# AE Robustness as Indistinguishable Decryption Leakage amid Multiple Failure Conditions

Ganyuan Cao[a]

EPFL, Lausanne, Switzerland

**Abstract.** Robustness has emerged as a critical criterion for authenticated encryption, alongside confidentiality and integrity. In this study, we revisit AEAD robustness by focusing on descriptive errors when multiple failure conditions exist. We introduce new notion, IND-CCLA and IND-sf-CCLA, that expands on classical security notions defined for AEAD by incorporating the indistinguishability of decryption leakage including text-based values and descriptive errors.

We highlight that simply outputting a single error message when decryption fails is insufficient to guarantee robustness, as leakage can undermine this approach. We examine error flags used when validating a ciphertext during the decryption process, and investigate whether it is possible to merge multiple error flags into one to mitigate this security risk. This helps to prevent the resulted leakage from giving adversaries additional advantage in future attacks, particularly when parts of the failure-checking mechanism have implementation flaws or disabled by an adversary through implementation-level attacks.

We provide a concrete proof of the robustness of Encode-then-Encipher (EtE) paradigm using our notions, demonstrating its capability to validate multiple failure conditions using a single error flag. We briefly revisit generic compositions for AE to show the practical relevance of our notions. We further present a transformation from our notion to a simulatable one, supporting future research on composable security regarding decryption leakage.

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

## Contents



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


Date of this document: 2024-06-21.

# 1 Introduction

## 1.1 Background and Motivation

Robustness in authenticated encryption has been defined in various ways. The most widely accepted definition is encapsulated in the term *robust authenticated encryption* (RAE), first introduced in [HKR15]. In line with the principles of RAE, we assert that an AEAD scheme is considered robust if it ensures both confidentiality and authenticity even when a nonce is accidentally misused or when there is information leakage due to an authenticity-check failure or other scheme-specific failures. Moreover, an AEAD scheme qualifies as an RAE scheme when it allows users to freely choose any expansion factor $\tau$, which determines the length of the ciphertext in relation to the plaintext, with the level of authenticity dependent on the selected $\tau$ parameter.

The security of a robust authenticated encryption (RAE) scheme is initially formalized as a *pseudorandom injection* (PRI) in [HKR15]. However, PRI addresses security in the presence of decryption leakage in a very generalized manner. It does not extensively explore scenarios where multiple conditions for decryption failures may be involved.

Descriptive error messages are frequently exploited in attacks, with the notable example being the padding oracle attack introduced by Vaudenay [Vau02], which has been further developed to target SSL/TLS [CHVV03, PRS11], IPsec [DP07, DP10], and other protocols. While outputting a single error message for all types of decryption failures appears promising as a countermeasure, it may prove ineffective due to the potential leakage of plaintext and error flags that are used when validating a ciphertext during decryption.

Various channels provide leakage. Practical implementations of encryption schemes may contain flaws in their failure-checking mechanisms, thereby inadvertently leaking information to adversaries. Moreover, real-world attacks on cryptographic systems extend beyond theoretical vulnerabilities to include side-channel attacks [AFP13, BB03], and implementation-level exploiting such as buffer overflows. Information acquired through these channels should also be treated as leakage.

In the context of "robustness", a cryptographic scheme should maintain security despite facing these challenges. Thus, in this paper, we explore whether it is feasible not only to indicate a single error upon decryption failure but also to limit the use of only one error flag when validating a ciphertext.

## 1.2 Related Work

The security of a RAE scheme was initially formalized as a pseudorandom injection (PRI), characterized by indistinguishability from a random injection $\pi_{N,A,\tau}$. In this context, the oracle returns a plaintext $M$ if there is a corresponding ciphertext $C$ such that $C = \pi_{N,A,\tau}(M)$. Otherwise, it returns a plaintext that fails the authenticity check, with the help of a decryption simulator. This notion generalizes decryption leakage without extensively addressing scenarios involving multiple failure conditions.

Several other works have introduced notions to formalize security under decryption leakage, addressing leakage ranging from error messages to plaintext. In [BDPS14], Boldyreva et al. examined scenarios where an encryption scheme may output multiple errors, focusing on the impact of error messages rather than the leaked plaintext. They introduced the notion of IND-CVA, which provides an IND-CPA adversary with an additional oracle to verify the validity of queried ciphertexts. They also proposed the *error invariance* (INV-ERR) notion, which ensures that no efficient adversary can generate more than one of the possible error messages. Additionally, [BDPS14] extended their study to capture stateful security.

In [ABL+14], Andreeva et al. presented the notion of *release-of-unverified-plaintext* (RUP) and linked it to the ciphertext integrity (INT). Specifically, in IND-RUP, the decryption algorithm always outputs a bitstring $M$. The adversary's objective is to trick the validation function into accepting a forged ciphertext, given access to both an encryption oracle and a decryption oracle.

Barwell et al., in [BPS15], introduced the notion of *subtle AE* (SAE), which incorporates the *error indistinguishability* (ERR-CCA) notion alongside IND-CPA and INT-CTXT. The ERR-CCA notion involves a leakage function that reveals the actual leakage. The security requirement is that an adversary should not be able to distinguish between leakage under the same key and different keys. However, we believe that this notion may not fully capture the indistinguishability of leakage itself, and there is some overlap with the integrity notions.

## 1.3 Our Contribution

We introduce a novel notion, denoted as IND-CCLA, to formalize the robustness of authenticated encryption. The IND-CCLA notion extends classical AE notions, such as IND-CCA3 [Shr04], by incorporating a leakage characterization function inspired by subtle AE [BPS15]. This augmentation captures leakage occurring when decryption fails. We refine the definition of the leakage characterization function to better separate it from the integrity notion, emphasizing the indistinguishability of the leakage itself.

Our model considers not only the plaintext but also any intermediate value used during decryption as potential leakage, and we particularly focus on error flags used to verify the validity of a ciphertext as leakage. We stipulate that an adversary should not be able to distinguish the leaked text from a random bitstring of the minimum possible leakage length. Building on this requirement, we introduce two sub-notions, IND-CCLA1 and IND-CCLA2, regarding the disclosure of error messages. In IND-CCLA1, following [BDPS14], we require that an adversary should not be able to trigger any error other than a predefined one within the error space.

IND-CCLA2 introduces a more stringent security requirement: the decryption function must output a single error message, and only one error flag (or similarly, one condition

predicate) is used to validate a ciphertext. We term this property *error unicity*. Using this notion, we answer the question:

*Is outputting a single error message upon decryption failures sufficient for robustness?*

We demonstrate that this level of security ensures an adversary gains no meaningful information as long as at least one failure condition is satisfied. Consequently, even if some failure checks have implementation flaws or are disabled (whether through side channels or implementation-level attacks), the resulted leakage does not provide the adversary with additional advantages in future attacks.

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 use our notions to analyze the stateful security of the Encode-then-Encipher (EtE) paradigm [BR00], the mainstream method for constructing robust AE, assuming the use of a counter as a nonce. This analysis allows us to formally demonstrate the ability of EtE to handle multiple failure conditions, highlighting EtE as a promising approach for constructing robust AE. We further revisit the generic compositions [BN00], including Encrypt-then-MAC and MAC-then-Encrypt, to provide motivating examples for our notions.

Furthermore, we present a transformation from our notion of leakage indistinguishability to leakage simulatability. As Maurer indicated in [Mau11], the composition property of game-based notions is unclear. Moreover, existing frameworks for composable security, such as *Universal Composability* (UC) [Can01] by Canetti and Constructive Cryptography (CC) [Mau11] by Maurer, are generally based on simulation-based proof. Transforming to simulatability allows us to facilitate future studies on security composability concerning decryption leakage.

# 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 $\ell$-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 $\varepsilon$ 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 $x \xleftarrow{\$} S$ 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 $\Pr[\mathcal{A}^{\text{INIT},\text{O}_1,\text{O}_2,\cdots} \Rightarrow b]$ as the probability that the adversary $\mathcal{A}$ outputs a value $b$ after the INIT procedure and queries to the oracle $\text{O}_1, \text{O}_2, \cdots$. We denote $\Pr[\text{G}(\mathcal{A}) \Rightarrow b]$ as the probability that a game G outputs $b$ when the adversary $\mathcal{A}$ plays game G. For simplicity, we define $\Pr[\text{G}(\mathcal{A})] := \Pr[\text{G}(\mathcal{A}) \Rightarrow 0]$. For notion simplicity, we interchangeably use the notation $\Delta_{\mathcal{A}}(\text{O}_L; \text{O}_R)$ and

$$\Delta_{\mathcal{A}} \begin{pmatrix} \text{O}_L \\ \text{O}_R \end{pmatrix} := \Pr[\mathcal{A}^{\text{O}_L} \Rightarrow 0] - \Pr[\mathcal{A}^{\text{O}_R} \Rightarrow 0]$$

to denote $\mathcal{A}$'s advantage in distinguishing between the oracles $\text{O}_L$ and $\text{O}_R$.

We let $\mathbf{Adv}^{\text{x}}_{\Pi}(\mathcal{A}_x)$ denote adversary $\mathcal{A}_x$'s advantage in breaking security notion $\mathsf{X}$ of a scheme $\Pi$. We say security notion $\mathsf{X}$ implies security notion $\mathsf{Y}$, denote $\mathsf{X} \to \mathsf{Y}$, if $\mathbf{Adv}^{\text{y}}_{\Pi}(\mathcal{A}_y) \leq c \cdot \mathbf{Adv}^{\text{x}}_{\Pi}(\mathcal{A}_x)$ for some constant $c > 0$.

## 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 *nonce-based* RAE (nRAE) in Definition 1 and *stateful* RAE (sRAE) in Definition 2.

**Definition 1** (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 $\Pi.\mathcal{D}_K^{N,A,\tau}(C) = \perp$.

**Definition 2** (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 $\Pi.\mathcal{D}_K^{A,\tau;\mathsf{st}_{\mathcal{D}}}(C) = \perp$.

# 3 Security Notions

We introduce the notions IND-CCLA and IND-sf-CCLA to formalize the robustness of nonce-based and stateful authenticated encryption (AE) schemes, respectively. Our notions naturally extend common AE security notions, such as IND-CCA3 [Shr04] for nonce-based schemes, and IND-sfCCA [BKN04] for stateful schemes. We incorporate the expansion parameter $\tau$, defined for robust AE (RAE) schemes, into our definitions. For fixed-expansion schemes, the parameter $\tau$ can be omitted.

## 3.1 Scope of Leakage

We broadly categorize leakage into two types: *text-based information* and *descriptive errors*. While some studies focus solely on plaintext as leakage, we highlight the importance of considering intermediate decryption values, such as a reconstructed initialization vector (IV), as leakage. However, in practical analysis, these forms of leakage may be overlooked. To provide a clearer understanding of what constitutes leakage, we characterize the *implementation* of a decryption scheme as a combination of a sequence of operations and condition predicates. We then define the output of each operation to be the leaked text-based information as in Definition 3.

**Definition 3** (Text-based Leakage)**.** Let a tuple $(\phi_1, \phi_2, \ldots, \phi_n)$ be a sequence of operations representing the implementation of a decryption function, such that $\phi_i$ is executed before $\phi_j$ if $i < j$. We define $(M_1, M_2, \ldots, M_n)$ as the text-based leakage values, where $M_i \leftarrow \phi_i(\cdot)$ for $i \in [n]$ meaning that $M_i$ is the output of the operation $\phi_i$.

We consider two types of errors: *implicit error flags* and *explicit error messages*. In Definitions 1 and 2, the decryption function $\mathcal{D}$ is defined to yield only one explicit error message, denoted as $\perp$. This explicit error message, $\perp$, is deliberately disclosed to the adversary. For example, $\perp$ might correspond to the message `"decryption failed"`, which is visibly displayed as output.

**Definition 4** (Implicit Error Flags)**.** 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 let $\perp_i = v_i(\cdot)$ for $i \geq 1$ be the *implicit error flags*.

In practice, each $\perp_i$ can be considered as a boolean variable, e.g. $\perp_1$ = `"cond1 = false"`. Although an adversary only sees $\perp$ as decryption output, they can still find values of these $\perp_i$'s, for example, in memory. Notably, several side-channel attacks like Lucky13 [AFP13] can also be used to infer which $\perp_i$ has been triggered.

We omit discussing errors that the adversary can trivially determine without querying, such as when a ciphertext is shorter than the minimum length supported by the scheme or when a ciphertext is not a multiple of the block size. Such queries do not provide the adversary with any additional advantage in distinguishing or forging since the result of such queries is already known to them.

Leakage Characterization Function. Inspired by the SAE notion introduced by Barwell et al. in [BPS15], we employ a *Leakage Characterization Function* $\mathcal{L}$ to capture unintentional leakage. (The term "characterization" is used here to distinguish it from the "simulator" discussed in Section 6). We define the leakage characterization function $\mathcal{L}$ for an AEAD scheme $\Pi$ as specified in Definition 5. We present the definition for a nonce-based scheme, and the nonce space $\mathcal{N}$ can be substituted with the decryption state space $\mathcal{ST}_{\mathcal{D}}$ for stateful schemes.

**Definition 5.** The leakage characterization function $\mathcal{L}$ for an AEAD scheme $\Pi$ 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 $\mathcal{S}_{\perp} = \{\perp_i\}_{i \geq 1}$ 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. $\mathcal{L}_{\Pi_K}^{N,A,\tau}(C) = \sqcup$ if there is no leaked text *and* $|\mathcal{S}_{\perp}| = 1$. This holds *regardless of* whether $\Pi.\mathcal{D}_K^{N,A,\tau}(C) = \perp$ or not for a queried ciphertext $C$.
2. $\mathcal{L}_{\Pi_K}^{N,A,\tau}(C) = \top$ if $C$ is a valid ciphertext *and* $\mathcal{L}_{\Pi;K}^{N,A,\tau}(C) \neq \sqcup$.
3. $\mathcal{L}_{\Pi_K}^{N,A,\tau}(C) = (M, \perp)$ if there is a leaked bitstring $M$ with $|M| > 0$ *and* $|\mathcal{S}_{\perp}| = 1$.
4. $\mathcal{L}_{\Pi_K}^{N,A,\tau}(C) = (M_i, \perp_i) \in \{0,1\}^{\ell} \times \mathcal{S}_{\perp}$ where $\ell \geq 0$ and $|\mathcal{S}_{\perp}| \geq 2$. We let $M_i$ be the last obtained bitstring before $\perp_i$. If there is no bitstring available before $\perp_i$, then $M_i$ is set to the empty string $\varepsilon$. It is assumed that $\Pi.\mathcal{D}$ halts upon encountering $\perp_i$.
5. The correctness is defined by: if $\Pi.\mathcal{D}_K^{N,A,\tau}(C) = \perp$, then $\mathcal{L}_{\Pi_K}^{N,A,\tau}(C) \neq \top$, and if $\mathcal{L}_{\Pi_K}^{N,A,\tau}(C) = \top$, then $\Pi.\mathcal{D}_K^{N,A,\tau}(C) \neq \perp$.

*Remark* 1. In Definition 5, 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.

For notation simplicity, we use $M \in \{0,1\}^{\ell}$ to represent all the bitstrings obtained before an error $\perp_i$ or when $|\mathcal{S}_{\perp}| = 1$. Essentially, for a set of bitstrings $\{M_1, \ldots, M_n\}$, we can rewrite as $M = M_1 || \ldots || M_n$, and require the indistinguishability of $M$ as a whole.

Furthermore, in Bulletpoint 4, we acknowledge that some schemes do not halt immediately upon detecting a failure. However, this does not impact our subsequent discussion, as our security notion either mandates the existence of a single error flag or ensures the negligibility of triggering an error beyond a predefined one, which is typically the first error.

**Example 1.** We show two examples of the outputs of the leakage characterization function as follows. Here we consider tag-based schemes and omit the expansion parameter $\tau$.

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 $\mathcal{L}_{\Pi_K}^{N,A}(C) = \sqcup$.
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 $\mathcal{L}_{\Pi_K}^{N,A}(C) \in \{(\varepsilon, \perp_1), (M, \perp_2)\}$, 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 Obfuscation. If multiple error flags result from condition predicates applied to the same leaked plaintext $M$, the scheme can "merge" these error flags into a single error flag using logical operators without causing additional plaintext leakage. We present this observation under the assumption that $v(\cdot) = \mathsf{true}$ leads to successful decryption. The observation similarly applies if $v(\cdot) = \mathsf{true}$ leads to unsuccessful decryption by replacing $\wedge$ with $\vee$.

**Observation 1.** *Let $\perp_i = v_i(M)$ and $\perp_j = v_j(M)$ for $i \neq j$, where $v_i$ and $v_j$ are condition predicates evaluated on the plaintext $M$. There exists a combined error flag $\perp_{i,j} = v_{i,j}(M)$ such that $v_{i,j}(M) = v_i(M) \wedge v_j(M)$. Specifically, if all condition predicates are evaluated on the same plaintext $M$, then we can define an error flag $\perp^* = v^*(M)$ where $v^*(M) = v_1(M) \wedge v_2(M) \wedge \ldots \wedge v_n(M)$ for $n = |\mathcal{S}_\perp|$.*

In this case, all predicates $v_i$ are evaluated concurrently, and if any $v_i(M) = \mathsf{false}$, then $v^*(M) = \mathsf{false}$. When observing $\perp^*$ (for instance, in memory), an adversary $\mathcal{A}$ is unable to ascertain the evaluation of each $v_i$. On the contrary, while a single error $\perp$ is adopted as decryption output for all types of failures, $\mathcal{A}$ can still exploit side channels, such as timing differences [AFP13], to deduce which predicate was not evaluated, thereby inferring evaluation of $v_i$.

Moreover, even if one of the checks has flaws, or if $\mathcal{A}$ attempts to disable it through implementation-level attacks so that $v_i$ always returns $\mathsf{true}$, $\mathcal{A}$ cannot verify this flaw or confirm the success of disabling the check by examining $\perp_i$, because a corresponding $\perp_i$ for a specific predicate $v_i$ no longer exists. Notably, when $v_i(\cdot) = \mathsf{true}$ for all inputs where $i \in S \subsetneq [n]$, by observing $\perp^* = \bigwedge_{i\in[n]} v_i(M)$, the adversary $\mathcal{A}$ still cannot verify this as long as there exists at least one $i \in [n] \setminus S$ such that $v_i(\cdot) = \mathsf{false}$, since the evaluation of each $v_i$ remains oblivious when observing $\perp^*$.

## 3.2 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 $\mathcal{L}(\cdot)$ to capture the information leaked during a decryption failure. This notion is defined for a nonce-based scheme.

**Definition 6** (IND-CCLA$x$)**.**

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

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 $\mathcal{L}(\cdot) = \sqcup$ to indicate no leakage at all, or if $\mathcal{L}(\cdot) = \top$ to indicate a valid ciphertext. In these cases, the adversary has zero advantage in distinguishing by leakage, and we return the same result. Otherwise, the oracle samples a bitstring $M_\lambda$ uniformly at random with a length equal to the minimum plaintext leakage $\ell^*_\lambda$, as defined by the scheme with respect to the tuple $(N, A, \tau)$ and the ciphertext length $|C|$ (or $\varepsilon$ if the length is 0). For instance, if $\mathcal{L}(\cdot) \in \{(M_1, \perp_1), (M_2, \perp_2)\}$ with $|M_1| < |M_2|$, then the minimum length $\ell^*_\lambda = |M_1|$. Alternatively, if $\mathcal{L}(\cdot) = (M, \perp)$, then $\ell^*_\lambda = |M|$. Also, we stress that $\ell^*_\lambda$ can be customized for desired leakage length e.g. setting $\ell^*_\lambda = 0$.

Error Invariance and Unicity. Based on $x_\perp \in \{\perp_i, \perp\}$, we define two sub-notions regarding the disclosure of error messages, named IND-CCLA1 and IND-CCLA2. For simplicity, we use IND-CCLA to refer to both IND-CCLA1 and IND-CCLA2 when a result applies to both notions.

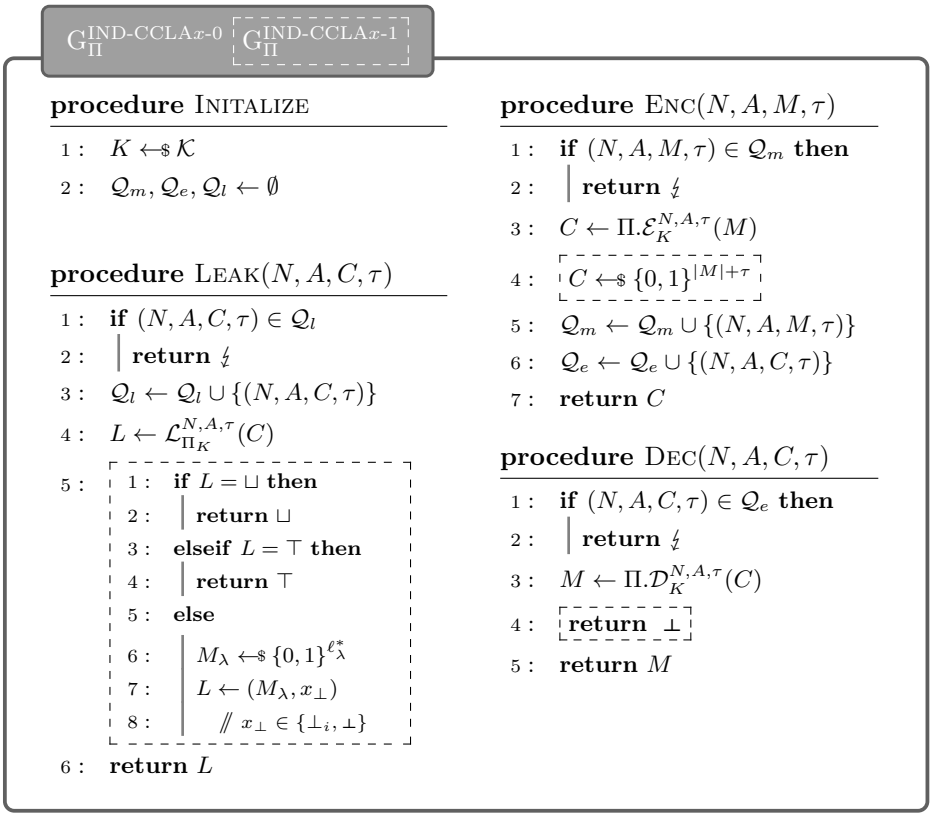


**Figure 1:** IND-CCLA$x$ games for a nonce-based (robust) AE scheme $\Pi$. The dot-boxed parts are exclusive to $G_{\Pi}^{\text{IND-CCLA}x\text{-}1}$. We define $\ell_\lambda^*$ as the minimum achievable plaintext leakage length concerning a tuple $(N, A, \tau)$ and the ciphertext length $|C|$. We let $\ell_\lambda^* \geq 0$ 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 $\perp_i \in \mathcal{S}_\perp$. This sub-notion aims to ensure:

   - The adversary cannot distinguish between the leaked text and a random bitstring of the minimum leakage length defined by the scheme.
   - The adversary cannot trigger any error flag other than $\perp_i$.

   This notion can be seen as a variant of the error invariance (INV-ERR) notion in [BDPS14], with the additional requirement of indistinguishable text disclosure.

2. IND-CCLA2 (*Error Unicity*): The tuple $(M_\lambda, \perp)$ is returned in the ideal world. This notion additionally requires $\mathcal{L}(\cdot)$ to disclose only one error. With this notion, we examine whether it is possible for a scheme to merge multiple condition predicates into one using logic operators (as in Observation 1) without incurring distinguishable text-based values, so that we can use a single error flag to validate all the failure conditions. Some practical considerations with this notion include:

   - An adversary cannot derive meaningful information (including text-based values and descriptive errors) from leakage if any failure condition is met.
   - If a scheme fails to verify some of failure conditions due to implementation flaws, it remains undetectable to an adversary observing the leakage.

- If an adversary attempts to disable some of checks (through some side channel), it cannot confirm if it succeeds through the leakage.

At a high level, IND-CCLA2 strictly requires that only one error flag is allowed, even if there are multiple failure conditions, meaning $|\mathcal{S}_\perp| = 1$. Conversely, IND-CCLA1 permits the existence of multiple error flags, allowing $|\mathcal{S}_\perp| \geq 2$.

If there is no leakage at all, i.e., $\mathcal{L}(\cdot) = \sqcup$, it trivially satisfies both IND-CCLA1 and IND-CCLA2 security. Additionally, when $|\mathcal{S}_\perp| = 1$, IND-CCLA1 converges to IND-CCLA2 since there is no other error flag to distinguish from the only error flag $\perp_1 \in \mathcal{S}_\perp$. To better differentiate between these two sub-notions, we define $\mathcal{L}$ to output $\perp$ when $|\mathcal{S}_\perp| = 1$, as discussed in Remark 1. Furthermore, a scheme with $|\mathcal{S}_\perp| \geq 2$ cannot be IND-CCLA2 because almost any query would result in $\perp_1$ being output, distinguishable from $\perp$.

However, to incentivize the development of single-error schemes, in Proposition 1 we demonstrate 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 $M_\lambda$ 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. 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 $\mathrm{G}_\Pi^{\text{IND-CCLA}x\text{-}0}$ in Figure 1 respectively, and we let $\$^{\mathcal{L}}$ be the Leak oracle as in Figure $\mathrm{G}_\Pi^{\text{IND-CCLA}x\text{-}1}$ 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 7** (IND-EPL$x$)**.**

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

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, returning the invalid symbol $\lightning$ for such queries. For IND-CCLA, the adversary is restricted from using the output of Enc to query Dec and is also prohibited from repeating a query to Enc or Leak with the same tuple. For IND-EPL, the only restriction is to avoid repeating queries to Leak with identical tuples.

It is pointless to query from Enc to Leak because the oracle Leak produces $\top$ for a valid ciphertext in both the real and ideal worlds. Similarly, in IND-EPL, forwarding

queries from Enc to Dec serves no purpose since decryption is executed honestly in both the real and ideal settings.

We do allow the adversary to repeat the nonce to capture security when a nonce might be misused. Additionally, we allow the adversary to query with variable stretch parameters, meaning the adversary can query with $\tau_1 \neq \tau_2$ in different queries. While using a small stretch parameter may allow the adversary to trivially win the INT-CTXT game, it still captures the *best achievable* security with respect to the selected stretch parameter.

## 3.3 IND-sf-CCLA Security

We describe the game for IND-sf-CCLA notion for stateful (robust) AE as in Figure 2. This notion captures the security in presence of out-of-order ciphertext delivery. 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 8.

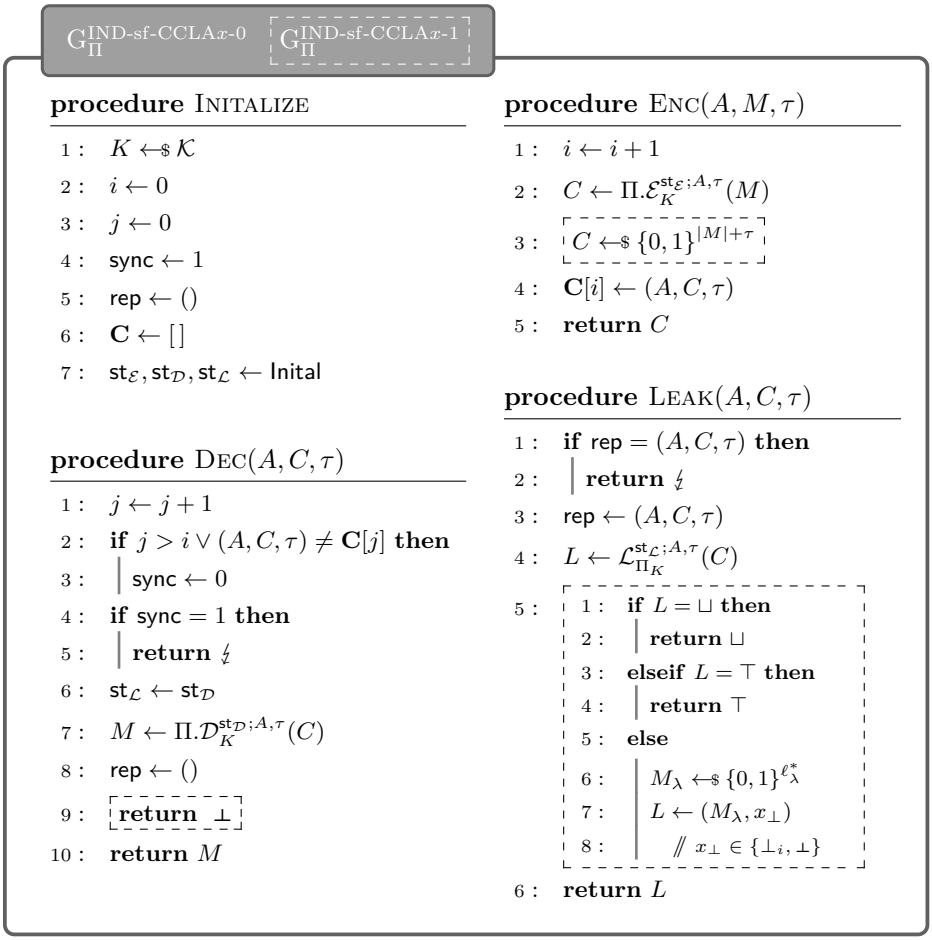


**Figure 2:** IND-sf-CCLA$x$ games for a stateful (robust) AE scheme $\Pi$. The boxed parts are exclusively to game $\mathrm{G}_{\Pi}^{\text{IND-sf-CCLA}x\text{-}1}$. We define $\ell_\lambda^*$ as the minimum achievable plaintext leakage length concerning a tuple $(A, \tau)$ and the ciphertext length $|C|$. 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 $\mathcal{L}_{\Pi_K}^{\mathsf{st}_{\mathcal{L}};A,\tau}(C) = \top$. We use $\mathsf{rep}$ to ensure the adversary does not make prohibited queries.

**Definition 8** (IND-sf-CCLA$x$)**.**

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

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 $\mathrm{G}_{\Pi}^{\text{IND-sf-CCLA}x\text{-}0}$ in Figure 2. Additionally, we use $\$^{\mathcal{L}}$ to denote the oracle Leak in game $\mathrm{G}_{\Pi}^{\text{IND-sf-CCLA}x\text{-}1}$ in Figure 2. We define IND-sf-EPL as follows.

**Definition 9** (IND-sf-EPL$x$)**.**

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

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 prohibiting queries to Dec with in-order ciphertexts from Enc, we also prohibit the adversary from making *consecutive* repeated queries to the Leak oracle. This is because two queries to Leak with the same tuple and the same state yield the same result, allowing the adversary to trivially win the game. Therefore, we require that there must be one query to Dec between two successive queries to Leak.

This restriction aligns with real-world scenarios, as leakage can only occur when 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. These prohibited queries are defined for both the IND-sf-CCLA and IND-sf-EPL notions. We let the oracles return the invalid symbol $\lightning$ for these prohibited queries.

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

## 3.4 Separation and Relations

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. 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-CCLA$x$ *adversary* $\mathcal{A}$*, there exist an* IND-CPA *adversary* $\mathcal{A}_{cpa}$*, an* INT-CTXT *adversary* $\mathcal{A}_{int}$ *and an* IND-EPL$x$ *adversary* $\mathcal{A}_{epl}$ *such that*

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

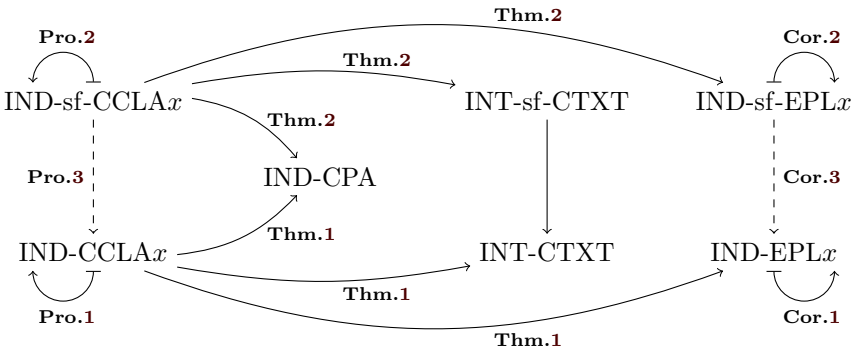


**Figure 3:** An illustration of implications between notions. We use $A \to B$ to denote that notion $A$ implies notion $B$. We use $Ax \dashrightarrow Bx$ to denote that $Ax$ implies $Bx$ only when $x$ is the same value for both $Ax$ and $Bx$. We use $Ax \mapsto Ax$ to denote $A2$ imples $A1$.

*Proof.* We rewrite the advantage as

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

By definition, we have that

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

for an IND-EPL$x$ adversary $\mathcal{A}_{epl}$.

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 $M_\lambda$ 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

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

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

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

□

Similarly, we can decompose IND-sf-CCLA notion into IND-CPA plus INT-sf-CTXT plus IND-sf-EPL. 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-CCLA$x$ *adversary $\mathcal{A}$, there exist an* IND-CPA *adversary $\mathcal{A}_{cpa}$, an* INT-sf-CTXT *adversary $\mathcal{A}_{int}$ and an* IND-sf-EPL$x$ *adversary $\mathcal{A}_{epl}$ such that*

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

*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-CCLA$x$ *implies* IND-CCLA$x$ *for* $x \in \{1, 2\}$.

**Corollary 3.** IND-sf-EPL$x$ *implies* IND-EPL$x$ *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 $\not\Rightarrow$ INT-CTXT). We consider an EtM scheme $\Pi$ where $C = M \oplus K_E$ and $T = C \oplus K_M$, with final output as $C||T$. From Example 1, we know $\mathcal{L}(\cdot) = \sqcup$. Thus both oracles will output $\sqcup$ meaning that $\mathbf{Adv}_{\Pi}^{\text{IND-EPL}}(\mathcal{A}) = 0$. 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 $\not\Rightarrow$ IND-EPL2). We consider the EtM paradigm 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 $\mathcal{L}(\cdot) = (M, \perp)$ 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 $\not\Rightarrow$ IND-EPL1). We consider EEM but also with the "decryption first" configuration. Thus we have $\mathcal{L}(\cdot) \in \{(M, \perp_1), (M, \perp_2)\}$, 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.

□

## 3.5 Comparison with Existing Notions

We make a brief comparison between our notion and established notions to underscore the advantages of our notion, specifically RAE security from [HKR15], the error invariance (INV-ERR) from [BDPS14], subtle AE from [BPS15], and plaintext awareness (PA) from [ABL+14].

RAE Security. In the context of RAE security, the comparison is made with a random injection holistically, whereas our notions emphasize the indistinguishability of the leakage itself. RAE formalizes leakage in a generalized manner, assuming that plaintext is always leaked upon decryption failure, and does not address scenarios involving multiple errors.

RAE claims to achieve the *best-possible* security concerning a queried tuple $(N, A, M, \tau)$ (or $(N, A, C, \tau)$ respectively). Indeed, RAE makes sense even when the stretch parameter $\tau$ is small. For example, when $\tau = 0$, all ciphertexts are considered valid, thus comparing the actual decryption $\mathcal{D}_K(\cdot)$ with the reverse of the injection $\pi^{-1}(\cdot)$. The adversary may not distinguish between these two worlds, yet it produces a valid ciphertext.

This fails to explicitly demonstrate which security goals, e.g. confidentiality, authenticity, and indistinguishability of leakage, can be achieved concerning a specific stretch parameter $\tau$. This should be conveyed through proofs, providing guidance on the appropriate stretch parameter for security. Our notions decompose these goals to capture each one individually,

enabling us to demonstrate what security can be achieved with a given stretch parameter, as discussed in Remark 2.

Regarding plaintext leakage, a specific requirement of RAE is to ensure that the leaked plaintext has a length $|M| \neq |C| - \tau$ for a queried ciphertext $C$ and an expansion parameter $\tau$. Our notions can also accommodate this requirement by additionally stipulating that $\ell^*_\lambda \neq |C| - \tau$.

Given the similarities between our notions and SAE, we can leverage the result of the equivalence between RAE and SAE as demonstrated in [BPS15, Theorem 14]. To eliminate the influence of multiple error flags, we restate the theorem under the condition that $|\mathcal{S}_\perp| = 1$. A key difference between our notions and SAE is that SAE compares $\mathcal{L}_K$ and $\mathcal{L}_{K'}$ for a different key $K'$. Thus they use $\mathcal{L}_{K'}$ as the leakage simulator for RAE in the ideal world for the proof. To better align their result with our notions, we modify the leakage simulator for RAE to the oracle $\$^{\mathcal{L}}$, with no loss of security.

**Proposition 5.** RAE *is equivalent to* IND-CCLA2 *when* $|\mathcal{S}_\perp| = 1$. *Specifically,*

$$\left|\mathbf{Adv}^{\mathrm{RAE}}_{\Pi}(\mathcal{A}) - \mathbf{Adv}^{\text{IND-CCLA2}}_{\Pi}(\mathcal{A})\right| \leq \frac{q}{2^{\tau-1}} + \frac{r^2 + r}{2^{\tau+m+1}}$$

*where $r$ is the number misused nonce of $\mathcal{A}$'s queries, $q$ is the number of queries, and $m$ is the length of the shortest string in the message space.*

*Proof.* We write the RAE advantage as

$$\mathbf{Adv}^{\mathrm{RAE}}_{\Pi}(\mathcal{A}) = \Delta_{\mathcal{A}}\begin{pmatrix}\mathcal{E}_K, \mathcal{D}_K, \mathcal{L}_K \\ \pi,\ \pi^{-1},\ \$^{\mathcal{L}}\end{pmatrix}$$

where $\pi$ is a random injection. We let $\mathcal{L}_K$ simulate the leakage. Thus we have that

$$\begin{aligned}\left|\mathbf{Adv}^{\mathrm{RAE}}_{\Pi}(\mathcal{A}) - \mathbf{Adv}^{\text{IND-CCLA2}}_{\Pi}(\mathcal{A})\right| &= \left|\Delta_{\mathcal{A}}\begin{pmatrix}\mathcal{E}_K, \mathcal{D}_K, \mathcal{L}_K \\ \pi,\ \pi^{-1},\ \$^{\mathcal{L}}\end{pmatrix} - \Delta_{\mathcal{A}}\begin{pmatrix}\mathcal{E}_K, \mathcal{D}_K, \mathcal{L}_K \\ \$^{\mathcal{E}},\ \perp,\ \$^{\mathcal{L}}\end{pmatrix}\right| \\ &\leq \Delta_{\mathcal{A}}\begin{pmatrix}\$^{\mathcal{E}},\ \perp,\ \$^{\mathcal{L}} \\ \pi,\ \pi^{-1}, \$^{\mathcal{L}}\end{pmatrix} \qquad (1)\end{aligned}$$

Note that Equation 1 is essentially the difference between PRI and MRAE, which has be characterized in [HKR15, Theorem 1]. □

Error Invariance. In [BDPS14], Boldyreva et al. explored the situation where a decryption scheme generates multiple error messages and introduced the error invariance (INV-ERR) notion such that an adversary should not see $\perp_j \in \mathcal{S}_\perp$ other than a predefined error $\perp_i \in \mathcal{S}_\perp$. A scheme is considered INV-ERR secure if there exists a $\perp_i \in \mathcal{S}_\perp$ such that $\mathbf{Adv}^{\text{INV-ERR}}_{\Pi,\perp_i}(\mathcal{A}) \leq \mathsf{negl}$. However, the actual implementations of schemes can be vulnerable to not only cryptographic attacks but also system or software attacks. Thus, ensuring computational infeasibility with respect to a specific error may be insufficient, as an adversary might exploit side channels to bypass the error check to see $\perp_j$. This reflects the intuition of our notion of error unicity.

Since we assume a decryption scheme producing only a single error message $\perp$, we draw a parallel to our leakage characterization function, considering the set of implicit error flags $\mathcal{S}_\perp$ as analogous to the error space in their notions.

In IND-EPL1, we stipulate that the adversary cannot induce an error flag $\perp_j$ with $\perp_i \neq \perp_j$, where $\perp_i \in \mathcal{S}_\perp$ represents the predefined error flag. This requirement resonates with the error invariance, while also factoring in the leaked text. For IND-EPL2, we essentially require that $|\mathcal{S}_\perp| = 1$, automatically achieving error invariance.

**Proposition 6.** IND-EPL *implies* INV-ERR.

*Proof (Sketch).* We let $\mathcal{A}$ be an INV-ERR adversary against an AEAD scheme $\Pi$ with error space $\mathcal{S}_\perp$. We first assume $|\mathcal{S}_\perp| \geq 2$. We can then 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 $\mathbf{Adv}^{\text{INV-ERR}}_{\Pi}(\mathcal{A}) \leq \mathbf{Adv}^{\text{IND-EPL2}}_{\Pi}(\mathcal{B})$. □

SUBTLE AE. In [BPS15], Barwell et al. introduced the concept of a leakage simulator function. They define the leakage function $\Lambda$ as $\Lambda : \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{C} \to \{\top\} \cup \mathcal{S}_\lambda$ where $\mathcal{S}_\lambda$ 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. With the leakage simulator function, they define subtle AE (SAE) as

$$\mathbf{Adv}^{\text{SAE}}_{\Pi}(\mathcal{A}) = \Delta_{\mathcal{A}} \begin{pmatrix} \mathcal{E}_K, \mathcal{D}_K, \Lambda_K \\ \$^{\mathcal{E}}, \perp, \Lambda_{K'} \end{pmatrix}$$

for $K \leftarrow\!\!\$\ \mathcal{K}$ and $K' \leftarrow\!\!\$\ \mathcal{K}$ with $K \neq K'$. Similarly, they extract the notion of *error simulatability* (ERR-CCA) from their notion as

$$\mathbf{Adv}^{\text{ERR-CCA}}_{\Pi}(\mathcal{A}) = \Delta_{\mathcal{A}} \begin{pmatrix} \mathcal{E}_K, \mathcal{D}_K, \Lambda_K \\ \mathcal{E}_K, \mathcal{D}_K, \Lambda_{K'} \end{pmatrix}.$$

We note that the definition provided by Barwell et al. is highly generalized, owing to the extensive scope of $\mathcal{S}_\lambda$. Barwell et al. claimed that they aimed to offer authors the flexibility to define leakage according to their specific requirements. While a broader definition of leakage, as suggested in their later work [BMOS17], can indeed provide adversaries with a greater advantage, this flexibility may also lead to the inadvertent neglect of certain vulnerabilities. For example, in the RIV scheme [AFL+16], Abed et al. defined leakage solely as the plaintext $M$, neglecting the possibility of an adversary manipulating the initialization vector (IV) through the tag $T$.

Moreover, comparing to $\Lambda_{K'}$ with a different key $K'$ does not sufficiently capture the impact of the leakage itself, as it overlaps with integrity notions. For a valid ciphertext $C$, $\Lambda_K(C)$ outputs $\top$, while $\Lambda_{K'}(C)$ is almost certain to produce an output other than $\top$, making it distinguishable. This necessitates considering an integrity adversary when evaluating the advantage of a leakage adversary. This issue is particularly evident when there is no leakage, as an adversary should have zero advantage in distinguishing based on leakage in such cases, but we must still account for an integrity adversary. To address this, we allow the oracles in both the real and ideal worlds to return $\top$ or $\sqcup$ when the leakage function produces such an output. In Proposition 7, we demonstrate that IND-EPL implies ERR-CCA when all ciphertexts passed to LEAK are invalid and leakage is possible.

**Proposition 7.** IND-EPL *implies* ERR-CCA *when* $\mathcal{L}^{N,A,\tau}_{\Pi_K}(C) \notin \{\top, \sqcup\}$ *for all tuples* $(N, A, C, \tau)$ *queried by the adversary.*

*Proof.* (ERR-CCA $\not\to$ IND-EPL1). We consider the example in [BPS15]. Suppose a scheme that is ERR-CCA secure, then there is a simple variant that upon triggering an error returns $\perp_1$ or $\perp_2$ depending on the first ciphertext bit. Then it is trivially not IND-EPL1.

(IND-EPL1 $\to$ ERR-CCA). Given an ERR-CCA adversary $\mathcal{A}$, we can construct an IND-EPL1 adversary $\mathcal{B}$ as follows. For each query to LEAK made by $\mathcal{A}$, we have $\mathcal{B}$ forward to its oracle LEAK. Note that a query from $\mathcal{A}$ eventually yields $(M, \perp_j)$ to be distinguished from $(M', \perp'_j) = \Lambda_{K'}(\cdot)$. Then $\mathcal{B}$ can use it to distinguish from $(M_\lambda, \perp_i)$.

(ERR-CCA $\not\to$ IND-EPL2). The result is trivial if $|\mathcal{S}_\perp| \geq 2$ by Corollary 1. Now suppose $|\mathcal{S}_\perp| = 1$ and $\Lambda$ implements a permutation. An ERR-CCA adversary has 0 advantage in distinguish $\Lambda_K(\cdot)$ and $\Lambda_{K'}(\cdot)$. However, an IND-EPL adversary can observe the repeated output from LEAK to distinguish.

(IND-EPL2 $\to$ ERR-CCA). Suppose $|\mathcal{S}_\perp| \geq 2$, then the result is trivial following Corollary 1. Now suppose $|\mathcal{S}_\perp| = 1$. Similarly, a query from the ERR-CCA adversary $\mathcal{A}$ eventually yields $(M, \perp)$ to be distinguished from $(M', \perp) = \Lambda_{K'}(\cdot)$. Then $\mathcal{B}$ can use it to distinguish from $(M_\lambda, \perp)$. □

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 decryption simulator. For our following discussion on EtE, we can essentially consider the indistinguishability of plaintext as a random bitstring. We define it as in Definition 10.

**Definition 10** (PA2)**.** Let $\widetilde{\mathcal{D}}$ be the decryption function without any check for failure such that $\widetilde{\mathcal{D}}$ always output a plaintext, then

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

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

Intuitively, PA2 emphasizes the indistinguishability of *plaintexts* when no checks are performed. In contrast, our IND-EPL2 notion focuses on the indistinguishability, or more precisely, the uniqueness, of the *error* itself. Moreover, IND-EPL2 reveals whether a ciphertext is valid and which error(s) it triggers if invalid, a detail not disclosed by PA2. This feature aims to prevent the exploitation of descriptive error flags in further attacks (with examples in Section 5). Additionally, even if the plaintext is indistinguishable, descriptive errors may still disclose information following our discussion in Remark 3. In Proposition 8, we separate our IND-EPL2 notion from PA2 security.

However, we recognize the importance of PA2 security in ensuring the robustness of AE, as it guarantees the confidentiality of the plaintext when the only check for failure is also disabled. Therefore, we conclude that an AE scheme should achieve both IND-EPL2 and PA2 to be considered robust.

**Proposition 8.** PA2 *does not imply* IND-EPL2*, and* IND-EPL2 *does not imply* PA2*.*

*Proof.* (IND-EPL2 $\not\to$ PA2). We consider an EtM scheme $\Pi$ as in Example 1. We know $\mathbf{Adv}^{\text{IND-EPL2}}_{\Pi}(\mathcal{A}) = 0$ since $\mathcal{L}(\cdot) = \sqcup$. However, it is trivial to break PA2 security by changing the tag of a ciphertext obtained from a previous encryption query.

(PA2 $\not\to$ IND-EPL2). We consider the EtE paradigm with a tweakable cipher and $|\mathcal{S}_\perp| = 2$ (with detailed discussion in Section 4). Then it is PA2 secure if the tweakable cipher is secure as a tweakable PRP and no repeated tweak is queried. However, it is not IND-EPL2 secure since for almost every query $\perp_1$ will be output to be distinguished from $\perp$. □

One may also observe that IND-EPL2 is strictly stronger than PA2 for EtE when $|\mathcal{S}_\perp| \geq 2$. In Remark 3, we further explain why $|\mathcal{S}_\perp|$ matters in certain special cases.

# 4 Robustness of Encode-then-Encipher

Encode-then-Encipher (EtE) paradigm, initially proposed by Bellare and Rogaway in [BR00], stands as the predominant approach for constructing robust AE. In [HKR15],

Hoang et al. proved that EtE with a tweakable VIL cipher achieves the security as a PRI. Despite this, there remains a dearth of study on the stateful security of EtE, particularly in scenarios involving multiple errors. Notably, in [ST13], Shrimpton and Terashima briefly touched upon the robustness of EtE, acknowledging its capacity to accommodate multiple errors. Here we provide a formal treatment of this property through our notions.

In [BMM+15], Badertscher et al. showed the security of EtE 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 message length. We then follow the idea of [BMM+15] by also using counter as nonce and prove the robustness of EtE from a game-based perspective with our notion.

## 4.1 EtE with Tweakable Cipher

We consider 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 $\mathcal{T} = \mathcal{N} \times \mathcal{AD} \times \mathbb{N}$. We define a robust AE scheme $\Pi = (\mathcal{E}, \mathcal{D})$ using EtE as follows. Let $C = \Pi.\mathcal{E}_K^{N,A,\tau}(M) = \widetilde{E}_{K;N,A,\tau}(M||0^\tau)$ and return $C$ as ciphertext. Let $M' = \widetilde{E}^{-1}_{K;N,A,\tau}(C)$. Then if $M'$ ends with $\tau$ zeros, $\Pi.\mathcal{D}_K^{N,A,\tau}(C)$ returns $M'$ excluding ending $\tau$ zeros as plaintext. Otherwise, $\Pi.\mathcal{D}_K^{N,A,\tau}(C)$ returns $\perp$.

The security of a tweakable block cipher is defined as (strong) indistinguishability from *tweakable* random permutation ($(\pm)\widetilde{\mathrm{PRP}}$), 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 $\widetilde{\pi}_T(\cdot)$ 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*

$$\Pr[\mathcal{A}^{\widetilde{\pi}_T(\cdot)} \Rightarrow 0] - \Pr[\mathcal{A}^{\$(\cdot)} \Rightarrow 0] = 0$$

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

*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. 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. Also, we provide a generalized result assuming the tweakable cipher is a blackbox thus omitting the possible intermediate values used in a specific construction e.g. a constructed IV in PIV [ST13].

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

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

*where* $\tau$ *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* $\Pi$ *making* $q$ *encryption queries, there is a* $\widetilde{\text{PRP}}$ *adversary* $\mathcal{A}_{tprp}$ *against the tweakable* VIL *cipher* $\widetilde{E}$ *such that*

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

*Proof (Sketch).* We consider three games $\text{G}_0$ – $\text{G}_2$ 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

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

Then we can bound $\Pr[\text{G}_0(\mathcal{A})] - \Pr[\text{G}_1(\mathcal{A})]$ by a $\widetilde{\text{PRP}}$ adversary $\mathcal{A}_{tprp}$. Following Lemma 1, we know that $\Pr[\text{G}_1(\mathcal{A})] - \Pr[\text{G}_2(\mathcal{A})] = 0$ since we assume counter does not repeat. □

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

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

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

*Proof (Sketch).* We consider two games $\text{G}_0$ and $\text{G}_1$ 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

$$\mathbf{Adv}^{\text{INT-sf-CTXT}}_{\Pi}(\mathcal{A}) = \Pr[\mathcal{A} \text{ wins } \text{G}_0] - \Pr[\mathcal{A} \text{ wins } \text{G}_1] + \Pr[\mathcal{A} \text{ wins } \text{G}_1].$$

Similarly, we can bound $\Pr[\mathcal{A} \text{ wins } \text{G}_0] - \Pr[\mathcal{A} \text{ wins } \text{G}_1]$ 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 $\text{G}_1$ when its query yields a bitstring ending with $\tau$ zeros, which is of probability at most $\frac{q}{2^\tau}$. □

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

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

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

*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 $M_\lambda$ 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

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

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 IND-CPA and IND-EPL security are independent of the stretch parameter $\tau$, and it only affects the level of authenticity provided. In [HII+22], Hosoyamada et al. investigated the CCA security of EtE under short stretch through an analysis of Rocca [SLN+21]. In [Kha24], Khairallah further generalized this study to the CCA security of PRI. Our findings converge with theirs in cases of unsuccessful decryption, underscoring the irrelevance of stretch parameter in our results.

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 EtE paradigm is its ability to leverage existing redundancy in the plaintext. For example, if the plaintext has been encoded prior to stretching or follows a specific format, such redundancy can be utilized to enhance authenticity. We define the *density* of message space $\mathcal{M}$ to measure how redundant the message space is as in Definition 11.

**Definition 11** ($\delta$-dense)**.** Let $v_\delta : \{0,1\}^\ell \to \{\mathsf{true}, \mathsf{false}\}$ be a predicate for $\ell \in \mathbb{N}$. We say $\mathcal{M} \subseteq \{0,1\}^\ell$ is $\delta$-dense with respect to the predicate $v$ if

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

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

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

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

LEAKAGE WITH MULTIPLE ERRORS. Suppose that $\mathcal{M}$ is $\delta$-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 $\delta$-dense, then for any* IND-sf-EPL1 *adversary $\mathcal{A}$ against the* EtE *construction $\Pi$ making $q$ leakage queries, there is a $\widetilde{\pm\text{PRP}}$ adversary $\mathcal{A}_{stprp}$ against the tweakable* VIL *cipher $\widetilde{E}$ such that*

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

*if* $\perp_1 =$ *"*$M'[\ell-\tau,\ell] \neq 0^\tau$*", or*

$$\mathbf{Adv}^{\text{IND-sf-EPL1}}_{\Pi}(\mathcal{A}) \leq \mathbf{Adv}^{\widetilde{\pm\text{PRP}}}_{\widetilde{E}}(\mathcal{A}_{stprp}) + \delta \tag{3}$$

*if* $\perp_1 =$ *"*$v_\delta(M'[0,\ell-\tau-1]) = \mathsf{false}$*", where* $\ell = |M'|$.

Leakage with Merged Error. The condition predicates both pertain to the same leaked plaintext $M'$. Based on Observation 1, we can merge them into a single leakage tuple $(M', \perp)$. Essentially, the stretching process can be viewed as a mapping $\psi : \mathcal{M} \to \mathcal{M}^*$. Consequently, we derive a new predicate $v_{\delta^*} : \mathcal{M}^* \to \{\mathsf{true}, \mathsf{false}\}$ on the updated message space $\mathcal{M}^*$, where $\delta^* = \frac{\delta}{2^\tau}$. This allows us to obtain a unified error $\perp^* = v_{\delta^*}(M')$.

Indeed, if $M'$ satisfies one of the condition predicates, it might display a specific characteristic, such as ending with $\tau$ zeros. However, the probability that the oracle in the ideal world produces such a bitstring is equivalent, providing the adversary with no more advantage than a random guess. Consequently, the adversary has no advantage in distinguishing based *solely* on the leaked plaintext.

Despite this, the adversary can break IND-sf-EPL1 security by identifying $\perp_2$ to recognize the real world when one of the condition predicate is satisfied (since $\perp_1$ is always output in the ideal world). After merging the two errors into one, $\perp$ is output instead of $\perp_2$, preventing the adversary from distinguishing based on the error flag. This reduction in the adversary's advantage removes the term $\frac{q}{2^\tau}$ from Equation 2, and $\delta$ from Equation 3, leading to the bound for IND-EPL2 advantage as stated in Corollary 6.

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

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

*Remark* 3. Following Lemma 4 and Corollary 6, the indistinguishability of leakage is unaffected by the stretch parameter $\tau$ when $|\mathcal{S}_\perp| = 1$. However, Corollary 5 shows that $\tau$ indeed plays a role in distinguishing when $|\mathcal{S}_\perp| \geq 2$. Although the importance of $|\mathcal{S}_\perp|$ might seem abstract, it has practical implications, as discussed in Section 3.2.

We broaden the scope of "encoding" to encompass any values that needs to be verified upon decryption. For instance, this can include verifying that decrypted social security numbers match with the users. When $|\mathcal{S}_\perp| \geq 2$, an adversary may infer these values by observing the leaked plaintext and the absence of the corresponding $\perp_i$. On the other hand, if $|\mathcal{S}_\perp| = 1$, the adversary remains oblivious to this, even if it partially satisfies the success condition.

This also highlights our separation of PA2 from IND-EPL2 in Proposition 8. Even if the plaintext appears indistinguishable from random bits, it might still "accidentially" satisfy some of the success conditions, leading to descriptive errors that reveal some information about those values to be verified (especially when these values are short).

Security under Nonce-Misuse. In Lemma 2 to Corollary 6, we actually assume an nonce-respecting adversary. The security under nonce-misuse of EtE has been discussed in many works including [HKR15, BR00, BMM+15]. Here we restate the security showing that IND-CPA and IND-EPL2 security in presence of an nonce-misusing adversary are still independent of the stretch parameter, but instead depend on the message length.

**Lemma 5.** *For any nonce-misusing* IND-CPA *adversary* $\mathcal{A}$ *against the* EtE *construction* $\Pi$ *making* $q$ *encryption queries, there is a* $\widetilde{\text{PRP}}$ *adversary* $\mathcal{A}_{tprp}$ *against the tweakable* VIL *cipher* $\widetilde{E}$ *such that*

$$\mathbf{Adv}^{\text{IND-CPA}}_{\Pi}(\mathcal{A}) = \mathbf{Adv}^{\widetilde{\text{PRP}}}_{\widetilde{E}}(\mathcal{A}_{tprp}) + \frac{q^2}{2^\ell}$$

*where* $\ell = \arg\min_{(N,A,M,\tau)\in\mathcal{Q}}\{|C| : C = \Pi.\mathcal{E}_K^{N,A,\tau}(M)\}$ *and* $\mathcal{Q}$ *is the set of encryption queries made by* $\mathcal{A}$.

*Proof.* We consider the games $G_0$ – $G_2$ as in proof of Lemma 2. In this case, the adversary can fix $(N, A, \tau)$ but query with different $M$. Thus we have a fixed tweak this time, which means we have a fixed permutation. Thus the behaviors of $G_1$ and $G_2$ differ when $G_2$ samples a repeated bitstring, which happens with probability at most $\frac{q^2-q}{2^\ell} \leq \frac{q^2}{2^\ell}$. □

**Lemma 6.** *For any nonce-misusing* IND-EPL2 *adversary* $\mathcal{A}$ *against the* EtE *construction* $\Pi$ *making* $q$ *leakage queries, there is a* $\widetilde{\pm\text{PRP}}$ *adversary* $\mathcal{A}_{stprp}$ *against the tweakable* VIL *cipher* $\widetilde{E}$ *such that*

$$\mathbf{Adv}^{\text{IND-EPL2}}_{\Pi}(\mathcal{A}) = \mathbf{Adv}^{\widetilde{\pm\text{PRP}}}_{\widetilde{E}}(\mathcal{A}_{stprp}) + \frac{q^2}{2^\ell}$$

*where* $\ell = \arg\min_{(N,A,C,\tau)\in\mathcal{Q}}\{|M| : (M, \perp) = \mathcal{L}^{N,A,\tau}_{\Pi_K}(C), |M| > 0\}$ *where* $\mathcal{Q}$ *is the set of leakage queries made by* $\mathcal{A}$.

*Proof.* The proof follows a similar proof to Lemma 5. □

# 5 Revisiting Generic Compositions

We present a revisiting on the generic compositions, specifically Encrypt-then-MAC (EtM) and MAC-then-Encrypt (MtE), as motivating examples for our IND-EPL2 notion. In these examples, we assume that the plaintext has been encoded prior to encryption (or MAC).

Encode-then-Encrypt-then-MAC (EEM). We define two predicates $v_1$ and $v_2$ for the success conditions where

$$v_1(C||T) = \text{“}\mathsf{MAC}.\mathcal{V}(C, T) = \mathsf{true}\text{”}$$

and

$$v_2(M) = \text{“}\textsc{Decode}(M) \neq \mathsf{inval}\text{”}.$$

We first assume that $\mathcal{L}^{N,A}_{\Pi_K}(C||T) \in \{(\varepsilon, \perp_1), (M, \perp_2)\}$ for an invalid ciphertext $C$, where $\perp_1 = v_1(C||T)$ and $\perp_2 = v_2(M)$. Following [BDPS14, Theorem 5], if the encryption scheme is IND-CPA and the MAC scheme is SUF-CMA, then EEM achieves IND-CCLA1 security by setting $\perp_i = \perp_1$ in the definition.

Suppose $v_1(C||T) = \mathsf{true}$ for all $(C, T)$. This could occur due to a flaw in the authenticity-check mechanism or for instance, if an adversary $\mathcal{A}$ manages to manipulate the return value of the verification $\mathcal{V}$ to always be true by exploiting a memory overflow. To show why $|\mathcal{S}_\perp|$ matters, we only assume $\perp_1$ or $\perp_2$ is disclosed for now. By observing $\perp_2$, $\mathcal{A}$ can infer that the only reason for failure is with incorrect encoding.

In this case, $\mathcal{A}$ can, for example, perform a padding oracle attack [Vau02] to recover the plaintext. Additionally, given the leakage of $M$, $\mathcal{A}$ can further manipulate $M$ to pass other checks if the encryption scheme is malleable as discussed in [Rog11]. Note that $\mathcal{A}$ no longer needs to worry about authenticity now, as it knows that the authenticity check has flaws or has been successfully disabled by observing the appearance of $\perp_2$.

Now, let us assume a scheme that merges two errors to obtain a single error flag $\perp^* = v_1(C) \wedge v_2(M)$. Note that $\mathcal{A}$ can obtain a ciphertext $C$ that decrypts to $M$ with $v_2(M) = \mathsf{true}$ from a previous encryption query. By modifying the tag of $C$ to an obviously invalid one, $\mathcal{A}$ can infer whether $v_1(C||T) = \mathsf{true}$ for all $(C, T)$ by observing for successful decryption, since $\mathcal{A}$ can control when $v_2(M) = \mathsf{true}$. This renders such error merging ineffective. In this case, we have $\mathcal{L}^{N,A}_{\Pi_K} = (M, \perp)$ (since we need to first obtain $M$ for $v_2$), which is trivially distinguishable.

**Proposition 9.** EEM *is not* IND-CCLA2 *secure.*

Encode-then-MAC-then-Encrypt (EME). Notably, EEM falls in the scope of *Encrypt-with-Redundancy* framework. Its authenticity has been discussed by An and Bellare in [AB01]. We analyze the security from the perspective of decryption leakage. We consider two condition predicates, $v_1$ and $v_2$, similar to the EEM example. However, we change $v_1$ to $v_1(M,T) =$ "$\mathsf{MAC}.\mathcal{V}(M,T) = \mathsf{true}$".

By setting $M' = M||T$, we allow both predicates to apply to $M'$ (i.e. first part of $M'$ is checked for encoding). Indeed, according to Observation 1, we can merge the two error flags into one $\perp^* = v_1(M') \wedge v_2(M')$. However, this merging is also ineffective since following a similar reason to EEM, that is, altering a portion of the ciphertext (with tag) does not disrupt the correct encoding. Consequently, the adversary can control when $v_2$ evaluates to true.

**Proposition 10.** EME *is not* IND-CCLA2 *secure.*

These examples highlight our intuition behind the combinatorial security of the indistuingishability of the leaked text-values and the uniqueness of error flag. Even if error flags are merged, the leaked text values might still aid in distinguishing them, rendering the merging of error flags ineffective.

Discussion on Recent Construction. This may be a common problem for tag-based AE schemes. RIV [AFL+16], inspired by SIV [RS06], employs a Feistel structure [Fei73] and incorporates an additional tag in their scheme. Specifically, the extra tag in RIV makes IV malleable thus allows predictable IV attacks [DR11]. RIV calculates $IV = F_1(N,A,C) \oplus T$ for decrypting $C$, where $F$ is a PRF. The authenticity is later verified by computing $IV' = F_2(N,A,M)$ and check if $IV = IV'$. Let $(N,A,C,T)$ be the output of a previous encryption query. Under CBC mode, the leaked plaintext reveals $M_1 \oplus M_1' = T \oplus T'$ when the adversary queries Leak with $(N,A,C,T')$ and $T \neq T'$, where $M_1$ and $M_1'$ are the first blocks of $C$ corresponding to $T$ and $T'$ respectively.

Unfortunately, RIV did not consider the IV as part of the leakage in their analysis. According to our definition, we have $\mathcal{L}_{\Pi_K}^{N,A}(C||T) = (IV||M)$. It is trivial that the IV is distinguishable from a random bitstring by querying with the same $(N,A,C)$ but a different tag $T'$, highlighting such a vulnerability.

# 6 Transformation to Simulatability

We observe that transforming a game-based proof with *real-or-ideal* oracles into a simulation-based proof is natural and straightforward. This is because security in a simulation-based proof is essentially characterized by the advantage of a distinguisher $\mathbf{D}$, which distinguishes between an adversary $\mathcal{A}$ interacting with the real functionality $\mathbf{Real}_{\mathcal{F}}$ and a simulator $\mathcal{S}$ interacting with an ideal functionality $\mathbf{Ideal}_{\mathcal{F}}$.

Notably, Degabriele and Fischlin, in [DF18], have shown the equivalence between IND-CPA and *encryption simulatability* (ES), and they have also demonstrated that *decryption simulatability with integrity* (DS-I) implies INT-CTXT. It remains for us to show that we can transform IND-EPL into a simulatable one.

It is natural to ask: what should an adversary know when decryption fails? Ideally, the adversary should learn nothing other than the length of the leaked content (since it is defined by the scheme) and the fact that decryption has failed. Therefore, we should be able to emulate the leakage without any knowledge of its contents except for its length. We consider an emulator $\mathsf{L}$ for the leakage, as depicted in Figure 4. Note that the only input provided to the emulator is the ciphertext length $|C|$ (and possibly the stretch parameter $\tau$). With this information, the minimum leakage length $\ell_\lambda^*$ should be computable. We then simulate the leakage content by sampling a random bitstring of that length and return $(M_\lambda, \perp)$ as the result. Note that we return $\perp$ as an error since we are simulating the

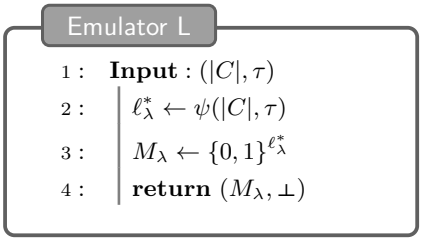


**Figure 4:** An emulator $\mathsf{L}$ for the leakage. We let $\psi$ be the function that evaluates the leakage length with respect to $|C|$ and $\tau$.

behavior of a decryption function that outputs a single error for all types of failures, and the descriptive error flags $\perp_i \in \mathcal{S}_\perp$ provide information beyond the fact that decryption fails. This actually better explains our purpose behind proposing the IND-EPL2 notion.

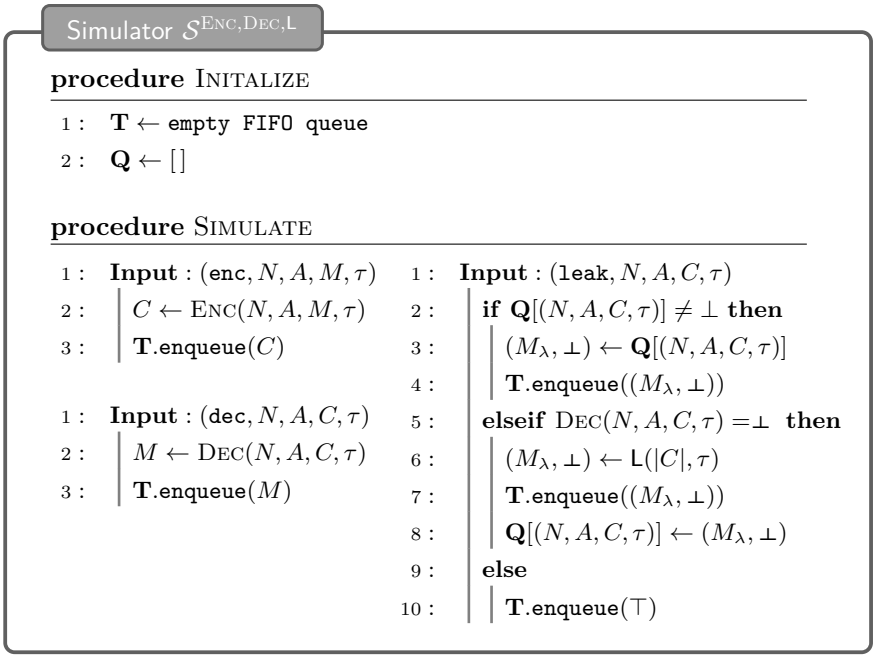


**Figure 5:** A simulator $\mathcal{S}$ for the adversary $\mathcal{A}$. We let $\texttt{enc}, \texttt{dec}$ and $\texttt{leak}$ indicate $\mathcal{A}$'s queries to the oracles Enc, Dec and Leak respectively. We let the queue $\mathbf{T}$ be the transcript of results. We use the lookup table $\mathbf{Q}$ to record the queries made, allowing us to manage repeated leakage queries by outputting the same result.

Next, we consider the simulator $\mathcal{S}$ for the adversary $\mathcal{A}$, as described in Figure 5. We fix the honest execution of $\mathcal{E}_K(\cdot)$ and $\mathcal{D}_K(\cdot)$, allowing the simulator $\mathcal{S}$ to determine when there is a decryption failure. For each of $\mathcal{A}$'s queries to Enc and Dec, $\mathcal{S}$ simply repeats the query. For $\mathcal{A}$'s queries to Leak, $\mathcal{S}$ first determines if the decryption fails. In such cases, $\mathcal{S}$ forwards $|C|$ and $\tau$ to the emulator $\mathsf{L}$ to obtain the simulated leakage. A queue $\mathbf{T}$ is used to track the transcript of the query results.

When defining $\mathcal{S}$ and $\mathsf{L}$, we omit the case when $\mathcal{L}(\cdot) = \sqcup$. This is because the emulator $\mathsf{L}$ and the simulator $\mathcal{S}$ should not have access to leakage function $\mathcal{L}_{\Pi_K}(\cdot)$ (though $\mathcal{S}$ can infer the output of $\mathcal{L}_{\Pi_K}(\cdot)$ when decryption is successful through $\mathcal{D}_K(\cdot)$). However, when $\mathcal{L}_{\Pi_K}(\cdot) = \sqcup$, distinguishing via leakage is impossible since there is no leakage.

We can then similarly define leakge simulatability as the advantage of a distinguisher

**D** in distinguishing between the transcripts by $\mathcal{A}$ and $\mathcal{S}$, as in Definition 12.

**Definition 12** (Leakage Simultatability (LS))**.**

$$\mathbf{Adv}^{\mathrm{LS}}_{\Pi}(\mathbf{D}) := \Pr[\mathcal{A}^{\mathcal{E}_K,\mathcal{D}_K,\mathcal{L}_K} \Rightarrow \mathbf{T}] - \Pr[\mathcal{S}^{\mathcal{E}_K,\mathcal{D}_K,\mathsf{L}} \Rightarrow \mathbf{T}]$$

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

**Proposition 11.** LS *is equivalent to* IND-EPL2.

*Proof.* We rewrite the advantages as

$$\begin{aligned}\left|\mathbf{Adv}^{\mathrm{LS}}_{\Pi}(\mathbf{D}) - \mathbf{Adv}^{\text{IND-EPL2}}_{\Pi}(\mathcal{A})\right| &= \left|\Delta_{\mathbf{D}}\begin{pmatrix}\mathcal{E}_K,\mathcal{D}_K,\mathcal{L}_K\\ \mathcal{E}_K,\mathcal{D}_K,\ \mathsf{L}\end{pmatrix} - \Delta_{\mathcal{A}}\begin{pmatrix}\mathcal{E}_K,\mathcal{D}_K,\mathcal{L}_K\\ \mathcal{E}_K,\mathcal{D}_K,\ \$^{\mathcal{L}}\end{pmatrix}\right| \\ &\leq \Delta\begin{pmatrix}\mathcal{E}_K,\mathcal{D}_K,\ \mathsf{L}\\ \mathcal{E}_K,\mathcal{D}_K,\$^{\mathcal{L}}\end{pmatrix}\end{aligned}$$

Observe that the transcript $\mathbf{T}_{\mathcal{A}}$ of $\mathcal{A}$ and the transcript $\mathbf{T}_{\mathcal{S}}$ of $\mathcal{S}$ only differ by the result of $\mathsf{L}$ and $\$_{\mathcal{L}}$. Additionally, $\mathcal{S}$ only call $\mathsf{L}$ when decryption fails. Otherwise, $\top$ is enqueued. In case of decryption failure, both $\mathcal{L}$ and $\$_{\mathcal{L}}$ sample a random bitstring of length $\ell^*_\lambda$, which yields 0 advantage in distinguishing. □

Notably, in [DF18], Degabriele and Fischlin extended their study to capture stateful security via simulatability, in the presence of multiple ciphertext fragments. It should be of no complication to also extend our LS notion to capture stateful security. They also explored the composability of their notions into a simulatable channel. Numerous frameworks address security composability, including the renowned *Universal Composability* (UC) framework by Canetti [Can01]. Additionally, Maurer introduced the Constructive Cryptography (CC) framework [Mau11], which we believe may provide a better example regarding channel composability.

As indicated in [Mau11], an AE scheme can essentially be considered as a composition of a confidential channel **CONF**, where only message length is leaked but adversary injection into the channel is allowed, and an authentic channel **AUTH**, where the adversary cannot inject messages, but the message content is available. Our transformation to leakage simulatability can be viewed as a simplified version of CC by replacing the simulator for an adversary who can observe the communication or inject messages in an Alice-Bob setting with an adversary making oracle queries. Thus, converting our results to the CC framework should not be complicated. Specifically, a channel should be defined such that every message injected by the adversary yields $(M_\lambda, \perp)$ from the emulator $\mathsf{L}$. We leave it as future work to formalize such a channel and study the composable security concerning leakage.

# 7 Conclusion and Future Work

In this work, we introduce a new notion, IND-CCLA, to formalize the robustness of AEAD schemes. This notion extends commonly accepted notions such as IND-CCA3 by incorporating an additional oracle to address security concerns arising from leakage due to decryption failures.

We particularly consider the situation involving multiple failure conditions and introduce the concept of error unicity, which requires that only a single error is disclosed, either implicitly through leakage or explicitly through decryption, even in the presence of multiple failure conditions. Our findings show that it is not enough to just output a single error upon decryption failure; the system must also ensure that only one error can be leaked. This approach guarantees robustness, even if the failure-checking mechanism has implementation

flaws, or if adversaries use side-channel or implementation-level attacks to gain knowledge of errors and disable failure checks.

We further extend this notion to IND-sf-CCLA, which captures stateful security when adversaries query out-of-order ciphertexts. We demonstrate the robustness of the Encode-then-Encipher (EtE) paradigm, a mainstream method for constructing robust AE, by proving its capability to handle multiple failure conditions with a single error.

Future work includes developing a more rigorous notion beyond error unicity to address security under multiple failures. While our current focus is on leakage related to decryption, it is also essential to formalize security concerning encryption leakage to ensure the indistinguishability of intermediate values used in encryption, as discussed in [BMOS17]. This would further strengthen AE robustness. Additionally, we provide a brief example of converting our game-based notion to a simulation-based notion. Future research should include an in-depth study of leakage simulatability to advance the understanding of composable security concerning decryption leakage and to model the discrepancy between the decryption function's behavior and the actual leakage incurred.

## References

[AB01] Jee Hea An and Mihir Bellare. Does encryption with redundancy provide authenticity? In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 512–528. Springer, Heidelberg, May 2001. doi:10.1007/3-540-44987-6_31.

[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.

[AFP13] Nadhem J Al Fardan and Kenneth G Paterson. Lucky thirteen: Breaking the tls and dtls record protocols. In *2013 IEEE symposium on security and privacy*, pages 526–540. IEEE, 2013.

[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.

[BB03] David Brumley and Dan Boneh. Remote timing attacks are practical. In *USENIX Security 2003*. USENIX Association, August 2003.

[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.

[BMOS17] Guy Barwell, Daniel P. Martin, Elisabeth Oswald, and Martijn Stam. Authenticated encryption in the face of protocol and side channel leakage. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 693–723. Springer, Heidelberg, December 2017. doi:10.1007/978-3-319-70694-8_24.

[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.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001. doi:10.1109/SFCS.2001.959888.

[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.

[DF18] Jean Paul Degabriele and Marc Fischlin. Simulatable channels: Extended security that is universally composable and easier to prove. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part III*, volume 11274 of *LNCS*, pages 519–550. Springer, Heidelberg, December 2018. doi:10.1007/978-3-030-03332-3_19.

[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.

[DR11] T Duong and J Rizzo. Beast: Surprising crypto attack against https. *Blog, September*, 42:45–47, 2011.

[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.

[Rog11] Phillip Rogaway. Evaluation of some blockcipher modes of operation. *Cryptography Research and Evaluation Committees (CRYPTREC) for the Government of Japan*, 630, 2011.

[RS06] Phillip Rogaway and Thomas Shrimpton. A provable-security treatment of the key-wrap problem. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 373–390. Springer, Heidelberg, May / June 2006. doi:10.1007/11761679_23.

[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.

[SLN+21] Kosei Sakamoto, Fukang Liu, Yuto Nakano, Shinsaku Kiyomoto, and Takanori Isobe. Rocca: An efficient AES-based encryption scheme for beyond 5g. *IACR Trans. Symm. Cryptol.*, 2021(2):1–30, 2021. doi:10.46586/tosc.v2021.i2.1-30.

[ST13] Thomas Shrimpton and R. Seth Terashima. A modular framework for building variable-input-length tweakable ciphers. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 405–423. Springer, Heidelberg, December 2013. doi:10.1007/978-3-642-42033-7_21.

[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

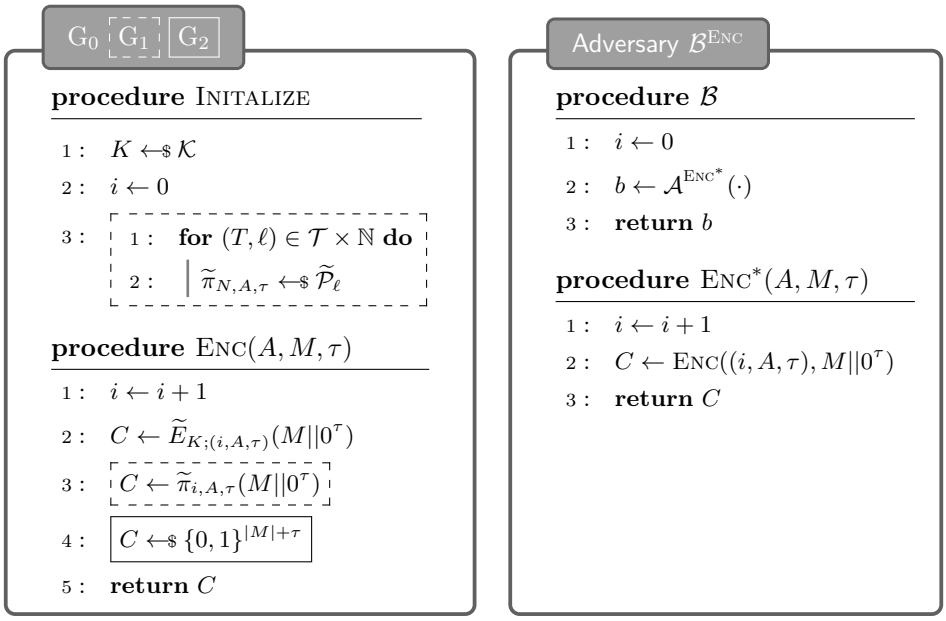


**Figure 6: Left**: Games $G_0$ – $G_2$ for proof of Lemma 2. Dot-boxed code is exclusive to $G_1$ and Frame-boxed code is exclusive to $G_2$. **Right**: $\widetilde{\text{PRP}}$ adversary $\mathcal{B}$ for proof for proof of Lemma 2. For notation simplicity, we let $T = (N, A, \tau)$ and we let $\mathcal{T} = \mathcal{N} \times \mathcal{AD} \times \mathbb{N}$.

*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 $\tau$ 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 $\text{Enc}^*$ for $\mathcal{A}$ such that for each encryption query made by $\mathcal{A}$, we let $\mathcal{B}$ append $\tau$ 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

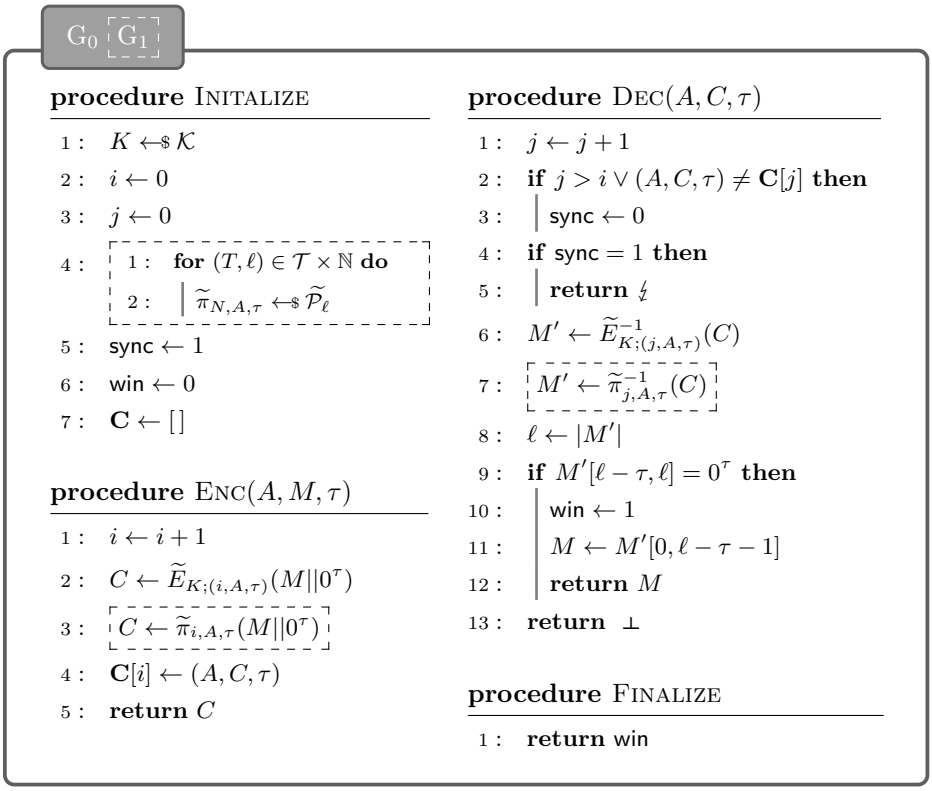


**Figure 7:** Games $G_0$ – $G_1$ for proof of Lemma 3. Dot-boxed code is exclusive to $G_1$.

*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 $\tau$ 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 $\tau$ 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 $(A, C, \tau)$ 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 bitstring ends with $\tau$ zeros, which is of probability $\frac{q}{2^\tau}$. In the other case, $(A, C, \tau)$ has

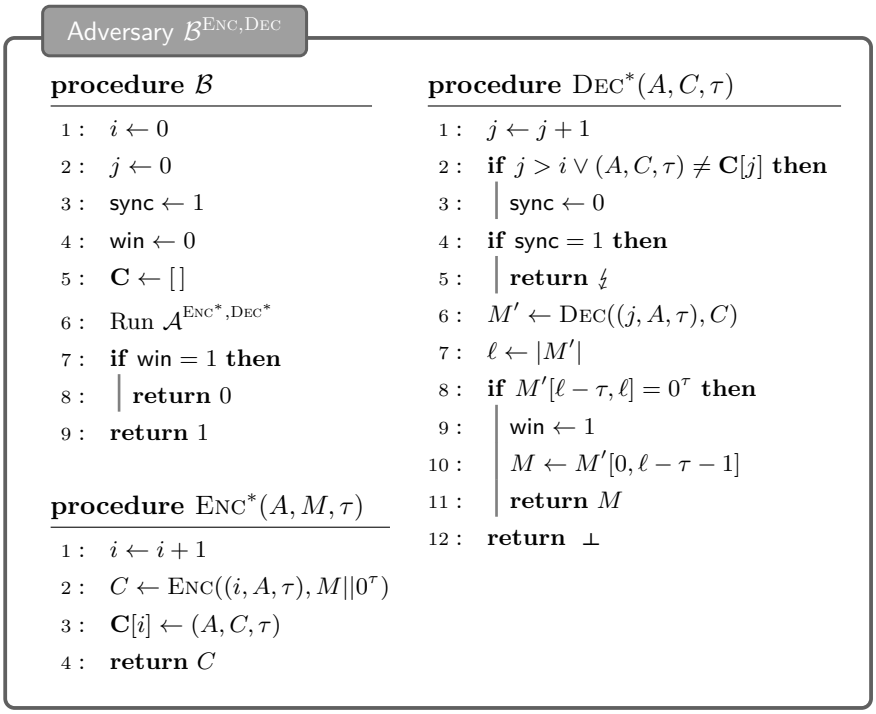


**Figure 8:** $\widetilde{\pm\text{PRP}}$ adversary $\mathcal{B}$ for proof of Lemma 3.

never been an output from ENC, then $C$ is valid only if $C$ deciphers to a bitstring with $\tau$ 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 $\text{G}_0$ – $\text{G}_2$ as in Figure 9 for the proof. In $\text{G}_0$, $\mathcal{A}$'s queries are answered with $\widetilde{E}$ and $\widetilde{E}^{-1}$ respectively. In $\text{G}_1$, $\mathcal{A}$'s queries are answered with $\widetilde{\pi}$ and $\widetilde{\pi}^{-1}$ respectively. In game $\text{G}_2$, a bitstring $M_\lambda$ 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 $\text{ENC}^*$ and $\text{DEC}^*$ 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

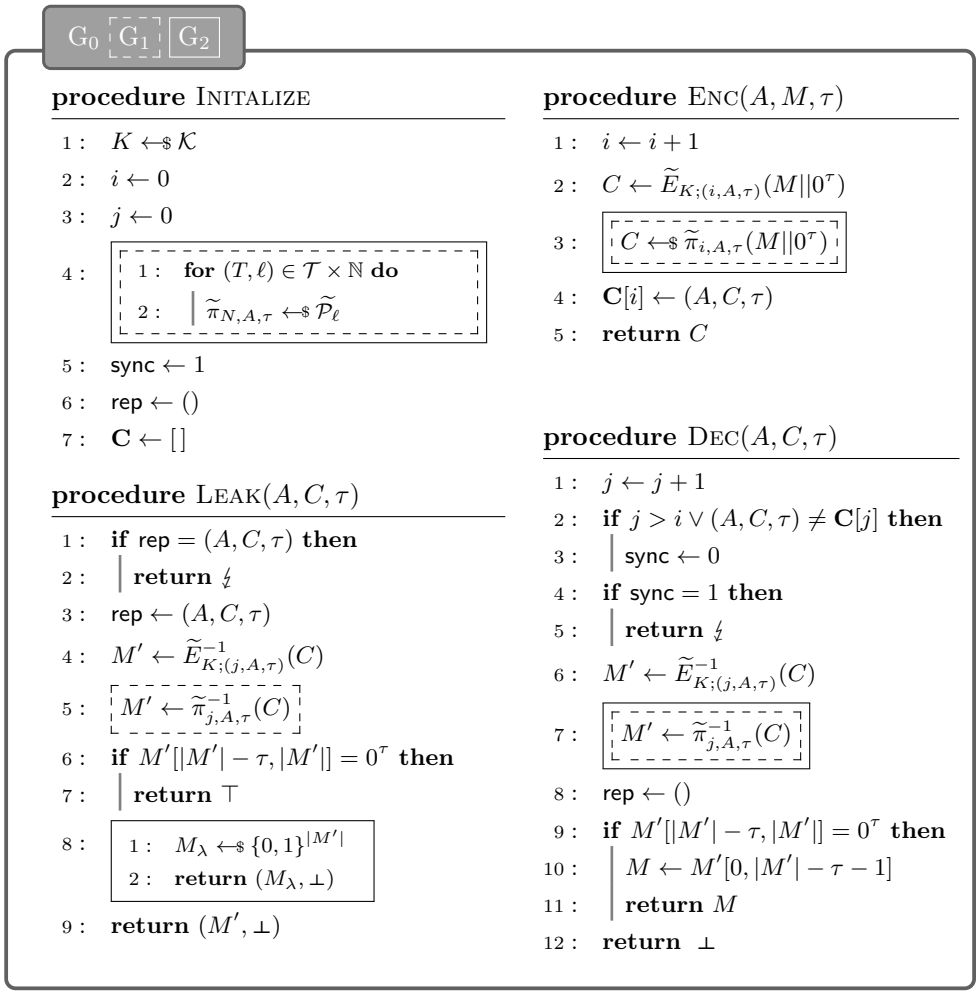


**Figure 9:** Game $G_0$ – $G_2$ for the proof of Lemma 4. Dot-boxed code is exclusive to $G_1$. Frame-boxed code is exclusive to $G_2$. Doubly-boxed code is for both $G_1$ and $G_2$.

returns $(M', \bot)$ 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

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

□

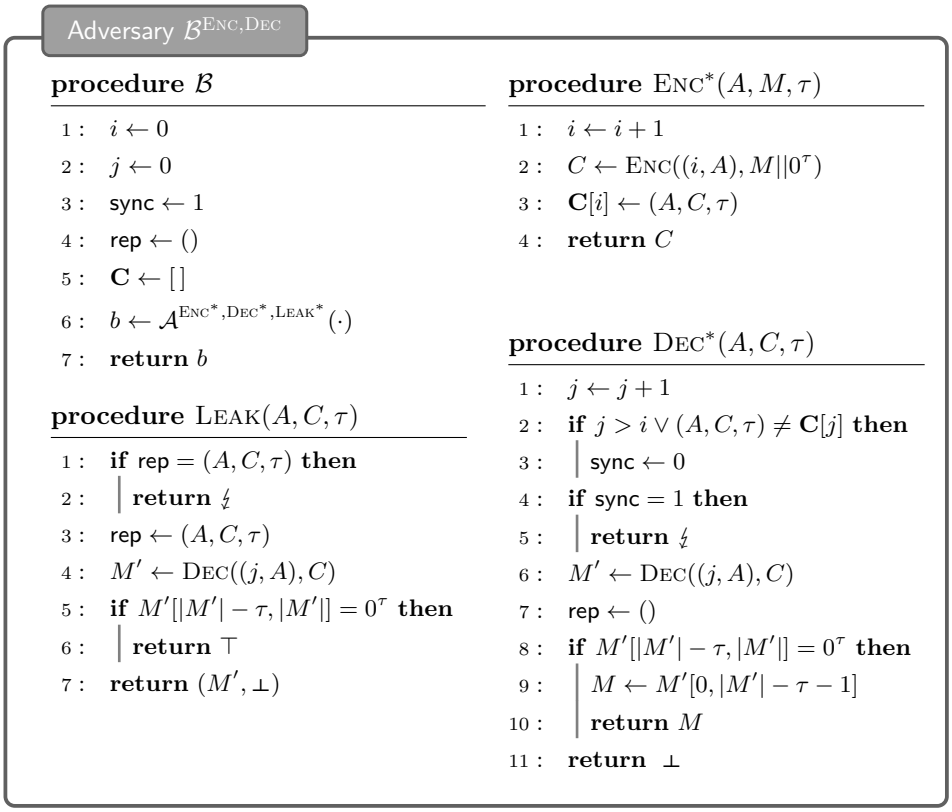


**Figure 10:** $\widetilde{\pm\text{PRP}}$ adversary $\mathcal{B}$ for the proof of Lemma 4.