Security of COFB against Chosen Ciphertext Attacks

Mustafa Khairallah

Nanyang Technological University
Singapore, Singapore
mustafa.khairallah@ntu.edu.sg

Abstract. COFB is a lightweight Authenticated Encryption with Associated Data (AEAD) mode based on block ciphers. It was proposed in CHES 2017 and is the basis for GIFT-COFB, a finalist in the NIST lightweight standardization project. It comes with provable security results that guarantee its security up to the birthday bound in the nonce-respecting model. However, the designers offer multiple versions of the analysis with different details and the implications of attacks against the scheme are not discussed deeply. In this article, we look at a group of possible forgery and privacy attacks against COFB. We show that the security for both forgery and privacy is bounded by the number of forgery attempts. We show the existence of forgery and privacy attacks with success probability $q_d/2^{n/2}$, given q_d forgery attempts. In particular, we show an attack with $2^{n/2}$ attempts using only a single known-plaintext encryption query against COFB. While these attacks do not contradict the claims made by the designers of GIFT-COFB, they show its limitations in terms of the number of forgery attempts. They also show that, while COFB generates a 128-bit tag, it behaves in a very similar manner to an AEAD scheme with 64-bit tag. As a result of independent interest, our analysis provides a contradiction to the main theorem of *Journal of Cryptology volume 33, pages 703–741 (2020)*, which includes an improved security proof of COFB compared to the CHES 2017 version. Finally, we discuss the term $nq_d/2^{n/2}$ that appears in the security proof of GIFT-COFB and CHES 2017, showing why there is a security gap between the provable results and the attacks. We emphasize that the results in this article do not threaten the security of GIFT-COFB in the scope of the NIST lightweight cryptography requirements or the claims made by the designers in the specification document of the design.

Keywords: COFB · GIFT · Block Cipher · NIST · AEAD · Authenticated Encryption · Forgery · CCA · Indistinguishability · Privacy

1 Introduction

Lightweight cryptography is the field of cryptology that deals with designing algorithms with a small footprint or low computational complexity targeted towards constrained devices, *e.g.,* micro-controllers and low area/power integrated circuits. Over the past few years, the National Institute for Standardization and Technology (NIST), USA, has been running a lightweight cryptography standardization project. One of the categories of the project called for Authenticated Encryption with Associated Data (AEAD) algorithms where the amount of data that can be processed under one key is at least $2^{50} - 1$ bytes and the cryptanalytic attacks against the algorithms are of at least 2^{112} computational complexity [nis18]. The project received 57 proposals, 56 of them were selected as round 1 candidates, then narrowed down to 32 in round 2. In March 2021, 10 proposals were announced as finalists. Among these candidates, GIFT-COFB [BCI+20] is a block cipher-based proposal and will be the focus of this article.

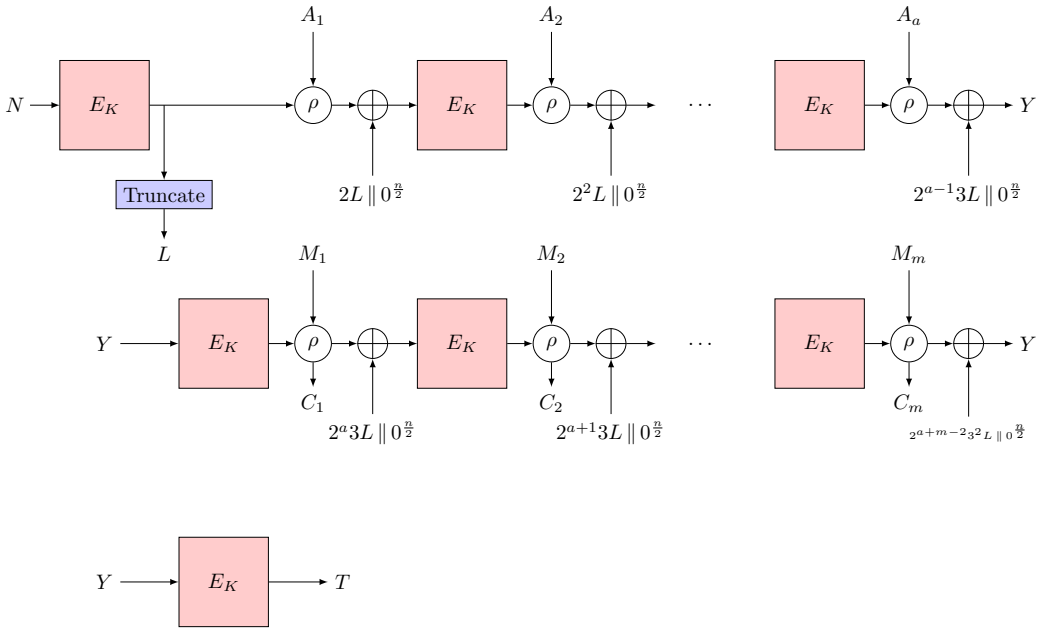

Figure 1: The COFB mode of operation.

The GIFT-COFB mode (depicted in Figure 1) is an instance of the COmbined FeedBack (COFB) mode, which is an AEAD mode proposed in CHES 2017 [CIMN17] as a lightweight algorithm based on n-bit Block Ciphers (BC). It is claimed to offer the integrity of ciphertexts up to $2^{n/2}/n$ forgery attempts and privacy up to $2^{n/2}$ data complexity in the nonce-respecting model. The integrity limit comes from a bound on the adversary's success probability in producing a forgery of the form $nq_d/2^{n/2}$ where q_d is the number of forgery attempts made by the adversary. Interestingly, this bound relies only on the number of forgery attempts and is independent of the amount of data encrypted by the algorithm or the computational capabilities of the adversary. It is typical to see similar terms when it comes to generic attacks based on the authentication tag size. In particular, an AEAD scheme that generates a τ-bit tag can be attacked by simply guessing the correct tag corresponding to a given ciphertext. The attack's success probability relies on the number of forgery attempts ($q_d/2^\tau$) and after 2^τ the adversary would have guessed the correct tag. However, COFB has a tag size of n bits, so the appearance of terms of the form $q_d/2^{n/2}$ and $nq_d/2^{n/2}$ is interesting. We also note that not all AEAD modes that are secure up to the Birthday Bound (upBB) suffer from such issues. For example, the GCM [MV04] is probably the most famous BC-based AEAD mode secure upBB but the security bound is of the form of $\sigma^2/2^n$, where σ is the total amount of data processed by the algorithm given a certain key. While $\sigma/2^{n/2} \approx \sigma^2/2^n$ when $\sigma \approx 2^{n/2}$, $\sigma/2^{n/2}$ is significantly larger than $\sigma^2/2^n$ when σ is small. These observations raise some research questions. Most notably:

1. *Can we break the* COFB *algorithm with only* $2^{n/2}/n$ *forgery attempts and negligible (or 0) encryption queries?*

2. *Can we show that* COFB *behaves as a scheme with a tag that is shorter than* n *bits, even when an* n*-bit tag is generated?*

These two questions are not answered by the security proofs of COFB. Provable security is a critical tool in studying the security of new designs. It provides mathematical guarantees for their security. However, it does not always take the adversary's point of

view and it often does not consider what happens when the provable security bounds are reached. It may lead to conservative bounds that cannot be matched by attacks in practice. Besides, analyzing the schemes helps understand and verify the security proofs, understand the different assumptions that the designers may have used or implied, and identify errors, if any.

Given a BC with n-bit block and k-bit key, COFB expands the internal state by only $n/2$ bits compared to the state of the BC ($n + k$ bits). The designers assume that the BC is secure in the standard Pseudo-Random Permutation (PRP) model and that it behaves as a Pseudo-Random Function (PRF) up to the bound derived in the PRP-PRF switching lemma [BR06]. They assume that an adversary makes q_e encryption queries that involve σ_e blocks (σ_e invocations of the BC) and q_d forgery attempts that involve σ_d blocks. The authors define COFB-R, a variant of COFB where all the BC calls are replaced by calls to one random function R sampled uniformly from the set of all random functions from n bits to n bits. The authors presented a provable security bound that suggests that the success probability of any single-key forgery adversary $\mathcal{A}$ against COFB-R as an AEAD scheme is bounded by

$$\Pr[\mathcal{A} \text{ forges COFB-R}] \leq \frac{4\sigma_e + 0.5nq_d}{2^{n/2}} + \frac{q_d + (q_e + \sigma_e + \sigma_d)\sigma_e}{2^n}.$$

Subsequently, an extended version of COFB has been published in the Journal of Cryptology (JoC) in 2020 [CIMN20]. The provable security bound in this version implies a different probability bound:

$$\Pr[\mathcal{A} \text{ forges COFB-R}] \leq \frac{4\sigma_e}{2^{n/2}} + \frac{q_d + (q_e + \sigma_e + 2\sigma_d)\sigma_d}{2^n}.$$

As part of the efforts surrounding the NIST lightweight cryptography project, the designers of GIFT-COFB [BCI+20] provided the following bound in the *Cryptology ePrint Archive report 2020/738*:

$$\Pr[\mathcal{A} \text{ forges COFB-R}] \leq \frac{1}{2^{n/2}} + \frac{(n+4)q_d}{2^{n/2+1}} + \frac{q_d + \sigma_e^2 + (q_e + \sigma_e + \sigma_d)\sigma_e}{2^n}.$$

While the three bounds share a lot of similarities, some of the strategies used in each security proof are different, which leads to the differences. More importantly, the bound from JoC 2020 is dominated by terms of the form $\sigma^2/2^n$. If this bound is correct, then the question is if it is possible to adopt that bound for GIFT-COFB, improving its security claims. Another major difference between the different versions is the nonce size, where [CIMN17] and [CIMN20] used an $n/2$-bit nonce, while [BCI+20] used an n-bit nonce. This has an effect on one of the attacks presented. We note that the NIST Lightweight cryptography call for submission [nis18] requires the nonce size to be at least 96 bits ($3n/4$ bits). In Section 5, we discuss the effect of the nonce size and the applicability to similar algorithms, specifically, HyENA, another design that has been a second-round candidate for the NIST lightweight standardization process and holds a lot of resemblance to COFB.

Contributions In this article, we analyze the security of COFB. We present the first CCA attack on the scheme with complexity $2^{n/2}$ and success probability $q_d/2^{n/2}$ and the first forgery attack with complexity $2^{n/2}$ to operate with a single encryption query. Not only do the attacks treat the BC as a black box, but they work even if the BC is replaced with a random function. We show that the behavior of COFB is very close to that of an algorithm with half the tag size. Table 1 shows a summary of the forgery attacks presented in the paper against COFB, their complexity, and whether they lead to CCA distinguishability attacks. Besides, our analysis shows an oversight in the JoC security proof and contradicts the main theorem of [CIMN20]. We show that the adversary's advantage for INT-CTXT

Table 1: The designers' claims on the data limits for successful forgery against COFB

Attack/Security Claims	Decryption Complexity	Encryption Complexity	IND$-CCA
COFB bound on forgeries	$2^{n/2}/n$	-	-
Block-Dropping (Section 4) [Kha20]	$2^{n/2}$	$2^{n/2+1}$	-
Optimized Block-Dropping (Section 4)	$2^{n/2}$	$O(1)$	-
Prefix-Suffix (Section 4)	$2^{n/2}$	$2^{n/4}$	-
Mask Enumeration (Section 5) with n-bit nonce	$2^{n/2}$	1	✓
Mask Enumeration (Section 5) with t-bit nonce	$2^{n/2}$	2^{n-t}	✓

is lower-bounded by $q_d/2^{n/2}$. Hence, it cannot be upper-bounded by a bound of the form $\sigma^2/2^n$. Finally, we discuss the bounds of the form $nq_d/2^{n/2}$ and analyze why such bounds appear in the security proof. We also formalize a CCA distinguishability attack that was proposed previously against a class of nonce-respecting AEAD schemes [Mè19].

Outline The paper is organized as follows: Section 2 includes the preliminaries needed to understand the paper. Section 3 presents a short discussion on the relation between integrity and chosen-ciphertext privacy from an attacker's point of view. Section 4 includes a group of forgery attacks on COFB. Section 5 presents a CCA distinguishability/privacy attack on COFB. Section 6 discusses the oversight in [CIMN20] and the logarithmic gap form the birthday bound in the security claims of COFB. Section 7 includes a brief introduction of related work and the paper is concluded in Section 8.

2 Preliminaries

2.1 Definitions and Security Notions

Notations We use ϵ to refer to the empty bit-string of length 0. $\{0,1\}^b$ is the set of all bit-strings of length b, while $\{0,1\}^*$ is the set of all arbitrary length bit-strings including the empty string ϵ. $|X|$ is the length of a bit-string X, where $|\epsilon| = 0$. Given two bit-strings X and Y, $X\|Y$ is the concatenation of the two strings. Given a bit-string X, where $|X| \geq l$, then $\lfloor X \rfloor_l$ is the bit-string consisting of the l leftmost bits of X, *i.e.,* l-bit truncation. $(X_1, .., X_l) \xleftarrow{n} X$ divides the bit-string X into l strings, each consists of n bits, except potentially the l^{th} block which can be smaller than n bits. We also refer to this operation as n-bit parsing. $\perp$ is a special symbol used to indicate the ciphertext is not authentic. $X \xleftarrow{\$} \mathcal{X}$ indicates that X is a random variable picked from $\mathcal{X}$ according to the uniform distribution. We use superscript to indicate either the order or type of variable, *e.g.,* X^i indicates the i^{th} instance of X, X^c mean challenge X and X^f means forgery X. We use subscript to indicate the block order within a multi-block variable. For example, if $|X| = 2n$, then $X = X_1\|X_2$.

Adversaries An adversary models an attacker. It is a program/routine with access to one or more oracles. The adversary makes queries to the oracle(s) and receives responses, and succeeds if it manages to break a given security goal.

Authenticated Encryption with Associated Data An Authenticated Encryption (AE) scheme [BN00] is a symmetric key algorithm that provides both privacy and authenticity. An AEAD scheme [Rog02] is similar except that both algorithms take an extra input called associated data A which is a public portion of the message, used for authentication only. In this paper, we focus on nonce-based AEAD, sometimes known as NAE.

Nonce-Based AEAD Syntax A nonce-based AEAD scheme, in the context of this paper, is defined as a three-tuple $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. $\mathcal{K} = \{0,1\}^k$ is the key space. Given a key $K \in \mathcal{K}$, $\mathcal{E}$ and $\mathcal{D}$ are defined as follows:

$$(C, T) \leftarrow \Pi.\mathcal{E}(K, N, A, M)$$

$$M \text{ or } \perp \leftarrow \Pi.\mathcal{D}(K, N, A, C, T)$$

where $N \in \{0,1\}^t$ is a public parameter corresponding to the plaintext/ciphertext, usually referred to as *nonce*. $C, M \in \mathcal{M} \subseteq \{0,1\}^*$, $A \in \mathcal{A} \subseteq \{0,1\}^*$ and $T \in \{0,1\}^\tau$. For simplicity, we drop Π when we are concerned with only one scheme, or the scheme in question is understood from the context. We also can refer to $\mathcal{E}(K, N, A, M)$ as $\mathcal{E}_K(N, A, M)$ or $\mathcal{E}_K^{N,A}(M)$ and the same for $\mathcal{D}$. Both $\mathcal{E}$ and $\mathcal{D}$ are deterministic algorithms, such that $\mathcal{D}_K^{N,A}(\mathcal{E}_K^{N,A}(M)) = M$.

Nonce-Based AEAD Security A nonce-based AEAD scheme requires that N is unique for all the invocations of $\mathcal{E}$. The nonce-based AEAD schemes are required to achieve at least two security notions: privacy and authenticity. We define both as follows:

Privacy The privacy of an AEAD scheme is defined using indistinguishability of ciphertexts from random strings, where the adversary sends a message and the oracle selects whether to encrypt the message using the AEAD scheme or output a string of random bits, based on a random selection bit b. The adversary has to distinguish between these two possibilities. Such adversaries can be classified into Chosen-Plaintext Adversaries (CPA) and Chosen-Ciphertext Adversaries (CCA), depending on whether the adversary interacts with only the encryption algorithm or both the encryption and decryption algorithms, respectively. In order to define the advantage of an indistinguishability adversary, we define a game that models the behavior and interaction between the adversary and the oracle, as shown in Figure 2. The IND$-CPA adversary is not allowed to interact with **Dec**. In fact, the depiction IND$-CPA in Figure 2 is equivalent to that in [Rog02], the inclusion of the **Dec** oracle is useful in explaining one of the attacks presented later. Adversaries cannot perform trivial decryption queries which are responses from earlier encryption queries and cannot repeat queries.

An adversary $\mathcal{A}$ interacts with the game by requesting q_e queries to **Enc** and q_d queries to **Dec**. The queries to **Enc** and **Dec** can be interleaved. At the end of the exchange, $\mathcal{A}$ calls **Finalize** with its guess of d and wins if **Finalize** returns 1. The IND$-CCA advantage of $\mathcal{A}$ against an AEAD scheme is defined as

$$\mathbf{Adv}_{\mathtt{aead}}^{\mathtt{ind\$-cca}}(\mathcal{A}) = \Pr[d = 0 | b = 0] - \Pr[d = 0 | b = 1],$$

which measures how much better can the adversary do compared to an adversary that randomly guesses b. We note that the adversaries can be characterized by how many queries they perform, where q_e is the number of encryption queries and q_d is the number of decryption queries. Besides, we define σ_e and σ_d as the sum of the length of encryption and decryption queries, respectively, which can be measured in bits, bytes, or bit-blocks, depending on the context. We use $\sigma = q_e + q_d + \sigma_e + \sigma_d$ as the total data complexity. We also use σ in cases where we are referring to the asymptotic data complexity of the

Initialize
$K \xleftarrow{\$} \mathcal{K}$
$b \xleftarrow{\$} \{0,1\}$
$\mathcal{S} \leftarrow \phi$
$\boxed{\mathcal{N} \leftarrow \phi}$
Dec(N, A, C, T)
If$(N, A, C, T) \in \mathcal{S}$ **then**
  $M \leftarrow \perp$
Else then
  $M \leftarrow \mathcal{D}_K^{N,A}(C, T)$
Return M
Enc(N, A, M)
$\boxed{\textbf{If } N \in \mathcal{N} \textbf{ then}}$
$\boxed{\textbf{Return } \perp}$
If $b = 0$ **then**
  $(C, T) \leftarrow \mathcal{E}_K^{N,A}(M)$
Else then
  $(C, T) \xleftarrow{\$} \{0,1\}^{|M|+\tau}$
$\mathcal{S} \leftarrow \mathcal{S} \cup \{(N, A, C, T)\}$
$\boxed{\mathcal{N} \leftarrow \mathcal{N} \cup \{N\}}$
Return(C, T)
Finalize(d)
Return $b = d$

Figure 2: The IND$-CCA security game as used in the paper.

adversary without distinction between encryption and decryption. Let the maximum running time of adversaries be t, then we write the upper bound of the advantage of any IND$-CCA adversary as

$$\mathbf{Adv}^{\mathtt{ind\$-cca}}_{\mathtt{aead}}(q_e, \sigma_e, q_d, \sigma_d, t) = \max_{\mathcal{A}} \mathbf{Adv}^{\mathtt{ind\$-cca}}_{\mathtt{aead}}(\mathcal{A})$$

Authenticity/Integrity A nonce-based AEAD scheme Π is required to ensure the integrity of ciphertexts (INT-CTXT). An adversary $\mathcal{A}$ breaks the INT-CTXT security if it submits a decryption query (N, A, C, T) to **Verify** in Figure 3 whose decryption $M \neq \perp$ and (C, T) has never been returned by a query **Enc**(N, A, M). Formally, the INT-CTXT advantage of $\mathcal{A}$ against an AEAD scheme is defined by

$$\mathbf{Adv}^{\mathtt{int-ctxt}}_{\mathtt{aead}}(\mathcal{A}) = \Pr[\mathcal{A} \text{ forges } \Pi]$$

where $\mathcal{A}$ forges Π is the event that any response to a query $\mathcal{A}$ makes to **Dec** is not $\perp$. We write the upper bound on the INT-CTXT advantage as

$$\mathbf{Adv}^{\mathtt{int-ctxt}}_{\mathtt{aead}}(q_e, \sigma_e, q_d, \sigma_d, t) = \max_{\mathcal{A}} \mathbf{Adv}^{\mathtt{int-ctxt}}_{\mathtt{aead}}(\mathcal{A})$$

In [CIMN17, BCI+20, CIMN20], the authors use a unified AEAD security notion for both privacy and authenticity. The adversary $\mathcal{A}$ is defined as the combination of both the privacy and integrity adversaries, *i.e.*, it wins if it can beat either of IND$-CPA or INT-CTXT games. This unified notion is actually similar to IND$-CCA, except that it considers a successful forgery as a win for the adversary, while IND$-CCA allows forgeries and requires the adversary to still perform a distinguishing attack. Note that unless the

Initialize
$K \xleftarrow{\$} \mathcal{K}$
$\mathcal{S} \leftarrow \phi$
$\boxed{\mathcal{N} \leftarrow \phi}$
Enc(N, A, M)
$\boxed{\textbf{If } N \in \mathcal{N} \textbf{ then}}$
$\boxed{\quad \textbf{Return } \perp}$
$(C, T) \leftarrow \mathcal{E}_K^{N,A}(M)$
$\mathcal{S} \leftarrow \mathcal{S} \cup \{(N, A, C, T)\}$
$\boxed{\mathcal{N} \leftarrow \mathcal{N} \cup \{N\}}$
Return (N, A, C, T)
Verify(N, A, C, T)
If$(N, A, C, T) \in \mathcal{S}$ **then**
$M \leftarrow \perp$
Else then
$M \leftarrow \mathcal{D}_K^{N,A}(C, T)$
Return $(M \neq \perp)$

Figure 3: The INT-CTXT game.

unified adversary is able to forge a ciphertext, *i.e.,* it can break INT-CTXT, all the calls to **Dec** in Figure 2 output $\perp$ and only calls to **Enc** output non-trivial values. If no adversary can break the unified notion, then no adversary can break IND$-CCA. The formalization of IND$-CCA is meant to emphasize the effect of successful forgery on privacy.

Nonce Misuse/Repetition The procedures presented in Figures 2 and 3 ensure that the nonces are never repeated during queries to **Enc**. In practice, implementing such countermeasure can be expensive. Hence, in some applications, we need to consider the implication of repeating the nonce. The IND$-CCA and INT-CTXT games, in this case, are similar to Figures 2 and 3, but do not implement the framed lines, which ensure that nonces cannot be repeated. In this paper, we consider schemes that are *insecure* against adversaries that can repeat nonces. When we say a scheme is IND$-CPA-insecure (IND$-CCA-insecure) against nonce-misuse adversaries, we mean that there is an adversary $\mathcal{A}_m$ that can adaptively choose a pair of queries

$$\textbf{Enc}(N, A^1, M^1)$$

and

$$\textbf{Enc}(N, A^2, M^2),$$

and based on the responses can guess the value of b, such that

$$\textbf{Adv}_{\mathtt{aead}}^{\mathtt{ind\$-cpa}}(\mathcal{A}_m) \approx 1.$$

2.2 The COFB mode

The COFB mode, shown in Figure 1, is an NAE scheme. Both encryption and decryption are similar, where the only difference is the linear feedback function that absorbs the plaintext during encryption and the ciphertext during decryption. The scheme starts by initializing its internal state by encrypting the nonce N. Not only is $E_K(N)$ used to initialize the internal state, but also to generate an $n/2$-bit mask L, as $\lfloor (E_K(N) \rfloor_{n/2}$. After the initial state and the mask are generated, the associated data A is divided into a blocks of n bits each and

absorbed as shown Figure 1. We define the state after absorbing the last block of A to be Y_a. We define $X_i = E_K(Y_{i-1})$. The message M is also divided into m blocks of n bits each and absorbed by applying a linear feedback function $(S_{i+a}, C_i) = \rho(X_{i+a-1}, M_i)$ and the new state S_{i+a+1} is masked into $Y_{i+a} = S_{i+a} \oplus 3^{p_a} \cdot 2^{a+i-1} \cdot L \| 0^{n/2}$, where the multiplication is done over GF$(2^{n/2})$ and can be represented by a cheap Linear Feedback Shift Register (LFSR). The tag is generated as follows: $Y_{a+m} = S_{a+m} \oplus 3^{p_a+p_m} \cdot 2^{a+m-2} \cdot L \| 0^{n/2}$ is calculated, then T is calculated as $E_K(Y_{a+m})$. $p_a = 1$ if the last block of A is n bits and 2, otherwise. Similarly, $p_m = 1$ is the last block of M is n bits, and 2, otherwise.

2.2.1 Combined Feedback

The combined feedback function ρ used in the COFB mode is a linear transformation from $2n$ bits to $2n$ bits. Given an output of the BC X_i and a plaintext block M_i, it outputs $C_i = X_i \oplus M_i$ and $S_i = M_i \oplus G(X_i)$, where G is a linear permutation over n bits. During decryption, it takes a ciphertext block C_i instead and outputs $M_i = X_i \oplus C_i$ and $S_i = X_i \oplus G(X_i) \oplus C_i$. If an adversary has access to a known-plaintext-ciphertext pair (M_i, C_i), X_i and S_i can be found by $M_i \oplus C_i$ and $M_i \oplus G(M_i \oplus C_i)$, respectively. Before applying the next BC call, S_i is masked to Y_i using the mask L as shown in Figure 1.

3 Converting an INT-CTXT adversary into an IND$-CCA adversary

Bellare and Namprempre [BN00] proved a relation between CCA indistinguishability, CPA indistinguishability and ciphertext integrity. We can also see that an IND$-CPA adversary $\mathcal{A}_p$ that runs in time t and makes q_e encryption queries can be used to construct an IND$-CCA adversary $\mathcal{A}$ that runs in time $O(t)$ and makes q_e encryption queries and 0 decryption queries. $\mathcal{A}$ runs $\mathcal{A}_p$ and outputs whatever $\mathcal{A}_p$ outputs. On the other hand, not every INT-CTXT adversary against an NAE-Scheme can be used to construct an IND$-CCA adversary. For example, consider the INT-CTXT adversary in Figure 4 which flips 1 bit of a ciphertext generated from an encryption query and tries to guess the corresponding tag of the new ciphertext. As long as $d = 0$, $\mathcal{A}$ gains no information from the calls to **Verify**. $\mathcal{A}$ still needs to define an IND$-CCA distinguisher. An important observation is that in the example in Figure 4, $\mathcal{A}$ cannot use N as the nonce of an **Enc** query, as that would not be nonce respecting. Hence, $\mathcal{A}$ is limited in how it can use the information it gains from $\mathcal{A}_c$ even if the forgery succeeds. Due to limitations like this, it is not always clear how to use an INT-CTXT adversary to construct an IND$-CCA adversary.

	INT-CTXT$^{\mathcal{A}_c}$
1 :	**Initialize**
2 :	$(C, T) \leftarrow$ **Enc**(N, A, M)
3 :	**For** i in $1 \cdots q_d$
4 :	$\quad T_f \xleftarrow{\$} \{0,1\}^\tau$
5 :	$\quad d \leftarrow$ **Verify**$(N, A, C \oplus 1, T_f)$
6 :	$\quad$ **If** $d = 1$ **then**
7 :	$\quad\quad$ **Return** d
8 :	**Return** d

Figure 4: A simple INT-CTXT adversary against an NAE-scheme.

This issue is relevant for at least two scenarios:

1. If the concrete security against INT-CTXT adversaries is much lower that the concrete security against IND$-CPA adversaries, this may affect the concrete security against IND$-CCA adversaries negatively, compared to IND$-CPA adversaries. In other words, if $\mathbf{Adv}^{\mathtt{int-ctxt}}_{\mathtt{aead}}(q_e, \sigma_e, q_d, \sigma_d, t) \gg \mathbf{Adv}^{\mathtt{ind\$-cpa}}_{\mathtt{aead}}(q_e, \sigma_e, t)$, and there exists an INT-CTXT adversary $\mathcal{A}_c$ and an IND$-CCA adversary $\mathcal{A}$ such that $\mathbf{Adv}^{\mathtt{ind\$-cca}}_{\mathtt{aead}}(\mathcal{A}) \approx \mathbf{Adv}^{\mathtt{int-ctxt}}_{\mathtt{aead}}(\mathcal{A}_c) \approx \mathbf{Adv}^{\mathtt{int-ctxt}}_{\mathtt{aead}}(q_e, \sigma_e, q_d, \sigma_d, t)$, then the concrete security against IND$-CCA adversaries is significantly less than that against IND$-CPA adversaries.

2. IND$-CPA adversaries have no access to **Dec** or **Verify**. Hence, the number of decryption queries q_d is irrelevant to IND$-CPA security. In practice, we are interested in different possible attacks with different complexities. One scenario of interest is attacks that use a small (constant) number of encryption queries q_e, and the attack complexity is dominated by the number of verification/decryption queries, as they determine how many failed forgeries can be allowed in practice. Additionally, some protocols may tolerate occasional forgeries (INT-CTXT) but not privacy fails (IND$-CCA).

To illustrate, consider an NAE-scheme whose advantages are bounded by

$$\mathbf{Adv}^{\mathtt{ind\$-cpa}}_{\mathtt{aead}}(q_e, \sigma_e, t) \leq \frac{q_e + t}{2^{128}}$$

and

$$\mathbf{Adv}^{\mathtt{int-ctxt}}_{\mathtt{aead}}(q_e, \sigma_e, q_d, \sigma_d t) \leq \frac{q_d}{2^{64}} + \frac{q_e + t}{2^{128}}.$$

Without a dedicated IND$-CCA analysis, we do not know if the advantage is close to that of INT-CTXT, which is quite low compared to the IND$-CPA bound. This example can be a scheme that provides 128-bit IND$-CPA security but that only has a 64-bit tag. In a discussion on the NIST lightweight cryptography forum [Mè19], Alexandre Mège proposed an IND$-CCA adversary against any NAE scheme that process the plaintext in blocks of n bits, with tag size τ. The adversary requires only 1 encryption query and and has success probability $q_d/2^{\tau}$ for q_d decryption queries. To construct this adversary, we start from a tag-guessing INT-CTXT adversary, *i.e.* an adversary that asks for the decryption of a random ciphertext and tries to guess the corresponding tag. If the tag size is τ bits then the probability of successful random guessing is $q_d/2^{\tau}$. Additionally, the scheme is IND$-CPA-insecure against nonce-misuse adversaries. Consider the pairs (A, M) and $(A^{'}, M^{'})$ carefully selected by the adversary to satisfy a property related to the structure of the scheme in question. Given knowledge about two tuples (N, A, M, C, T) and $(N, A^{'}, M^{'}, C^{'}, T^{'})$, such that $(C, T) = \mathcal{E}_K^{N,A}(M)$ and $(C^{'}, T^{'}) = \mathcal{E}_K^{N,A^{'}}(M^{'})$, the adversary can guess b with overwhelming probability.

For simplicity, we formalize the attack only to schemes that satisfy the following property:

There is an integer $n > 0$, *s.t.* if:

1. $|A| = |A^{'}|$.
2. $|M| = |M^{'}| = 2n$.
3. $M \neq M^{'}$.
4. $(C, T) = \mathcal{E}_K^{N,A}(M)$ and $(C^{'}, T^{'}) = \mathcal{E}_K^{N,A^{'}}(M^{'})$.

then,

$$\lfloor M \rfloor_n = \lfloor M^{'} \rfloor_n \text{ implies } \lfloor C \rfloor_n = \lfloor C^{'} \rfloor_n$$

and

$$\lfloor M \rfloor_n \neq \lfloor M^{'} \rfloor_n \text{ implies } \lfloor C \rfloor_n \neq \lfloor C^{'} \rfloor_n.$$

Figure 5 depicts an adversary against schemes that satisfy this property. The adversary tries to guess the tag for a tuple (N, A, C, T). Once the adversary guesses the correct tag, it receives the corresponding message M. The adversary transforms (A, M) into $(A^{'}, M^{'})$ which satisfies the property. The running time of this adversary is probabilistic with worst case runtime $q_d = 2^{\tau}$. Since the sampling in step 7 never repeats the same value, then after 2^{τ} iterations all the values of T will have been tried, *i.e.*, the number of iterations cannot exceed 2^{τ}. To sum up, the attack requires 2^{τ} forgery attempts and only one encryption query. We emphasize that this attack, as depicted in Figure 5, applies only to schemes that satisfy the mentioned property. However, it is easy to change it to other types of schemes as well. The critical issue that makes a scheme vulnerable to this attack strategy is whether the scheme is also vulnerable to nonce-misuse adversaries.

IND$-CCA$^{\mathcal{MG}}$

1 : **Initialize**
2 : $M \leftarrow \perp$
3 : $C \xleftarrow{\$} \{0,1\}^{2n}$
4 : $N \xleftarrow{\$} \{0,1\}^{t}$
5 : $\mathcal{T} \leftarrow \phi$
6 : **While** $M = \perp$
7 : $\quad T \xleftarrow{\$} \{0,1\}^{\tau} \setminus \mathcal{T}$
8 : $\quad \mathcal{T} \leftarrow \mathcal{T} \cup \{T\}$
9 : $\quad M \leftarrow \mathbf{Dec}(N, A, C, T)$
10 : $(A^{'}, M^{'}) \leftarrow (A, M \oplus 0^{n} \| 1^{n})$
11 : $(C^{'}, T^{'}) \leftarrow \mathbf{Enc}(N, A^{'}, M^{'})$
12 : **If** $\lfloor C \rfloor_n = \lfloor C^{'} \rfloor_n$ **then**
13 : $\quad d \leftarrow 0$
14 : **Else then**
15 : $\quad d \leftarrow 1$
16 : **return Finalize**(d)

Figure 5: A formalization of the IND$-CCA adversary proposed in [Mè19].

4 Forgery Attacks against COFB

In this section, we present 3 forgery adversaries against COFB that require around $2^{n/2}$ forgery attempts (**Verify** queries). The first adversary is an application of the analysis framework proposed in [Kha20]. Hence, it is not optimized for COFB in particular, but for a wide class of AEAD schemes. We call this adversary the Block-Dropping ($\mathcal{BD}$) adversary and it is depicted in Figure 6. The second adversary is an optimization for the case of COFB, called Optimized Block-Dropping ($\mathcal{OBD}$) adversary and it is depicted in Figure 7. The third adversary is a Mask-Collision ($\mathcal{MC}$) adversary and it is depicted in Figure 8.

	$\text{INT-CTXT}^{\mathcal{BD}}$
1 :	**Initialize**
2 :	$d \leftarrow$ **False**
3 :	$M \xleftarrow{\$} \{0,1\}^{2n}$
4 :	$\mathcal{N} \leftarrow \phi$
5 :	**While** $d =$ **False and** $\{0,1\}^t \setminus \mathcal{N} \neq \phi$
6 :	$\quad N \xleftarrow{\$} \{0,1\}^t \setminus \mathcal{N}$
7 :	$\quad \mathcal{N} \leftarrow \mathcal{N} \cup \{N\}$
8 :	$\quad (C,T) \leftarrow \mathbf{Enc}(N, \epsilon, M)$
9 :	$\quad (M_1, M_2) \xleftarrow{n} M$
10 :	$\quad C^f \leftarrow M_2 \oplus M_1 \oplus C_1 \oplus G(M_1 \oplus C_1) \oplus G(M_2 \oplus C_2)$
11 :	$\quad d \leftarrow \mathbf{Verify}(N, \epsilon, C^f, T)$
12 :	**Return** d

Figure 6: The Block-Dropping INT-CTXT adversary based on the framework from [Kha20].

4.1 Block-Dropping

The $\mathcal{BD}$ and $\mathcal{OBD}$ adversaries, depicted in Figures 6 and 7, both try to drop a block from a ciphertext that has been legitimately generated using a chosen plaintext. The $\mathcal{BD}$ adversary exploits the fact that when the mask of the COFB scheme is set to $0^{n/2}$, the mask value does not change from one block to another. In such case, the encryption is very similar to the CBC encryption. The adversary then assumes that the mask is always $0^{n/2}$, which, for a chosen-plaintext **Enc** query, gives him a candidate for the internal state at each point of the execution. We know that for any plaintext block M_i, the output of the previous BC call $X_{i-1} = M_i \oplus C_i$, while the input to next BC call is $Y_i = M_i \oplus G(X_{i-1}) \oplus c \cdot L \| 0^{n/2}$ for some constant c. By assuming that $L = 0$, we have $Y_i = M_i \oplus G(X_{i-1})$ and $Y_{i+1} = M_{i+1} \oplus G(X_i)$. For the forgery, the adversary chooses M^f such that $Y_{i+1} = M^f \oplus G(X_{i-1})$ and sets C^f to $M^f \oplus X_{i-1}$. In order for this forgery attempt to work, the assumption that $L = 0$ must be true. Since L is generated using a PRF (a truncated output of a PRP call), this assumption holds with $2^{-n/2}$. In order to have a high success probability, the $\mathcal{BD}$ adversary needs to ask for $2^{n/2}$ chosen-plaintext **Enc** queries of at least 2 blocks and try the forgery on each of them. We note that $\mathcal{BD}$ may not always succeed, depending on the size of the nonce and the design of the BC. $L = \lfloor (E_K(0^{n-t} \| N)) \rfloor_{n/2}$. If t is large enough, *e.g.* $t \approx n$, then the probability that there is at least one value of N such that $L = 0$ is high. However, if t is small, this is not guaranteed. In practice, $128 \leq t \leq 64$, while $n = 128$, so this is not an issue. In particular, GIFT-COFB uses $t = n = 128$, in which case $\mathcal{BD}$ succeeds with overwhelming probability.

The $\mathcal{OBD}$ adversary uses a slightly more complex assumption, but improves the attack complexity. $\mathcal{OBD}$ starts by asking for one chosen-plaintext query of 2 blocks (M_1, M_2), and it also starts by assuming that $3^2 \cdot L = \Delta = 0$. Hence, for the first forgery trial, $\mathcal{OBD}$ behaves like $\mathcal{BD}$. However, if the first trial fails, $\mathcal{OBD}$ changes the assumption instead of asking for a new **Enc**. It assumes that $3^2 \cdot L = 1$. In general, the assumption for each forgery attempt is $3^2 \cdot L = \Delta$, where Δ is incremented by 1 from the previous attempt. Note that $3^2 \cdot L$ is the value of the mask after processing an empty string of associated data. In the case of $\mathcal{OBD}$,

$$X_0 = M_1 \oplus C_1$$

$$Y_1 = M_1 \oplus G(X_0) \oplus 2 \cdot \Delta \| 0^{n/2}$$

$$X_1 = M_2 \oplus C_2$$

$$Y_2 = M_2 \oplus G(X_1) \oplus 2 \cdot 3 \cdot \Delta \| 0^{n/2}$$

	INT-CTXT$^{\mathcal{OBD}}$
1 :	**Initialize**
2 :	$d \leftarrow$ **False**
3 :	$M \xleftarrow{\$} \{0,1\}^{2n}$
4 :	$N \xleftarrow{\$} \{0,1\}^{t}$;$\Delta \leftarrow 0^{n/2}$
5 :	$(C,T) \leftarrow$ **Enc**(N, ϵ, M)
6 :	$(M_1, M_2) \xleftarrow{n} M$
7 :	**While** $d =$ **False**
8 :	$C^f \leftarrow 3^2 \cdot \Delta \| 0^{n/2} \oplus M_2 \oplus M_1 \oplus C_1 \oplus G(M_1 \oplus C_1) \oplus G(M_2 \oplus C_2)$
9 :	$d \leftarrow$ **Verify**(N, ϵ, C^f, T)
10 :	$\Delta \leftarrow \Delta + 1$
11 :	**Return True**

Figure 7: An optimization of the adversary in Figure 6.

Hence, $\mathcal{OBD}$ chooses M^f such that $Y_2 = M^f \oplus G(X_0) \oplus 3 \cdot \Delta \| 0^{n/2}$ and sets C^f to $M^f \oplus X_0$. Thus,

$$\begin{aligned} C^f &= M_2 \oplus G(X_1) \oplus 2 \cdot 3 \cdot \Delta \| 0^{n/2} \oplus G(X_0) \oplus X_0 \oplus 3 \cdot \Delta \| 0^{n/2} \\ &= M_2 \oplus G(M_2 \oplus C_2) \oplus G(M_1 \oplus C_1) \oplus M_1 \oplus C_1 \oplus (2 \cdot 3 \oplus 3) \cdot \Delta \| 0^{n/2} \\ &= M_2 \oplus G(M_2 \oplus C_2) \oplus G(M_1 \oplus C_1) \oplus M_1 \oplus C_1 \oplus 3^2 \cdot \Delta \| 0^{n/2} \end{aligned}$$

Unlike, $\mathcal{BD}$, $\mathcal{OBD}$ tries to guess the correct mask value, not wait for the mask to satisfy a given assumption. Hence, regardless of the nonce size, $\mathcal{OBD}$ always terminates with a successful forgery. Its success probability is 1 with probabilistic runtime. The worst case complexity is 2^{64} forgery attempts with only one encryption query. The description of both $\mathcal{BD}$ and $\mathcal{OBD}$ can be slightly modified to have a success probability of less than 1 where $\mathcal{BD}$ asks for q_d queries to **Enc** and **Verify**, while $\mathcal{OBD}$ queries **Enc** once and **Verify** q_d times. In such case, the INT-CTXT advantage of both adversaries is

$$\frac{q_d}{2^{n/2}}.$$

4.2 Mask Collision

The mask collision ($\mathcal{MC}$) adversary, depicted in Figure 8, merges two parts of different ciphertexts generated legitimately. The adversary takes two ciphertexts of two or more blocks, then merges the first part of the first ciphertext with the second part of the second ciphertext through a *"bridge"* block. Let the two queries used to perform the forgery be $(C^i, T^i) \leftarrow$ **Enc**(N^i, ϵ, M^i) and $(C^j, T^j) \leftarrow$ **Enc**(N^j, ϵ, M^j), where M^i and M^j consist of two blocks, each. For the i^{th} query, we have

$$X_1^i = M_2^i \oplus C_2^i$$

and

$$Y_2^i = G(M_2^i \oplus C_2^i) \oplus M_2^i \oplus 2 \cdot 3 \cdot \Delta^i$$

where $\Delta^i = 3^2 \cdot L^i$. $\mathcal{MC}$ assumes that $L^i = L^j$, and defines the forged ciphertext as

$$C^f = C_1^i \| C_x$$

where

$$Y_2^j = G(M_2^i \oplus C_2^i) \oplus M_x \oplus 2 \cdot 3 \cdot \Delta^i = G(M_2^j \oplus C_2^j) \oplus M_2^j \oplus 2 \cdot 3 \cdot \Delta^j,$$

INT-CTXT$^{\mathcal{MC}}$

Initialize
$\mathcal{N} \leftarrow \phi$
For i in $1 \cdots 2^{n/4}$
  $N^i \xleftarrow{\$} \{0,1\}^t \setminus \mathcal{N}$
  $\mathcal{N} \leftarrow \mathcal{N} \cup \{N^i\}$
  $M^i \xleftarrow{\$} \{0,1\}^{2n}$
  $(C^i, T^i) \leftarrow \mathbf{Enc}(N^i, \epsilon, M^i)$
For i in $1 \cdots 2^{n/4}$
  For j in $i+1 \cdots 2^{n/4}$
    $(M_1^i, M_2^i) \xleftarrow{n} M^i$
    $(M_1^j, M_2^j) \xleftarrow{n} M^j$
    $(C_1^i, C_2^i) \xleftarrow{n} C^i$
    $(C_1^j, C_2^j) \xleftarrow{n} C^j$
    $C^f \leftarrow C_1^i \| M_2^i \oplus C_2^i \oplus G(M_2^i \oplus C_2^i) \oplus G(M_2^j \oplus C_2^j) \oplus M_2^j$
    $d \leftarrow \mathbf{Verify}(N^i, \epsilon, C^f, T^j)$
    If $d =$ **True then Return** d
return d

Figure 8: The Mask-Collision INT-CTXT Adversary.

$$M_x = G(M_2^i \oplus C_2^i) \oplus G(M_2^j \oplus C_2^j) \oplus M_2^j$$

and

$$C_x = X_1^i \oplus M_x = M_2^i \oplus C_2^i \oplus G(M_2^i \oplus C_2^i) \oplus G(M_2^j \oplus C_2^j) \oplus M_2^j$$

Essentially, $\mathcal{MC}$ tries to transform the internal state during the forgery attempt from the internal state of the i^{th} **Enc** query, to that of the j^{th} **Enc** query. This works if there is a collision between the masks of the i^{th} and j^{th} queries, which is a birthday collision. Since it is a collision over $n/2$ random bits, the collision occurs with high probability when the number of calls to **Enc**, q_e is more than $2^{n/4}$. However, in order to detect the collision, a successful forgery must occur. The number of potential forgeries is $q_d = \binom{q_e}{2}$. When $q_e = 2^{n/4}$, $q_d = 2^{n/2+1} - 2^{n/4}$. We note that when the collision occurs, the verification step of the attack always succeeds with probability 1. Hence, the success probability of the attack after q_e and q_d encryption and decryption queries is bounded by the following inequality.

$$\Pr[\text{INT-CTXT}^{\mathcal{MC}}(q_e, \sigma_e, q_d, \sigma_d, t)] \leq \frac{\binom{q_e}{2} q_d}{2^n}$$

For the parameters in Figure 8

$$\Pr[\text{INT-CTXT}^{\mathcal{MC}}(2^{n/4}, \sigma_e, 2^{n/2}, \sigma_d, t)] \leq \frac{\frac{2^{n/2}-2^{n/4}}{2} \times 2^{n/2}}{2^n}$$

$$= \frac{2^{n/2} - 2^{n/4}}{2^{n/2+1}} = 0.5 - 2^{-n/4-1}.$$

5 Chosen-Ciphertext Distinguishing Attack against COFB

The INT-CTXT adversaries presented in Section 4 rely on a simple observation: the chosen plaintext queries give the adversary a somewhat restricted access to the inputs and outputs of the underlying block cipher. In particular, the equation $X_{i-1} = M_i \oplus C_i$ reveals the output of the $(i-1)^{th}$ BC call, while the equation $Y_i = G(M_i \oplus C_i) \oplus M_i \oplus c \cdot L \| 0^{n/2}$,

for some constant c, reveals half of the input to the i^{th} BC call. However, the way the adversaries are constructed relies on satisfying an assumption on the mask L corresponding a nonce N that has been already queried during a query **Enc**. In the cases of $\mathcal{BD}$ and $\mathcal{MC}$ the assumption results in a non-negligible number of queries to **Enc**, with $q_e = 2^{n/2}$ for $\mathcal{BD}$ and $\sim 2^{n/4}$ for $\mathcal{MC}$. $\mathcal{OBD}$ overcomes this limitation by relaxing the assumption on L, where L can take any value. However, it still relies on forging using a nonce that has been queried before. In Figure 9, we present an IND$-CCA adversary $\mathcal{L}$ requires 1 query to **Enc**, and

$$\mathbf{Adv}^{\mathtt{ind\$-cca}}_{\mathtt{aead}}(\mathcal{L}) = \frac{2^t - 1}{2^t} \times \frac{1}{2^{n-t}} \times \frac{q_d}{2^{n/2}} - \frac{1}{2^n}$$

	IND$-CCA$^{\mathcal{L}}$
1 :	**Initialize**
2 :	$N \xleftarrow{\$} \{0,1\}^t$
3 :	$M \xleftarrow{\$} \{0,1\}^n$
4 :	$(C,T) \leftarrow \mathbf{Enc}(N, \epsilon, M)$
5 :	$M^f \leftarrow \perp$
6 :	$L \leftarrow 0^{n/2}$
7 :	**While** $M^f = \perp$
8 :	$V \leftarrow T$
9 :	$P \leftarrow G(M \oplus C) \oplus M \oplus 2 \cdot 3^2 \cdot L \| 0^{n/2}$
10 :	$L^f \leftarrow \lfloor V \rfloor_{n/2}$
11 :	$0^{n-t} \| N^f \leftarrow P \& 0^{n-t} \| 1^t$
12 :	$T^f \leftarrow V$
13 :	$A^f \leftarrow P \oplus 3 \cdot L^f \| 0^{n/2}$
14 :	$C^f \leftarrow P \oplus 3^2 \cdot L^f \| 0^{n/2} \oplus G(V) \oplus V$
15 :	$M^f \leftarrow \mathbf{Dec}(N^f, A^f, C^f, T^f)$
16 :	$L \leftarrow L + 1$
17 :	$M^r \xleftarrow{\$} \{0,1\}^n$
18 :	$M^0 \leftarrow M^f \| M^r$
19 :	$(C,T) \leftarrow \mathbf{Enc}(N^f, A^f, M^0)$
20 :	**If** $\lfloor C \rfloor_n = C^f$ **then**
21 :	**Return Finalize**(0)
22 :	**Else then**
23 :	**Return Finalize**(1)

Figure 9: The $\mathcal{L}$ IND$-CCA against COFB.

The idea of $\mathcal{L}$ is to guess one input-output pair of the underlying BC. Given the equations governing the internal state of the COFB scheme, $\mathcal{L}$ only needs to guess $n/2$ bits, to obtain the pair (P, V) such that $V = E_K(P)$. $\mathcal{L}$ then assumes the forgery nonce $0^{n-t} \| N^f = P$. Based on this assumption, $L^f = \lfloor E_K(N^f) \rfloor_{n/2}$. $\mathcal{L}$ constructs an associated data block and a ciphertext block such that the input and output to every BC call during the forgery attempt are always P and V, respectively. If the forgery is successful, $\mathcal{L}$ receives M^f as the single-block decrypted plaintext corresponding to C^f. If the forgery is unsuccessful, the adversary guesses a different value for L. Once the forgery is successful, $\mathcal{L}$ knows the state of the COFB scheme after absorbing the nonce N^f and associated data block A^f. As longs as N^f has never appeared in any query to **Enc**, it can be used in a subsequent query while maintaining that $\mathcal{L}$ is nonce-respecting. To sum up, in order for the attack to succeed, three conditions need to be satisfied:

1. $\mathcal{L}$ must guess the correct mask value L of the first encryption query. The probability of satisfying this condition for a given forgery is $2^{-n/2}$

2. For the correct guess of L, it must be possible to use P as a nonce. In particular, P must be of the form $0^{n-t}\|N^f$. The probability of satisfying this condition is 2^{t-n}.

3. N^f must have not been used before as a nonce during any encryption query. $\mathcal{L}$, as defined in Figure 9, makes only a single query to **Enc**. Hence, the probability of that the nonce N^f is fresh is $1 - 2^{-t} \sim 1$.

Since $\mathcal{L}$ tries all possible values of L, the first condition will be eventually satisfied with at most $2^{n/2}$ calls to **Dec**. The second and third conditions, however, may not necessarily be satisfied in Figure 9. In order to study their effect on the complexity, we separate the analysis into two cases: when $n = t$ and when $n > t$.

5.1 Full-block nonce

When the nonce size $t = n$, then the attack becomes simpler. $\mathcal{L}$ succeeds with overwhelming probability. Once the correct value of L is guessed, the corresponding value P can always be used as N^f. Moreover, the probability that $N^f \neq N$ is $1 - 2^{-n}$. This leads to the advantage

$$\mathbf{Adv}^{\mathtt{ind\$-cca}}_{\mathtt{aead}}(\mathcal{L}) \approx \frac{q_d}{2^{n/2}}$$

While this case is the simplest to analyze, it is relevant in practice as the choice $n = t$ is common when $n = 128$ and is, indeed, the case for GIFT-COFB, the NIST lightweight cryptography finalist that uses the COFB scheme with GIFT as a BC. Hence, it can be used to conclude that the advantage IND$-CCA adversaries against GIFT-COFB is at least $\approx q_d/2^{n/2}$.

5.2 Partial-block nonce

In case the nonce size $t < n$, then $\mathcal{L}$ needs to make sure that P can be used as a nonce. The way nonce is defined in [CIMN20] requires $P = 0^{n-t}\|N^f$. At the same time, we know that P is of the form

$$G(M_i \oplus C_i) \oplus M_i \oplus c \cdot L \| 0^{n/2}$$

where c is a small constant. The only unknown variable in this expression is L. The conditions needed to satisfy $P = 0^{n-t}\|N^f$ apply to the leftmost $n - t$ bits. Assuming that the internal states at different points of execution of the first call to **Enc** are uniformly distributed, based on the properties of the BC and the random choice of the chosen plaintext, then one way to increase the probability of satisfying these conditions is increasing the number of blocks in the query to **Enc**, or increasing the number of queries to **Enc**. The maximum length allowed by COFB [BCI+19] for the plaintext is $\sim 2^{51}$ blocks. If the nonce length is between 128 bits and 78 bits, we perform the same attack but the initial encryption is performed using a message of 2^{n-t} blocks, where for each guess of L, the adversary will find one candidate for P with high probability. If the nonce is shorter than 78 bits, we need to construct a more complex adversary.

Applicability to HyENA HyENA [CDJN19] was a round 2 candidate of the NIST lightweight cryptography standardization project and it holds a lot of resemblance to the COFB mode. The main differences are:

1. It uses a different feedback function known as hybrid feedback.

2. Before calling the BC for the last time the internal state is split in two halves and swapped.
3. The nonce is limited to $3n/4$ bits concatenated with an $(n/4)$-bit constant during the first call to the BC.

These differences mean that applying the attack to HyENA requires that initial encryption query is of length $2^{n/4}$ and the way to construct the messages is different, but can still be performed.

Potential Remedy Besides the nonce size, the IND$-CCA attack presented requires the ability to predict both the internal state and mask corresponding to a nonce N, where a successful guess of an $n/2$ bit value leaks both. Currently, one PRF is used to generate both the mask and the initial internal state. In order prevent this, we can use two different PRF constructions to generate each value. For example, we can use $L = \text{truncate}(E_K(N))$, which is the same as the current situation, while the initial state (the state XORed with the first associated data block) can be $E_K(E_K(N))$, *i.e.* adding an extra BC call after the mask generation. However, while this may prevent the presented IND$-CCA attack, it will not affect the security bounds and may introduce additional problems. Hence, this issue requires an independent study, outside the scope of this paper.

Effective Tag Size of GIFT-COFB The second phase of the attack presented in Section 5 holds some resemblance to the attack described in Section 3 [Mè19]. They mainly differ in two points:

1. The attack in [Mè19] targets algorithms with short tags. It works with $2^{n/2}$ forgery attempts against algorithms with $n/2$-bit tags. Our attack complexity is a function of the mask size rather than the tag size. Given an $n/2$-bit mask, the attack works with $2^{n/2}$ forgery attempts, even if the tag size is larger than $n/2$ bits.
2. The attack in [Mè19] requires only decryption queries, and only one encryption query. Our attack requires one additional encryption query at the beginning. However, this encryption query may consist of one block (if the nonce size is large enough) and the plaintext can only be known, not necessarily chosen. In practice, this limitation is very mild and the adversary can achieve it in many cases.

Given these observations, it seems that the tag size of GIFT-COFB offers little immunity compared to algorithms with half the tag size, and by keeping the tag size n instead of truncating it to $n/2$ (or a value in between) seems to offer very minimal security advantage.

Setting the tag size for COFB instances. The attacks presented in this article show that the attack complexity against COFB is mainly dominated by the mask size*, i.e.,* an instance of COFB with mask size ζ can be attacked for both privacy and integrity with advantage of the form

$$\frac{q_d}{2^\zeta}$$

Besides, the existence of attacks that are bounded by the birthday bound for the number of encryption queries is well-established for privacy through attacks on the PRP-PRF switching for privacy and through the work of Inoue and Minematsu [IM21] which shows forgery attacks with advantage

$$\frac{\sigma_e^2}{2^n}$$

With the accumulation of these results, we find it reasonable to set the tag size $\tau = \zeta$. For most applications, setting $\tau = n/2$ will offer almost no security degradation compared to the current instance of GIFT-COFB.

Generalization to many forgeries The main goal of this paper is to challenge the limits of the security of COFB and its main real-world instance GIFT-COFB. However, the attack can be enhanced to allow many forgeries from a single plaintext query. If the **Enc** query of $\mathcal{L}$ includes m plaintext blocks, then after successfully guessing L, *i.e.*, after the first successful forgery, the adversary has m choices for N^f and can forge many blocks of A^f and C^f. Even with the single block encryption query, the adversary has only a single choice for N^f, but can use it with different lengths of A^f and M^f. In this respect, COFB is even worse than a generic scheme with $n/2$-bit tag, as such scheme allows one forgery per $2^{n/2}$ attempts, while COFB allows many forgeries once the $2^{n/2}$-attempt threshold is met.

6 Discussion of the Security Proofs of COFB

As discussed in Section 1, there are three security proofs in the literature that cover COFB. The attacks proposed in this paper do not contradict the proofs in [BCI+20] and [CIMN17], whose bounds are dominated by the term $nq_d/2^{n/2}$. However, it does contradict Theorem 2 in [CIMN20]. This is due to an error in calculating the probability of a collision between the internal state values in encryption and decryption queries. Let Y_i^j be the input to the BC at block i in the encryption query j and $Y_{i'}^{j'}$ be the input to the BC at block i' in the decryption query j'. The proof of Theorem 2 of [CIMN20] bounds the probability of a collision of the form $Y_i^j = Y_{i'}^{j'}$ by

$$\Pr[\exists i, j, i', j',\ s.t.\ Y_i^j = Y_{i'}^{j'}] \leq \frac{(q_e + \sigma_e)\sigma_f}{2^n}$$

However, this assumes that the internal states are completely random. In reality, the adversary has a lot of control over $n/2$ bits of the state. For example, the adversary can force a collision on half the state by forcing half the input of the BC during a decryption query to a value that has been observed during encryption. However, once the adversary makes such decision, the probability of a full collision becomes $2^{-n/2}$. The adversary can keep changing the masked half of the state during decryption until the guess is correct. This observation is confirmed by the attacks presented in this article. It has been communicated to the authors of [CIMN20] and has been verified by them.

Due to the aforementioned oversight, we focus the rest of the of discussion on the security proof of GIFT-COFB [BCI+20]. The authors give the following bound for COFB-R:

$$\mathbf{Adv}^{\mathsf{aead}}_{\mathsf{COFB-R}}(q_e, \sigma_e, q_d, \sigma_d, t) \leq \frac{1}{2^{n/2}} + \frac{(n+4)q_d}{2^{n/2+1}} + \frac{q_d + \sigma_e^2 + (q_e + \sigma_e + \sigma_d)\sigma_e}{2^n}$$

which is dominated by the term $0.5nq_d/2^{n/2}$. This indicates a logarithmic gap from all the best known attacks, which have an advantage $O(q_d/2^{n/2})$. However, we argue that this bound is an artefact of the proof technique used. In other words, it is impossible to find an adversary with advantage $O(nq_d/2^{n/2})$. First, we look at the security proof in [BCI+20]. The term $nq_d/2^{n/2}$ is the upper-bound on the probability of an event that is considered a win for the adversary. This event is defined as follows:

- Let the forgery attempt f share the first p_i block cipher calls with the encryption query i, with $N^f = N^i$. In other words, $p_i + 1$ includes the first difference in the associated data and/or ciphertext between the forgery attempt f and the encryption query i, such that $Y^f_{p_i+1} \neq Y^i_j$ We set $p_i = -1$ if no such encryption query exists. The bad event **B4** is defined as

  $$\exists (s, i_1, j)\ s.t.\ Y^f_{p_i+1} = Y^{i_1}_j,$$

  where $j \neq 0$.

Table 2: Expected values of σ_e and q_f according to different values of $\mathsf{mcoll}_{\mathsf{R}}(\mathbf{Y})$ when $n = 128$.

$\mathsf{mcoll}(\mathbf{Y}_R)$	σ_e	q_f
1	1	2^{64}
2	$2^{32.5}$	2^{63}
4	$2^{49.15}$	2^{62}
8	$2^{57.91}$	2^{61}
16	$2^{62.77}$	2^{60}
21	$2^{64.07}$	$2^{59.61}$
32	$2^{65.68}$	2^{59}
64	$2^{67.59}$	2^{58}

In order to bound the probability of such event, we consider an adversary that can control the unmasked half of the internal state, and expects a collision on the masked half with one or more of the internal states during encryption queries. As such, Pr[**B4**] is maximized when the collision targets a value $Y_j^{i_1}$ whose unmasked half is part of the multi-collision with the largest size $\mathsf{mcoll}(\mathbf{Y}_R)$, where $\mathbf{Y}$ is the set of all the inputs to the BC during encryption. In other words,

$$\Pr[\mathbf{B4}] \leq \frac{\mathsf{mcoll}(\mathbf{Y}_R)q_d}{2^{n/2}}.$$

We note that this bound is not independent of the number and complexity of encryption queries, as the expected value of $\mathsf{mcoll}(\mathbf{Y}_R)$ depends on the amount of encrypted data. For example, when $\sigma_e \approx 2^{n/4}$, the expected value of $\mathsf{mcoll}(\mathbf{Y}_R) \approx 2$, while at $\sigma_e \approx 2^{64}$, the expected value is 21. Table 2 gives expected values of σ_e required to get a multi-collision of a certain size and the corresponding q_d.

In [BCI+20], the authors bound $\mathsf{mcoll}(\mathbf{Y}_R) < n$, which leads to the simplified term $0.5nq_d/2^{n/2}$. However, this simplification hides a non-negligible dependence on the encryption complexity. Given these observations on how the bound is constructed, we conjecture that the logarithmic gap between the security bound and the best known attacks is an artefact of the security proof and that the term dependent on the number of decryption queries/forgery attempts in the security bound of COFB can be improved to $q_d/2^n$, but we leave proving such bound to future work.

7 Related Work

After the attacks presented in this paper were initially published, other researchers have also studied the security proofs of COFB. Inoue and Minematsu [IM21] showed a forgery attack that is dominated by encryption queries. It requires about $2^{n/2}$ encryption queries and one forgery attempt. Later, Inoue *et. al.* [IIM22] showed an IND$-CPA attack that requires $2^{n/2}$ encryption queries that works when $n = t$. These results are independent of the results presented here. These works can be seen as complementary to this paper, investigating different attack strategies on COFB.

8 Conclusions

In this article, we have analyzed the COFB algorithm showing that it is secure against IND$-CCA adversaries at most up to $2^{n/2}$ forgery attempts. We presented new forgery attacks and the first IND$-CCA attack against COFB with negligible encryption complexity

and $2^{n/2}$ forgery attempts. We show that COFB behaves in a similar manner to a generic AEAD scheme with $n/2$-bit tag, and in some scenarios even worse. As a byproduct, we have identified an oversight in [CIMN20]. However, we emphasize that the attacks do not threaten [BCI+20], [CIMN17] or the security of GIFT-COFB according to the requirements of the NIST lightweight cryptography standardization project.

Acknowledgments

The author would like to thank the reviewers of ToSC for their insightful comments and feedback. The author would also like to thank the designers of GIFT-COFB and the authors of [CIMN20] for their comments on an earlier draft. This work was funded under the MALEC project, Temasek Laboratories NTU grant DSOCL17101.

References

[BCI+19] Subhadeep Banik, Avik Chakraborti, Tetsu Iwata, Kazuhiko Minematsu, Mridul Nandi, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Yosuke Todo. GIFT-COFB. NIST Lightweight Cryptography Project, 2019. https://csrc.nist.gov/Projects/Lightweight-Cryptography/Round-1-Candidates.

[BCI+20] Subhadeep Banik, Avik Chakraborti, Tetsu Iwata, Kazuhiko Minematsu, Mridul Nandi, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Yosuke Todo. GIFT-COFB. Cryptology ePrint Archive, Report 2020/738, 2020. https://eprint.iacr.org/2020/738.

[BN00] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In Tatsuaki Okamoto, editor, *Advances in Cryptology — ASIACRYPT 2000*, pages 531–545, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[BR06] Mihir Bellare and Phillip Rogaway. The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006*, pages 409–426, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[CDJN19] Avik Chakraborti, Nilanjan Datta, Ashwin Jha, and Mridul Nandi. HyENA. NIST Lightweight Cryptography Project, 2019. https://csrc.nist.gov/Projects/Lightweight-Cryptography/Round-1-Candidates.

[CIMN17] Avik Chakraborti, Tetsu Iwata, Kazuhiko Minematsu, and Mridul Nandi. Blockcipher-based authenticated encryption: How small can we go? In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 277–298. Springer, 2017. https://link.springer.com/chapter/10.1007/978-3-319-66787-4_14.

[CIMN20] Avik Chakraborti, Tetsu Iwata, Kazuhiko Minematsu, and Mridul Nandi. Blockcipher-based authenticated encryption: How small can we go? In *Journal of Cryptology*, pages 703–741. Springer, 2020. https://link.springer.com/article/10.1007/s00145-019-09325-z.

[IIM22] Akiko Inoue, Tetsu Iwata, and Kazuhiko Minematsu. Analyzing the Provable Security Bounds of GIFT-COFB and Photon-Beetle. Cryptology ePrint Archive, Report 2022/001, 2022. https://ia.cr/2022/001.

[IM21] Akiko Inoue and Kazuhiko Minematsu. GIFT-COFB is Tightly Birthday Secure with Encryption Queries. Cryptology ePrint Archive, Report 2021/737, 2021. https://ia.cr/2021/737.

[Kha20] Mustafa Khairallah. Weak Keys in the Rekeying Paradigm: Application to COMET and mixFeed. *IACR Transactions on Symmetric Cryptology*, 2019(4):272–289, Jan. 2020.

[MV04] David McGrew and John Viega. The galois/counter mode of operation (gcm). *submission to NIST Modes of Operation Process*, 20:0278–0070, 2004.

[Mè19] Alexandre Mège, Nov 2019. https://groups.google.com/a/list.nist.gov/g/lwc-forum/c/2a0H-HQHgqU/m/EtjdRFSmBQAJ.

[nis18] Submission Requirements and Evaluation Criteria for the Lightweight Cryptography Standardization Process, 2018. https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/final-lwc-submission-requirements-august2018.pdf.

[Rog02] Phillip Rogaway. Authenticated-Encryption with Associated-Data. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, CCS '02, page 98–107, New York, NY, USA, 2002. Association for Computing Machinery.