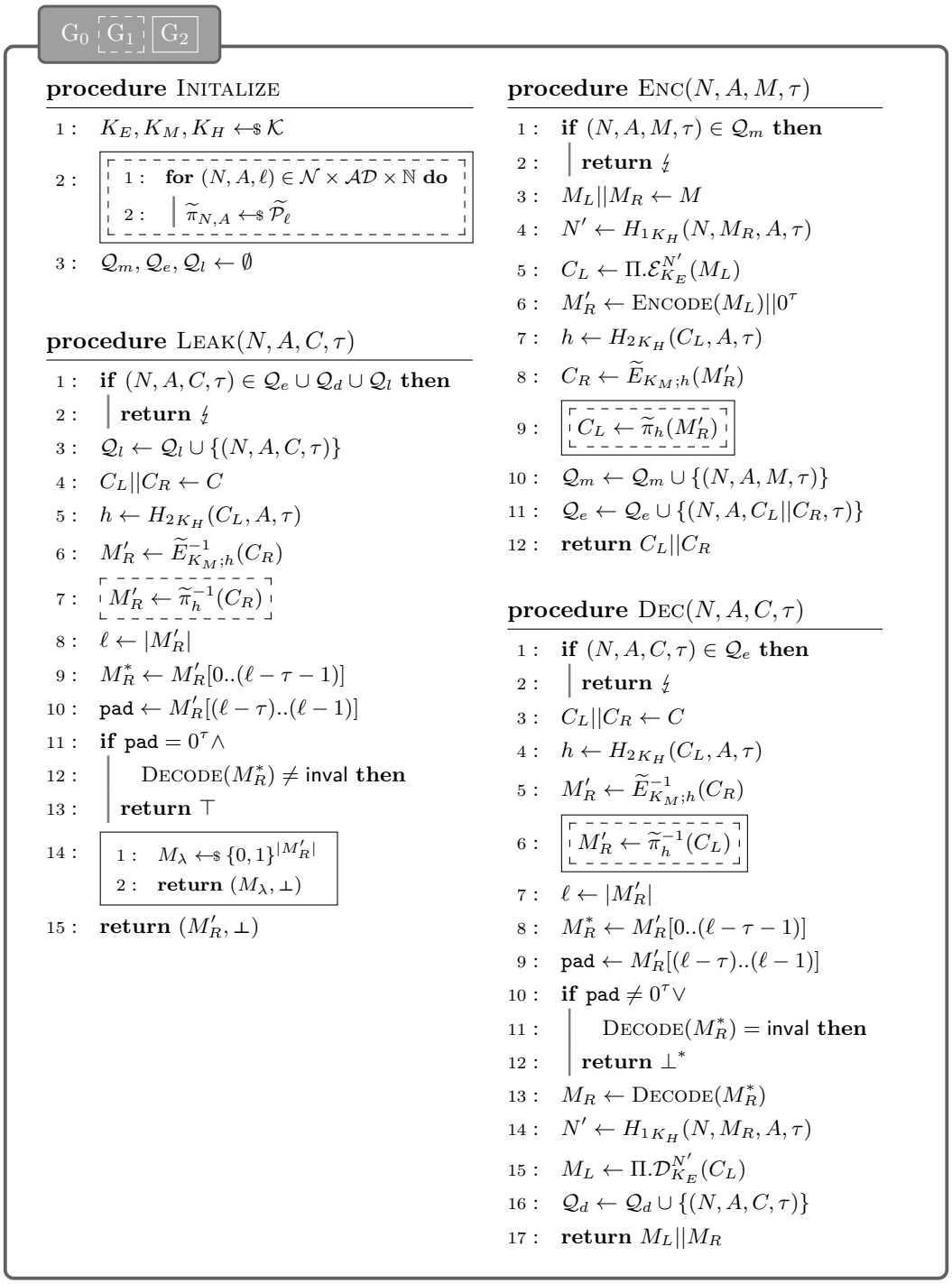

Figure 14: Game G_0 – G_2 for the proof of Lemma 7. Dot-boxed code is exclusive to G_1. Frame-boxed code is exclusive to G_2. Doubly-boxed code is for both G_1 and G_2.

Adversary $\mathcal{B}^{\text{Enc},\text{Dec}}$

procedure

$K_E, K_H \leftarrow\!\!\$\ \mathcal{K}$
$\mathcal{Q}_m, \mathcal{Q}_e, \mathcal{Q}_l \leftarrow \emptyset$
$b \leftarrow \mathcal{A}^{\text{Enc}^*,\text{Dec}^*,\text{Leak}^*}(\cdot)$
return b

procedure $\text{Dec}^*(N, A, C, \tau)$

if $(N, A, C, \tau) \in \mathcal{Q}_e$ **then**
  return $\lightning$
$C_L||C_R \leftarrow C$
$h \leftarrow H_{2K_H}(C_L, A, \tau)$
$M'_L \leftarrow \text{Dec}(h, C_R)$
$\ell \leftarrow |M'_R|$
$M^*_R \leftarrow M'_R[0..(\ell - \tau - 1)]$
$\mathsf{pad} \leftarrow M'_R[(\ell - \tau)..(\ell - 1)]$
if $\mathsf{pad} \neq 0^\tau \vee$
  $\text{Decode}(M^*_R) = \mathsf{inval}$ **then**
  return $\perp$
$M_R \leftarrow \text{Decode}(M^*_R)$
$N' \leftarrow H_{1K_H}(N, M_R, A, \tau)$
$M_L \leftarrow \Pi.\mathcal{D}^{N'}_{K_E}(C_L)$
$\mathcal{Q}_d \leftarrow \mathcal{Q}_d \cup \{(N, A, C, \tau)\}$
return $M_L||M_R$

procedure $\text{Enc}^*(N, A, M, \tau)$

if $(N, A, M, \tau) \in \mathcal{Q}_m$ **then**
  return $\lightning$
$(M_L, M_R) \leftarrow \text{SplitMsg}(M)$
$N' \leftarrow H_{1K_H}(N, M_L, A, \tau)$
$C_R \leftarrow \Pi.\mathcal{E}^{N'}_{K_E}(M_R)$
$M'_L \leftarrow 0^\tau||M_L$
$h \leftarrow H_{2K_H}(C_R, A, \tau)$
$\mathcal{Q}_m \leftarrow \mathcal{Q}_m \cup \{(N, A, M, \tau)\}$
$\mathcal{Q}_e \leftarrow \mathcal{Q}_e \cup \{(N, A, C_L||C_R, \tau)\}$
$C_L \leftarrow \text{Enc}(h, M'_L)$
return $C_L||C_R$

procedure $\text{Leak}^*(N, A, C, \tau)$

if $(N, A, C, \tau) \in \mathcal{Q}_e \cup \mathcal{Q}_d \cup \mathcal{Q}_l$ **then**
  return $\lightning$
$\mathcal{Q}_l \leftarrow \mathcal{Q}_l \cup \{(N, A, C, \tau)\}$
$C_L||C_R \leftarrow \text{SplitCtx}(C, \tau)$
$h \leftarrow H_{2K_H}(C_L, A, \tau)$
$M'_R \leftarrow \text{Dec}(h, C_R)$
$\ell \leftarrow |M'_R|$
$M^*_R \leftarrow M'_R[0..(\ell - \tau - 1)]$
$\mathsf{pad} \leftarrow M'_R[(\ell - \tau)..(\ell - 1)]$
if $\mathsf{pad} = 0^\tau \wedge$
  $\text{Decode}(M^*_R) \neq \mathsf{inval}$ **then**
  return $\top$
return $(M'_R, \perp)$

Figure 15: $\widetilde{\pm\text{PRP}}$ adversary $\mathcal{B}$ for proof of Lemma 7. Here Enc^*, Dec^* and Leak^* are simulated oracles for the IND-EPL2 adversary $\mathcal{A}$.