AE Robustness as Indistinguishable Decryption Leakage under Multiple Failure Conditions

Ganyuan Cao\textsuperscript{a} \textsuperscript{\textcopyright}

EPFL, Lausanne, Switzerland

Abstract. Robustness has emerged as an important criterion for authenticated encryption, alongside the requirements of confidentiality and integrity. We introduce a novel notion, denoted as \(\text{IND-CCLA} \), to formalize the robustness of authenticated encryption from the perspective of decryption leakage. This notion is an augmentation of common notions defined for AEAD schemes by considering indistinguishability of potential leakage due to decryption failure including candidate plaintext and error messages, particularly in the presence of multiple failure conditions. With this notion, we study the disparity between a single-error decryption function and the actual leakage incurred during decryption. We introduce the concept of error unicity to require that only one error is disclosed, whether explicitly via decryption or implicitly via leakage, even there are multiple failure conditions. This aims to mitigate the security issue caused by disclosing multiple errors via leakage. We further extend this notion to \(\text{IND-sf-CCLA} \) to formalize the stateful security involving out-of-order ciphertext. We provide a concrete proof on the robustness of Encode-then-Encipher paradigm through our notions to show its ability to admit multiple failure conditions. Additionally, we briefly show a transformation from our notion to a simulatable one, which can aid future study on composable security concerning decryption leakage.

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

1 Introduction

1.1 Background and Motivation

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

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

\textsuperscript{a}The author is jointly affiliated at EPFL and ETH Zürich

E-mail: ganyuan.cao@epfl.ch (Ganyuan Cao)

This work is licensed under a “CC BY 4.0” license.

Date of this document: 2024-05-13.
padding oracle attack introduced by Vaudenay [Vau02], which has been further developed to target SSL/TLS [CHVV03, PRS11], IPsec [DP07, DP10], and other systems. While adopting a unified error message in the decryption function for all error types appears promising as a mitigation, there may still be leakage that grants adversaries an additional advantage. That is because decryption functions typically reveal plaintext only upon successful completion of all verification steps, whereas leakage may occur prior to their completion, potentially exposing partial plaintext and error messages, which makes those attacks possible.

Furthermore, there has been limited exploration into the stateful security involving out-of-order ciphertexts with PRI. With our notions, we aim to study the disparity between a single-error decryption function and the actual leakage incurred during decryption including candidate plaintexts and error messages, while also allowing for an analysis that captures stateful security in the presence of out-of-order ciphertexts.

1.2 Related Work

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

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

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

In [ABL+14], Andreeva et al. introduced the notion of release-of-unverified-plaintext (RUP) and associated it with the ciphertext integrity (INT) notion. Specifically, in IND-RUP, the decryption algorithm always outputs a bitstring $M$. The adversary’s goal is to make the validation function to accept a forged ciphertext given an encryption oracle and a decryption oracle.

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

1.3 Our Contribution

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

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

In IND-CCLA2, we impose a stronger security requirement that there should be only one error disclosed (whether implicitly by leakage or explicitly by decryption), even there may be multiple failure conditions. We call this property as error unicity. With this notion, we aim to align the actual leakage as closely as possible to the behavior of a single-error decryption function, thus to mitigate the security issue caused by disclosing errors via leakage. This notion concurrently ensures that an adversary learns no meaningful information as long as one of the failure conditions is met, thus to guarantee the security if there is a flaw in one of the checks, or that check is compromised by the adversary. Thus we consider this as an important property in terms of “robustness”.

We then extend this notion to a stronger version, IND-sf-CCLA, to formalize stateful security in scenarios involving out-of-order ciphertext delivery for stateful AE schemes. We follow our notions to analyze the stateful security of the Encode-then-Encipher (EtE) paradigm [BR00], which is the mainstream way to construct robust AE, by assuming the use of counter as nonce. This also allows us to formally show the ability of EtE to accommodate multiple failure conditions, which makes EtE a promising method to construct robust AE.

We further present a transformation of our notion of leakage indistinguishability to leakage simulatability. As indicated by Maurer in [Mau11], the composition property of game-based notions is unclear. Additionally, existing frameworks for composable security, including Universal Composability (UC) [Can01] by Canetti, and Constructive Cryptography (CC) [Mau11] by Maurer, are generally based on simulation-based proof. A transformation to simulatability make it easier for the future study on security composable 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 \( x = (x_1, \ldots, x_\ell) \in S^+ \) 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 \( T \) 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 \( T \) to an empty table, we use the notation \( T \leftarrow \varepsilon \). To assign a value \( V \) to a key \( K \) in \( T \), we use the notation \( T[K] \leftarrow V \). If a value has previously been
assigned to $K$ in $T$, it will be overwritten by $V$. To read a value associated with a key $K$ in $T$ and assign it to $V$, we use the notation $V \leftarrow T[K]$. If there is no value associated with $K$ in $T$, $V$ will be assigned the special value $\perp$.

Let $S$ be a finite set. We define the notation $x \leftarrow S$ to represent the selection of a value from the set $S$ uniformly at random, which we then assign to the variable $x$. For an algorithm $A$, we use the notation $y \leftarrow A^{O_1, O_2, \ldots}$ to denote running $A$ given access to oracles $O_1, O_2, \ldots$, and then assigning of the output of $A$ 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 $A$ 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 $\Pr[A^{\text{INIT}\cdot O_1, O_2, \ldots}] = b]$ as the probability that the adversary $A$ outputs a value $b$ after the INIT procedure and queries to the oracle $O_1, O_2, \ldots$. We denote $\Pr[G(A) = b]$ as the probability that a game $G$ outputs $b$ when the adversary $A$ plays the game $G$. For simplicity, we define $\Pr[G(A)] := \Pr[G(A) \Rightarrow 0]$.

For notion simplicity, we interchangeably use the notation $\Delta_A(O_L; O_R)$ and

$$\Delta_A(O_L) := \Pr[A^{O_L} \Rightarrow 0] - \Pr[A^{O_R} \Rightarrow 0]$$

to denote $A$’s advantage in distinguishing between the oracles $O_L$ and $O_R$. We use the notation $\$O$ to refer to an oracle that, on an input $X$, selects a value $Y$ uniformly at random from the space of all possible outputs with $|Y| = |Y'|$ where $O(X) = Y$, and then returns $Y'$. We implicitly assume that $\$O$ effectively employs lazy sampling, meaning that whenever a repeated input $X$ is queried, $\$O(X)$ always returns the same output, and otherwise samples a fresh uniform value.

We let $\text{Adv}^\Pi_1(A_y)$ denote adversary $A_y$’s advantage in breaking security notion $\Pi$ of a scheme $\Pi$. We say security notion $X$ implies security notion $Y$, denote $X \Rightarrow Y$, if $\text{Adv}^\Pi_1(A_y) \leq c \cdot \text{Adv}^\Pi_1(A_y)$ 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 $\Pi = (E, D)$ specifies two algorithms

$$E : K \times N \times AD \times N \times M \rightarrow C$$

and

$$D : K \times N \times AD \times N \times C \rightarrow M \cup \{ \perp \}$$
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 $(K, N, A, \tau, M) \in \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{M} \times \mathcal{M}$, returns a ciphertext $C \leftarrow \Pi_\mathcal{E}^N_{K,A,\tau}(M)$ such that $C \in \mathcal{C}$ and $|C| = |M| + \tau$. The decryption algorithm $\mathcal{D}$ takes a five-tuple $(K, N, A, \tau, C) \in \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{M} \times \mathcal{C}$, and returns a message $M \leftarrow \Pi_\mathcal{D}^N_{K,A,\tau}(C)$ such that $M \in \mathcal{M} \cup \{\perp\}$. If there is no $M \in \mathcal{M}$ such that $C = \Pi_\mathcal{E}^N_{K,A,\tau}(M)$, then $\Pi_\mathcal{D}^N_{K,A,\tau}(C) = \perp$.

**Definition 2** (Stateful RAE (sRAE)). A stateful robust authenticated encryption (sRAE) scheme is a tuple $\Pi = (\mathcal{E}, \mathcal{D})$ specifies two *stateful* algorithms

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

and

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

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 encryption states. The encryption algorithm $\mathcal{E}$ takes a five-tuple $(K, A, \tau, M; st_\mathcal{E}) \in \mathcal{K} \times \mathcal{AD} \times \mathcal{N} \times \mathcal{M} \times \mathcal{ST}_\mathcal{E}$, returns a ciphertext-state pair $(C; st'_\mathcal{E}) \leftarrow \Pi_\mathcal{E}^A_{K,A,\tau}(M)$, such that $C \in \mathcal{C}$ and $|C| = |M| + \tau$. The decryption algorithm $\mathcal{D}$ takes a five-tuple $(K, A, \tau, C; st_\mathcal{D}) \in \mathcal{K} \times \mathcal{AD} \times \mathcal{N} \times \mathcal{C} \times \mathcal{ST}_\mathcal{D}$, and returns a message-state pair $(M; st'_\mathcal{D}) \leftarrow \Pi_\mathcal{D}^A_{K,A,\tau}(C)$ such that $M \in \mathcal{M} \cup \{\perp\}$. If there is no $M \in \mathcal{M}$ such that $C = \Pi_\mathcal{E}^A_{K,A,\tau}(M)$, then $\Pi_\mathcal{D}^A_{K,A,\tau}(C) = \perp$.

### 3 Security Notions

We introduce the notion IND-CCLA to formalize the robustness of a nonce-based AE scheme, and the notion IND-sf-CCLA for a stateful AE scheme. Our notions can be seen as a natural extension from common notions used to formalize AE security including IND-CCA3 [Sho04] for nonce-based schemes and IND-sfCCA [BKN04] for stateful schemes. We include the expansion parameter $\tau$ defined for RAE scheme in the definitions of our notions. For fixed-expansion schemes, the parameter $\tau$ can be discarded.

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

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

We omit the discussion for errors which adversary trivially knows the result even without querying, for example, ciphertext is shorter than the minimum length supported by a scheme, or ciphertext is not a multiple of the block size etc. Such query does not grant the adversary with extra advantage in distinguishing or forging since the adversary trivially knows the result of such a query.
Remark 1. Inspired by SAE notion introduced by Barwell et al. in [BPS15], we use a leakage simulator function $L$ to capture those inadvertent leakage. We improve the definition to capture the leakage of multiple candidate plaintexts and multiple error flags. We define the leakage simulator function $L$ for an AEAD scheme $\Pi$ as in Definition 3. We present the definition with respect to a nonce-based scheme, the nonce space $\mathcal{N}$ can be replaced with decryption state space $\mathcal{S}_D$ for the definition for stateful schemes.

**Definition 3.** The leakage simulator function $L$ 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

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

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 $\Pi.D$, such that the following conditions hold:

1. $L^{N,A,T}_{\Pi,K}(C) = \top$ if there is no leaked plaintext and $|\mathcal{S}_\perp| = 1$. This holds regardless of whether $\Pi.D^{N,A,T}_{\Pi,K}(C) = \perp$ or not for a queried ciphertext $C$.
2. $L^{N,A,T}_{\Pi,K}(C) = \top$ if $C$ is a valid ciphertext and $L^{N,A,T}_{\Pi,K}(C) \neq \bot$.
3. $L^{N,A,T}_{\Pi,K}(C) = (M, \perp)$ if there is a leaked bitstring $M$ with $|M| > 0$ and $|\mathcal{S}_\perp| = 1$.
4. $L^{N,A,T}_{\Pi,K}(C) = (M_i, \perp_i) \in \{0,1\}^\ell \times \mathcal{S}_\perp$, 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 $\perp_i$, $M_i$ is set to the empty string $\varepsilon$. It is assumed that $\Pi.D$ halts upon encountering $\perp_i$.
5. The correctness is defined by: if $\Pi.D^{N,A,T}_{\Pi,K}(C) = \perp$, then $L^{N,A,T}_{\Pi,K}(C) \neq \top$, and if $L^{N,A,T}_{\Pi,K}(C) = \top$, then $\Pi.D^{N,A,T}_{\Pi,K}(C) \neq \perp$.

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

Also, we stress that $M \in \{0,1\}^\ell$ is a bitstring intending to capture all the bitstrings obtained before an error $\perp_i$ or when $|\mathcal{S}_\perp| = 1$. This includes intermediate values during the decryption, such as a reconstructed initialization vector (IV) etc. Essentially, for a set of plaintexts (or rather, bitstrings) $M_1, \ldots, M_n$, we can rewrite them as $M = M_1 || \ldots || M_n$, and require the indistinguishability of $M$ as a whole.

Indeed, in Bulletpoint 4, there are schemes that does not halt immediately when an error is detected. However, this does not affect our following discussion since in our security notion we either require the existence of only one error flag or 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 simulator 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 $L^{N,A}_{\Pi,K}(C) = \top$.
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 $\mathcal{L}^{N,A}_{\Pi_k}(C) \in \{\varepsilon, \bot, (M, \bot)\}$, where $\bot$ indicates a failure on the authenticity check with tag, $\bot$ indicates an error in decoding, and $M \in \{0,1\}^*$ denotes the plaintext in incorrect format.

**Error Merging.** If multiple error flags are results of the predicates on the same leaked plaintext $M$, then the scheme can “merge” those error flags into one error flag using logic operators without incurring extra plaintext leakage. Consequently, $M$ fails all the checks simultaneously if one of the failure conditions is met, thereby circumventing the need for checks across multiple phases. We state the observation assuming without loss of generality that $v(\cdot) = \text{true}$ leads to successful decryption. The observation also holds if we assume $v(\cdot) = \text{true}$ leads to unsuccessful decryption by changing $\land$ to $\lor$.

**Observation 1.** Let $\bot_i = v_i(M)$ and $\bot_j = v_j(M)$ with $i \neq j$, where $v_i$ and $v_j$ are condition predicates on plaintext, and $M$ is the candidate plaintext to be verified. Then there is a merged error flag $\bot_{i,j} = v_{i,j}(M)$ where $v_{i,j}(M) = v_i(M) \land v_j(M)$. Notably, if all the condition predicates are on the same candidate plaintext $M$, then there is an error flag $\bot' = v'(M)$ where $v'(M) = v_1(M) \land v_2(M) \ldots \land v_n(M)$ and $n = |S_\bot|$.

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

### 3.1 IND-CCLA Security

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

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

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

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

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

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

1. **IND-CCLA1 (Error Invariance):** The tuple $(M_\lambda, \bot)$ is returned in the ideal world for a $\bot \in S_\bot$. Our goal with this sub-notion is to ensure that:
   - The adversary cannot distinguish between the leaked plaintext and a random bitstring of the minimum leakage length defined by the scheme.
   - The adversary cannot trigger an error flag except for $\bot$.

2. **IND-CCLA2 (Unicity):** The tuple $(M_\lambda, \bot)$ is returned in the ideal world for a $\bot \in S_\bot$. Our goal with this sub-notion is to ensure that:
   - The adversary cannot distinguish between the leaked plaintext and a random bitstring of the minimum leakage length defined by the scheme.
   - The adversary cannot trigger an error flag except for $\bot$.
2. IND-CCLA2 (Error Unicity): The tuple \((M_\lambda, \bot)\) is returned in the ideal world. With this notion, in addition to ensuring that indistinguishability of the leaked plaintext, we require that \(L()\) also discloses only one error just like decryption even there are multiple failure conditions. This allows us to align the actual leakage as closely as possible with the behavior of a single-error decryption function, thus to mitigate the security issue caused by disclosing multiple implicit error flags via leakage. Several practical considerations arise from this notion includes:

- The adversary cannot obtain meaningful plaintext as long as one of the failure conditions is met.

- In the event that the scheme fails to verify a failure condition (e.g., due to implementation flaws), the adversary remains oblivious to such a flaw unless it successfully passes all other checks.

- If the adversary successfully passes a check (either by satisfying the condition predicate or bypassing the check through a side channel), it should not be confident that its strategy effectively breaks the check.

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

Observe that if there is not leakage at all i.e., $L(\cdot) = \bot$, then it trivially satisfies both IND-CCLA1 and IND-CCLA2 security. Also when $|S_\perp| = 1$, then IND-CCLA1 converges to IND-CCLA2 since there is no other error flag to be distinguished from the only error flag $\bot_1 \in S_\perp$. Thus to better syntactically separate between these two sub-notions, we define $L$ to output $\bot$ when $|S_\perp| = 1$ as also discussed in Remark 1. Additionally, a scheme with $|S_\perp| \geq 2$ cannot be IND-CCLA2 since almost for any query $\bot_1$ will be output to be distinguished from $\bot$.

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

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

**Proof (Sketch).** (IND-CCLA2 $\rightarrow$ IND-CCLA1). Suppose we have an adversary $A$ that breaks IND-CCLA1 security. Then $A$’s query to Leak yields $(M, \bot_i)$ with $|M| \geq |M_\lambda|$ or $\bot_i \neq \bot_1$ to be distinguished from $(M_\lambda, \bot_i)$ where $M_\lambda$ is the random bitstring of the minimum leakage length. In all the cases, we can use $A$ to distinguish from $(M_\lambda, \bot)$.

(IND-CCLA1 $\not\rightarrow$ IND-CCLA2). Consider an AE scheme that is IND-CCLA1 secure with two error flags $\bot_1$ and $\bot_2$. It yields immediate distinguishing since $\bot_1$ will be output to be distinguished from $\bot$ 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 $x_\perp \in \{\bot_1, \bot_2\}$ respectively. For simplicity, we slightly abuse the notations to let $E_K$, $D_K$ and $L_K$ be ENC, DEC and LEAK in game $G^\text{IND-CCLA}_x$ in Figure 1 respectively, and we let $S^\text{E}$ be the LEAK oracle as in Figure $G^\text{IND-CCLA}_x$ in Figure 1. For notation simplicity, we use IND-EPL to denote both IND-EPL1 and IND-EPL2 if a result applies to both notions.

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

\[
\text{Adv}^\text{IND-EPL}_x(A) := \Delta_A \left( E_K, D_K, L_K \right) \quad \text{for key } K \leftarrow \mathcal{K} \text{ and } x \in \{1, 2\}.
\]

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

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

**Prohibited & Pointless Queries.** We specify the following generally prohibited queries to prevent trivial wins. We let the oracles return the invalid symbol $\dagger$ for those prohibited queries.

For IND-CCLA, the adversary is restricted from using the output of ENC to query DEC, and it is also prohibited from repeating a query to ENC or LEAK with the same tuple. For IND-EPL, the only requirement 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 \( \text{Enc} \) to \( \text{Dec} \) serves no purpose since decryption is executed honestly in both the real and ideal settings.

Also, we do allow the adversary to repeat the nonce to capture the security when a nonce is possibly misused. Additionally, we stress that, we allow an adversary to query with variable stretch parameter, that is, the adversary can query with \( \tau_1 \neq \tau_2 \) in different queries. Indeed, with small stretch, the adversary may trivially win the INT-CTXT game. However, this still captures the best achievable security with respect to a selected stretch parameter.

### 3.2 IND-sf-CCLA Security

We describe the game for IND-sf-CCLA notion for stateful (robust) AE as in Figure 2. This notion 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 6.

**Definition 6 (IND-sf-CCLA\(_x\)).**

\[
\text{Adv}_{\Pi}^{\text{IND-sf-CCLA}\_x}(A) := \text{Pr}[G_{\Pi}^{\text{IND-sf-CCLA}\_x-0}(A)] - \text{Pr}[G_{\Pi}^{\text{IND-sf-CCLA}\_x-1}(A)]
\]

for \( x \in \{1, 2\} \).

**Proposition 2.** IND-sf-CCLA\(_2\) implies IND-sf-CCLA\(_1\) for a scheme that includes at least two implicit error flags i.e., \( |S_\perp| \geq 2 \).

**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 \( \text{Enc}, \text{Dec} \) and \( \text{Leak} \) respectively in game \( G_{\Pi}^{\text{IND-sf-CCLA}\_x-0} \) in Figure 2. Additionally, we use \( \$^L \) to denote the oracle \( \text{Leak} \) in game \( G_{\Pi}^{\text{IND-sf-CCLA}\_x-1} \) in Figure 2. We define IND-sf-EPL as follows.

**Definition 7 (IND-sf-EPL\(_x\)).**

\[
\text{Adv}_{\Pi}^{\text{IND-sf-EPL}\_x}(A) := \Delta_A \left( \mathcal{E}_K, \mathcal{D}_K, \mathcal{L}_K, \mathcal{L}_K, \mathcal{L}_K, \$^L \right)
\]

for key \( K \leftarrow \mathcal{K} \) and \( x \in \{1, 2\} \).

**Corollary 2.** IND-sf-EPL\(_2\) implies IND-sf-EPL\(_1\) for a scheme that includes at least two implicit error flags i.e., \( |S_\perp| \geq 2 \).

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

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

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

Similarly, it is pointless to query in-order ciphertext from \( \text{Enc} \) to \( \text{Leak} \) since the oracle yields \( \perp \) in both real and ideal world.
3.3 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 \( A \), there exist an IND-CPA adversary \( A_{cpa} \), an INT-CTXT adversary \( A_{int} \) and an IND-EPL\( x \) adversary \( A_{opt} \) such that

\[
\text{Adv}_{\Pi}^{\text{IND-CCLA}\, x}(A) \leq \text{Adv}_{\Pi}^{\text{IND-CPA}}(A_{cpa}) + \text{Adv}_{\Pi}^{\text{INT-CTXT}}(A_{int}) + \text{Adv}_{\Pi}^{\text{INT-EPL}\, x}(A_{opt}).
\]
Figure 3: An illustration of implications between notions. We use $A \rightarrow B$ to denote that notion $A$ implies notion $B$. We use $Ax \leftrightarrow Bx$ to denote that $Ax$ implies $Bx$ only when $x$ is the same value for both $Ax$ and $Bx$. We use $Ax \leftrightarrow Ax$ to denote $A2$ implies $A1$.

**Proof.** We rewrite the advantage as

$$\text{Adv}^{\text{IND-CCCA}_x}(A) = \Delta_A \left( \mathcal{E}_K, D_K, \mathcal{L}_K \right) + \Delta_A \left( \mathcal{E}_K, D_K, \mathcal{L}_K \right) .$$

By definition, we have that

$$\text{Adv}^{\text{IND-CCCA}_x}(A) \leq \text{Adv}^{\text{IND-CCCA}_x}(B) .$$

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

$$\text{Adv}^{\text{IND-CCCA}_x}(A) \leq \text{Adv}^{\text{IND-CPA}}(A_{\text{epl}}) + \text{Adv}^{\text{INT-CTXT}}(A_{\text{int}}) + \text{Adv}^{\text{IND-EPL}_x}(A_{\text{epl}}) .$$

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-sfCCLA advantage in [BKN04] and we follow the definition of IND-sfCCLA advantage in [BKN04].

**Theorem 2.** For $x \in \{1, 2\}$, for any IND-sf-CCLA adversary $A$, there exist an IND-CPA adversary $A_{\text{epl}}$, an INT-sf-CTXT adversary $A_{\text{int}}$ and an IND-sf-EPL adversary $A_{\text{epl}}$ such that

$$\text{Adv}^{\text{IND-sf-CCLA}_x}(A) \leq \text{Adv}^{\text{IND-CPA}}(A_{\text{epl}}) + \text{Adv}^{\text{INT-sf-CTXT}}(A_{\text{int}}) + \text{Adv}^{\text{IND-sf-EPL}_x}(A_{\text{epl}}) .$$

**Proof.** The proof follows a similar proof of Theorem 1 by replacing IND-CCA3 with IND-sfCCA.
Implication between Notions. The following set of relationships is inherently obvious. We present them here to provide completeness and we omit proofs since they are trivial.

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

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

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

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

**Proof (Sketch).** (IND-EPL \( \not\Rightarrow \) INT-CTXT). We consider an EtM scheme where \( C = M \oplus K_E \) and \( T = C \oplus K_M \), with final output as \( C || T \). From Example 1, we know \( L(\cdot) = \bot \). Thus both oracles will output \( \bot \) meaning that IND-EPL advantage is 0. Nevertheless, an adversary can forge a valid ciphertext by querying the encryption oracle to obtain \( C || T \) and returning \( C \oplus 1^n || T \oplus 1^n \) as forgery.

(IND-CCA3 \( \not\Rightarrow \) 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 \( L(\cdot) = (M, \bot) \) 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 \( L(\cdot) \in \{(M, \bot_1), (M, \bot_2)\} \), where \( \bot_1 \) indicates the authenticity failure, and \( \bot_2 \) indicates incorrect encoding. By also changing the tag for a valid ciphertext from a previous encryption query, the adversary can distinguish \( M \) from a random bitstring.

3.4 Comparison with Existing Notions

We make a brief comparison 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 [ABL14].

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

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

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

Given that our notions share a similarity with the subtle AE, a result of the equivalence between RAE and SAE has been showed in [BPS15, Theorem 14]. To eliminate the influence of multiple error flags, we restate the theorem under the condition that \( |S_\perp| = 1 \).

Additionally, a key difference between our notion and SAE is that SAE compares \( \pi \) where \( A \) to its oracle \( \Pi \) with error space \( \Pi \). Specifically, \( S_\perp \) is the number misused nonce of \( A \)'s queries, \( q \) is the number of queries, and \( m \) is the length of the shortest string in the message space.

\[ \text{Proof.} \] We write the RAE advantage as

\[ \text{Adv}_{\Pi}^{\text{RAE}}(A) = \Delta_A \left( \varepsilon_K, D_K, \mathcal{L}_K \right) \]

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

\[ \left| \text{Adv}_{\Pi}^{\text{RAE}}(A) - \text{Adv}_{\Pi}^{\text{IND-CCLA2}}(A) \right| = \left| \Delta_A \left( \varepsilon_K, D_K, \mathcal{L}_K \right) - \Delta_A \left( \varepsilon_K, D_K, \mathcal{L}_K \right) \right| \leq \Delta_A \left( \varepsilon, \perp, \mathcal{L} \right) \]

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

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

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

\[ \text{Proposition 6.} \] IND-EPL implies INV-ERR.

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

Now if \( |S_\perp| = 1 \), then \( \text{Adv}_{\Pi}^{\text{INV-ERR}}(A) = 0 \) since there is no \( \perp_j \in S_\perp \) to be distinguished from \( \perp_i \). However, \( B \) may still distinguish by leaked plaintext \( M \) to break IND-EPL2. Thus we have \( \text{Adv}_{\Pi}^{\text{INV-ERR}}(A) \leq \text{Adv}_{\Pi}^{\text{IND-EPL2}}(B) \).
**Subtle AE.** In [BPS15], Barwell et al. introduced the concept of a leakage simulator function. They define the leakage function $\mathcal{L}$ as $\mathcal{L} : K \times N \times AD \times C \to \{\top\} \cup S_\lambda$ where $S_\lambda$ 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

$$\text{Adv}_{\Pi}^{\text{SAE}}(A) = \Delta_A \left( \mathcal{E}_K, D_K, \mathcal{L}_K \right)$$

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

$$\text{Adv}_{\Pi}^{\text{ERR-CCA}}(A) = \Delta_A \left( \mathcal{E}_K, D_K, \mathcal{L}_K \right).$$

We observe that the definition is in a very generalized way due to the wide scope of $S_\lambda$. Barwell et al. claimed that it is intended to give authors the flexibility to define the leakage with such a configuration. Nevertheless, such flexibility may let authors overlook certain vulnerabilities. For instance, in RIV [AFL+16], Abed et al. defined the leakage only as the plaintext $M$. However, their design actually includes an IV which can be malleable by the tag $T$.

Additionally, comparison to $\mathcal{L}_{K'}$ with a different key $K'$ does not adequately capture the impact of the leakage itself due to an overlap with the integrity notion. For a valid ciphertext $C$, $\mathcal{L}_{K'}(C)$ yields $\top$ while $\mathcal{L}_{K'}(C)$ almost for sure outputs something other than $\top$ to be distinguished. This requires us to always consider an INT-CTXT adversary when bounding the advantage of a leakage adversary. This is more apparent when there is no leakage. An adversary should have 0 advantage in distinguishing by leakage when there is no leakage but we have to consider an integrity adversary then. Thus to resolve this issue, we let the oracles in both real and ideal world to return $\top$ or $\perp$ when the leakage function yields such an output. In Proposition 7, we show that IND-EPL implies ERR-CCA when all the ciphertexts to LEAK are not valid.

**Proposition 7.** IND-EPL implies ERR-CCA when $\mathcal{L}_K(C) \neq \top$ and $\mathcal{L}_K(C) \neq \perp$ for all ciphertexts $C$ queried by the adversary.

**Proof.** (ERR-CCA $\not\rightarrow$ IND-EPL1). We consider the example in [BPS15]. Suppose a scheme that is INV–ERR, then there is a simple variant that upon triggering an error returns $\bot_1$ or $\bot_2$ depending on the first ciphertext bit. Then it is trivially IND-EPL1.

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

(IND-EPL2 $\rightarrow$ ERR-CCA). Suppose $|S_\bot| \geq 2$ by Corollary 1. Now suppose $|S_\bot| = 1$ and $\mathcal{L}$ implements a permutation. An ERR-CCA adversary has 0 advantage in distinguish $\mathcal{L}_K(\cdot)$ and $\mathcal{L}_{K'}(\cdot)$. However, an IND-EPL adversary can observe the repeated output from LEAK to distinguish.

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

**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 example on EtE, we can essentially consider the indistinguishability of plaintext as a random bitstring. We then define it in Definition 8.

**Definition 8 (PA2).** Let \( \tilde{D} \) be the decryption function without any check for failure such that \( \tilde{D} \) always output a plaintext, then

\[
\text{Adv}_{PA2}^{0}(A) := \Delta_{A}(\tilde{E}_{K}, \tilde{D}_{K}; \tilde{E}_{K}, \tilde{D})
\]

for key \( K \leftarrow \mathbb{K} \).

Notably, PA2 security considers the indistinguishability of the plaintext when all checks are bypassed, whereas our IND-EPL2 notion stresses the indistinguishability (or rather, uniqueness) of the error message itself, thus to prevent a particular error flag to be used for some side channel attacks. As discussed in many studies including [Vau02, CHVV03, PRS11], the revelation of the error message alone can lead to significant attacks. In Proposition 8, we separate our IND-EPL2 notion from PA2 security. However, we acknowledge the significance of PA2 security in ensuring the AE robustness as it guarantees the confidentiality of the plaintext when all check mechanisms are circumvented. We conclude that an AE scheme should achieve both IND-EPL2 and PA2 to be considered robust.

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

**Proof.** (IND-EPL2 \( \not\rightarrow \) PA2). We consider Encrypt-then-MAC paradigm as in Example 1. We know the IND-EPL2 advantage is 0 since \( L(\cdot) = \bot \). However, it is trivial to break PA2 security by changing the tag of a ciphertext obtained from a previous encryption query.

(PA2 \( \not\rightarrow \) IND-EPL2). We consider the Encode-then-Encipher paradigm with \( |S_{\bot}| = 2 \). Thus \( L(C) \in \{(M, \bot_{1}), (M, \bot_{2})\} \) where \( \bot_{1} \) and \( \bot_{2} \) denotes two failures on encoding, and \( M \) is the deciphered string. Then it is PA2 secure if the cipher is secure as a pseudorandom permutation. However, it is not IND-EPL2 secure since for almost every query \( \bot_{1} \) will be output to be distinguished from \( \bot \).

\[\square\]

4 Stateful Security of Encode-then-Encipher

Encode-then-Encipher (EtE), proposed by Bellare and Rogaway in [BR00], is the mainstream way to construct robust AE. In [HKR15], Hoang et al. proved that EtE with VIL cipher achieves the security as a pseudorandom injection (PRI). However, there has been limited studies about the stateful security of EtE in presence of multiple errors. Notably, robustness of EtE as the ability to admit multiple errors has been briefly mentioned by Shrimpton and Terashima in [ST13]. We here present a formal treatment of this property through our notions.

In [BMM+15], Badertscher et al. showed the security of the Encode-then-Encipher from the view of composable security with the Constructive Cryptography (CC) framework proposed by Maurer in [Mau11], by constructing a random injection channel (RIC) from a uniform random injection (which ideally models a VIL cipher) and an insecure channel. The RIC models an ideal world in which a counter is used as nonce, and the adversary only has knowledge of the associated data and the message length. We then follow the idea of [BMM+15] by also using counter as nonce and prove the security and robustness of EtE from a more generalized game-based perspective with our notion.
4.1 EtE with Tweakable Cipher

We consider a tweakable cipher $E : \mathcal{K} \times T \times M \to \mathcal{C}$ as described in [LRW02]. Here we set the tweak space $T = N \times \mathcal{AD} \times \mathbb{N}$. We define a robust AE scheme $\Pi = (E, D)$ using EtE as follows. Let $C = \Pi \cdot E^{N,\cdot,A,T}(M) = E_{K,N,A,T}(M||0^\tau)$ and return $C$ as ciphertext. Let $M' = E_{K,N,A,T}^{-1}(C)$. Then if $M'$ ends with $\tau$ zeros, $\Pi \cdot D^{N,\cdot,A,T}(C)$ returns $M'$ excluding ending $\tau$ zeros as plaintext. Otherwise, $\Pi \cdot D^{N,\cdot,A,T}(C)$ returns $\bot$.

The security of a tweakable block cipher is defined as (strong) indistinguishability from tweakable random permutation $((\pm)\text{PRP})$, which is a random permutation parameterized by tweak $T$. To adapt this notion to a VIL cipher, we introduce an additional length parameter. Let $\hat{T}$ represent the set of all tweakable permutations on $\{0, 1\}^\ell$. For each pair $(T, \ell) \in \hat{T} \times \mathbb{N}$, we define $\pi_{\hat{T}}(\cdot)$ as a tweakable permutation sampled independently and uniformly at random from $\hat{T}$.

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

$$\Pr[A_{\hat{T}}(\cdot) \Rightarrow 0] - \Pr[A^\hat{T}(\cdot) \Rightarrow 0] = 0$$

for every $\ell \geq 0$, where $\pi_{\hat{T}} : \{0, 1\}^\ell \to \{0, 1\}^\ell$ is a tweakable random permutation and $\hat{S}$ 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 $1/2^\ell$ 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 $1/2^\ell$ at each query. Consequently, both oracles exhibit the same distribution, meaning that the adversary has 0 advantage in distinguishing between these two oracles. $\square$

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 $A$ 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 $A$ against the EtE construction $\Pi$ making $q_d$ decryption queries, there is a $\pm$PRP adversary $A_{\text{stprp}}$ against the tweakable VIL cipher $E$ such that

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

where $\tau$ is the minimum expansion parameter queried by $A$.

**Proof.** The proof follows by combining Theorem 2 with Lemmas 2, 3 and 4. $\square$

**Lemma 2.** For any IND-CPA adversary $A$ against the EtE construction $\Pi$ making $q$ encryption queries, there is a PRP adversary $A_{\text{sprp}}$ against the tweakable VIL cipher $E$ such that

$$\text{Adv}_{\Pi}^{\text{IND-CPA}}(A) = \text{Adv}_{E}^{\text{PRP}}(A_{\text{sprp}}).$$
Proof (Sketch). We consider three games \( G_0 - G_2 \) where adversary’s queries are answered with the tweakable VIL cipher \( \tilde{E} \), a tweakable random permutation \( \tilde{\pi} \), and a random bitstring of length \(|M| + \tau\) respectively. We have that

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

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

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

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

where \( \tau \) is the minimum expansion parameter queried by \( \mathcal{A} \).

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

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

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

We first define the leakage simulator function \( \mathcal{L} \) for the EtE paradigm. Consider that during the decryption, \( M' = \tilde{E}^{-1}_{K,N,A,\tau}(\mathcal{C}) \) 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 \( \mathcal{L}_{\Pi,N}^{N,A,\tau}(\mathcal{C}) = (M',\bot) \) for an invalid ciphertext \( \mathcal{C} \).

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

\[
\text{Adv}_{\Pi}^{\text{IND-sf-EPL2}}(\mathcal{A}) = \text{Adv}_{\tilde{E}}^{\pm \tilde{\text{PRP}}}(\mathcal{A}_{\text{stprp}}).
\]

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

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

Similarly, we bound \( \Pr[G_0(\mathcal{A})] - \Pr[G_1(\mathcal{A})] \) by a \( \tilde{\text{PRP}} \) adversary \( \mathcal{A}_{\text{tprp}} \). 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. \( \square \)
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 [HHI+22], Hosoymada 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 Encode-then-Encipher paradigm is that: when there exists redundancy in the plaintext, such redundancy can be exploited to establish or enhance authenticity. We define the density of message space $\mathcal{M}$ to measure how redundant the message space is as in Definition 9.

**Definition 9** ($\delta$-dense). Let $v : \{0,1\}^\ell \rightarrow \{\text{true, false}\}$ 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
\[
\Pr[\forall M \in \mathcal{M} : v(M) = \text{true}] \leq \delta.
\]

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 $A$ against the EtE construction $\Pi$ making $q$ decryption queries, there is a $^\pm\text{PRP}$ adversary $A_{\text{stprp}}$ against the tweakable VIL cipher $\tilde{E}$ such that
\[
\text{Adv}_{\Pi}^{\text{INT-sf-CTXT}}(A) \leq \text{Adv}_{\tilde{E}}^{^\pm\text{PRP}}(A_{\text{stprp}}) + \frac{\delta q}{2^\ell}.
\]

**Leakage with Multiple Errors.** Suppose that $\mathcal{M}$ is $\delta$-dense, the possible leakage tuples are $(M', \bot_1)$ and $(M', \bot_2)$. Fixing $\bot_1$ as the error flag to be distinguished, for IND-EPL1 security which requires that the adversary should not see the error flag $\bot_2$, 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 $A$ against the EtE construction $\Pi$ making $q$ leakage queries, there is a $^\pm\text{PRP}$ adversary $A_{\text{stprp}}$ against the tweakable VIL cipher $\tilde{E}$ such that
\[
\text{Adv}_{\Pi}^{\text{IND-sf-EPL1}}(A) \leq \text{Adv}_{\tilde{E}}^{^\pm\text{PRP}}(A_{\text{stprp}}) + \frac{q}{2^\ell}
\] (2)

if $\bot_1$ implies $M'$ does not ends with $\tau$ zeros, or
\[
\text{Adv}_{\Pi}^{\text{IND-sf-EPL1}}(A) \leq \text{Adv}_{\tilde{E}}^{^\pm\text{PRP}}(A_{\text{stprp}}) + \delta
\] (3)

if $\bot_1$ implies $v(M') = \text{false}$.

**Leakage with Merged Error.** Note that the condition predicates both concern the same leaked plaintext $M'$. Following Observation 1, we can merge them into one leakage tuple $(M', \bot)$. If $M'$ satisfies one of the condition predicates, the leaked plaintext may exhibit certain property e.g., with $\tau$ ending zeros. Nevertheless, the probability that the oracle in ideal world generates such a bitstring is the same, yielding the adversary no more advantage than flipping a coin. Thus the adversary has no advantage in distinguishing only by leaked plaintext.
Nevertheless, the adversary can break $\text{IND-sf-EPL1}$ security by looking for $\perp_2$ to identify the real world when one of the checks passes (since $\perp_1$ is always output in the ideal world). After merging two errors into one, $\perp$ will be output instead of $\perp_2$ thus the adversary can no longer distinguish by error flag. This allows to reduce the adversary’s advantage by removing the term $\frac{q^2}{2^\ell}$ from Equation 2, and $\delta$ from Equation 3, to obtain the bound for $\text{IND-sf-EPL2}$ advantage as in Corollary 6.

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

$$\text{Adv}_H^{\text{IND-sf-EPL2}}(A) = \text{Adv}_E^{\pm\text{PRP}}(A_{\text{stprp}}) + \frac{q^2}{2^\ell}$$

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

**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 $\frac{q^2}{2^\ell}$. \hfill $\square$

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

$$\text{Adv}_H^{\text{IND-CPA}}(A) = \text{Adv}_E^{\text{PRP}}(A_{\text{tprp}}) + \frac{q^2}{2^\ell}$$

where $\ell = \min_{(N,A,M,\tau) \in Q} \{|C| : (M, \perp) = \Pi.E^N_{K,A,\tau}(C)\}$ where $Q$ is the set of queries made by $A$.

**Proof.** The proof follows a similar proof to Lemma 5. \hfill $\square$

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

$$\text{Adv}_H^{\text{IND-CPA}}(A) = \text{Adv}_E^{\pm\text{PRP}}(A_{\text{stprp}}) + \frac{q^2}{2^\ell}$$

where $\ell = \min_{(N,A,C,\tau) \in Q} \{|M| : (M, \perp) = \Pi.E^N_{K,A,\tau}(C)\}$ where $Q$ is the set of queries made by $A$.

**Proof.** The proof follows a similar proof to Lemma 5. \hfill $\square$

### 5 Revisiting EEM Robustness

**Some Observation on Standard EEM.** We revisit the EEM paradigm as in Example 1. We consider two predicates $v_1$ and $v_2$ where

$$v_1(C,T) = \text{"MAC.}\mathcal{V}(C,T) = \text{true}$$

and

$$v_2(M) = \text{"DECODE}(M) \neq \text{inval}.$$


We first consider the case where \( \mathcal{L}(\cdot) \in \{ (\perp, \perp_1), (M, \perp_2) \} \) for an invalid ciphertext \( C \), where \( \perp_1 = v_1(C) \) and \( \perp_2 = v_2(M) \), and \( M \) represents the bitstring that has not undergone the decoding process. Following the result in [BDPS14], it is easy to see that if the encryption scheme is IND-CPA and the MAC scheme is SUF-CMA, then \( \text{EEM} \) is IND-CCLA1 secure by setting \( \perp_i = \perp_1 \) in the notion definition.

Now suppose that the scheme fails to verify the validity of the tag such that \( v_1(C, T) = \text{true} \) for all \( C \) and \( T \), then there is an unavoidable leakage of plaintext (in incorrect encoding) since \( v_2 \) is on \( M \). Thus seeing \( (M, \perp_2) \), the adversary \( A \) immediately knows that the tag check is broken and the only error now is with the encoding. Then \( A \) can continue with other attacks based on the error message and leaked plaintext. Similarly, if the adversary manages to forge a tag, then \( A \) is certain that its strategy is successful by observing the appearance of \( (M, \perp_2) \). Then it can adopt that in the future forgery.

Now suppose that the scheme merges two errors to obtain \( \perp' = v_1(C) \land v_2(M) \). Note that \( A \) can actually obtain a ciphertext \( C' \) decrypted to \( M \) with \( v_2(M) = \text{true} \) from prior encryption query. By changing tag of \( C \) to an “obvious” invalid one, \( A \) can infer that when \( v_1(C, T) = \text{true} \) for all \( C \) and \( T \) since it knows \( v_2(M) = \text{true} \), making such a merge meaningless.

**Proposition 9.** \( \text{EEM} \) is not IND-CCLA2 secure.

**Discussion on Other Constructions.** Several constructions have been proposed aiming to boost the AE robustness from various perspectives.

RIV [AFL+16], inspired by SIV [RS06], utilizes the Feistel structure [Fei73] and incorporates an extra tag in their scheme. However, we believe that tag-based schemes generally suffer from tag malleabilization issue, resulting in decryption distinguishability. Specifically, the extra tag in RIV makes it vulnerable to predictable IV attacks [DR11].

RIV computes \( IV = F(N, A, C) \oplus T \) for decrypting \( C \), where \( F \) is a PRF. By querying \( T_1, T_2 \) with \( T_1 \neq T_2 \) with fixed \( (N, A, C) \), the leaked plaintext reveals \( M_1 \oplus M_2 = T_1 \oplus T_2 \) in CBC mode. Additionally, RIV should consider IV as a part of leakage whereas it only considered leaked plaintext. This makes RIV also vulnerable when CTR mode is used by querying different \( C_1, C_2 \) and \( T_1, T_2 \) such that the same IV is produced.

There are also works utilizing \( \texttt{EtE} \) to establish authenticity. In [ST13], Shrimpton and Terashima proposed PIV which utilizes three-round Feistel structure to obtain a tweakable block cipher. In Degabriele and Karađžić proposed SIV and showed the idea of authenticity from nonce. Following our result, \( \texttt{EtE} \) paradigm can be generally considered robust. Future work on \( \texttt{EtE} \) should include how to build VIL cipher from blockcipher mode of operation to allow easier transform from schemes that are used more commonly in practice.

6 Transformation to Simulatability

We observe that the transformation from game-based proof with real-or-ideal oracles to simulation-based proof is natural and straightforward. That is because the security with simulation-based proof is essentially bounded by a distinguisher \( D \) which distinguishes the world where an adversary \( A \) interacting with the real functionality \( \text{Real}_f \) from the world where a simulator \( S \) interacting with an ideal functionality \( \text{Ideal}_f \). Here we present a mapping between our game-based notions with simulation-based notions to show the simulatability.

Notably, in [DF18], Degabriele and Fischlin have shown the equivalence between IND-CPA and encryption simulatability (ES), and the implication from decryption simulatability with integrity (DS-I) to IND-CTXT. It leaves us to show that we can turn IND-EPL notions to simulatability.
22 A Characterization of AE Robustness

Figure 4: An emulator \( L \) for the leakage. We let \( \psi \) be the function that evaluates the leakage length with respect to \(|C|\) and \( \tau \).

Figure 5: A simulator \( S \) for the adversary \( A \). We let the queue \( T \) be the transcript of results.

As for ideal functionality, we should be able to emulate the leakage without any knowledge of the leakage contents except its length (when there is decryption failure). Thus we consider an emulator \( L \) of the leakage as 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 \( \ell^* \) should be computable. We then simulate the leakage by sampling a random bitstring of that length.

Now we consider the simulator \( S \) of the adversary \( A \) as described in Figure 5. We fix the honest execution of \( E_K(\cdot) \) and \( D_K(\cdot) \), which allows the simulator \( S \) to determine when there is a decryption failure. For each of \( A \)'s query to \( \text{Enc} \) and \( \text{Dec} \), we let \( S \) simply repeat such a query. For \( A \)'s query to \( \text{Leak} \), we let \( S \) first determine if the decryption fails. In such a case, \( S \) simply forwards \(|C|\) and \( \tau \) to the emulator \( L \) to obtain the simulated leakage. We use a queue \( T \) to keep track the transcript of the query results. Note that when defining \( S \) and \( L \), we omit the case when \( L(\cdot) = \perp \). That is because the emulator \( L \) should not have the access to \( L \). Nevertheless, when \( L(\cdot) = \perp \), distinguishing via leakage is impossible since there is no leakage.

We can then similarly define leakage simulatability as the advantage of a distinguisher \( D \) in distinguishing between the transcripts by \( A \) and \( S \), as in Definition 10.

Definition 10 (Leakage Simultatability (LS)).

\[
\text{Adv}^{\text{LS}}_D(\Pi) := \Pr[A^E_K \cdot D_K \cdot L \Rightarrow T] - \Pr[S^E_K \cdot D_K \cdot L \Rightarrow T]
\]

for key \( K \leftarrow \mathcal{K} \).
Proposition 10. LS is equivalent to IND-EPL2.

Proof. We rewrite the advantages as
\[
\left| \text{Adv}^{LS}_\Pi (D) - \text{Adv}^{\text{IND-EPL2}}_\Pi (A) \right| = \left| \Delta_D \left( \frac{\mathcal{E}_K, \mathcal{D}_K, \mathcal{L}_K}{\mathcal{E}_K, \mathcal{D}_K, \mathcal{L}_K} \right) - \Delta_A \left( \frac{\mathcal{E}_K, \mathcal{D}_K, \mathcal{L}_K}{\mathcal{E}_K, \mathcal{D}_K, \$\mathcal{L}} \right) \right| \\
\leq \Delta \left( \frac{\mathcal{E}_K, \mathcal{D}_K, \mathcal{L}}{\mathcal{E}_K, \mathcal{D}_K, \$\mathcal{L}} \right)
\]

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

Notably, in [DF18], Degabriele and Fischlin also extended their study to capture stateful security via simulatability, as well as in presence of multiple ciphertext fragments. They further studied the composability of their notions to a simulatable channel. There are many frameworks on the security composability including the famous Universal Composability (UC) framework by by Canetti in [Can01]. Maurer also introduced Constructive Cryptography [Mau11] (CC) framework, which we believe may serve a better example in terms of channel composability.

As indicated in [Mau11], an AE scheme can essentially be considered as a composition of a confidential channel $\text{CONF}$ where only message length is leaked but injection into channel from adversary is allowed, and an authentic channel $\text{AUTH}$ where the adversary cannot inject its message but message content is available. Our transformation to leakage simulatability can be considered as a simplified version of CC by replacing the simulator for an adversary who can observe the communication or inject its message in Alice-Bob setting, to an adversary making oracle queries. Thus converting our results to CC should not be of complication. Specifically, a channel for leakage should be “constructable” from $\text{CONF}$ due to the need of ciphertext length and independence from authenticity. We leave it as future work to be done.

7 Conclusion and Future Work

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

We further extend this notion to IND-sf-CCLA to capture the stateful security where out-of-order ciphertexts are queried by the adversary. We prove the robustness of the Encode-then-Encipher (EtE) paradigm which is a mainstream way to construct robust AE, by showing its ability to accommodate multiple failure conditions.

Our study evaluates the disparity between a single-error decryption function and the actual leakage incurred during the decryption. This shows that even though a scheme only
discloses a single error in the decryption function, the actual leakage may still grant the adversary more advantage than decryption. Future work to be done includes introducing a more rigorous notion than error unicity to capture security under multiple failures. In this work, we focus on the leakage with respect to decryption. Security about encryption leakage should also be be formalized to capture the indistinguishability of the intermediate values used in encryption, thus to bolster AE robustness. We present a brief example of converting our game-based notion to simulation-based notion. Future work that can be done includes a more careful study on leakage simulatability thus to facilitate the study on composable security concerning decryption leakage.

References


A Detailed Proofs

A.1 Proof of Lemma 2

Proof. We consider three games \( G_0 \) – \( G_2 \) as in Figure 6 and we use a counter \( i \) as nonce. In \( G_0 \), the encryption is done with the tweakable VIL cipher \( \tilde{E} \) and the oracle first appends \( \tau \) zeros after \( M \) and returns \( \tilde{E}_{K, i, A} (M|0^\tau) \) as output. In \( G_1 \), the oracle samples a tweakable random permutation \( \tilde{\pi} \) and returns \( \tilde{\pi}_{i, A} (M|0^\tau) \) 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

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

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

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

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

\[
\Pr[G_1(A)] - \Pr[G_2(A)] = 0.
\]

Finally, we have that

\[
\text{Adv}^{\text{IND-CPA}}_{\Pi}(A) = \text{Adv}^{\text{PRP}}_{\tilde{E}}(B).
\]

\( \square \)
A.2 Proof of Lemma 3

\begin{center}
\begin{tabular}{ll}
\textbf{procedure} & \textbf{INITIALIZE} \\
1: & $K \leftarrow \mathcal{K}$ \\
2: & $i \leftarrow 0$ \\
3: & $j \leftarrow 0$ \\
4: & \hphantom{1:} $\begin{array}{l}
0: \quad \text{for } (T, \ell) \in T \times \mathbb{N} \text{ do} \\
1: \quad \hat{\pi}_{N, A, \tau} \leftarrow \hat{\pi}_t \\
2: \quad \text{sync} \leftarrow 1 \\
3: \quad \text{win} \leftarrow 0 \\
4: \quad C \leftarrow [] \\
\end{array}$ \\
5: & \hphantom{1:} end for \\
6: & \textbf{return} $C$
\end{tabular}
\end{center}

\begin{center}
\begin{tabular}{ll}
\textbf{procedure} & \textbf{ENC}(A, M, \tau) \\
1: & $i \leftarrow i + 1$ \\
2: & $C \leftarrow E_{K_i(A, \tau)}(M|0^\tau)$ \\
3: & $C' \leftarrow \pi_{i, A, \tau}(M|0^\tau)$ \\
4: & $C[i] \leftarrow (A, C, \tau)$ \\
5: & \textbf{return} $C$
\end{tabular}
\end{center}

\begin{center}
\begin{tabular}{ll}
\textbf{procedure} & \textbf{DEC}(A, C, \tau) \\
1: & $j \leftarrow j + 1$ \\
2: & if $j > i \land (A, C, \tau) \neq C[j]$ then \\
3: & \hphantom{1:} \text{sync} \leftarrow 0 \\
4: & if sync = 1 then \\
5: & \hphantom{1:} \text{return } \dagger \\
6: & \hat{M}' \leftarrow E_{K_i(A, \tau)}^{-1}(C) \\
7: & \hat{M}' \leftarrow \pi_{j, A, \tau}^{-1}(C) \\
8: & \ell \leftarrow |\hat{M}'| \\
9: & if $\hat{M}'[\ell - \tau, \ell] = 0^\tau$ then \\
10: & \hphantom{1:} \text{return } \dagger \\
11: & \hat{M} \leftarrow \hat{M}'[0, \ell - \tau - 1] \\
12: & \text{return } M \\
13: & \text{return } \dagger \\
\end{tabular}
\end{center}

\begin{center}
\begin{tabular}{ll}
\textbf{procedure} & \textbf{FINALIZE} \\
1: & \textbf{return } \text{win}
\end{tabular}
\end{center}

\textbf{Figure 7:} Games $G_0 - G_1$ for proof of Lemma 3. Dot-boxed code is exclusive to $G_1$.

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

$$\text{Adv}^{\text{INT-of-CTXT}}_{\text{R}A}(A) = \Pr[G_0(A) \Rightarrow 1] - \Pr[G_1(A) \Rightarrow 1] + \Pr[G_1(A) \Rightarrow 1].$$

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

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

Now we bound the probability that $A$ wins in $G_1$. We consider two cases of $A$’s queries. In first case, $A$ queries a tuple $(A, C, \tau)$ that is the output of the oracle $\text{ENC}$. In this case, $A$ 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 $A$ wins if the deciphered bitstring ends with $\tau$ zeros, which is of probability $\frac{1}{2^\tau}$. In the other case, $(A, C, \tau)$ has
procedure $B$

1: $i \leftarrow 0$
2: $j \leftarrow 0$
3: sync ← 1
4: win ← 0
5: $C \leftarrow []$
6: Run $A^{Enc, Dec^*}$
7: if win = 1 then
8: return 0
9: return 1

procedure $Enc^*(A, M, \tau)$

1: $i \leftarrow i + 1$
2: $C \leftarrow Enc((i, A, \tau), M||0^\tau)$
3: $C[i] \leftarrow (A, C, \tau)$
4: return $C$

procedure $Dec^*(A, C, \tau)$

1: $j \leftarrow j + 1$
2: if $j > i \lor (A, C, \tau) \neq C[j]$ then
3: sync ← 0
4: if sync = 1 then
5: return j
6: $M' \leftarrow Dec((j, A, \tau), C)$
7: $\ell \leftarrow |M'|$
8: if $M'[^{\ell - \tau, \ell}] = 0^\tau$ then
9: win ← 1
10: $M \leftarrow M'[0, \ell - \tau - 1]$
11: return $M$
12: return $\perp$

Figure 8: $\pm\text{PRP}$ adversary $B$ 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

\[ \Pr[G_1(A) \Rightarrow 1] \leq \frac{q}{2^\tau}. \]

Finally, we have that

\[ \text{Adv}_{\Pi}^{\text{INT-sf-CTXT}}(A) \leq \text{Adv}_{\bar{E}}^{\pm\text{PRP}}(B) + \frac{q}{2^\tau}. \]

A.3 Proof of Lemma 4

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

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

Now we show that we can construct an $\pm\text{PRP}$ adversary $B$ as in Figure 10. For $A$’s encryption and decryption queries, we can construct simulated oracles $Enc^*$ and $Dec^*$ as described in proofs of Lemma 2 and 3. For the leakage query, $B$ forwards the ciphertext queried by $A$ to its oracle $Dec$. Then $B$ 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 $A$ where $M'$ is the deciphered bitstring. We let $B$ return the same bit $b$ that $A$ returns. We then have that

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

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 $A$. 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

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

Finally, we have that

$$\text{Adv}^{\text{IND}-sf-\text{EPL}^2}_{\Pi}(A) = \text{Adv}^{\pm \text{PRF}}_{E}(B).$$

$\square$
Procedure $B$

1: $i \leftarrow 0$
2: $j \leftarrow 0$
3: sync $\leftarrow 1$
4: rep $\leftarrow ()$
5: $C \leftarrow []$
6: $b \leftarrow A^{Enc*, Dec*, Leak*}(.)$
7: return $b$

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

1: $i \leftarrow i + 1$
2: $C \leftarrow \text{Enc}((i, A), M||0')$
3: $C[i] \leftarrow (A, C, \tau)$
4: return $C$

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

1: $j \leftarrow j + 1$
2: if $j > i \lor (A, C, \tau) \neq C[j]$ then
3: sync $\leftarrow 0$
4: if sync $= 1$ then
5: return $\bot$
6: $M' \leftarrow \text{Dec}((j, A), C)$
7: rep $\leftarrow ()$
8: if $M'[[M'] - \tau, |M'|] = 0'$ then
9: $M \leftarrow M'[0, |M'| - \tau - 1]$
10: return $M$
11: return $\bot$

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