Decryption Indistinguishability under Chosen Control Flow

Ganyuan Cao[1]

EPFL ganyuan.cao@epfl.ch

Abstract. Security proofs for cryptographic primitives typically assume operations are executed in the correct sequence; however, insecure implementations or software-level attacks can disrupt control flows, potentially invalidating these guarantees. To address this issue, we introduce a new security notion, IND-CFA, which formalizes decryption security in the presence of adversarially controlled execution flows. Using this notion, we investigate the control flows under which a cryptographic scheme remains secure, providing insights into secure implementation practices. We revisit the Encrypt-then-MAC paradigm, underscoring the crucial role of operation sequencing in ensuring the security of authenticated encryption schemes built using this method. Additionally, we provide a detailed analysis of the Encode-then-Encipher (EtE) paradigm, a widely adopted approach for constructing robust AE schemes, revealing its vulnerability to adversarial control flows that can enable attackers to infer low-entropy values in the presence of multiple failure conditions.

Keywords: Decryption Leakage, Adversarial Control Flow, Security Proof

Contents

1 Introduction

1.1 Background and Motivation

Provable security for cryptographic primitives is generally established on the assumption that operations are executed in correct order, but their implementation can introduce weaknesses. When the execution flow of cryptographic algorithms is poorly implemented or adversarially controlled, it can undermine their security properties. In AEAD schemes, manipulation of execution flow can lead to leaked error messages or temporary values, especially when multiple failures exist. Vaudenay's padding oracle attack [Vau02] is a classic example, with similar vulnerabilities exploited in SSL/TLS [CHVV03, PRS11] and IPsec [DP07, DP10].

Although many security notions [HKR15, ABL+14, BDPS14, BPS15] have been proposed to address security under decryption leakage, there has been limited work on what execution flow may incur such unexpected leakage. Thus, our goal is to propose a notion to formalize the decryption security of AEAD schemes under adversarially-controlled execution flows, and determine how to securely implement such schemes to prevent these vulnerabilities.

1.2 Related Work

To the best of our knowledge, there is limited work that directly formalizes security in the presence of chosen control flow attacks. While symbolic execution and formal methods, such as those facilitated by tools like Cryptol [GKM+04], EasyCrypt [BGHZ11], and CryptoHOL [BLS20], may offer indirect analyses, these approaches do not fully address the intricacies of such attacks, and the use of these tools generally requires a background knowledge of formal verification. Given that control flow attacks result in decryption leakages that can provide adversaries with significant information, we draw connections to established notions that model security in the face of decryption leakage.

In [HKR15], Hoang et al. introduced the notion of *robust* authenticated encryption (RAE), with security defined through the concept of *pseudorandom injection* (PRI), ensuring the indistinguishability of any leaked plaintext. While this approach safeguards the confidentiality of leaked plaintext, it does not thoroughly examine the impact of descriptive errors arising during decryption.

Boldyreva et al., in [BDPS14], examined scenarios where encryption schemes could yield multiple error messages. They proposed the IND-CVA notion, which provides an IND-CPA adversary with access to an additional oracle for verifying queried ciphertexts. Furthermore, they introduced the *error invariance* (INV-ERR) notion, ensuring that no efficient adversary can generate more than one distinct error message. However, this notion can be misleading when it comes how to handle errors securely. Simply outputting a non-specific error message for all types of failures may trivially achieve INV-ERR security, but real-world implementations remain susceptible to side-channel attacks. For instance, the timing-based Lucky13 attack [AFP13] allows adversaries to infer whether the encoding is correct, even when only a generic error message is provided as output. This highlights the importance of analyzing the verification process itself, rather than focusing solely on the error messages.

Andreeva et al. presented the notion of *release-of-unverified-plaintext* (RUP) in [ABL+14], linking it to ciphertext integrity (INT) and plaintext awareness (PA) to address confidentiality of leaked plaintext. In their notion, the decryption algorithm always outputs a bitstring M, bypassing any failure checks. As noted in [ABL+14], plaintext awareness appears overly stringent, as most schemes fail to achieve it, except for the Encode-then-Encipher paradigm [BR00]. Rather than striving for maximal security guarantees, we approach the problem from a different angle, focusing on the degree of security retained under adversarial control flow, providing insights into the secure implementation of encryption schemes.

In [BPS15], Barwell et al. introduced the concept of *subtle* AE, integrating the notion of *error indistinguishability* (ERR-CCA) alongside IND-CPA and INT-CTXT. The ERR-CCA notion incorporates a leakage function that discloses any information leaked during decryption. While this approach allows scheme designers the flexibility to define what constitutes leakage, we argue that such broad generalizations could lead to the oversight of specific vulnerabilities.

Moreover, these notions often overlook a crucial aspect: they typically discuss either leaked plaintext or error messages in isolation, without considering the *relationship* between the two. What remains underexplored is how specific plaintext values lead to successful or failed verification, a factor that enables more potent attacks than simply relying on either the plaintext or error messages alone.

1.3 Our Contribution

In this work, we focus on the security of AEAD decryption, as the decryption process usually involves a more complex execution flows that are more susceptible to manipulation due to the presence of multiple verification steps. We conduct a security analysis to examine the level of protection that can be preserved under adversarially-controlled execution flows. Specifically, our contributions are outlined as follows:

– **New Security Notions**: We analyze the *implementation* of an AEAD scheme's decryption algorithm in a structured manner as a sequence of *operations*, which may leak text values, and *condition predicates*, which may leak descriptive errors. We introduce a novel security notion, IND-CFA, that allows for a *non-adaptive adversary* to influence the control flow during decryption. This enables us to assess whether an AEAD scheme remains secure under such manipulated control flows. Instead of focusing on the strongest possible security, we pose the question:

  Under what control flow is security still guaranteed?

  This approach helps us provide guidance on how to securely implement cryptographic primitives and offers insights into software-level protections, such as maintaining control flow integrity when an AEAD scheme is vulnerable under adversarial control.

  Building on this, we compose our notion with IND-CPA to introduce AE-CFA security, demonstrating that our new notion is sufficient for AE security. We also compare our notion with existing security notions to highlight its relevance and advantages.

– **Revisiting Encode-then-Encrypt-then-MAC (EEM)**: We revisit two key approaches to defining decryption within the Encode-then-Encrypt-then-MAC (EEM) paradigm. The first, introduced in [BN00], adopts a "decryption-first" strategy, as implemented in `PyCryptodome` [Leg]. The second, presented in [BDPS14], follows a "verification-first" approach, which is employed in TLS [MP14]. Through our security notion, we demonstrate that these two execution sequences provide distinct levels of security. We highlight the critical importance of the "verification-first" approach from [BDPS14], which ensures that decryption does not proceed if authentication fails. Furthermore, we stress the necessity of preserving control flow integrity within this scheme to guarantee strong security.

– **Analysis on Encode-then-Encipher**: We provide a concrete analysis of the security of the Encode-then-Encipher (EtE) paradigm [BR00], a widely used approach for constructing robust authenticated encryption. Although EtE is proven secure under strong security models such as *Robust Authenticated Encryption* (RAE) [HKR15], we demonstrate that it has a vulnerability where an adversary can control the order of condition predicates to infer the values of low-entropy inputs to these predicates when multiple condition predicates are involved. However, we also observe that EtE has the capability to validate multiple failure conditions with a single predicate, making it a cost-free patch for better security.

2 Preliminaries

2.1 Notation

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

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

2.2 Game-Based Proof

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

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

$$\Delta_{\mathcal{A}} \binom{\mathrm{O}_L}{\mathrm{O}_R} := \Pr[\mathcal{A}^{\mathrm{O}_L} \Rightarrow 0] - \Pr[\mathcal{A}^{\mathrm{O}_R} \Rightarrow 0]$$

to denote $\mathcal{A}$'s advantage in distinguishing between the oracles O_L and O_R.

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

2.3 Authenticated Encryption with Associated Data (AEAD)

Definition 1 (Nonce-Based AEAD)**.** A nonce-based AEAD scheme is a tuple $\mathsf{AEAD} = (\mathtt{Enc}, \mathtt{Dec})$ specifies two algorithms

$$\mathtt{Enc} : \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathbb{N} \times \mathcal{M} \rightarrow \mathcal{C}$$

and

$$\mathtt{Dec} : \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathbb{N} \times \mathcal{C} \rightarrow \mathcal{M} \cup \{\perp\}$$

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, and $\mathcal{AD} \subseteq \{0,1\}^*$ is the space of associated data. The encryption algorithm $\mathtt{Enc}$ takes a four-tuple $(K, N, A, M) \in \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathbb{N} \times \mathcal{M}$, returns a ciphertext $C \leftarrow \mathsf{AEAD}.\mathtt{Enc}^{N,A}_K(M)$ such that $C \in \mathcal{C}$. The decryption algorithm $\mathtt{Dec}$ takes a four-tuple $(K, N, A, C) \in \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{C}$, and returns a message $M \leftarrow \mathsf{AEAD}.\mathtt{Dec}^{N,A,\tau}_K(C)$ such that $M \in \mathcal{M} \cup \{\perp\}$. If there is no $M \in \mathcal{M}$ such that $C = \mathsf{AEAD}.\mathtt{Enc}^{N,A}_K(M)$, then $\mathsf{AEAD}.\mathtt{Dec}^{N,A}_K(C) = \perp$.

3 Security under Adversarial Control Flow

3.1 Implementation of A Scheme

In contrast to previous works that define leakage through a simulator function, we propose a more concrete approach by directly analyzing the implementation itself, treating the full execution trace as the leakage for a more precise interpretation. Specifically, we model the implementation of a decryption algorithm as a sequence of operations and condition predicates, as outlined in Definition 2.

Definition 2. An AEAD decryption implementation is a Turing machine $\mathtt{Impl}$ with tape alphabet $\Gamma = (\{0,1\}^*)^+ \cup \{\mathsf{true}, \mathsf{false}\} \cup \{\#\}$ where $(\{0,1\}^*)^+$ is space of tuples of bitstrings of arbitrary length and $\#$ is a special end-of-input marker such that:

- **Input**: On input $(K, N, A, C) \in \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{C}$, the tuple is written to the input tape of the machine. A working tape $\mathcal{T}$ is initalized as an empty tape.
- **Transition**: The machine proceeds through a sequence of operations and condition predicates as transition functions that write results to the *working tape* $\mathcal{T}$ of the machine.
  - An operation $\pi_i : \{0,1\}^* \rightarrow \{0,1\}^*$ takes the bitstring from the current state, writes the result M_i to the working tape $\mathcal{T}$, and transitions to the next state.
  - A condition predicate $\omega_i : \{0,1\}^* \rightarrow \{\mathsf{true}, \mathsf{false}\}$ checks the validity of the current bitstring. If ω_i detects a valid input, it writes true to the working tape $\mathcal{T}$ and proceeds to the next state. If ω_i detects an invalid input, it writes false to the working tape, then the machine then halts, and outputs the current state of the working tape $\mathcal{T}$ including all intermediate results up to that point.

  A sequence of transitions consists of any number of operations π_i and condition predicates ω_i in an arbitrary order, executed in sequence, for example, $\mathcal{F} = (\pi_1, \pi_2, \omega_1)$ or $\mathcal{F} = (\pi_1, \pi_2, \omega_1, \pi_3, \omega_2)$. We call such a sequence a *control flow*.

– **Output**: If the machine does not halt due to any condition predicate ω_i, it means that all predicates have been successfully passed, and the current working tape $\mathcal{T}$ is output as the result. Otherwise, if any ω_i fails, the working tape up to that point is output. We call the output working tape $\mathcal{T}$ a *transcript*.

Let $\mathcal{T}$ represent the output of the AEAD decryption implementation $\texttt{Impl}$, executed under the control flow $\mathcal{F}$ with the input (K, N, A, C). We denote this as $\mathcal{T} \leftarrow \texttt{Impl}_{\mathcal{F}}(K, N, A, C)$.

Example 1. We show two examples of the outputs of the machine $\texttt{Impl}$ as follows.

1. Encrypt-then-MAC (EtM) [BN00]: We follow the definition in [BDPS14], where no decryption is performed if the MAC verification fails. This leads to the following sequence of operations $\mathcal{F} = (\omega_1, \pi_1)$ where ω_1 := "$r \leftarrow \mathsf{MAC}.\texttt{Vfy}_{K_m}(C, T)$" for $r \in \{\mathsf{true}, \mathsf{false}\}$, and π_1 := "$M \leftarrow \mathsf{SE}.\texttt{Dec}^N_{K_e}(C)$". The working tuple $\mathcal{T}$ initially starts as an empty tape. If authentication check fails i.e., $r = \mathsf{false}$, then ω_1 write false to $\mathcal{T}$ and $\texttt{Impl}$ halts to return $\mathcal{T} = [\mathsf{false}]$ as the output. Otherwise if $r = \mathsf{true}$, we first write true to $\mathcal{T}$, and then append the decrypted message M to $\mathcal{T}$. The final output is $\mathcal{T} = [\mathsf{true}, M]$.

2. Encode-then-Encrypt-then-MAC (EEM) [BKN04]: In this paradigm, the plaintext is first encoded using, for instance, PKCS padding [Hou09]. In this case, we consider a control flow $\mathcal{F} = (\omega_1, \pi_1, \omega_2, \pi_3)$ where ω_1 := "$r \leftarrow \mathsf{MAC}.\texttt{Vfy}_{K_m}(C, T)$", π_1 := "$M^* \leftarrow \mathsf{SE}.\texttt{Dec}^N_{K_e}(C)$", ω_2 checks whether M^* follows the correct encoding, and π_2 decodes M^*. Then for a valid ciphertext, the transcript $\mathcal{T} = [\mathsf{true}, M^*, \mathsf{true}, M]$. For a ciphertext with a valid tag but decrypted to a bitstring with invalid encoding, the transcript $\mathcal{T} = [\mathsf{true}, M', \mathsf{false}]$. For a ciphertext with invalid tag, the transcript $\mathcal{T} = [\mathsf{false}]$.

3.2 IND-CFA Security

We propose a new notion, ***Ind****istinguishability (of Decryption) under* ***C****hosen (Control)* ***F****low* ***A****ttack*, denoted by IND-CFA, for AEAD schemes, as shown in Figure 1. In this game, we allow the adversary to *non-adaptively* select a control flow, which is then executed by $\texttt{Impl}$ in the oracle DEC, and the complete execution trace (transcript) is revealed to the adversary.

Definition 3 (IND-CFA)**.** An AEAD scheme is ε-IND-CFA secure *with respect to a adversarial-chosen control flow* $\mathcal{F}$ if for any PPT adversary $\mathcal{A}$, it has $\mathbf{Adv}^{\text{IND-CFA}}_{\mathsf{AEAD},\mathcal{F}}(\mathcal{A}) \leq \varepsilon$ where

$$\mathbf{Adv}^{\text{IND-CFA}}_{\mathsf{AEAD},\mathcal{F}}(\mathcal{A}) := \Pr[\mathrm{G}^{\text{IND-CFA-0}}_{\mathsf{AEAD}}(\mathcal{A})] - \Pr[\mathrm{G}^{\text{IND-CFA-1}}_{\mathsf{AEAD}}(\mathcal{A})]$$

OBSERVATION ON THE NOTION. We adopt the *real-or-ideal* oracle model for DEC. The adversary's objective is to distinguish between the real transcript $\mathcal{T}_R \leftarrow \texttt{Impl}_{\mathcal{F}}(K, N, A, C)$, produced by the AEAD decryption implementation under an adversary-controlled flow $\mathcal{F}$, and an ideal transcript $\mathcal{T}_I$ corresponding to the same control flow $\mathcal{F}$.

Let $\mathcal{F}^*$ represent the original control flow defined by the AEAD scheme, and let $\mathcal{F}$ be the control flow chosen by the adversary. We assume that for all $\delta \in \mathcal{F}$, it holds that $\delta \in \mathcal{F}^*$. This implies that the adversary can modify the order of operations and condition predicates in the control flow, or omit an operation or condition predicate, but cannot introduce new operations or predicates. Additionally, we assume that the adversary does not query a flow $\mathcal{F} = (\ldots, \omega_i, \ldots, \pi_i, \ldots, \pi_j, \ldots)$ where ω_i requires the output of π_i or π_j as input, or π_i requires the output of π_j as input.

To simplify the handling of deterministic operations, we assumed deterministic steps, such as parsing or XORing, are merged into the randomness-inducing operations. Essentially, these steps can be treated as sub-steps of a broader operation that eventually

$G^{\text{IND-CFA-0}}_{\mathsf{AEAD}}$ $\boxed{G^{\text{IND-CFA-1}}_{\mathsf{AEAD}}}$

```
procedure Init
1 : $K \leftarrow\$ \mathcal{K}$
2 : $\mathcal{Q}_e \leftarrow \emptyset$
3 : $\mathcal{F} \leftarrow \mathcal{A}(\cdot)$

procedure Enc$(N, A, M)$
1 : $C \leftarrow \mathsf{AEAD}.\mathtt{Enc}_K^{N,A}(M)$
2 : $\mathcal{Q}_e \leftarrow \mathcal{Q}_e \cup \{(N, A, C)\}$
3 : return $C$

procedure Dec$(N, A, C)$
1 : if $(N, A, C) \in \mathcal{Q}_e$ then
2 :     return $\lightning$
3 : $\mathcal{T} \leftarrow \mathtt{Impl}_{\mathcal{F}}(K, N, A, C)$
4 : ┌─────────────────────────────────────────────┐
    │ 1 : $(\pi_1, \ldots, \pi_t, \omega_1, \ldots) \leftarrow \mathcal{F}$ │
    │ 2 : for $i = 1 \ldots t$ do                  │
    │ 3 :     $m_i \leftarrow \mathcal{S}_{\pi_i}(\cdot)$ │
    │ 4 : $\mathcal{T} \leftarrow [m_1, \ldots, m_t, \mathsf{false}]$ │
    └─────────────────────────────────────────────┘
5 : $\mathcal{Q}_d \leftarrow \mathcal{Q}_d \cup (N, A, C)$
6 : return $\mathcal{T}$
```

Figure 1: IND-CFA games for a nonce-based AEAD scheme. The dot-boxed parts are exclusive to $G^{\text{IND-CFA-1}}_{\mathsf{AEAD}}$.

produces random-looking output. For simplicity, we prohibit the adversary from forwarding a query from Enc to Dec since it yields trivial distinguishing.

Given an adversary-chosen control flow $\mathcal{F} = (\pi_1, \ldots, \pi_t, \omega_1, \ldots)$, we define the ideal transcript corresponding to this flow as $\mathcal{T}_I = [m_1, \ldots, m_t, \mathsf{false}]$. We draw on concepts from the *simulation-based proof* by letting each m_i to be generated by a *simulator* $\mathcal{S}_{\pi_i}$ for the operation π_i. For instance, $\mathcal{S}_{\pi_i}$ may take input as a message length $|m|$ and output a bitstring $c \leftarrow\$ \{0,1\}^{\psi(|m|)}$ for some function ψ.

In addition to ensuring the indistinguishability of the transcript's values, we require that the output of the first condition predicate ω_1 is false. This leads to the following security requirements, even under adversary-controlled execution flows:

1. The information disclosed to the actual adversary $\mathcal{A}$ does not exceed what is disclosed to the simulator $\mathcal{S}$.

2. The first condition predicate is not satisfied, and execution terminates upon failure.

Role of Intermediate Values. Importantly, our notion also reveals intermediate values, such as a reconstructed IV, alongside the final plaintext, and require the indistinguishability of these values as well. This requirement becomes crucial in attacks like the one targeting RIV [AFL+16], which is proven SAE-secure [BPS15] under the assumption that the only final plaintext M is considered as leakage.

Specifically, RIV constructs the IV as $IV \leftarrow F_K(N, H, C) \oplus T$ for a PRF F, where $M \leftarrow \mathsf{SE}.\mathtt{Dec}_{K_e}^{IV}(C)$ is the plaintext. Given two distinct pairs (C, T) and (C', T'), the adversary can control when they produce the same IV. In CTR mode, this allows the adversary to distinguish based on $C_1 \oplus C'_1 = M_1 \oplus M'_1$, where C_1 and M_1 represent the first block of ciphertext and plaintext, respectively.

Notably, this attack would not be feasible if the IV remained hidden, as the output of the PRF F would appear random and the adversary has negligible advantage in "guessing" when IV matches.

Composition for AE Security. It is straightforward to see how IND-CFA security can be composed with IND-CPA security to align with established security notions for authenticated encryption (AE), such as the IND-CCA3 notion introduced by Shrimpton [Shr04], as well as the combination of IND-CPA and INT-CTXT [BN00].

In Definition 4, we introduce a new security notion called ***AE** Security under **C**hosen (Control) **F**low **A**ttack*, denoted by AE-CFA. For notation simplicity, we abuse the notation slightly by using $\textsc{Enc}_K$ to represent the oracle $\textsc{Enc}$ depicted in Figure 1 but the adversary is not allowed to repeat a query with the same (N, A, M) tuple, and $\$^{\textsc{Enc}}$ denotes the process of sampling a random bitstring $C \leftarrow\!\!\$\ \{0,1\}^{\psi(|M|)}$ where ψ is a function dependent on the plaintext length $|M|$. We use $\texttt{Impl}^R_{\mathcal{F}}$ to refer to the execution that generates the real transcript, and $\texttt{Impl}^I_{\mathcal{F}}$ for the execution that generates the ideal transcript.

Definition 4 (AE-CFA Advantage)**.**

$$\mathbf{Adv}^{\text{AE-CFA}}_{\mathsf{AEAD},\mathcal{F}}(\mathcal{A}) := \Delta_{\mathcal{A}} \begin{pmatrix} \textsc{Enc}_K, \texttt{Impl}^R_{\mathcal{F}} \\ \$^{\textsc{Enc}}, \ \texttt{Impl}^I_{\mathcal{F}} \end{pmatrix}$$

for $K \leftarrow\!\!\$\ \mathcal{K}$ and under a control flow $\mathcal{F}$.

Theorem 1. AE-CFA *implies* IND-CPA *and* IND-CFA. *Specifically, for any* AE-CFA *adversary* $\mathcal{A}_{ae}$, *there is an* IND-CPA *adversary* $\mathcal{A}_{cpa}$ *and an* IND-CFA *adversary* $\mathcal{A}_{cfa}$ *such that*

$$\mathbf{Adv}^{\text{AE-CFA}}_{\mathsf{AEAD},\mathcal{F}}(\mathcal{A}_{ae}) \leq \mathbf{Adv}^{\text{IND-CPA}}_{\mathsf{AEAD}}(\mathcal{A}_{cpa}) + \mathbf{Adv}^{\text{IND-CFA}}_{\mathsf{AEAD},\mathcal{F}}(\mathcal{A}_{cfa})$$

Proof. We write the advantage as

$$\begin{aligned}
\mathbf{Adv}^{\text{AE-CFA}}_{\mathsf{AEAD},\mathcal{F}}(\mathcal{A}_{ae}) &= \Delta_{\mathcal{A}_{ae}} \begin{pmatrix} \textsc{Enc}_K, \texttt{Impl}^R_{\mathcal{F}} \\ \$^{\textsc{Enc}}, \ \texttt{Impl}^I_{\mathcal{F}} \end{pmatrix} \\
&= \Delta_{\mathcal{A}_1} \begin{pmatrix} \textsc{Enc}_K, \texttt{Impl}^R_{\mathcal{F}} \\ \textsc{Enc}_K, \texttt{Impl}^I_{\mathcal{F}} \end{pmatrix} + \Delta_{\mathcal{A}_2} \begin{pmatrix} \textsc{Enc}_K, \texttt{Impl}^I_{\mathcal{F}} \\ \$^{\textsc{Enc}}, \ \texttt{Impl}^I_{\mathcal{F}} \end{pmatrix} \\
&= \mathbf{Adv}^{\text{IND-CFA}}_{\mathsf{AEAD},\mathcal{F}}(\mathcal{A}_{cfa}) + \Delta_{\mathcal{A}_3} \begin{pmatrix} \textsc{Enc}_K, \texttt{Impl}^I_{\mathcal{F}} \\ \$^{\textsc{Enc}}, \ \texttt{Impl}^I_{\mathcal{F}} \end{pmatrix}
\end{aligned}$$

Note that for any tuple (N, A, C), the oracle $\texttt{Impl}^I_{\mathcal{F}}$ always replies with a transcript of random bitstrings and false. This can be perfectly simulated by an IND-CPA adversary $\mathcal{A}_{cpa}$. Thus we have that

$$\Delta_{\mathcal{A}_3} \begin{pmatrix} \textsc{Enc}_K, \texttt{Impl}^I_{\mathcal{F}} \\ \$^{\textsc{Enc}}, \ \texttt{Impl}^I_{\mathcal{F}} \end{pmatrix} \leq \mathbf{Adv}^{\text{IND-CPA}}_{\mathsf{AEAD}}(\mathcal{A}_{cpa})$$

which concludes the proof. □

To show that our notion is sufficient to capture AE security, we show that IND-CFA implies INT-CTXT in Proposition 1, which leads to the result that AE-CFA implies AE security in Corollary 1.

Proposition 1. IND-CFA *implies* INT-CTXT.

Proof. We fix $\mathcal{F}$ to be the original control flow. Let $\mathcal{A}$ be an INT-CTXT adversary. Let (N, A, C) be $\mathcal{A}$'s query such that $\mathsf{AEAD}.\mathsf{Dec}^{N,A}_K(C) = M \neq \perp$. In this case, an IND-CFA adversary $\mathcal{B}$ queries such a tuple, yielding a transcript $\mathcal{T}$ ending with M and no false in $\mathcal{T}$ be distinguished from the ideal tuple. Thus we have that $\mathbf{Adv}^{\text{INT-CTXT}}_{\mathsf{AEAD}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{IND-CFA}}_{\mathsf{AEAD},\mathcal{F}}(\mathcal{B})$. □

Corollary 1. AE-CFA *implies* AE.

Proof. The proof follows by combining Theorem 1 and Proposition 1. □

3.3 Comparison with Existing Notions

We briefly compare our notion with established ones to highlight its advantages. To our knowledge, there has been limited work addressing the security of symmetric cryptographic primitives under control flow attacks. However, since our notion captures a comprehensive security model that considers both descriptive errors and candidate plaintexts that may be leaked when decryption fails, we focus our comparison with notions that address these aspects.

3.3.1 Error Invariance (INV-ERR).

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 $\perp_i \in \mathcal{S}_\perp$.

Definition 5 (INV-ERR Advantage)**.**

$$\mathbf{Adv}^{\text{INV-ERR}}_{\mathsf{AEAD},\perp_i}(\mathcal{A}) := \Pr[\mathcal{A}^{\textsc{Enc},\textsc{Dec}} \Rightarrow \perp_j \in \mathcal{S}_\perp \mid \perp_j \neq \perp_i]$$

An AEAD scheme is considered INV-ERR secure if there exists a $\perp_i \in \mathcal{S}_\perp$ such that $\mathbf{Adv}^{\text{INV-ERR}}_{\mathsf{AEAD},\perp_i}(\mathcal{A}) \leq \mathsf{negl}$.

However, the composition of $\mathcal{S}_\perp$ is not clearly defined, as it may refer to either the output of the decryption algorithm or the result of each condition predicate. As noted in [BDPS14], a scheme is trivially INV-ERR secure if $|\mathcal{S}_\perp| = 1$. However, this can lead to the misconception that simply outputting a generic error message for all types of failures is sufficient for security. In reality, side-channel attacks, such as those described in [AFP13], can still infer whether a specific error has been triggered, rendering this approach ineffective.

INV-ERR can be seen as a variant of our notion where an adversary queries with the original control flow $\mathcal{F}$ and any value produced by the execution of the decryption is not disclosed to the adversary. In Proposition 2, we show that IND-CFA is strictly stronger than INV-ERR.

Proposition 2. IND-CFA *implies* INV-ERR *under the original control* $\mathcal{F}$*.*

Proof. (IND-CFA $\rightarrow$ INV-ERR) Assuming the error space $|\mathcal{S}_\perp| \geq 2$, let $\mathcal{A}$ be an INV-ERR adversary, and fix $\perp_1$ as the error corresponding to the first condition predicate ω_1 that is to be distinguished. We can now construct an IND-CFA adversary $\mathcal{B}$ as follows: for each decryption query (N, A, C) made by $\mathcal{A}$, we let $\mathcal{B}$ forwards it to its oracle $\textsc{Dec}$ with the tuple (N, A, C). Notably, $\mathcal{A}$ eventually queries a tuple (N, A, C) that results in an error other than $\perp_1$. In this case, the real transcript observed by $\mathcal{B}$ is $\mathcal{T}_R = [m_1, \ldots, m_t, \mathsf{true}, \ldots]$, which must be distinguished from the ideal transcript $\mathcal{T}_I = [m_1, \ldots, m_t, \mathsf{false}]$. Thus we have that $\mathbf{Adv}^{\text{INV-ERR}}_{\mathsf{AEAD},\perp_1}(\mathcal{B}) \leq \mathbf{Adv}^{\text{IND-CFA}}_{\mathsf{AEAD},\mathcal{F}}(\mathcal{A})$.

(INV-ERR $\not\rightarrow$ IND-CFA) Encode-then-Encrypt-then-MAC with “decryption first” configuration as a counterexample. Details are discussed in Section 4. □

3.3.2 Plaintext Awareness (PA) and RUP.

Plaintext Awareness. In [ABL+14], Andreeva et al. introduced *plaintext awareness* (for symmetric encryption) 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 simplicity and to better align with our notion, we consider the indistinguishability of plaintext as a random bitstring. We define it as in Definition 6.

Definition 6 (PA2 Advantage)**.** Let $\widetilde{\mathcal{D}}$ be the decryption function without any check for failure such that $\widetilde{\mathcal{D}}$ always output a plaintext, then

$$\mathbf{Adv}^{\mathrm{PA2}}_{\mathsf{AEAD}}(\mathcal{A}) := \Delta_{\mathcal{A}}\left(\textsc{Enc}_K, \widetilde{\textsc{Dec}}_K; \textsc{Enc}_K, \$^{\widetilde{\textsc{Dec}}}\right)$$

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

As also highlighted by Andreeva et al. in [ABL+14], the PA2 notion is particularly strong. Many schemes, such as OCB [RBBK01], GCM [MV04], and SIV [RS06], generally fail to meet this security, with Encode-then-Encipher (EtE) [BR00] being an exception. This observation underscores our approach of not pursuing the strongest level of security, but rather examining how a scheme performs under various control flow scenarios to determine its robustness.

Note that PA2 can be viewed as a variant of IND-CFA in which the adversary queries using a modified control flow, $\mathcal{F} = (\pi_1, \ldots, \pi_i)$. This means that all operations from the original control flow $\mathcal{F}^*$ are executed in the original order, with all condition predicates are omitted. In Proposition 3, we show that IND-CFA implies PA2 under the control flow $\mathcal{F}$.

Proposition 3. IND-CFA *implies* PA2 *under the control flow* $\mathcal{F} = (\pi_1, \ldots, \pi_i)$.

Proof. We fix $\mathcal{F} = (\pi_1, \ldots, \pi_i)$ as the control flow to be executed in the oracle Dec in the game IND-CFA. We consider an PA2 adversary $\mathcal{A}$, and we show that we can construct an IND-CFA adversary $\mathcal{B}$. For each query (N, A, C) made by $\mathcal{A}$, we let $\mathcal{B}$ forward such a query. Finally, $\mathcal{A}$'s query will finally yield an output M that is distinguishable from a random bitstring. Similarly, $\mathcal{B}$ can observe $m_t = M$ in $\mathcal{T}_R = [m_1, \ldots, m_t]$ to distinguish from the ideal transcript. Thus we have that $\mathbf{Adv}^{\mathrm{PA2}}_{\mathsf{AEAD}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{IND\text{-}CFA}}_{\mathsf{AEAD},\mathcal{F}}(\mathcal{A})$. □

INT-RUP. In [ABL+14], Andreeva et al. also discussed the INT-RUP notion, where the adversary goal is to forge a valid ciphertext given an encryption oracle, a decryption oracle $\widetilde{\textsc{Dec}}$ that always decrypt without verification, and a verification oracle Vfy that verifies the validity of the ciphertext.

Definition 7 (INT-RUP Advantage)**.**

$$\mathbf{Adv}^{\mathrm{INT\text{-}RUP}}_{\mathsf{AEAD}}(\mathcal{A}) := \Pr[\mathcal{A}^{\textsc{Enc},\widetilde{\textsc{Dec}},\textsc{Vfy}} \Rightarrow (N, A, C) \mid \textsc{Vfy}_K(N, A, C) = 1]$$

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

It is easy to see that our notion implies INT-RUP under the original control flow since the full execution trace has been provided to the adversary.

Proposition 4. IND-CFA *implies* INT-RUP *under the original control flow* $\mathcal{F}$.

Proof. We consider an PA2 adversary $\mathcal{A}$, and we show that we can construct an IND-CFA adversary $\mathcal{B}$. For each query (N, A, C) made by $\mathcal{A}$ to $\widetilde{\textsc{Dec}}$, we let $\mathcal{B}$ forward such a query to its oracle Dec. We let $\mathcal{T} = [m_1, \ldots, m_t, r_1, \ldots]$ be the transcript obtained by $\mathcal{B}$. Then we let $\mathcal{B}$ return m_t to $\mathcal{A}$. For each query made by $\mathcal{A}$ to Vfy, we let $\mathcal{B}$ checks whether false is in the returned transcript. If so, $\mathcal{B}$ returns false to $\mathcal{A}$. Otherwise, $\mathcal{B}$ returns true to $\mathcal{A}$.

Finally, $\mathcal{A}$ will query a tuple (N, A, C) such that $\textsc{Vfy}_K(N, A, C) = \mathsf{true}$. By forwarding this tuple, $\mathcal{B}$ will observe a transcript without false to distinguish from the ideal transcript. Thus we have that $\mathbf{Adv}^{\mathrm{INT\text{-}RUP}}_{\mathsf{AEAD}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{IND\text{-}CFA}}_{\mathsf{AEAD},\mathcal{F}}(\mathcal{A})$. □

4 Revisiting Encode-then-Encrypt-then-MAC

We revisit the Encode-then-Encrypt-then-MAC (EEM) construction, which is generally considered authenticated encryption (AE) secure, to demonstrate the significance of operation ordering. Specifically, there are two distinct approaches to defining the decryption process in EEM. The first, as outlined in [BN00], performs decryption before verifying the tag (Figure 2, Left), while the second, from [BDPS14], mandates that decryption should not occur if the authentication check fails (Figure 2, Right).

```
$\mathsf{EEM}_1.\mathtt{Dec}^N_{K_e,K_m}(C)$
1 : $C'||T \leftarrow C$
2 : $M' \leftarrow \mathsf{SE.Dec}^N_{K_e}(C)$
3 : if $\mathsf{false} \leftarrow \mathsf{MAC.Vfy}_{K_m}(C,T)$ then
4 :   return $\perp$
5 : if $\mathsf{false} \leftarrow \mathsf{Encode.Check}(M')$ then
6 :   return $\perp$
7 : $M \leftarrow \mathsf{Encode.Decode}(M')$
8 : return $M$
```

```
$\mathsf{EEM}_2.\mathtt{Dec}^N_{K_e,K_m}(C)$
1 : $C'||T \leftarrow C$
2 : if $\mathsf{false} \leftarrow \mathsf{MAC.Vfy}_{K_m}(C,T)$ then
3 :   return $\perp$
4 : $M' \leftarrow \mathsf{SE.Dec}^N_{K_e}(C)$
5 : if $\mathsf{false} \leftarrow \mathsf{Encode.Check}(M')$ then
6 :   return $\perp$
7 : $M \leftarrow \mathsf{Encode.Decode}(M')$
8 : return $M$
```

Figure 2: Two methods of executing EEM: the left side follows the definition in [BN00], while the right side follows [BDPS14]. The differences in control flows are highlighted.

Note that both implementations achieve error invariance (INV-ERR) when the MAC scheme is SUF-CMA-secure, as established in [BDPS14]. Let $\mathcal{F}_1$ and $\mathcal{F}_2$ represent the control flows of EEM_1 and EEM_2, respectively. In the case of $\mathcal{F}_1$, by simply modifying the tag, the real transcript becomes $\mathcal{T}_1^R = [M', \mathsf{false}]$. Here, the adversary can distinguish this transcript from the ideal transcript $\mathcal{T}_1^I$ by M'. This distinction supports our proof of Proposition 2, showing that the IND-CFA notion is strictly stronger than INV-ERR.

Also, observe that, even if M' is not leaked, an adversary can still manipulate the control flow, forcing the encoding check to be executed first, since M' has already been available before both of condition predicates. This would result in a transcript $\mathcal{T}_1^R = [\mathsf{true}]$.

Proposition 5. EEM_1 *is not* IND-CFA *secure.*

On the other hand, in the case of $\mathcal{F}_2$, with an invalid tag, the real transcript becomes $\mathcal{T}_2^R = [\mathsf{false}]$, which is indistinguishable from the ideal transcript $\mathcal{T}_2^I = [\mathsf{false}]$, provided the MAC scheme is SUF-CMA-secure, as discussed in [BDPS14]. Since the encoding check depends on M', it cannot be meaningfully executed when M' is unavailable and the availability of M' depends on whether the verification of tag is successful.

Proposition 6. *For any* IND-CFA *adversary* $\mathcal{A}_{cfa}$*, there is an* SUF-CMA *adversary such that*

$$\mathbf{Adv}^{\text{IND-CFA}}_{\mathsf{EEM}_2,\mathcal{F}_2}(\mathcal{A}_{cfa}) \leq \mathbf{Adv}^{\text{SUF-CMA}}_{\mathsf{MAC}}(\mathcal{A}_{suf})$$

Remark 1 (Comment on Control Flow Integrity). From the discussion above, it is evident that the order of execution affects the security guarantees of cryptographic schemes. The intuition behind our notion is to assess whether a scheme remains secure when its control flow is either incorrectly implemented or manipulated by an adversary, and to offer guidance on secure implementation practices. If the scheme fails to maintain security under such conditions, software developers should implement additional techniques to safeguard the control flow's integrity or address the risks of leaked values from a software security standpoint.

5 Security of Encode-then-Encipher

The Encode-then-Encipher (EtE) paradigm, introduced by Bellare and Rogaway in [BR00], is widely recognized as a leading method for constructing authenticated encryption due to its robustness against decryption leakage and nonce misuse. This approach has been adopted in various studies [HKR15, ST13] to develop *robust authenticated encryption.*

We provide a proof of the security of EtE when applied with a tweakable VIL cipher, examining its control flow to reveal a potential vulnerability where the adversary can change the order of condition predicates to infer low-entropy values under multiple condition predicates. Additionally, we demonstrate that EtE allows for a single condition predicate to address multiple failure conditions, thereby mitigating the security issues arising from the use of multiple condition predicates.

5.1 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 an EtE construction $\mathsf{EtE} = (\mathtt{Enc}, \mathtt{Dec})$ as follow Let $C = \mathsf{EtE}.\mathtt{Enc}_K^{N,A,\tau}(M) = \widetilde{E}_{K;N,A,\tau}(M \| 0^\tau)$ and return C as ciphertext. Let $M^* = \widetilde{E}^{-1}_{K;N,A,\tau}(C)$. Then if M^* ends with τ zeros, $\mathsf{EtE}.\mathtt{Dec}_K^{N,A,\tau}(C)$ returns M^* excluding ending τ zeros as plaintext. Otherwise, $\mathsf{EtE}.\mathtt{Dec}_K^{N,A,\tau}(C)$ 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)**.** *Let $\mathcal{A}$ be an adversary that queries a distinct tweak T each time. Then, for any $\ell \geq 0$, we have*

$$\Pr\left[\mathcal{A}^{\widetilde{\Pi}_T(\cdot)} \Rightarrow 0\right] - \Pr\left[\mathcal{A}^{\$(\cdot)} \Rightarrow 0\right] = 0,$$

where $\widetilde{\Pi}_T$ denotes an oracle implementing a tweakable random permutation $\widetilde{\Pi}_T : \{0,1\}^\ell \to \{0,1\}^\ell$, and \$ denotes an oracle that samples a bitstring uniformly at random from $\{0,1\}^\ell$.

Proof. We first consider $\$(\cdot)$ i.e., the oracle that outputs a uniformly random bitstring in $\{0,1\}^\ell$ upon each query. The probability that any specific bitstring $L \in \{0,1\}^\ell$ is output by this oracle upon a query is $\frac{1}{2^\ell}$.

For the tweakable random permutation oracle $\widetilde{\Pi}_T$, given that each tweak T used by $\mathcal{A}$ is unique, a fresh permutation is sampled uniformly at random for each distinct tweak. This results in the probability that any input $M \in \{0,1\}^\ell$ is mapped to any specific output $L \in \{0,1\}^\ell$ at each query being $\frac{1}{2^\ell}$ as well.

Since both oracles, under the conditions specified, produce outputs with identical distributions, the adversary $\mathcal{A}$ has 0 advantage in distinguishing between the tweakable random permutation oracle and the oracle outputting random bits. □

5.2 Proof of Security

We show that EtE with a tweakable is AE-CFA secure, highlighting it as a promising way to construct robust AE. 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].

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

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

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

Proof. We consider three games $\mathrm{G}_0 - \mathrm{G}_2$ 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

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

We first bound $\Pr[\mathrm{G}_0(\mathcal{A})] - \Pr[\mathrm{G}_1(\mathcal{A})]$ by a $\widetilde{\text{PRP}}$ adversary $\mathcal{A}_{tprp}$.

Following Lemma 1, we know that the adversary has 0 advantage in distinguishing between G_1 and G_2 if each queried tweak is distinct. Now we assume the adversary fix (N, A, τ) but query with different M. This 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 $\frac{q^2-q}{2^\ell} \leq \frac{q^2}{2^\ell}$. □

Following the definition in Section 5.1, we can then consider the implementation of an EtE with a tweakable VIL cipher $\widetilde{E}$ as the following components: $\pi_1 :=$ "$M^* \leftarrow \widetilde{E}^{-1}_{K;N,A,\tau}(C)$" and $\omega_1 :=$ "$M^*[\ell-\tau-1\ldots\ell] = 0^\tau$" for $\ell = |M^*|$. Thus the only meaningful control flow is $\mathcal{F} = (\pi_1, \omega_1)$ since ω_1 depends on π_1.

We define the simulator $\mathcal{S}_{\pi_1}$ to operate as follows: given an input (N, A, C, τ), it first checks for an existing entry in its record. If a corresponding value m is found, it outputs that value; otherwise, the simulator $\mathcal{S}_{\pi_1}$ then sample a bitstring $m \leftarrow\!\!\$ \{0,1\}^{|C|}$ uniformly at random based on the input length $|C|$ and outputs m.

In Lemma 3, we demonstrate that EtE is secure under the IND-CFA notion.

Lemma 3. *For any* IND-CFA *adversary* $\mathcal{A}$ *against the* EtE *making* q *decryption queries, there is a* $\widetilde{\text{PRP}}$ *adversary* $\mathcal{A}_{tprp}$ *against the tweakable* VIL *cipher* $\widetilde{E}$ *such that*

$$\mathbf{Adv}_{\mathsf{EtE},\mathcal{F}}^{\text{IND-CFA}}(\mathcal{A}) \leq \mathbf{Adv}_{\widetilde{E}}^{\widetilde{\text{PRP}}}(\mathcal{A}_{tprp}) + \frac{q^2}{2^\ell} + \frac{q+1}{2^\tau}$$

where $\ell = \arg\min_{(N,A,C,\tau)\in\mathcal{Q}}\{|M^*| : M^* = \widetilde{E}^{-1}_{K;N,A,\tau}(C)\}$ *and* $\mathcal{Q}$ *is the set of decryption queries made by* $\mathcal{A}$, *under the control* $\mathcal{F} = (\pi_1, \omega_1)$ *where* $\pi_1 :=$ *"*$M^* \leftarrow \widetilde{E}^{-1}_{K;N,A,\tau}(C)$*" and* $\omega_1 :=$ *"*$M^*[\ell-\tau-1\ldots\ell] = 0^\tau$*".*

Proof. We consider three games $\mathrm{G}_0 - \mathrm{G}_2$. In G_0, we let ENC answer the query with $\widetilde{E}$ and let DEC answer with the transcript $\mathcal{T} = [M^*, r]$ where M^* and $r \in \{\mathsf{true}, \mathsf{false}\}$ are written by the π_1 and v_1 respectively. In G_1, we replace $\widetilde{E}$ and $\widetilde{E}^{-1}$ in ENC and DEC with a tweakable random permutation $\widetilde{\Pi}$ and $\widetilde{\Pi}^{-1}$ respectively. In G_2, we return the ideal transcript $\mathcal{T} = [M^*, \mathsf{false}]$ where $M^* \leftarrow\!\!\$ \{0,1\}^{|C|}$. Thus we have that

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

Similarly, we can bound $\Pr[\mathrm{G}_0(\mathcal{A})] - \Pr[\mathrm{G}_1(\mathcal{A})]$ by a $\widetilde{\text{PRP}}$ adversary $\mathcal{A}_{tprp}$.

We first assume the adversary does not fix (N, A, τ), which means each tweak queried to DEC is distinct. By Lemma 1, the adversary has 0 advantage in distinguish from M^*. Thus the transcript $\mathcal{T}_1$ generated in G_1 and the transcript $\mathcal{T}_2$ generated in G_2 only differs if $\mathcal{T}_1$ contains a true, meaning M^* ends with τ zeros. Since we have a fresh random permutation at each query, this happens with probability at most $\frac{1}{2^\tau}$.

Now we assume the adversary fix (N, A, τ) for tweak but query with different C. Then the behavior of G_1 and G_2 happens if one of the following two events happens. In the first event, the transcript from G_1 yields $\mathcal{T}_2 = [M^*, \mathsf{true}]$, which happens with probability at most $\frac{q}{2^\tau}$ given that G_1 implements a fixed random permutation this time. In the second case, the M^* sampled in G_2 repeats, which happens with probability $\frac{q^2}{2^\ell}$. By Union Bound, we obtain the bound above. □

Theorem 2. *For any* AE-CFA *adversary $\mathcal{A}$ against the* EtE *making q_e encryption queries and q_d decryption queries, there is a* $\widetilde{\pm\mathrm{PRP}}$ *adversary $\mathcal{A}_{stprp}$ against the tweakable* VIL *cipher $\widetilde{E}$ such that*

$$\mathbf{Adv}^{\text{AE-CFA}}_{\mathsf{EtE}}(\mathcal{A}) \leq 2 \cdot \mathbf{Adv}^{\widetilde{\pm\mathrm{PRP}}}_{\widetilde{E}}(\mathcal{A}_{stprp}) + \frac{q_e}{2^{\ell_1}} + \frac{q_d}{2^{\ell_2}} + \frac{q_d+1}{2^\tau}$$

where $\ell_1 = \arg\min_{(N,A,M,\tau)\in\mathcal{Q}}\{|C| : C = \mathsf{EtE.Enc}^{N,A,\tau}_K(M)\}$ *and* $\ell_2 = \arg\min_{(N,A,C,\tau)\in\mathcal{Q}}\{|M^*| : M^* = \widetilde{E}^{-1}_{K;N,A,\tau}(C)\}$.

Proof. The proof follows by combining Theorem 1 with Lemmas 2 and 3. □

Verification for 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 8.

Definition 8 (δ-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 δ-dense with respect to the predicate v if

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

In this case, we need to consider two condition predicates where $\omega_1 :=$ "$M^*[\ell - \tau - 1 \ldots \ell] = 0^\tau$" and $\omega_2 :=$ "$v_\delta(M^*[0 \ldots \ell - \tau - 1])$". for $\ell = |M^*|$. Then we can obtain two control flows by manipulating the order of ω_1 and ω_2. We let $\mathcal{F}_1 = (\pi_1, \omega_1, \omega_2)$ and $\mathcal{F}_2 = (\pi_1, \omega_2, \omega_1)$. Based on $\mathcal{F}_1$ and $\mathcal{F}_2$, we obtain two different IND-CFA advantages.

Corollary 2. *Assuming the message space $\mathcal{M}$ is δ-dense, then for any* IND-CFA *adversary $\mathcal{A}$ against* EtE*, there is a* $\widetilde{\pm\mathrm{PRP}}$ *adversary $\mathcal{A}_{stprp}$ against the tweakable* VIL *cipher $\widetilde{E}$ such that*

$$\mathbf{Adv}^{\text{IND-CFA}}_{\mathsf{EtE},\mathcal{F}_1}(\mathcal{A}) \leq \mathbf{Adv}^{\widetilde{\pm\mathrm{PRP}}}_{\widetilde{E}}(\mathcal{A}_{stprp}) + \frac{q^2}{2^\ell} + \frac{q+1}{2^\tau}$$

for $\mathcal{F}_1 = (\pi_1, \omega_1, \omega_2)$, and

$$\mathbf{Adv}^{\text{IND-CFA}}_{\mathsf{EtE},\mathcal{F}_2}(\mathcal{A}) \leq \mathbf{Adv}^{\widetilde{\pm\mathrm{PRP}}}_{\widetilde{E}}(\mathcal{A}_{stprp}) + \frac{q^2}{2^\ell} + (q+1)\delta$$

for $\mathcal{F}_2 = (\pi_1, \omega_2, \omega_1)$, where $\omega_1 =$ "$M^*[\ell - \tau - 1 \ldots \ell] = 0^\tau$" and $\omega_2 =$ "$v_\delta(M^*[0 \ldots \ell - \tau - 1])$".

Remark 2 (*Correlating Conditions and Values*). From Lemma 3, we observe that different control flows provide varying levels of advantage to the adversary. If a condition predicate that offers a greater advantage is evaluated before one with a smaller advantage, it becomes easier for the adversary to distinguish between the real and ideal world.

We broaden the definition of "encoded parts" to encompass any values passed to a condition predicate during decryption. For instance, this could involve verifying the CVV digits of a credit card. We represent such a condition predicate as $\omega =$ "$\mathsf{Dict}[\mathtt{id}] = M^*[i \ldots j]$" for some publicly known identifier `id` and secret dictionary Dict. In scenarios where a scheme is poorly implemented or the adversary takes control of the decryption flow, they may be able to infer these values by evaluating such a condition predicate and correlating the leaked plaintext with the error flag.

This risk exists because, even if the plaintext from a tuple (N, A, C, τ) queried by the adversary remains indistinguishable from a random bitstring, there is still a possibility that it "accidentally" passes the condition predicate, particularly when the value in question has low entropy.

Merge Condition Predicates. We observe an important property that EtE has that can be used to resolve the issue: if the condition predicates all pertain to the same leaked plaintext M^*, we can validate all the failure conditions with one condition predicate. Essentially, the stretching process can be viewed as a mapping $\psi : \mathcal{M} \to \mathcal{M}^*$.

We consider the two condition predicates described above. This allows us to 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 the single valid control flow $\mathcal{F} = (\pi, v_{\delta^*})$, which leads us to the result in Lemma 4. In this case, an adversary can no long choose to first evaluate the condition predicates that may discloses information.

Lemma 4. *For any* IND-CFA *adversary $\mathcal{A}$ against the* EtE *making q decryption queries, there is a* $\widetilde{\mathrm{PRP}}$ *adversary $\mathcal{A}_{tprp}$ against the tweakable* VIL *cipher $\widetilde{E}$ such that*

$$\mathbf{Adv}^{\text{IND-CFA}}_{\mathsf{EtE},\mathcal{F}}(\mathcal{A}) \leq \mathbf{Adv}^{\pm\widetilde{\mathrm{PRP}}}_{\widetilde{E}}(\mathcal{A}_{tprp}) + \frac{q^2}{2^\ell} + \frac{(q+1)\delta}{2^\tau}$$

where $\ell = \arg\min_{(N,A,C,\tau)\in\mathcal{Q}}\{|M^*| : M^* = \widetilde{E}^{-1}_{K;N,A,\tau}(C)\}$ and $\mathcal{Q}$ is the set of encryption queries made by $\mathcal{A}$, under the control flow $\mathcal{F} = (\pi_1, \omega_1)$ where $\omega_1 =$ "$M^*[\ell - \tau - 1 \ldots \ell] = 0^\tau \wedge v_\delta(M^*[0 \ldots \ell - \tau])$".

6 Conclusion and Future Work

In this work, we conducted a comprehensive security analysis of AEAD schemes under adversarially-chosen control flows, with a particular focus on vulnerabilities during decryption. We introduced the new notion of IND-CFA to formalize the security of decryption when the control flow is influenced by an adversary. This notion allows us to systematically evaluate how much security can still be retained under such adversarial conditions, addressing gaps left by previous works on decryption leakage. By capturing both plaintext leakage and descriptive errors, IND-CFA offers a more nuanced view of the security landscape.

We also revisited the Encrypt-then-MAC composition, a widely-used approach for authenticated encryption. Our analysis emphasizes the critical role of operation order, particularly the necessity of ensuring that decryption does not proceed if tag verification fails. This reveals a vulnerability when execution flow is controlled by an adversary, underscoring the importance of additional software-level protections, such as enforcing control flow integrity or protecting memory to prevent the leakage of intermediate values.

Additionally, we provided a concrete security proof for the Encode-then-Encipher paradigm, a prominent method for constructing robust authenticated encryption. Despite its established security under various robustness notions, we demonstrated that Encode-then-Encipher is susceptible to adversarial manipulation when multiple verification steps are involved. However, we highlighted its unique ability to perform a single verification across multiple failure conditions, offering a cost-effective patch that further strengthens its security.

In this work, we focused on decryption security due to the central role verification plays in the decryption process. Extending this analysis to encryption, where similar issues may arise, would be a natural next step. Specifically, we prohibit queries from encryption to decryption oracle to avoid trivial distinguishing. This could be lifted by setting a input-output relation between each encryption and its reverse decryption step, which the ideal functionality can refer to. Also, while our focus was on symmetric primitives, specifically AEAD schemes, control flow attacks also pose significant threats to public-key primitives. Addressing this challenge and extending our security notions to these broader contexts remains an important direction for future research.

References

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

[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.

[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.

[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.

[BGHZ11] Gilles Barthe, Benjamin Grégoire, Sylvain Heraud, and Santiago Zanella Béguelin. Computer-aided security proofs for the working cryptographer. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 71–90. Springer, Heidelberg, August 2011.

[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.

[BLS20] David A. Basin, Andreas Lochbihler, and S. Reza Sefidgar. CryptHOL: Game-based proofs in higher-order logic. *Journal of Cryptology*, 33(2):494–566, April 2020.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[GKM$^+$04] A. D. Gordon, J. Kelsey, J. McAllister, C. Vanderpool, D. Dutton, and J. Kelsey. Cryptol: A language for cryptographic specifications, 2004. Available at: `https://www.cryptol.net/`.

[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.

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

[Leg] Legrandin. Pycryptodome: A self-contained python package of low-level cryptographic primitives. `https://github.com/Legrandin/pycryptodome`. Accessed: 2024-09-09.

[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.

[MP14] David McGrew and Kyle Paterson. Encrypt-then-mac for tls and dtls. RFC 7366, 2014. Accessed: 2024-09-09.

[MV04] David A. McGrew and John Viega. The security and performance of the galois/counter mode of operation (full version). Cryptology ePrint Archive, Report 2004/193, 2004. `https://eprint.iacr.org/2004/193`.

[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.

[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.

[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.

[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.

[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.

[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.

A Detailed Proofs

A.1 Proof of Lemma 2

```
G_0  [G_1]  [G_2]

procedure Initalize
  $K \leftarrow\!\!\$\ \mathcal{K}$
  [ for $(T,\ell) \in \mathcal{T} \times \mathbb{N}$ do
      $\widetilde{\Pi}_{N,A,\tau} \leftarrow\!\!\$\ \widetilde{\mathcal{P}}_\ell$ ]

procedure Enc$(N, A, M, \tau)$
  $C \leftarrow \widetilde{E}_{K;N,A,\tau}(M||0^\tau)$
  [ $C \leftarrow \widetilde{\Pi}_{N,A,\tau}(M||0^\tau)$ ]
  [ $C \leftarrow\!\!\$\ \{0,1\}^{|M|+\tau}$ ]
  return $C$
```

```
Adversary $\mathcal{B}^{\text{Enc}}$

procedure $\mathcal{B}$
  $b \leftarrow \mathcal{A}^{\text{Enc}^*}(\cdot)$
  return $b$

procedure Enc$^*(N, A, M, \tau)$
  $C \leftarrow$ Enc$((N, A, \tau), M||0^\tau)$
  return $C$
```

Figure 3: 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 3 and we use a counter i as nonce. In G_0, the encryption is done with the tweakable VIL cipher $\widetilde{E}$ and the oracle first appends τ zeros after M and returns $\widetilde{E}_{K;N,A,\tau}(M||0^\tau)$ as output. In G_1, the oracle samples a tweakable random permutation $\widetilde{\Pi}$ and return $\widetilde{\Pi}_{N,A,\tau}(M||0^\tau)$ 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

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

We can then construct a $\widetilde{\text{PRP}}$ adversary $\mathcal{B}$ from $\mathcal{A}$ as in Figure 3. We construct the simulated encryption oracle Enc* for $\mathcal{A}$ such that for each encryption query made by $\mathcal{A}$, we let $\mathcal{B}$ append τ zeros after it and forward it to $\mathcal{B}$'s oracle Enc, then $\mathcal{B}$ forwards the response from Enc to $\mathcal{A}$. We then let $\mathcal{B}$ return the same b that $\mathcal{A}$ returns. We then have that

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

Following Lemma 1, we know that the adversary has 0 advantage in distinguishing between G_1 and G_2 if each queried tweak is distinct. Thus we assume the adversary fix (N, A, τ) but query with different M. This means G_1 implements a random permutation. Thus the behaviors of G_1 and G_2 differ when G_2 samples a repeated bitstring, which happens with probability at most $\frac{q^2-q}{2^\ell} \leq \frac{q^2}{2^\ell}$. Thus we have that

$$\Pr[G_1(\mathcal{A})] - \Pr[G_2(\mathcal{A})] \leq \frac{q^2}{2^\ell}$$

Finally, it yields that

$$\mathbf{Adv}^{\text{IND-CPA}}_{\mathsf{EtE}}(\mathcal{A}) \leq \mathbf{Adv}^{\widetilde{\text{PRP}}}_{\widetilde{E}}(\mathcal{B}) + \frac{q^2}{2^\ell}$$

which concludes the proof. □

A.2 Proof of Lemma 3

```
G_0  [G_1]  [G_2]

procedure INITALIZE
  1: $K \leftarrow\!\$\ \mathcal{K}$
  2: [[ 1: for $(T, \ell) \in \mathcal{T} \times \mathbb{N}$ do
        2:   $\widetilde{\Pi}_{N,A,\tau} \leftarrow\!\$\ \widetilde{\mathcal{P}}_\ell$ ]]

procedure ENC$(N, A, M, \tau)$
  1: $C \leftarrow \widetilde{E}_{K;(N,A,\tau)}(M||0^\tau)$
  2: [[ $C \leftarrow\!\$\ \widetilde{\Pi}_{N,A,\tau}(M||0^\tau)$ ]]
  3: return $C$

procedure DEC$(N, A, C, \tau)$
  1: $M^* \leftarrow \widetilde{E}^{-1}_{K;(N,A,\tau)}(C)$
  2: [ $M^* \leftarrow \widetilde{\Pi}^{-1}_{N,A,\tau}(C)$ ]
  3: [ $M^* \leftarrow\!\$\ \{0,1\}^{|C|}$ ]
  4: $\ell \leftarrow |M^*|$
  5: if $M^*[\ell - \tau - 1 \ldots \ell] \neq 0^\tau$ then
  6:   $\mathcal{T} \leftarrow [M^*, \mathsf{false}]$
  7: else
  8:   $\mathcal{T} \leftarrow [M^*, \mathsf{true}]$
  9: [ $\mathcal{T} \leftarrow [M^*, \mathsf{false}]$ ]
 10: return $\mathcal{T}$
```

Figure 4: Game $G_0 - G_2$ for the proof of Lemma 3. Dot-boxed code is exclusive to G_1. Frame-boxed code is exclusive to G_2. Doubly-boxed code is exclusively for both G_1 and G_2.

```
Adversary $\mathcal{B}^{\text{ENC},\text{DEC}}$

procedure $\mathcal{B}$
  1: $b \leftarrow \mathcal{A}^{\text{ENC}^*,\text{DEC}^*}(\cdot)$
  2: return $b$

procedure ENC$^*(N, A, M, \tau)$
  1: $C \leftarrow \text{ENC}((N, A, \tau), M||0^\tau)$
  2: return $C$

procedure DEC$^*(N, A, C, \tau)$
  1: $M^* \leftarrow \text{DEC}((N, A, \tau), C)$
  2: $\ell \leftarrow |M^*|$
  3: if $M^*[\ell - \tau - 1 \ldots \ell] \neq 0^\tau$ then
  4:   $\mathcal{T} \leftarrow [M^*, \mathsf{false}]$
  5: else
  6:   $\mathcal{T} \leftarrow [M^*, \mathsf{true}]$
  7: return $\mathcal{T}$
```

Figure 5: $\pm\widetilde{\text{PRP}}$ adversary $\mathcal{B}$ for the proof of Lemma 3.

Proof. We consider three games $G_0 - G_2$ as in Figure 4 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. The transcripts in G_0 and G_1 are defined as $\mathcal{T} = [M^*, r]$ where $r \in \{\mathsf{true}, \mathsf{false}\}$ depends on whether M^* ends with τ zeros. In game G_2, a bitstring M^* is sampled uniformly at random of length $|C|$ and the transcript $\mathcal{T} = [M^*, \mathsf{false}]$ is always returned. In G_2, we still answer $\mathcal{A}$'s encryption query with $\widetilde{\Pi}$. We have that

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

Now we show that we can construct an $\widetilde{\pm\mathrm{PRP}}$ adversary $\mathcal{B}$ as in Figure 5. We construct the simulated encryption oracle $\textsc{Enc}^*$ for $\mathcal{A}$ such that for each encryption query made by $\mathcal{A}$, we let $\mathcal{B}$ append τ zeros after it and forward it to $\mathcal{B}$'s oracle $\textsc{Enc}$, then $\mathcal{B}$ forwards the response from $\textsc{Enc}$ to $\mathcal{A}$. For the simulated decryption oracle $\textsc{Dec}^*$, we let $\mathcal{B}$ forward the tuple queried by $\mathcal{A}$ to its oracle $\textsc{Dec}$ to get M^*. Depending if M^* ends with τ zeros, we let $\mathcal{B}$ return $\mathcal{T} = [M^*, \mathsf{true}]$ and $\mathcal{T} = [M^*, \mathsf{false}]$ respectively. We then have that We then have that

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

We first assume the adversary does not fix (N, A, τ), which means each tweak queried to $\textsc{Dec}$ is distinct. Similarly, following Lemma 1, the adversary has 0 advantage in distinguish from M_*. Thus the transcript $\mathcal{T}_1$ generated in G_1 and the transcript $\mathcal{T}_2$ generated in G_2 only differs if $\mathcal{T}_1$ contains a true, meaning M^* ends with τ zeros. Since we have a fresh random permutation at each query, this happens with probability at most $\frac{1}{2^\tau}$.

Now we assume the adversary fix (N, A, τ) but query with different C. Then the behavior of G_1 and G_2 happens if one of the following two events happens. In the first event, the transcript from G_1 yields $\mathcal{T}_2 = (M^*, \mathsf{true})$, which happens with probability at most $\frac{q}{2^\tau}$ since we fix a random permutation in G_1 this time. In the second case, the M^* sampled in G_2 repeats, which happens with probability $\frac{q^2}{2^\ell}$. By Union Bound, we obtain the bound

$$\Pr[\mathrm{G}_1(\mathcal{A})] - \Pr[\mathrm{G}_2(\mathcal{A})] \leq \frac{q^2}{2^\ell} + \frac{q}{2^\tau} + \frac{1}{2^\tau}$$

Finally, we have that

$$\mathbf{Adv}^{\text{IND-CFA}}_{\mathsf{EtE}}(\mathcal{A}) \leq \mathbf{Adv}^{\widetilde{\pm\mathrm{PRP}}}_{\widetilde{E}}(\mathcal{B}) + \frac{q^2}{2^\ell} + \frac{q+1}{2^\tau}$$

which concludes the proof. □