OAE-RUP: A Strong Online AEAD Security Notion and its Application to SAEF*

Amit Singh Bhati[1], Elena Andreeva[2], and Damian Vizár[3]

[1] COSIC, KU Leuven, Belgium
[2] Technical University of Vienna, Austria
[3] CSEM, Switzerland.
amitsingh.bhati@esat.kuleuven.be, elena.andreeva@tuwien.ac.at, damian.vizar@csem.ch

Abstract. Release of unverified plaintexts (RUP) security is an important target for robustness in AE schemes. It is also highly crucial for lightweight (LW) implementations of online AE schemes on memory-constrained devices. Surprisingly, very few online AEAD schemes come with provable guarantees against RUP integrity and not one with any well-defined RUP confidentiality.

In this work, we first propose a new strong security notion for online AE schemes called OAE-RUP that captures security under blockwise processing of both encryption (which includes nonce-misuse) and decryption (which includes RUP). Formally, OAE-RUP combines the standard RUP integrity notion INT-RUP with a new RUP confidentiality notion sOPRPF (strong Online PseudoRandom Permutation followed by a pseudorandom Function). sOPRPF is based on the concept of "strong online permutations" and can be seen as an extension of the well-known CCA3 notion (Abed et al., FSE 2014) that captures arbitrary-length inputs.

An OAE-RUP-secure scheme is resistant against nonce-misuse as well as leakage of unverified plaintexts where the integrity remains unaffected, and the confidentiality of any encrypted plaintext is preserved up to the leakage of the longest prefix with the leaked plaintexts and the leakage of the length of the longest prefix with the nonce-repeating ciphertexts.

We then prove the OAE-RUP security of the SAEF mode. SAEF is a ForkAE mode (Asiacrypt 2019) that is optimized for authenticated encryption of short messages and processes the message blocks sequentially and in an *online* manner. At SAC 2020, it was shown that SAEF is also an *online nonce misuse-resistant* AE (OAE), offering enhanced security against adversaries that make blockwise adaptive encryption queries. It has remained an open question if SAEF also resists attacks against blockwise adaptive decryption adversaries or, more generally, when the decrypted plaintext is released before verification (RUP).

Our proofs are conducted using the coefficients H technique, and they show that, without any modifications, SAEF is OAE-RUP secure up to the birthday bound, i.e., up to $2^{n/2}$ processed data blocks, where n is the block size of the forkcipher.

* This is the full version of the original work published in SCN 2024.

Keywords: Authenticated encryption · forkcipher · lightweight cryptography · provable security · online · release of unverified plaintext · OAE-RUP

1 Introduction

Authenticated Encryption. An authenticated encryption (AE) scheme provides both confidentiality and authenticity for messages. Most AE schemes today take two additional inputs besides the plaintext data: an associated data AD and a nonce N. The associated data is some information, such as a packet header, that is sent in the clear but needs to be authenticated, while the nonce is a unique value that helps to avoid the need for either keeping a state or using a random value. The formalization of nonce-based authenticated encryption with associated data (AEAD) was introduced by Rogaway in 2002 [40].

The design and analysis of AEADs have recently received a lot of attention from the research community, mainly motivated by the past CAESAR competition (2014–2018) [14], the recent NIST lightweight cryptography standardization process (2018–2023) [36] and the new NIST call for Accordion Cipher (2023–) [37]. In all these, robust or strong security for AEADs has been a clear target.

Nonce-misuse resistance and security against release of unverified plaintext (RUP) are the two main notions that are recognized for strongly secure practical AEAD schemes. While nonce-misuse deals with protection against nonce repetitions, RUP ensures a limited damage when unverified decryption is leaked. More precisely, integrity preservation despite RUP was considered as *critical* for defence in depth security [14], whereas some limited RUP confidentiality damage was still acceptable.

A side advantage of RUP-secure schemes is their ability to serve as an extra security layer against implementation flaws. When an implementation fail to verify a tag for a given ciphertext and leaks the unverified plaintext, RUP confidentiality ensures that if the ciphertext is not valid i.e., it has been tampered during transmission, the resulting decrypted plaintext will resemble meaningless gibberish (with limited information leakage about the original plaintext) and hence should be detectable.

Release of Unverified Plaintext (RUP). RUP security captures the scenario where applications may require decrypting and releasing the data before checking its authenticity due to memory limitations, real-time requirements, or other factors. This exposes them to potential attacks that exploit the unverified data to break the confidentiality or integrity of the scheme. Video/audio media streaming, voice-over IP and disk encryption are some examples of applications with real-time requirements where the data may need to be released before the authentication tag is verified to improve the quality or speed of the communication.

RUP security is more important for designing and analyzing AE schemes for lightweight applications as they may have to trade-off some security for efficiency or functionality. Lightweight applications are common in resource-constrained devices, such as IoT sensors, smart cards, RFID tags, etc. These devices often use lightweight AE schemes designed to be efficient and simple but still secure in situations where they have to release decrypted data before verifying its authenticity due to constraints such as low latency, long messages, high speed, etc.

The significance of RUP security extends beyond scenarios where plaintext is directly exposed without verification. Even in systems where direct disclosure is prohibited, attackers can exploit side channels to deduce properties of the plaintext, as evidenced by the real-world attacks [2, 3, 17, 42].

Therefore, it is essential to have formal models and proofs of RUP security that capture this setting and practical schemes that achieve RUP security without sacrificing performance or simplicity. The AEAD syntax and formal RUP security definitions were introduced by Andreeva et al. [5] in Asiacrypt 2014. First, they defined the security notion of plaintext awareness PA in two variants, PA1 and PA2, and then proposed to combine PA1 with IND-CPA to achieve confidentiality of an AE(AD) scheme. To achieve integrity of ciphertexts under RUP, they used INT-CTXT in the RUP setting, also known as the INT-RUP notion.

In the same work of Asiacrypt 2014, it has also been concluded that AE schemes with bijective encryption (excluding the tag) cannot be PA1 secure, which makes PA1 a quite strong security notion. There are only a few AE(AD) schemes that can achieve PA1 security, such as SIV [41], BTM [33], and HBS [34]. Almost all of these PA1 secure schemes require two passes over the plaintext and thus are offline. This conflicts with one of the main goals of achieving online (a.k.a. one-pass or blockwise processing) property under the RUP setting.

RUP security complements the conventional security models that assume the data is always verified before release. It also poses new challenges and open problems for researchers and practitioners of cryptography, as many existing schemes are shown insecure [21, 25, 31] or inefficient [8, 28, 44] in the RUP setting.

RUP Security of Online AEs. Online, a.k.a. blockwise processing encryption and decryption, are considered important properties of an AE scheme for lightweight applications where it is important to encrypt/decrypt the ciphertext blocks quickly with constant latency (e.g., real-time streaming protocols or optical transport networks (OTNs)), or where a constant memory implementation is needed. These applications are common in consumer-grade IoT devices, which have tight cost constraints and often make the devices leak parts of the unverified plaintext when decrypting long messages. Hence, such applications would benefit from a lightweight AEAD scheme with RUP security results.

In 2015, Abed et al. [1] considered an AE variation of the OPRP-CCA [11] encryption security notion called CCA3 [26] as a weaker alternative to PA1. CCA3 claims to achieve confidentiality of online AE schemes (but accepts only block-aligned plaintexts, i.e., no support for incomplete final plaintext blocks) in the RUP setting where the nonce can be reused, and leakage of the common prefix is acceptable. CCA3 is a weaker notion than IND-CPA+PA1, which only applies to block-aligned online AE schemes and guarantees confidentiality up to the leakage of the longest prefix. The state of the art does not answer the following important question

How to define RUP confidentiality security together with nonce-misuse resistance for online AEAD schemes that process arbitrary length inputs?

The most common and simple approach for designing an online AE is the feedback or CBC-style approach. Feedback was also one of the popular design approaches in the NIST lightweight competition and was used in 8 out of 32 2^{nd} round candidates, namely GIFT-COFB [9], HyENA [20], COMET [27], mixFeed [22], Romulus-N [32], TinyJAMBU [43], SAEB [35] and SAEF [7]. Feedback-approach-based modes are very useful for applications with stringent constraints on memory and hardware due to their small state size, making them perfect targets for the RUP security analysis.

In [21], Chakraborti et al. showed a general impossibility result that any blockcipher-based feedback AEAD mode with rate 1 is not INT-RUP secure, which implies that rate-1 blockcipher modes GIFT-COFB, HyENA are not INT-RUP secure. This leaves us to focus on feedback AEAD modes based on other primitives or on blockcipher but with a rate lower than 1. Recently, in [24], Dutta et al. studied the INT-RUP security of blockcipher based feedback AEAD modes SAEB and TinyJAMBU and showed that SAEB in its current form is not INT-RUP secure whereas TinyJAMBU (with rate 1/4 in its input size) is INT-RUP secure.

SAEF ForkAE Mode. SAEF [6,7] is a forkcipher-based sequential and online nonce-based AEAD mode optimized for processing short messages and, therefore, suitable for lightweight applications where the predominant message size is just a few blocks. SAEF was originally proposed in Asiacrypt 2019 as part of the ForkAE family of forkcipher modes, which was also a second-round candidate of the NIST lightweight competition. SAEF has been shown to achieve confidentiality and authenticity against nonce-respecting adversaries up to the birthday bound in [7]. Moreover, recently, in [4], SAEF was proven to be secure when the nonces are repeated up to leakage of common plaintext prefix lengths under an extended version of OAE [26] a.k.a. OAE1 in [30]. As shown and mentioned in [15,30], OAE1 secure schemes are prone to CPSS (chosen prefix secret suffix) attacks, however, their level of confidentiality guarantee can be sufficient in many applications, given that the higher-level security layer is carefully designed to avoid CPSS attacks. A consequence of SAEF's OAE1 security result is that SAEF resists attacks by blockwise adaptive adversaries in encryption and hence is suitable for lightweight encryption with low latency and low memory requirements. The latter results prove the *defence in depth* resistance of SAEF against nonce respecting and nonce repeating adversaries.

Despite its robustness features, it is not known if the SAEF mode resist attacks by blockwise adaptive nonce-misusing adversaries in decryption and thus if it is also suited for lightweight decryption with more stringent low latency and low memory requirements.

We note that the RUP security investigation of SAEF was also mentioned as one of the open problems in [4].

Our Contributions. Our contribution to this work is two-fold.

1. RUP confidentiality notion for online AEs: We note that a revised online AE confidentiality notion of CCA3 (named OAE) is defined in [26] to handle plaintexts whose lengths are a multiple of the underlying blocksize n. Hence, we first extend the formalism to arbitrary size messages to deal with messages that are not necessarily n-bit aligned and then adapt the CCA3 notion accordingly into sOPRPF (short for ***s****trong* ***O****nline* ***P****seudo****R****andom* ***P****ermutation followed by a pseudorandom* ***F****unction*). Our sOPRPF can also be seen as a natural extension of the OPRPF [4] notion from chosen plaintext attacks to chosen ciphertext attacks. Informally, sOPRPF security provides confidentiality of plaintexts (up to the leakage of the longest common prefix) against nonce-misusing adversaries that also observe unverified plaintexts.
   We note that just OPRPF security is not sufficient here for the confidentiality of plaintexts under decryption leakage as it does not capture chosen ciphertext attacks, and the stronger IND-CPA+PA1 security has been shown unachievable in [5] for any "online permutation" based online AE that allows nonce repetitions. This makes sOPRPF the best choice available for online AEs.
   We then define a joint notion of RUP security for online AE schemes called OAE-RUP as sOPRPF+INT-RUP and positively answer the first open question highlighted above. OAE-RUP is a stronger notion that also implies OAE [4] security. In simple words, an OAE-RUP-secure scheme is resistant against nonce-misuse as well as leakage of unverified plaintexts where the integrity remains unaffected, and the confidentiality of any encrypted plaintext is preserved up to the leakage of the longest prefix with the leaked plaintexts and the leakage of the length of the longest prefix with the nonce-repeating ciphertexts.
2. As our next contribution, we investigate the OAE-RUP security of SAEF and positively answer the second open question highlighted above (and in [4]). More concretely, we show that the SAEF mode is provably OAE-RUP secure without requiring design modifications. The integrity of SAEF under RUP remains intact, whereas the confidentiality is degraded but preserved up to the leakage of the longest common prefix. We use coefficients H technique [38] as the main tool for the analysis and prove that SAEF is OAE-RUP secure up to $2^{n/2}$ blocks of processed data in total, where n is the block size of the underlying forkcipher.

We highlight that syntax-wise, sOPRPF may not apply to all types of secure online AE schemes. Therefore, we also define a weaker yet generalized version of RUP confidentiality that is simple, intuitive, and applicable to all types of online AE schemes called *confidentiality resilience under RUP* (CR-RUP) in Appendix B. A CR-RUP-secure scheme preserves confidentiality (against nonce-respecting adversaries) of messages that have no common prefix with any released unverified plaintext. CR-RUP can be seen analogous to the nonce misuse resilience [8] (NMR) notion of security. Being weaker, it still captures a meaningful level of security. It says nothing about the security of plaintexts that are directly subject to leakage but will imply that plaintexts that are not subject to leakage and, with no common prefix to leaked plaintexts and unique nonces, are fully secure.

We also study the relationships and differences among popular AEAD notions to compare them with OAE-RUP and to argue its importance. We provide our detailed comparison analysis in Section 3.

A study of CR-RUP and sOPRPF security of other existing INT-RUP secure online AE schemes can be seen as an interesting problem for future research. Taking the new security results of this work into account, we now summarize the current provable security results of NIST LW candidates.

Table 1: Comparison of SAEF with NIST LW submissions with beyond nAE security claims. Here, white colored properties in the first column are unachievable by any online AE scheme.

	ESTATE [19], Romulus-M [32]	Spook [13]	Oribatida [16], LOCUS,LOTUS [18]	TinyJAMBU [43]	SAEF [4, 7, This work]
One-pass Encryption	✗	✓	✓	✓	✓
NMR [8]	✓	✓	✗	✓	✓
OAE [4]	✗	✗	✗	✗	✓
MRAE [41]	✓	✗	✗	✗	✗
One-pass Decryption	✓	✓	✓	✓	✓
INT-RUP [5]	✓	✗	✓	✓	✓
sOPRPF [This work]	✗	✗	✗	✗	✓
IND-CPA+PA1 [5]	✓	✗	✗	✗	✗

Security comparison of SAEF with other NIST LW Candidates. Among the 32 AE family candidates in the second round of the NIST lightweight competition, only 8 AE modes (including SAEF) come with claims above the conventional nAE security. We compare these modes concerning security properties beyond nAE, a.k.a. *defence in depth*, in Table 1. We provide a detailed explanation of Table 1 with revising all the properties and describing how the checkmarks are derived in Appendix A.

2 Preliminaries

Strings. All strings are treated as binary strings. The set of strings of any length is denoted by $\{0,1\}^*$, while the set of strings of length n (where n is a positive integer) is represented by $\{0,1\}^n$. We define $\{0,1\}^{\leq n}$ as $\bigcup_{i=0}^{n}\{0,1\}^i$. The set of all permutations of $\{0,1\}^n$ is represented by $\mathrm{Perm}(n)$, and the set of all functions with domain $\{0,1\}^m$ and range $\{0,1\}^n$ is denoted by $\mathrm{Func}(m,n)$.

For a string X of ℓ bits, $X[i]$ denotes the i-th bit of X for $i = 0,\ldots,\ell-1$ (counting from left to right). We define $X[i\ldots j] = X[i]\|X[i+1]\|\ldots\|X[j]$ for $0 \leq i < j < \ell$. The notation $\mathsf{left}_\ell(X)$ is used for the ℓ leftmost bits of X and $\mathsf{right}_r(X)$ for the r rightmost bits of X, ensuring $X = \mathsf{left}_\chi(X)\|\mathsf{right}_{|X|-\chi}(X)$

for all $0 \leq \chi \leq |X|$. The notation $(L, R) = \mathsf{lsplit}_n(X)$ represents splitting a string $X \in \{0,1\}^*$ into two parts such that $L = \mathsf{left}_{\min(|X|,n)}(X)$ and $R = \mathsf{right}_{|X|-|L|}(X)$. For $n \geq |X|$, we have $(X, \varepsilon) = \mathsf{lsplit}_n(X)$ where ε denotes the empty string with length 0. Given an integer $n > 0$ and $X \in \{0,1\}^*$, $X \| 10^*$ denotes $X \| 10^{n-(|X| \bmod n)-1}$. We also define $\mathsf{pad10}(X)$ which returns X if $|X| \equiv 0 \pmod n$ and $X \| 10^*$ otherwise.

String Partitioning. We fix an arbitrary integer n and refer to it as the block size. For any string X, $|X|_n = \lceil |X|/n \rceil$ denotes the number of n-bit blocks in X. The notation $X_1, \ldots, X_x, X_* \xleftarrow{n} X$ indicates the partitioning of X into n-bit blocks, with $X = X_1 \| \ldots \| X_x \| X_*$ where $|X_i| = n$ for $i = 1, \ldots, x$ and $0 < |X_*| \leq n$. Thus, $x = |X|_n - 1$.

Blocks. The set of all n-bit strings (or blocks) is denoted by B_n, equivalent to $\{0,1\}^n$. We define B_n^* as $\{\varepsilon\} \cup \bigcup_{i=1}^{\infty} \mathsf{B}_n^i$. A string X is considered "n-aligned" if $X \in \mathsf{B}_n^*$. The notation X_i represents the i-th n-bit block of an n-aligned string X. For two distinct n-aligned strings $X, Y \in \mathsf{B}_n^*$ with $|X|_n \leq |Y|_n$ without loss of generality, $\mathsf{llcp}_n(X, Y) = \max\{1 \leq i \leq |X|_n \mid X_j = Y_j \text{ for } 1 \leq j \leq i\}$ denotes the length of the longest common prefix (in n-bit blocks) of X and Y.

Miscellaneous. The notation $X \leftarrow\!\$\ \mathcal{X}$ signifies sampling an element X from a finite set $\mathcal{X}$ under the uniform distribution. The falling factorial $(p)_q$ is defined as $p \cdot (p-1) \cdot (p-2) \cdots (p-q+1)$ with $(p)_0 = 1$. A predicate $\mathsf{P}(x)$ is defined such that $\mathsf{P}(x) = 1$ if it is true for x, and $\mathsf{P}(x) = 0$ if false for x. Lexicographic comparison for integer tuples is used, e.g., $(i', j') < (i, j)$ if $i' < i$ or $i' = i$ and $j' < j$. The symbol $\perp$ denotes an undefined value or error. We consider $\mathcal{A}$ as an adversary (algorithm) attempting to distinguish between world $\mathcal{O}_{re}$ (or game G_{re}) and world $\mathcal{O}_{id}$ (or game G_{id}, respectively). The notation $\mathcal{A}^{\mathcal{O}_x}$ (or $\mathcal{A}^{\mathsf{G}_x}$) denotes the event where $\mathcal{A}$, after interacting with world $\mathcal{O}_x$ (or playing game G_x), returns $x = id$.

2.1 Syntax of AEAD under RUP setting

A nonce-based Authenticated Encryption with Associated Data (AEAD) scheme under the RUP setting is defined by the tuple $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{V})$. The key space $\mathcal{K}$ is a finite set. The encryption algorithm $\mathcal{E} : \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{M} \to \mathcal{C}$ deterministically maps a secret key K, a nonce N, associated data A, and a message M to a ciphertext $C = \mathcal{E}(K, N, A, M)$. Here, the domains for the nonce, associated data, and message are subsets of $\{0,1\}^*$. The decryption algorithm $\mathcal{D} : \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{C} \to \mathcal{M}$ is deterministic and maps a tuple (K, N, A, C) back to the message $M \in \mathcal{M}$. The verification algorithm $\mathcal{V} : \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{C} \to \{\top, \perp\}$ deterministically evaluates the tuple (K, N, A, C) and returns $\top$ for successful authentication or $\perp$ for an authentication failure.

For any message $M \in \mathcal{M}$, the domain condition $\{0,1\}^{|M|} \subseteq \mathcal{M}$ ensures that for any integer m, either all or none of the strings of length m are in $\mathcal{M}$. Furthermore, for all $(K, N, A, M) \in \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{M}$, the ciphertext length is given by $|\mathcal{E}(K, N, A, M)| = |M| + \theta$, where θ is a non-negative integer known as the stretch of Π. The final n bits of the stretch represents the n bits of authentication tag whereas the rest $\theta - n$ bits represent the ciphertext expansion due to the padding of the input. To ensure correctness, it is required that for all $(K, N, A, M) \in \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{M}$, the decryption and verification algorithms satisfy $M = \mathcal{D}(K, N, A, \mathcal{E}(K, N, A, M))$ and $\top = \mathcal{V}(K, N, A, \mathcal{E}(K, N, A, M))$.

For notational convenience, we define $\mathcal{E}_K(N, A, M) = \mathcal{E}(K, N, A, M)$, $\mathcal{D}_K(N, A, C) = \mathcal{D}(K, N, A, C)$, and $\mathcal{V}_K(N, A, C) = \mathcal{V}(K, N, A, C)$.

2.2 Security definitions under RUP setting

sOPRPF Confidentiality. To define the sOPRPF confidentiality notion, we need to recall the definition of an online permutation (an ideal object that captures the online/blockwise processing of plaintext/ciphertext data). An online permutation [11] $\pi \colon \mathsf{B}_n^* \to \mathsf{B}_n^*$ is a function that has the following properties for some positive integer n: (i) π preserves the length of the input; i.e., for any integer $m \geq 0$, the function π applied to mn-bit inputs, written as $\pi \colon \mathsf{B}_n^m \to \mathsf{B}_n^m$, is a permutation; (ii) π preserves blockwise prefix i.e., the number of common blocks (with block size n) at the start of any two inputs is the same for their corresponding outputs. More specifically, for each $M, M' \in \mathsf{B}_n^*$, we have that $\mathsf{llcp}_n(M, M') = \mathsf{llcp}_n(\pi(M), \pi(M'))$.

We use $\mathrm{OPerm}(n)$ to denote the set of all online permutations. Each $\pi \in \mathrm{OPerm}(n)$ can also be described as a collection $(\pi_M)_{M \in \mathsf{B}_n^*}$ of permutations, such that for any $M_1 \| M_2 \| \ldots \| M_r \in \mathsf{B}_n^r$ we get $\pi(M_1 \| M_2 \| \ldots \| M_r) = \pi_\varepsilon(M_1) \| \pi_{M_1}(M_2) \| \ldots \| \pi_{M_1 \| \ldots \| M_{r-1}}(M_r)$, where ε is the empty string. There is a one-to-one correspondence between $\mathrm{OPerm}(n)$ and the set of all such permutation collections. We use $\pi \leftarrow\$ \mathrm{OPerm}(n)$ to denote random sampling of an online permutation $\pi = (\pi_M)_{M \in \mathsf{B}_n^*}$ from the set $\mathrm{OPerm}(n)$ and define it by uniform random sampling of underlying $\pi_\varepsilon, \pi_{M_1}, \ldots, \pi_{M_1 \| \ldots \| M_{r-1}}$ from $\mathrm{Perm}(n)$ on demand to answer queries of the form $M_1 \| M_2 \| \ldots \| M_r$ (for more details on this lazy sampling, we refer the reader to [4] and [11]).

We now define the sOPRPF confidentiality of an AEAD scheme Π with two games, $\textbf{soprpf-real}_\Pi$ and $\textbf{soprpf-ideal}_\Pi$. In both games, the adversary $\mathcal{A}$ can make any number of chosen plaintext and chosen ciphertext queries to the encryption and decryption oracles, respectively. $\mathcal{A}$ can also use the same nonce more than once. We assume that $\mathcal{A}$ does not make empty message/ciphertext queries as there is no confidentiality to achieve in such a case. In the game $\textbf{soprpf-real}_\Pi$, the oracles use the encryption and decryption algorithms of Π with a randomly picked secret key. On the other hand, in the game $\textbf{soprpf-ideal}_\Pi$, the encryption oracle returns a random online permutation of the input padded with a random string as the tag, the decryption oracle first drops the tag and then returns the (unverified) inverse online permutation of the remaining ciphertext, and the verification oracle first decrypts the ciphertext and then regenerates and verifies the random string (as the tag) from it. More formally, upon an encryption query with inputs $(N, A, M) \in \mathcal{N} \times \mathcal{AD} \times \mathcal{M}$, the encryption oracle returns $P_{N,A}(M_L) \| \pi_{N,A,M_L,\lfloor |M_R|/n \rfloor}(\mathsf{pad10}(M_R)) \| f_{N,A,M}$ where

$M = M_L \| M_R$ with M_L being the longest prefix of M such that $|M_L|$ is divisible by n and $|M_R| \neq 0$, $P_{N,A} \leftarrow\$ \text{OPerm}(n)$ is a random online permutation for each pair (N, A), $\pi_{N,A,M_L,\lfloor |M_R|/n \rfloor} \leftarrow\$ \text{Perm}(n)$ is a random permutation for each quadruple $(N, A, M_L, \lfloor |M_R|/n \rfloor)$, and $f_{N,A,M} \leftarrow\$ \{0,1\}^{(|M| \bmod n)}$ is a random string for each triple (N, A, M). Similarly, upon a decryption query with inputs (N, A, C), where $C = C_L \| C_R \| C_T$ with C_L being the longest prefix of C such that $|C_L|$ is divisible by n, $|C_R| = n$ and $|C_T| \neq 0$, the decryption oracle returns $P^{-1}_{N,A}(C_L) \| \pi^{-1}_{N,A,P^{-1}_{N,A}(C_L),\lfloor |C_T|/n \rfloor}(C_R)$ where $P^{-1}_{N,A}$ and $\pi^{-1}_{N,A,P^{-1}_{N,A}(C_L),\lfloor |C_T|/n \rfloor}$ are the inverse permutations of $P_{N,A}$ and $\pi_{N,A,P^{-1}_{N,A}(C_L),\lfloor |C_T|/n \rfloor}$, respectively. The **soprpf** advantage of an adversary $\mathcal{A}$ against $\Pi = \text{SAEF}[\mathsf{F}]$ is defined as $\mathbf{Adv}^{\mathbf{soprpf}}_{\Pi}(\mathcal{A}) = \Pr[\mathcal{A}^{\mathbf{soprpf\text{-}real}_{\Pi}}] - \Pr[\mathcal{A}^{\mathbf{soprpf\text{-}ideal}_{\Pi}}]$.

The notion of sOPRPF is a stronger notion than OPRPF [4] and a weaker notion than IND-CPA+PA1 [5]. Roughly, it claims RUP confidentiality up to the longest common prefix. We note that OPRPF is not sufficient for the confidentiality of plaintexts with decryptional leakage (as it captures only CPA) and the stronger IND-CPA+PA1 security is not achievable by any online scheme that is based on the concept of "online permutation" and allows the nonce to be reused over queries which makes sOPRPF the best option at hand.

INT-RUP Authenticity [5]. Traditional requirements for the integrity of an AE scheme can be achieved by the INT-CTXT notion, where the adversary is allowed to make encryption and decryption queries, but the decryption oracle always returns $\perp$. However, under the RUP setting, where the adversary is allowed to observe the unverified plaintext, the integrity requirements as in INT-CTXT need to be modified. The following definition from the work of Andreeva et al. [5] presents the targeted notion of integrity under the RUP setting.

Let us define two games, $\mathbf{intrup\text{-}real}_{\Pi}$ and $\mathbf{intrup\text{-}ideal}_{\Pi}$. In both games, the adversary is given access to an encryption, a decryption, and a verification oracle. In the game $\mathbf{intrup\text{-}real}_{\Pi}$, all three oracles faithfully implement the corresponding algorithms of Π using the same randomly sampled secret key. Here the verification oracle returns $\top$ in case of a successful forgery, and $\perp$ otherwise. In the game $\mathbf{intrup\text{-}ideal}_{\Pi}$, the encryption and decryption oracles are same as in $\mathbf{intrup\text{-}real}_{\Pi}$ but the verification oracle always returns $\perp$.

Definition 1 (INT-RUP Advantage). *Let $\mathcal{A}$ be a computationally bounded adversary with access to an encryption, a decryption, and a verification oracle namely $\mathcal{E}_K, \mathcal{D}_K$, and $\mathcal{V}_K$ for Π for some $K \leftarrow\$ \mathcal{K}$. Let* $\mathbf{intrup\text{-}real}_{\Pi}$ *and* $\mathbf{intrup\text{-}ideal}_{\Pi}$ *be two games as defined above. The* INT-RUP *advantage of $\mathcal{A}$ against Π is then defined as*

$$\mathbf{Adv}^{\mathbf{intrup}}_{\Pi}(\mathcal{A}) = \Pr[\mathcal{A}^{\mathbf{intrup\text{-}real}_{\Pi}}] - \Pr[\mathcal{A}^{\mathbf{intrup\text{-}ideal}_{\Pi}}].$$

In other words, this advantage defines the probability that $\mathcal{A}$ forges, i.e., $\mathcal{A}$ comes up with a new ciphertext-tag pair which is not an output from the queries of encryption oracle $\mathcal{E}_K$, but when queried to verification oracle $\mathcal{V}_K$ it results into $\top$, i.e., the forgery is a success.

We note that the INT-RUP definition does not specify the adversary being nonce-respecting or -misusing as the main goal here is to capture insecurities from the release of unverified plaintexts. However, such a distinction is needed to combine this notion with confidentiality against nonce-misusing adversaries. We use INT-RUP(NR) and INT-RUP(NM) to represent the INT-RUP security against nonce-respecting and nonce-misusing adversaries, respectively. For the rest of the paper, we drop (NM) and use INT-RUP to denote the INT-RUP security against nonce-misusing adversaries for simplicity.

We now define a new online AE security notion for RUP called OAE-RUP as sOPRPF+INT-RUP.

Definition 2 (OAE-RUP Advantage). *Let $\mathcal{A}$ be a computationally bounded nonce-misusing adversary with access to an encryption, a decryption, and a verification oracle namely $\mathcal{E}_K, \mathcal{D}_K$, and $\mathcal{V}_K$ for Π for some $K \leftarrow\!\$\, \mathcal{K}$. The* OAE-RUP *advantage of $\mathcal{A}$ against Π is then defined as*

$$\mathbf{Adv}_{\Pi}^{\mathsf{OAE\text{-}RUP}}(\mathcal{A}) = \mathbf{Adv}_{\Pi}^{\mathbf{soprpf}}(\mathcal{A}) + \mathbf{Adv}_{\Pi}^{\mathbf{intrup}}(\mathcal{A}).$$

As by definition sOPRPF implies OPRPF[4] and INT-RUP implies INT-CTXT (under nonce-misuse), we have that OAE-RUP implies the OAE [4] security for online AEs. In other words, OAE-RUP jointly covers both nonce-misuse and RUP security of online AE schemes.

Informally, the OAE-RUP security of an AE scheme Π says that under nonce-misuse and leakage of unverified plaintexts, the integrity of Π remains intact whereas the confidentiality is degraded but preserved up to the leakage of common message prefixes.

2.3 Forkcipher

We use the forkcipher definition from Andreeva et al. [39]. A forkcipher F is a tweakable symmetric primitive that maps a key K, a tweak T, and an n-bit input M to two n-bit ciphertexts C_0 and C_1, which are independent permutations of M.

A forkcipher consists of two deterministic algorithms: 1) the encryption algorithm $\mathsf{F}\colon \{0,1\}^k \times \mathcal{T} \times \{0,1\}^n \times \{0,1,\mathsf{b}\} \rightarrow \{0,1\}^n \cup \{0,1\}^n \times \{0,1\}^n$ and the inversion algorithm $\mathsf{F}^{-1}\colon \{0,1\}^k \times \mathcal{T} \times \{0,1\}^n \times \{0,1\} \times \{\mathsf{i},\mathsf{o},\mathsf{b}\} \rightarrow \{0,1\}^n \cup \{0,1\}^n \times \{0,1\}^n$.

The encryption algorithm takes K, T, M, and an output selector s. It outputs C_0 if $s = 0$, C_1 if $s = 1$, or both C_0 and C_1 if $s = \mathsf{b}$. Notations $\mathsf{F}(K,T,M,s) = \mathsf{F}_K(T,M,s) = \mathsf{F}_K^T(M,s) = \mathsf{F}_K^{T,s}(M)$ are used interchangeably.

The inversion algorithm takes K, T, C, an input indicator b, and an output selector s. It outputs M if $s = \mathsf{i}$, the other ciphertext C' if $s = \mathsf{o}$, or both M and

[4] The existing definition of OPRPF in [4] models the last ciphertext block as an output of a random function, however, we consider it here as a random permutation (as invertibility is required to successfully decrypt a ciphertext for leakage).

C' if $s = \mathsf{b}$. Notations $\mathsf{F}^{-1}(K, T, M, b, s) = \mathsf{F}^{-1}{}_K(T, M, b, s) = \mathsf{F}^{-1}{}_K^T(M, b, s) = \mathsf{F}^{-1}{}_K^{T,b,s}(M)$ are used interchangeably.

A forkcipher F is correct if for any (K, T, M, β) with $K \in \{0,1\}^k$, $T \in \mathcal{T}$, $M \in \{0,1\}^n$, and $\beta \in \{0,1\}$, it satisfies: (i) $\mathsf{F}^{-1}(K, T, \mathsf{F}(K, T, M, \beta), \beta, \mathsf{i}) = M$, (ii) $\mathsf{F}^{-1}(K, T, \mathsf{F}(K, T, M, \beta), \beta, \mathsf{o}) = \mathsf{F}(K, T, M, \beta \oplus 1)$, (iii) $(\mathsf{F}(K, T, M, 0), \mathsf{F}(K, T, M, 1)) = \mathsf{F}(K, T, M, \mathsf{b})$ and (iv) $(\mathsf{F}^{-1}(K, T, C, \beta, \mathsf{i}), \mathsf{F}^{-1}(K, T, C, \beta, \mathsf{o})) = \mathsf{F}^{-1}(K, T, C, \beta, \mathsf{b})$.

We assume $\mathcal{T} = \{0,1\}^t$ for some t. Parameters k, n, and t refer to the key size, block size, and tweak size of the forkcipher, respectively.

Forkcipher Security. The security of F is defined by the indistinguishability between the real $\mathbf{prtfp\text{-}real}_{\mathsf{F}}$ and ideal $\mathbf{prtfp\text{-}ideal}_{\mathsf{F}}$ worlds when an adversary interacts using chosen ciphertext queries. In the real world, the forkcipher oracle implements the actual F algorithm. In the ideal world, the oracle uses two independent tweakable random permutations $\pi_{T,0}, \pi_{T,1} \leftarrow\$ \mathrm{Perm}(n)$ for each $T \in \mathcal{T}$. The adversary's advantage is:

$$\mathbf{Adv}^{\mathbf{prtfp}}_{\mathsf{F}}(\mathcal{A}) = \Pr[\mathcal{A}^{\mathbf{prtfp\text{-}real}_{\mathsf{F}}}] - \Pr[\mathcal{A}^{\mathbf{prtfp\text{-}ideal}_{\mathsf{F}}}].$$

2.4 Coefficients H Technique

The coefficients H is a simple but powerful proof technique by Patarin [38]. It is often used to prove the indistinguishability of a provided construction from an idealized object for an information-theoretic adversary. Coefficients H-based proofs use the concept of "transcripts". A transcript is defined as a complete record of the interaction of an adversary $\mathcal{A}$ with its oracles in the indistinguishability experiment. For example, if (M_i, C_i) represents the input and output of the i^{th} query of $\mathcal{A}$ to its oracle and the total number of queries made by $\mathcal{A}$ is q, then the corresponding transcript (denoted by τ) is defined as $\tau = \langle (M_1, C_1), \ldots, (M_q, C_q) \rangle$. The goal of an adversary $\mathcal{A}$ is to distinguish interactions in the real world $\mathcal{O}_{\mathsf{real}}$ from the ones in the ideal world $\mathcal{O}_{\mathsf{ideal}}$.

We denote the distribution of transcripts in the real and the ideal world by Θ_{real} and Θ_{ideal}, respectively. We call a transcript τ *attainable* if the probability of achieving τ in the ideal world is non-zero. Further, w.l.o.g. we also assume that $\mathcal{A}$ does not make any duplicate or prohibited queries. We can now state the fundamental Lemma of the coefficients H technique.

Lemma 1 (Fundamental Lemma of the Coefficients H Technique [38]). *Assume that the set of attainable transcripts is partitioned into two disjoint sets $\mathcal{T}_{\mathit{good}}$ and $\mathcal{T}_{\mathit{bad}}$. Also, assume there exist $\epsilon_1, \epsilon_2 \geq 0$ such that for any transcript $\tau \in \mathcal{T}_{\mathit{good}}$, we have $\frac{\Pr[\Theta_{\mathsf{real}}=\tau]}{\Pr[\Theta_{\mathsf{ideal}}=\tau]} \geq 1 - \epsilon_1$, and $\Pr[\Theta_{\mathsf{ideal}} \in \mathcal{T}_{\mathit{bad}}] \leq \epsilon_2$. Then, for any adversary $\mathcal{A}$, it holds that*

$$|\Pr[\mathcal{A}^{\mathcal{O}_{\mathsf{real}}}] - \Pr[\mathcal{A}^{\mathcal{O}_{\mathsf{ideal}}}]| \leq \epsilon_1 + \epsilon_2.$$

3 Relations Among AEAD Notions

In this section, we discuss the relationships and differences among popular AEAD security notions. For comparison, we consider two key parameters that define a security notion: the adversary's capabilities and the security goals achieved.

For simplicity, we denote an AEAD security notion that achieves X-type security (i.e., confidentiality and integrity of the processed messages, and integrity of the processed associated data) against Y-type adversaries as an (X, Y)-notion. To ensure a fair comparison for input-order-sensitive notions such as OAE, we assume that the AEAD schemes targeting any of these notions process inputs in the ordered form (N, A, M), where N, A, and M represent the nonce, associated data, and message, respectively.

We provide two tables, detailing the X and Y types, and a plot comparing the strengths of popular AEAD notions in Fig. 1. Here the y-axis defines the adversarial powers i.e., the Y types whereas the x-axis defines the achieved security goals i.e., the X types. A point on the plot is represented by a cell in the big colored table consisting the AEAD notions that includes IND-CPA+INT-CTXT [12], IND-CCA+INT-CTXT [12], nAE [40], NMR [8], AERUP [23], subtleAE [10], $\mathrm{RAE}_{\mathsf{sim}}$ [29], OAE [26], OAE-RUP as well as some of their weaker variants that claims security only for nonce-respecting (NR) queries and/or queries that are not subject to decryptional leakage (NL).

Abbreviations. In Fig. 1, Y types varies from $\mathsf{1}$ to $\mathsf{12}$ and are defined by the different combinations of powers that are given to the target adversaries. These combinations consist four different powers - nonce-misuse (NM), blockwise adaptive input processing for encryption (BE), observing the unverified plaintexts and therefore the leakage of prefixes (LP) and blockwise adaptive input processing for decryption (BD).

Various combinations are defined here by allowing some of the powers and restricting the others. For e.g., $Y = \mathsf{4}$ means that the target adversary is nonce-misusing and blockwise adaptive for encryption, however, cannot observe any decryptional leakage and cannot be blockwise adaptive for decryption.

Similarly, the X types varies from A to F and are defined by the different combinations of achieved security goals for different category of encrypted messages. More specifically, there are four different categories of encrypted messages - 1) messages that contain unique nonces and are not subject to leakage i.e., share no prefix with any leaked unverified plaintexts, 2) messages that contain repeated nonces but are still not subject to leakage, 3) messages that contain unique nonces but are subject to leakage and 4) messages that contain repeated nonces and are also subject to leakage. These categories are represented (in the same order) by the first column of the X types table in Fig. 1.

We consider four different types of well-defined security goals dubbed $\mathsf{t1}, \mathsf{t2}, \mathsf{t3}$ and $\mathsf{t4}$ that can be captured for the encrypted messages in various categories. None of the four security goals compromise on integrity i.e., full integrity of all encrypted messages in desired whereas the achieved confidentiality is different for all of them defined as 1) $\mathsf{t1}$ - confidentiality with only leakage of the length of

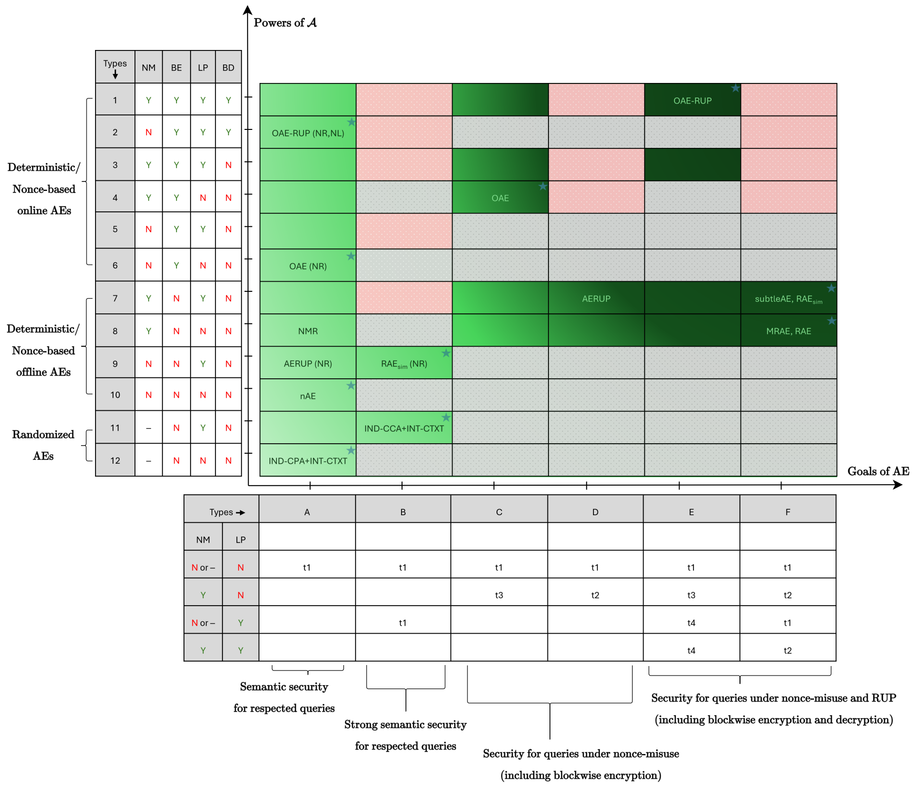

Fig. 1: Relations among popular AEAD notions.

the plaintext, 2) t2 - confidentiality as t1 but with additional leakage of message repetitions, 3) t3 - confidentiality as t2 but with additional leakage of the length of common prefixes with other encrypted plaintexts and 4) t4 - confidentiality as t3 but with additional leakage of the common prefixes with other decrypted plaintexts. For e.g., $X = \mathsf{C}$ means that the AEAD notion captures security as full integrity for all encrypted plaintexts, t1 confidentiality for encrypted plaintexts that contain unique nonces and are not subject to leakage and t3 confidentiality for encrypted plaintexts that contain repeated nonces but are still not subject to leakage. It says nothing about the confidentiality of other categories of encrypted plaintexts (if any).

We note that nonce-respecting and nonce-misusing are nonce-specific terms that only apply to nonce-based AEAD notions. Therefore, in Fig. 1, we use "–" in nonce-related categories to denote "not applicable" for randomized AEAD security notions and we use Y and N to represent yes and no, respectively.

How to Read Fig. 1. With all the abbreviations defined, (X, Y)-notion can now easily be understood. The position of an (X, Y)-notion in Fig. 1 is simply defined by the target adversary setting and the captured goals of it. Let us consider the following example - $\mathrm{RAE}_{\mathsf{sim}}$ [29] is a robust authenticated encryption security notion that by definition captures full integrity and confidentiality of encrypted plaintexts with one degradation that the ciphertexts of repeated plaintexts are same. This means when the nonce is unique, it achieves $\mathsf{t1}$ confidentiality and $\mathsf{t2}$, otherwise. This makes $X = \mathsf{F}$ for $\mathrm{RAE}_{\mathsf{sim}}$. Further, $\mathrm{RAE}_{\mathsf{sim}}$, by definition, resists against nonce-misuse and decryptional leakage but cannot be online and therefore does not support blockwise encryption and decryption. This makes $Y = \mathsf{7}$ which implies $\mathrm{RAE}_{\mathsf{sim}}$ an $(\mathsf{7}, \mathsf{F})$-notion.

Colors, Gradient and Stars. The red colored cells represent impossible combinations to be captured by any AEAD notion due to contradictions between the corresponding adversarial powers and achieved security goals. For e.g., the cell corresponding to $(\mathsf{1}, \mathsf{F})$-notion is red because the adversary is nonce-misusing and blockwise adaptive yet the security goals claims $\mathsf{t2}$ (i.e., confidentiality with leakage of only input length and repetition) for nonce-misused and leaked-prefix queries which is impossible as blockwise encryption additionally leaks repetitions of common prefixes.

The gray colored cells represent senseless combinations to be captured by any AEAD notion due to adversarial powers being incompatible/inconsistent with the achieved security goals. For e.g., the cell corresponding to $(\mathsf{2}, \mathsf{C})$-notion is gray because the adversary is required to be nonce-respecting yet the security claims includes $\mathsf{t3}$ security for queries with nonce-misuse.

We call the remaining cells that are not red or gray as the sensible notions. They are colored with a green gradient. The gradient shows the strength of an AEAD notion where going from the lighter to the darker area implies the strengthening of adversarial powers and/or achieved security goals.

The table shows that OAE-RUP and $\mathrm{RAE}_{\mathsf{sim}}$ are two of the strongest AEAD security notions in their categories. Comparing with each other, $\mathrm{RAE}_{\mathsf{sim}}$ gives adversaries less power but achieves stronger security goals when compared with OAE-RUP.

We also note that as per the definition of the gradient, the rightmost notion for a given Y-type in Fig. 1 is the strongest/best notion for that Y. The same is denoted by a blue starred cell in every row.

Position of sOPRPF and CR-RUP. We highlight that OAE-RUP and OAE-RUP (NR, NL) are the two best notions in their rows (i.e., under the target adversary settings). OAE-RUP (NR, NL) is a weaker variant of OAE-RUP that claims same security as OAE-RUP but only for the encrypted plaintexts that contain unique nonces and are not subject to decryptional leakage. We recall that OAE-RUP is defined as sOPRPF+INT-RUP and as per the definition of CR-RUP (see Appendix B), OAE-RUP (NR,NL) can be defined as CR-RUP+INT-RUP(NR).

4 SAEF and its OAE-RUP Security

SAEF (Sequential AE from a Forkcipher) is a nonce-based AEAD scheme utilizing a tweakable forkcipher F (as defined in Section 2.3) with $\mathcal{T} = \{0,1\}^t$ for a positive $t \leq n$. The scheme, denoted as SAEF[F] $= (\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{V})$, has a key space $\mathcal{K} = \{0,1\}^k$, a nonce space $\mathcal{N} = \{0,1\}^{t-4}$, and both the associated data (AD) and message spaces as $\{0,1\}^*$. The ciphertext expansion of SAEF is n bits. The encryption, decryption, and verification algorithms are detailed in Figure 3 whereas a pictorial diagram of SAEF's encryption process is provided in Figure 2. Unlike the earlier SAEF representation [7], which lacked explicit separation of decryption and verification functionalities, our current syntax distinctly separates these functions without altering the input-output behavior of the SAEF algorithm.

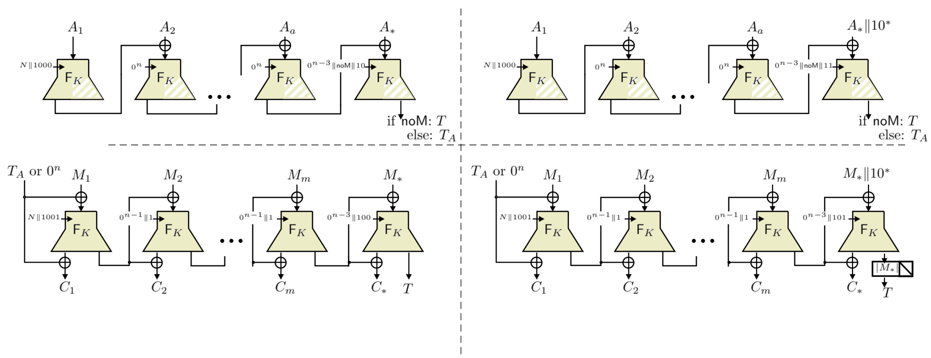

Fig. 2: The Encryption Algorithm of SAEF[F] Mode. The bit $\mathsf{noM} = 1$ if and only if $|M| = 0$. Here, the white hatching indicates that the output block is not computed. The diagram shows the processing of Additional Data (AD) and message in four scenarios: 1) Top Left: AD length is a multiple of n. 2) Top Right: AD length is not a multiple of n. 3) Bottom Left: Message length is a multiple of n. 4) Bottom Right: Message length is not a multiple of n.

SAEF processes an encryption query in blocks of n bits (in order), with first AD and then the message. It uses a single forkcipher call for each block. These forkcipher calls are tweaked by composing: (1) either the nonce followed by a 1-bit (for the first F call of the query) or the string 0^{t-3}, (2) a three-bit flag f.

This flag f is used to ensure proper domain separation for various "types" of blocks in the encryption algorithm. The values of f from the set $\{000, 010, 011, 110, 111, 001, 100, 101\}$ are respectively used when processing non-final AD block, the last n-bit long AD block, the last AD block of $< n$ bits, the last AD block of n bits to produce tag, the last AD block of $< n$ bits to produce tag, non-final message block, the last n-bit message block and the last message block of $< n$ bits.

The left (or right, respectively) output block of every F call is used as a chaining value to mask the input of the following F call in the case of AD processing (or both the input and output of the following F call in the case of message processing, respectively). The first F call of every query is not masked but contains the nonce in the tweak. The tag for a query is defined as the (possibly truncated) last "right" output block of F. In case of truncation, message padding is used for partial integrity check of the ciphertext. For a decryption (respectively verification) query, the processing of input blocks is similar to the encryption, but now with the chaining values in the message processing part are computed with the "inverse" F algorithm. These chaining values are used (similarly to the encryption algorithm) to compute the corresponding plaintext blocks (respectively to verify the final tag).

```
function E(K, N, A, M)
    A_1, ..., A_a, A_* <-n- A
    M_1, ..., M_m, M_* <-n- M
    noM ← 0
    if |M| = 0 then noM ← 1
    Δ ← 0^n; T ← N‖1
    for i ← 1 to a do
        T ← T‖000
        Δ ← F_K^{T,0}(A_i ⊕ Δ)
        T ← 0^{t−3}
    end for
    if |A_*| = n then
        T ← T‖noM‖10
        Δ ← F_K^{T,0}(A_* ⊕ Δ)
        T ← N‖1
    else if |A_*| > 0 then
        T ← T‖noM‖11
        Δ ← F_K^{T,0}((A_*‖10^*) ⊕ Δ)
        T ← N‖1
    end if
    for i ← 1 to m do
        T ← T‖001
        C_i, Δ ← F_K^{T,b}(M_i ⊕ Δ) ⊕ (Δ, 0^n)
        T ← 0^{t−3}
    end for
    if |M_*| = n then
        T ← T‖100
    else if |M_*| > 0 then
        T ← T‖101
    end if
    if noM = 1 then
        T ← Δ
        return T
    else
        C_*, T ← F_K^{T,b}(pad10(M_*) ⊕ Δ) ⊕ (Δ‖0^n)
    end if
    return C_1‖...‖C_m‖C_*‖left_{|M_*|}(T)
end function
```

```
function D(K, N, A, C)
    A_1, ..., A_a, A_* <-n- A
    C_1, ..., C_m, C_*, T <-n- C
    noM ← 0
    if |C| = n then noM ← 1
    Δ ← 0^n; T ← N‖1
    for i ← 1 to a do
        T ← T‖000
        Δ ← F_K^{T,0}(A_i ⊕ Δ)
        T ← 0^{t−3}
    end for
    if |A_*| = n then
        T ← T‖noM‖10
        Δ ← F_K^{T,0}(A_* ⊕ Δ)
        T ← N‖1
    else if |A_*| > 0 then
        T ← T‖noM‖11
        Δ ← F_K^{T,0}((A_*‖10^*) ⊕ Δ)
        T ← N‖1
    end if
    for i ← 1 to m do
        T ← T‖001
        M_i, Δ ← F−1_K^{T,0,b}(C_i ⊕ Δ) ⊕ (Δ, 0^n)
        T ← 0^{t−3}
    end for
    if |T| = n then
        T ← T‖100
    else if |T| > 0 then
        T ← T‖101
    end if
    if noM = 1 then
        return ε

    else
        M_*, T' ← F−1_K^{T,0,b}(C_* ⊕ Δ) ⊕ (Δ, 0^n)
    end if
    return M_1‖...‖M_m‖left_{|T|}(M_*)
end function
```

```
function V(K, N, A, C)
    A_1, ..., A_a, A_* <-n- A
    C_1, ..., C_m, C_*, T <-n- C
    noM ← 0
    if |C| = n then noM ← 1
    Δ ← 0^n; T ← N‖1
    for i ← 1 to a do
        T ← T‖000
        Δ ← F_K^{T,0}(A_i ⊕ Δ)
        T ← 0^{t−3}
    end for
    if |A_*| = n then
        T ← T‖noM‖10
        Δ ← F_K^{T,0}(A_* ⊕ Δ)
        T ← N‖1
    else if |A_*| > 0 then
        T ← T‖noM‖11
        Δ ← F_K^{T,0}((A_*‖10^*) ⊕ Δ)
        T ← N‖1
    end if
    for i ← 1 to m do
        T ← T‖001
        M_i, Δ ← F−1_K^{T,0,b}(C_i ⊕ Δ) ⊕ (Δ, 0^n)
        T ← 0^{t−3}
    end for
    if |T| = n then
        T ← T‖100
    else if |T| > 0 then
        T ← T‖101
    end if
    if noM = 1 ∧ C ≠ Δ then
        return ⊥

    else if noM = 0 then
        M_*, T' ← F−1_K^{T,0,b}(C_* ⊕ Δ) ⊕ (Δ, 0^n)
        T' ← left_{|T|}(T')
        P ← right_{n−|T|}(M_*)
        if T‖left_{n−|T|}(10^n) ≠ T'‖P then
            return ⊥
        end if
    end if
    return ⊤
end function
```

Fig. 3: The SAEF[F] AEAD scheme.

Security of SAEF. In [4], Andreeva et al. proved that SAEF achieves OAE confidentiality and integrity up to the birthday bound under Nonce-Misuse. However, there have been no investigations into the security of SAEF under the release of unverified plaintext (i.e., if the decrypted plaintext is released before the tag verification). We state the formal claim about confidentiality and integrity of SAEF under RUP in Theorem 1.

Theorem 1. *Let F be a tweakable forkcipher with $\mathcal{T} = \{0,1\}^t$. Then for any nonce-misuse adversary $\mathcal{A}$ who makes at most q_e encryption, at most q_d decryption, and at most q_v verification queries with $q_e + q_d \leq 2^{n-1}$ such that the total number of forkcipher calls induced by all the queries is at most σ, we have*

$$\mathbf{Adv}^{\mathbf{soprpf}}_{\mathrm{SAEF}[\mathsf{F}]}(\mathcal{A}) \leq \mathbf{Adv}^{\mathbf{prtfp}}_{\mathsf{F}}(\mathcal{B}) + \frac{3 \cdot \sigma^2}{2^{n+1}}$$
$$\mathbf{Adv}^{\mathbf{intrup}}_{\mathrm{SAEF}[\mathsf{F}]}(\mathcal{A}) \leq \mathbf{Adv}^{\mathbf{prtfp}}_{\mathsf{F}}(\mathcal{B}) + \frac{\sigma^2 + 4 \cdot q_d q_v}{2^n}$$

for some adversary $\mathcal{B}$, making at most 2σ queries, and running in the time given by the running time of $\mathcal{A}$ plus $\gamma \cdot \sigma$ for some "small" constant γ.

5 Proof of Theorem 1

Proof (Theorem 1). **Replacing F.** We first replace F with a pair of independent random tweakable permutations $\pi_0 = (\pi_{\mathsf{T},0} \leftarrow\$ \mathrm{Perm}(n))_{\mathsf{T}\in\{0,1\}^t}$ and $\pi_1 = (\pi_{\mathsf{T},1} \leftarrow\$ \mathrm{Perm}(n))_{\mathsf{T}\in\{0,1\}^t}$ and let $\mathrm{SAEF}[(\pi_0, \pi_1)]$ denote the SAEF mode that uses π_0, π_1 instead of F, which yields for the integrity case, $\mathbf{Adv}^{\mathbf{intrup}}_{\mathrm{SAEF}[\mathsf{F}]}(\mathcal{A}) \leq \mathbf{Adv}^{\mathbf{prtfp}}_{\mathsf{F}}(\mathcal{B}) + \mathbf{Adv}^{\mathbf{intrup}}_{\mathrm{SAEF}[(\pi_0,\pi_1)]}(\mathcal{A})$ and for the confidentiality case, $\mathbf{Adv}^{\mathbf{soprpf}}_{\mathrm{SAEF}[\mathsf{F}]}(\mathcal{A}) \leq \mathbf{Adv}^{\mathbf{prtfp}}_{\mathsf{F}}(\mathcal{B}) + \mathbf{Adv}^{\mathbf{soprpf}}_{\mathrm{SAEF}[(\pi_0,\pi_1)]}(\mathcal{A})$.

Now, the adversary, in the integrity case is left to distinguish between the games $\mathbf{intrup\text{-}real}_{\mathrm{SAEF}[(\pi_0,\pi_1)]}$ (let say the "real-int world") and $\mathbf{intrup\text{-}ideal}_{\mathrm{SAEF}[(\pi_0,\pi_1)]}$ (let say the "ideal-int world") and for the confidentiality case is left to distinguish between the games $\mathbf{soprpf\text{-}real}_{\mathrm{SAEF}[(\pi_0,\pi_1)]}$ (let say the "real-conf world") and $\mathbf{soprpf\text{-}ideal}_{\mathrm{SAEF}[(\pi_0,\pi_1)]}$ (let say the "ideal-conf world").

Transcripts. Following the coefficients H technique [38], we describe the interactions of $\mathcal{A}$ with its integrity (i.e., **intrup**) oracles in a transcript:

$$\tau = \langle (N^i, A^i, M^i, C^i)_{i=1}^{q_e}, (\bar{N}^i, \bar{A}^i, \bar{M}^i, \bar{C}^i)_{i=1}^{q_d}, (\tilde{N}^i, \tilde{A}^i, \tilde{C}^i, b^i)_{i=1}^{q_v} \rangle$$

and with its confidentiality (i.e., **soprpf**) oracles in a transcript:

$$\tau = \langle (N^i, A^i, M^i, C^i)_{i=1}^{q_e}, (\bar{N}^i, \bar{A}^i, \bar{M}^i, \bar{C}^i)_{i=1}^{q_d} \rangle \, .$$

For the i-th encryption query (N^i, A^i, M^i) with output C^i, SAEF processes A^i, M^i, and C^i in blocks $A^i_1, \ldots, A^i_{a^i}, A^i_*$, $M^i_1, \ldots, M^i_{m^i}, M^i_*$, and $C^i_1, \ldots, C^i_{m^i}, C^i_*, T^i$ (as defined in the SAEF algorithms, Figure 3). Here, a^i and m^i are the lengths of A^i and M^i in n-bit blocks, respectively ($a^i = |A^i|_n - 1$ and

$m^i = |M^i|_n - 1$). SAEF also uses internal chaining values as whitening masks, denoted by Δ. The masks for processing $A^i_1, \dots, A^i_{a^i}, A^i_*$ are $\Delta^i_1, \dots, \Delta^i_{a^i+1}$, and for $M^i_1, \dots, M^i_{m^i}, M^i_*$, they are $\Delta^i_{a^i+2}, \dots, \Delta^i_{a^i+m^i+2}$.

For the i-th decryption query $(\bar{N}^i, \bar{A}^i, \bar{C}^i)$ with output $\bar{M}^i$, SAEF processes $\bar{A}^i, \bar{C}^i$, and $\bar{M}^i$ in blocks $\bar{A}^i_1, \dots, \bar{A}^i_{\bar{a}^i}, \bar{A}^i_*$, $\bar{C}^i_1, \dots, \bar{C}^i_{\bar{m}^i}, \bar{C}^i_*, \bar{T}^i$, and $\bar{M}^i_1, \dots, \bar{M}^i_{\bar{m}^i}, \bar{M}^i_*$. Here, $\bar{a}^i$ and $\bar{m}^i$ are the lengths of $\bar{A}^i$ and $\bar{C}^i$ in n-bit blocks ($\bar{a}^i = |\bar{A}^i|_n - 1$ and $\bar{m}^i = |\bar{C}^i|_n - 2$). The whitening masks for $\bar{A}^i_1, \dots, \bar{A}^i_{\bar{a}^i}, \bar{A}^i_*$ are $\bar{\Delta}^i_1, \dots, \bar{\Delta}^i_{\bar{a}^i+1}$, and for $\bar{C}^i_1, \dots, \bar{C}^i_{\bar{m}^i}, \bar{C}^i_*$, they are $\bar{\Delta}^i_{\bar{a}^i+2}, \dots, \bar{\Delta}^i_{\bar{a}^i+\bar{m}^i+2}$.

For the i-th verification query $(\tilde{N}^i, \tilde{A}^i, \tilde{C}^i)$ with output $b^i \in \{\top, \bot\}$, SAEF processes $\tilde{A}^i$ and $\tilde{C}^i$ in blocks $\tilde{A}^i_1, \dots, \tilde{A}^i_{\tilde{a}^i}, \tilde{A}^i_*$ and $\tilde{C}^i_1, \dots, \tilde{C}^i_{\tilde{m}^i}, \tilde{C}^i_*, \tilde{T}^i$. Here, $\tilde{a}^i$ and $\tilde{m}^i$ are the lengths of $\tilde{A}^i$ and $\tilde{C}^i$ in n-bit blocks ($\tilde{a}^i = |\tilde{A}^i|_n - 1$ and $\tilde{m}^i = |\tilde{C}^i|_n - 2$). The whitening masks for $\tilde{A}^i_1, \dots, \tilde{A}^i_{\tilde{a}^i}, \tilde{A}^i_*$ are $\tilde{\Delta}^i_1, \dots, \tilde{\Delta}^i_{\tilde{a}^i+1}$, and for $\tilde{C}^i_1, \dots, \tilde{C}^i_{\tilde{m}^i}, \tilde{C}^i_*$, they are $\tilde{\Delta}^i_{\tilde{a}^i+2}, \dots, \tilde{\Delta}^i_{\tilde{a}^i+\tilde{m}^i+2}$.

Additional Information. To simplify proof analysis, we provide the adversary with encryption masks Δ^i_j and decryption masks $\bar{\Delta}^i_j$ in both integrity and confidentiality settings. Additionally, in the integrity setting, the adversary receives the internally computed plaintexts $\tilde{M}^i_j$ and verification masks $\tilde{\Delta}^i_j$, while in the confidentiality setting, the adversary receives the tag bits normally discarded by truncation, i.e., now $|T^i| = n$ for $1 \le i \le q_e$.

This additional information can only aid the adversary by increasing its advantage, thus can be considered here for upper bounding the adversarial advantage in the targeted setting. We note that this additional information is provided to the adversary (in the corresponding setting) when it has made all its queries and only the final response is pending.

Block-Tuple Representation. To streamline notation and ease analysis, we switch to a *block-tuple* representation by defining the i-th encryption query as $(\mathsf{T}^i_j, \Delta^i_j, X^i_j, Y^i_j)^{\ell^i}_{j=1}, T^i$, with $\ell^i = a^i + m^i + 2$. The j-th quadruple in this tuple represents the processing in the j-th forkcipher call, where T^i_j is the forkcipher tweak, Δ^i_j is the whitening mask, X^i_j is the associated data/plaintext block, and Y^i_j is the empty/ciphertext block. Specifically:

- For the first block, $\mathsf{T}^i_1 = N\|1\|F$ with $F \in \{0,1\}^3$ and $\Delta^i_1 = 0^n$. For $j > 1$ we have $\mathsf{T}^i_j = 0^{t-3}\|F$ with $F \in \{0,1\}^3$.
- If $|A| > 0$, for $1 \le j \le a^i$ we have $X^i_j = A^i_j$, $Y^i_j = \varepsilon$, and $F = 000$. For $j = a^i + 1$ we have $X^i_j = \mathsf{pad10}(A^i_*)$, $Y^i_j = \varepsilon$, and $F \in \{0,1\}^3$ as in Figure 3.
- If $|M| > 0$, for $a^i + 2 \le j < \ell^i$ we have $X^i_j = M^i_j$, $Y^i_j = C^i_j$, and $F = 001$. For $j = \ell^i$ we have $X^i_j = \mathsf{pad10}(M^i_*)$, $Y^i_j = C^i_*$, and $F \in \{0,1\}^3$ as in Figure 3.
- If $A = M = \varepsilon$ we have $j = \ell^i = 1$, $X^i_j = \mathsf{pad10}(\varepsilon)$, $Y^i_j = \varepsilon$, and $F = 111$.

Similarly, the block-tuple representation for decryption queries is $(\bar{\mathsf{T}}^i_j, \bar{\Delta}^i_j, \bar{X}^i_j, \bar{Y}^i_j)^{\bar{\ell}^i}_{j=1}, \bar{T}^i$ with $\bar{\ell}^i = \bar{m}^i + \bar{a}^i + 2$, and for verification queries, it is $(\tilde{\mathsf{T}}^i_j, \tilde{\Delta}^i_j, \tilde{X}^i_j, \tilde{Y}^i_j)^{\tilde{\ell}^i}_{j=1}, \tilde{T}^i, b^i$ with $\tilde{\ell}^i = \tilde{m}^i + \tilde{a}^i + 2$. We re-index the decryption queries from $q_e + 1$ to $q_e + q_d$ and the verification queries from $q_e + q_d + 1$

to $q_e + q_d + q_v$. This re-indexing allows us to drop the bars and tildes from variables, denoting them as $(\mathsf{T}^i_j, \Delta^i_j, X^i_j, Y^i_j)_{j=1}^{\ell^i}, T^i$ for $q_e + 1 \le i \le q_e + q_d$ and $(\mathsf{T}^i_j, \Delta^i_j, X^i_j, Y^i_j)_{j=1}^{\ell^i}, T^i, b^i$ for $q_e + q_d + 1 \le i \le q_e + q_d + q_v$.

The transcript of $\mathcal{A}$ consists of all query-response tuples, uniquely defined at the end of its interaction. This transcript remains invariant regardless of the order of encryption, decryption, and verification queries, making the re-indexing a valid step that simplifies notation in transcripts.

The block-tuple representation for SAEF mode was initially introduced in [4], with an equivalence proof showing how to reconstruct the original representation. We refer readers to [4, Appendix A] for the full proof of equivalence.

Blockwise Common Prefix of Queries. With the block-tuple notation, we can simply define the length of the longest common prefix between the i-th and i'-th query with $\ell^i \le \ell^{i'}$ as

$$\mathsf{llcp}_n(i, i') = \max\{1 \le u \le \ell^i \mid (\mathsf{T}^i_j, \Delta^i_j, X^i_j, Y^i_j) = (\mathsf{T}^{i'}_j, \Delta^{i'}_j, X^{i'}_j, Y^{i'}_j) \text{ for } 1 \le j \le u\}.$$

This definition covers common blockwise prefixes between all types of query pairs (e.g., between two encryption queries, an encryption and a decryption query, a decryption and a verification query, etc.). Informally, $\mathsf{llcp}_n(i, i')$ represents the number of internal chaining values Δs that are equal between the i-th and i'-th query. For instance, if the nonces N^i and $N^{i'}$ are different then $\mathsf{llcp}_n(i, i') = 0$. If we have two queries with $N^i = N^{i'}$ but the i'^{th} query has AD $A^{i'} = A^i \| M^i_1$ i.e., equal to the AD of the i^{th} query appended with its first message block (and the rest of these messages have no common prefix), we will still have $\mathsf{llcp}_n(i, i') = a^i + 1$, due to the inclusion of tweak strings in the block tuples.

We now define the length of the longest common blockwise prefix of a query *with all previous queries* as $\mathsf{llcp}_n(i) = \max_{1 \le i' < i} \mathsf{llcp}_n(i, i')$. We also note that for verification queries, all encryption and decryption queries are always taken into account (as per the convention of query indexing).

Extended Transcripts. Using block-tuple notation, we redefine the extended transcripts of $\mathcal{A}$ with its integrity oracles as follows:

$$\tau = \left\langle \left(\left(\mathsf{T}^i_j, \Delta^i_j, X^i_j, Y^i_j \right)_{j=1}^{\ell^i}, T^i \right)_{i=1}^{q_e+q_d}, \left(\left(\mathsf{T}^i_j, \Delta^i_j, X^i_j, Y^i_j \right)_{j=1}^{\ell^i}, T^i, b^i \right)_{i=q_e+q_d+1}^{q_e+q_d+q_v} \right\rangle$$

and with its confidentiality oracles, as follows:

$$\tau = \left\langle \left(\left(\mathsf{T}^i_j, \Delta^i_j, X^i_j, Y^i_j \right)_{j=1}^{\ell^i}, T^i \right)_{i=1}^{q_e+q_d} \right\rangle.$$

Note that the terms q_e, q_d, q_v, a, and m are random variables and may vary for different attainable transcripts. However, assuming the adversary can make at most σ block queries, we have $\sum_{i=1}^{q_e+q_d+q_v} (a^i + m^i + 2) = \sigma$ for the integrity setting and $\sum_{i=1}^{q_e+q_d} (a^i + m^i + 2) = \sigma' \le \sigma$ for the confidentiality setting.

5.1 Integrity Analysis

Sampling of Additional Information. In the real-int world, all additional information variables are computed by oracles that evaluate SAEF. In the ideal-int world, the encryption and decryption oracles also evaluate SAEF, defining the Δ_j^i masks for $1 \leq i \leq q_e + q_d$. However, the verification oracle in the ideal-int world does not perform any computations, leaving Δ_j^i for $q_e + q_d + 1 \leq i \leq q_e + q_d + q_v$ and $\tilde{M}_j^i$ for $1 \leq i \leq q_v$ undefined. Therefore, we need to define the sampling of these variables.

We set $\Delta_1^i = 0^n$ for $q_e + q_d + 1 \leq i \leq q_e + q_d + q_v$ and sample the remaining Δ_j^i masks uniformly and independently at random, except when such a mask is trivially defined due to a "common prefix" with a previous query. More formally, for the i-th query with $q_e + q_d + 1 \leq i \leq q_e + q_d + q_v$ and $1 < j \leq \mathsf{llcp}_n(i) + 1$, we let $\Delta_j^i = \Delta_j^{i'}$ for the smallest $i' < i$ such that the i'-th query has $\mathsf{llcp}_n(i) = \mathsf{llcp}_n(i, i')$. For the remaining block-tuples with $\mathsf{llcp}_n(i) + 1 < j \leq \ell^i$, Δ_j^i is sampled uniformly at random. Once these masks are sampled, we use the SAEF decryption algorithm with π_0 and these masks to compute $\tilde{M}_j^i$.

Coefficients H. Let Θ_{rein} and Θ_{idin} represent the distribution of the transcript in the real-int world and the ideal-int world, respectively. The proof relies on the fundamental lemma of the coefficients H technique as defined in Lemma 1. We represent the j-th block call of the i-th query in a transcript by the index tuple (i, j). An attainable transcript τ is considered bad if any of the following conditions occur:

BadT_1 (Input Collision): There exists $(i', j') < (i, j)$ such that $1 \leq i \leq q_e + q_d + q_v$, $\mathsf{llcp}_n(i) < j \leq \ell^i$ (not in the longest common prefix), and the (i, j) block call has a tweak-input collision with the (i', j') block call, i.e., $\mathsf{T}_j^i = \mathsf{T}_{j'}^{i'}$ and $X_j^i \oplus \Delta_j^i = X_{j'}^{i'} \oplus \Delta_{j'}^{i'}$.

BadT_2 (Mask Collision): There exists $(i', j') < (i, j)$ such that $1 \leq i \leq q_e + q_d + q_v$, $\mathsf{llcp}_n(i) < j < \ell^i$, and both block calls have the same tweaks $\mathsf{T}_j^i = \mathsf{T}_{j'}^{i'}$ and different inputs $X_j^i \oplus \Delta_j^i \neq X_{j'}^{i'} \oplus \Delta_{j'}^{i'}$, but the subsequent masks $\Delta_{j+1}^i = \Delta_{j'+1}^{i'}$ collide.

BadT_3 (Forgery): There exists $q_e + q_d + 1 \leq i \leq q_e + q_d + q_v$ such that for $j = \ell^i$ we have any of the following:

Case 1: The last bit of T_j^i is 0 and $\pi_{\mathsf{T}_j^i,1}(X_j^i \oplus \Delta_j^i) = T^i$.

Case 2: The last bit of T_j^i is 1, $\mathsf{right}_{n-|T^i|}(X_j^i) = 10^{n-|T^i|-1}$ and $\mathsf{left}_{|T^i|}(\pi_{\mathsf{T}_j^i,1}(X_j^i \oplus \Delta_j^i)) = T^i$.

Case 3: The last bit of T_j^i is 1, and there exists $q_e + 1 \leq i_d \leq q_e + q_d$ with $\mathsf{T}_{\ell^{i_d}}^{i_d} = 1$ and $|T^i| = |T^{i_d}|$ such that $\mathsf{right}_{n-|T^{i_d}|}(X_{\ell^{i_d}}^{i_d}) = 10^{n-|T^{i_d}|-1}$ and $\mathsf{left}_{|T^i|}(\pi_{\mathsf{T}_j^i,1}(X_j^i \oplus \Delta_j^i)) = T^i$.

We emphasize that Case 3 of BadT_3 is not required in existing integrity analyses [4, 7] of SAEF but is necessary under the RUP setting, where an adversary can observe unverified plaintexts. In essence, it captures a scenario where an adversary makes decryption queries with incomplete last blocks that only differ in the last complete ciphertext blocks ($\bar{C}_*$) to find an unverified padding matching 10^{w-1} for some $1 \le w \le n-1$. Once found, the adversary uses the same ciphertext with its tag part (of size $n-w$ bits) replaced with random strings as forgeries.

We define $\mathcal{T}_{\text{bad}}$ as the set of "bad" transcripts, a subset of attainable transcripts for which the predicate $\mathsf{BadT}(\tau) = (\mathsf{BadT}_1(\tau) \vee \mathsf{BadT}_2(\tau) \vee \mathsf{BadT}_3(\tau)) = 1$. Conversely, $\mathcal{T}_{\text{good}}$ represents the attainable transcripts not in $\mathcal{T}_{\text{bad}}$, thus considered "good" transcripts.

Lemma 2. *For $\mathcal{T}_{bad}$ above and $q_e + q_d \le 2^{n-1}$, we have*

$$\Pr[\Theta_{idin} \in \mathcal{T}_{bad}] \le \frac{\sigma^2}{2^n} + \frac{4 \cdot q_d q_v}{2^n}\,.$$

Lemma 3. *Let $\tau \in \mathcal{T}_{good}$, meaning τ is a good transcript. Then* $\frac{\Pr[\Theta_{rein}=\tau]}{\Pr[\Theta_{idin}=\tau]} \ge 1$.

With the well-defined bad events, both lemmas can be proved using standard probability analysis. We defer the proof of Lemma 2 and 3 to Appendix C. Combining the results of Lemma 2 and 3 (taking $\epsilon_1 = 0$) into Lemma 1, we obtain the upper bound $\mathbf{Adv}^{\mathbf{intrup}}_{\mathrm{SAEF}[(\pi_0,\pi_1)]}(\mathcal{A}) \le \frac{\sigma^2}{2^n} + \frac{4 \cdot q_d q_v}{2^n}$ and thus the integrity result of Theorem 1.

5.2 Confidentiality Analysis

Sampling of Additional Information. In the real-conf world, all additional information variables are computed internally by the SAEF encryption and decryption algorithms. However, in the ideal-conf world, these variables are not defined, as outputs are directly computed by an online permutation (or its inverse), a random permutation (or its inverse), and a random function, with tags being sampled directly to the desired length. Consequently, in the ideal-conf world, we sample the masks uniformly at random while ensuring consistency with SAEF's prefix preservation, and each authentication tag is extended by appending 0 to $n-1$ uniform random bits as needed.

Formally, for each $1 \le i \le q_e + q_d$, we set $\Delta^i_j = \Delta^{i'}_j$ for $1 \le j \le \mathsf{llcp}_n(i) + 1$, where i' is the smallest index less than i such that the i'-th query has $\mathsf{llcp}_n(i) = \mathsf{llcp}_n(i, i')$. For $\mathsf{llcp}_n(i) + 1 < j \le \ell^i$, the mask Δ^i_j is sampled uniformly at random. This approach to sampling Δ masks is not dependent on the sequential indexing of queries, meaning it applies equally well to an adaptive adversary with encryption and decryption queries in any order.

Coefficients H. Let Θ_{reco} and Θ_{idco} denote the distribution of the transcripts in the real-conf world and the ideal-conf world, respectively.

The proof relies on the fundamental lemma of the coefficients H technique, as defined in Lemma 1. We represent the j-th block call of the i-th query in a transcript by the index tuple (i, j). We consider an attainable transcript τ to be bad if any of the following conditions occur:

BadT_1 (Input Collision): There exists $(i', j') < (i, j)$ such that $1 \le i \le q_e + q_d$, $\mathsf{llcp}_n(i) < j \le \ell^i$, and the (i, j) block call has a tweak-input collision with the (i', j') block call, i.e., $\mathsf{T}^i_j = \mathsf{T}^{i'}_{j'}$ and $X^i_j \oplus \Delta^i_j = X^{i'}_{j'} \oplus \Delta^{i'}_{j'}$.

BadT_2 (Output Collision): There exists $(i', j') < (i, j)$ such that $1 \le i \le q_e + q_d$, $\mathsf{llcp}_n(i) < j < \ell^i$, and both block calls have the same tweaks $\mathsf{T}^i_j = \mathsf{T}^{i'}_{j'}$ and different inputs $X^i_j \oplus \Delta^i_j \ne X^{i'}_{j'} \oplus \Delta^{i'}_{j'}$. However, one of the outputs collides, i.e., one of the following is true:

I. $j = \mathsf{llcp}_n(i) + 1$ and **(i)** $\Delta^i_{j+1} = \Delta^{i'}_{j'+1}$ if $j < \ell^i$ and $j' < \ell^{i'}$, or **(ii)** $T^i = T^{i'}$ if $1 \le i \le q_e$, $j = \ell^i$ and $j' = \ell^{i'}$.

II. $j > \mathsf{llcp}_n(i) + 1$ and **(i)** $Y^i_j \oplus \Delta^i_j = Y^{i'}_{j'} \oplus \Delta^{i'}_{j'}$ if $j > a^i + 1$, or **(ii)** $\Delta^i_{j+1} = \Delta^{i'}_{j'+1}$ if $j < \ell^i$ and $j' < \ell^{i'}$, or **(iii)** $T^i = T^{i'}$ if $1 \le i \le q_e$, $j = \ell^i$ and $j' = \ell^{i'}$.

We define $\mathcal{T}'_{\text{bad}}$ as the set of "bad" transcripts, a subset of attainable transcripts where the predicate $\mathsf{BadT}'(\tau) = (\mathsf{BadT}'_1(\tau) \vee \mathsf{BadT}'_2(\tau))$ is true. Conversely, $\mathcal{T}'_{\text{good}}$ represents the attainable transcripts not in $\mathcal{T}'_{\text{bad}}$, thus considered "good" transcripts.

Lemma 4. *For $\mathcal{T}'_{bad}$ as defined above,*

$$\Pr[\Theta_{idco} \in \mathcal{T}'_{bad}] \le \frac{3 \cdot \sigma^2}{2^{n+1}} \,.$$

Lemma 5. *Let $\tau \in \mathcal{T}'_{good}$, meaning τ is a good transcript. Then,* $\frac{\Pr[\Theta_{reco}=\tau]}{\Pr[\Theta_{idco}=\tau]} \ge 1\,.$

With the defined bad events, both lemmas can be proven using standard probability analysis. The proofs are deferred to Appendix C. By combining Lemmas 4 and 5 (with $\epsilon_1 = 0$) into Lemma 1, we derive the upper bound $\mathbf{Adv}^{\mathbf{soprpf}}_{\mathrm{SAEF}[(\pi_0,\pi_1)]}(\mathcal{A}) \le \frac{3 \cdot \sigma^2}{2^{n+1}}$ thus establishing the confidentiality result of Theorem 1. □

6 Conclusion

We propose a RUP confidentiality notion for online AE schemes named sOPRPF which can be seen as the strong version of OPRPF [4] where the adversary is now allowed to see decryption of chosen ciphertexts as well. In terms of RUP security, sOPRPF captures confidentiality for online AE schemes up to the leakage of common prefix with the released unverified plaintexts. We define a strong AE security notion called OAE-RUP as sOPRPF+INT-RUP which is a stronger notion than OAE [4] and is the best achievable option available for online AE schemes jointly against nonce-misuse, blockwise adaptive and/or RUP adversaries. We also compare popular AEAD notions to OAE-RUP to highlight its relevance.

We prove that SAEF ensures OAE-RUP security provided the total data processed with a single key is $\ll 2^{n/2}$ blocks, where n is the block size of the underlying forkcipher. This means SAEF maintains reasonable security even when unverified plaintext is released. Specifically, SAEF's integrity remains intact while its confidentiality is preserved up to the longest common prefix. These security properties are significant for lightweight cryptography applications, where constrained devices might (or are required to) leak portions of unverified plaintext during decryption and could suffer from nonce repetitions.

We also propose *confidentiality resilience under RUP* (CR-RUP) as a generalized and intuitive notion of confidentiality for online AE schemes, similar to nonce misuse resilience (NMR) [8]. Although weaker than sOPRPF, CR-RUP ensures meaningful security by protecting plaintexts not directly subject to leakage.

Future research could explore the CR-RUP and sOPRPF security of other existing INT-RUP secure online AE schemes.

Acknowledgments

Amit Singh Bhati was supported by CyberSecurity Research Flanders with reference number VR20192203, in part by the Research Council KU Leuven C1 on Security and Privacy for Cyber-Physical Systems and the Internet of Things with contract number C16/15/058 and by the Flemish Government through FWO Project G.0835.16 A security Architecture for IoT. Elena Andreeva was supported by the Austrian Science Fund (FWF) SpyCoDe grant with number 10.55776/F8507-N.

References

1. Abed, F., Fluhrer, S., Forler, C., List, E., Lucks, S., McGrew, D., Wenzel, J.: Pipelineable on-line encryption. In: International Workshop on Fast Software Encryption. pp. 205–223. Springer (2014)
2. Al Fardan, N.J., Paterson, K.G.: Lucky thirteen: Breaking the TLS and DTLS record protocols. In: 2013 IEEE symposium on security and privacy. pp. 526–540. IEEE (2013)
3. AlFardan, N., Paterson, K.G.: Plaintext-recovery attacks against datagram TLS. In: Network and distributed system security symposium (NDSS 2012) (2012)
4. Andreeva, E., Bhati, A.S., Vizár, D.: Nonce-Misuse Security of the SAEF Authenticated Encryption Mode. In: Selected Areas in Cryptography. pp. 512–534. Springer (2021)
5. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., Yasuda, K.: How to securely release unverified plaintext in authenticated encryption. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 105–125. Springer (2014)
6. Andreeva, E., Lallemand, V., Purnal, A., Reyhanitabar, R., Roy, A., Vizár, D.: ForkAE v. Submission to NIST LwC Standardization Process (2019)

7. Andreeva, E., Lallemand, V., Purnal, A., Reyhanitabar, R., Roy, A., Vizár, D.: Forkcipher: a New Primitive for Authenticated Encryption of Very Short Messages. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 153–182. Springer (2019)
8. Ashur, T., Dunkelman, O., Luykx, A.: Boosting authenticated encryption robustness with minimal modifications. In: Annual International Cryptology Conference. pp. 3–33. Springer (2017)
9. Banik, S., Chakraborti, A., Inoue, A., Iwata, T., Minematsu, K., Nandi, M., Peyrin, T., Sasaki, Y., Sim, S.M., Todo, Y.: Gift-cofb. Cryptology ePrint Archive (2020)
10. Barwell, G., Page, D., Stam, M.: Rogue decryption failures: Reconciling AE robustness notions. In: Cryptography and Coding: 15th IMA International Conference, IMACC 2015, Oxford, UK, December 15-17, 2015. Proceedings 15. pp. 94–111. Springer (2015)
11. Bellare, M., Boldyreva, A., Knudsen, L., Namprempre, C.: Online ciphers and the hash-CBC construction. In: Annual International Cryptology Conference. pp. 292–309. Springer (2001)
12. Bellare, M., Namprempre, C.: Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 531–545. Springer (2000)
13. Bellizia, D., Berti, F., Bronchain, O., Cassiers, G., Duval, S., Guo, C., Leander, G., Leurent, G., Levi, I., Momin, C., Pereira, O., Peters, T., Standaert, F.X., Udvarhelyi, B., Wiemer, F.: Spook: Sponge-Based Leakage-Resistant Authenticated Encryption with a Masked Tweakable Block Cipher. IACR Transactions on Symmetric Cryptology **2020**(1), 295–349 (2020)
14. Bernstein, D.J.: Cryptographic competitions: CAESAR. `http://competitions.cr.yp.to`
15. Bhati, A.S., Andreeva, E., Vizar, D., Deprez, A., Pittevils, J., Roy, A.: New Results and Insights on ForkAE. NIST Lightweight Cryptography Workshop 2020
16. Bhattacharjee, A., List, E., López, C.M., Nandi, M.: The Oribatida Family of Lightweight Authenticated Encryption Schemes. Indian Statistical Institute Kolkata: Kolkata, India p. 2019 (2019)
17. Canvel, B., Hiltgen, A., Vaudenay, S., Vuagnoux, M.: Password interception in a SSL/TLS channel. In: Advances in Cryptology-CRYPTO 2003: 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003. Proceedings 23. pp. 583–599. Springer (2003)
18. Chakraborti, A., Datta, N., Jha, A., Mancillas-López, C., Nandi, M., Sasaki, Y.: INT-RUP Secure Lightweight Parallel AE Modes. IACR Transactions on Symmetric Cryptology pp. 81–118 (2019)
19. Chakraborti, A., Datta, N., Jha, A., Mancillas-López, C., Nandi, M., Sasaki, Y.: ESTATE: A lightweight and low energy authenticated encryption mode. IACR Transactions on Symmetric Cryptology pp. 350–389 (2020)
20. Chakraborti, A., Datta, N., Jha, A., Mitragotri, S., Nandi, M.: From combined to hybrid: Making feedback-based AE even smaller. IACR Transactions on Symmetric Cryptology pp. 417–445 (2020)
21. Chakraborti, A., Datta, N., Nandi, M.: INT-RUP analysis of block-cipher based authenticated encryption schemes. In: Cryptographers' Track at the RSA Conference. pp. 39–54. Springer (2016)
22. Chakraborty, B., Nandi, M.: The mf mode of authenticated encryption with associated data. Journal of Mathematical Cryptology **16**(1), 73–97 (2022)

23. Chang, D., Datta, N., Dutta, A., Mennink, B., Nandi, M., Sanadhya, S., Sibleyras, F.: Release of unverified plaintext: Tight unified model and application to ANYDAE. IACR Transactions on Symmetric Cryptology pp. 119–146 (2019)
24. Datta, N., Dutta, A., Ghosh, S.: INT-RUP security of SAEB and TinyJAMBU. In: International Conference on Cryptology in India. pp. 146–170. Springer (2022)
25. Datta, N., Luykx, A., Mennink, B., Nandi, M.: Understanding RUP Integrity of COLM. IACR Transactions on Symmetric Cryptology pp. 143–161 (2017)
26. Fleischmann, E., Forler, C., Lucks, S.: McOE: A Family of Almost Foolproof On-Line Authenticated Encryption Schemes. In: Fast Software Encryption. vol. 7549, pp. 196–215. Springer (2012)
27. Gueron, S., Jha, A., Nandi, M.: Comet: counter mode encryption with authentication tag. Second Round Candidate of the NIST LWC Competition (2019)
28. Hirose, S., Sasaki, Y., Yasuda, K.: Rate-one AE with security under RUP. In: Information Security: 20th International Conference, ISC 2017, Ho Chi Minh City, Vietnam, November 22-24, 2017, Proceedings 20. pp. 3–20. Springer (2017)
29. Hoang, V.T., Krovetz, T., Rogaway, P.: Robust authenticated-encryption AEZ and the problem that it solves. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 15–44. Springer (2015)
30. Hoang, V.T., Reyhanitabar, R., Rogaway, P., Vizár, D.: Online authenticated-encryption and its nonce-reuse misuse-resistance. In: Annual International Cryptology Conference. pp. 493–517. Springer (2015)
31. Imamura, K., Minematsu, K., Iwata, T.: Integrity analysis of authenticated encryption based on stream ciphers. International Journal of Information Security **17**, 493–511 (2018)
32. Iwata, T., Khairallah, M., Minematsu, K., Peyrin, T.: Duel of the Titans: The Romulus and Remus Families of Lightweight AEAD Algorithms. IACR Transactions on Symmetric Cryptology **2020**(1), 43–120 (2020)
33. Iwata, T., Yasuda, K.: BTM: A single-key, inverse-cipher-free mode for deterministic authenticated encryption. In: International Workshop on Selected Areas in Cryptography. pp. 313–330. Springer (2009)
34. Iwata, T., Yasuda, K.: HBS: A single-key mode of operation for deterministic authenticated encryption. In: International Workshop on Fast Software Encryption. pp. 394–415. Springer (2009)
35. Naito, Y., Matsui, M., Sugawara, T., Suzuki, D.: SAEB: a lightweight blockcipher-based AEAD mode of operation. Cryptology ePrint Archive (2019)
36. NIST: DRAFT Submission Requirements and Evaluation Criteria for the Lightweight Cryptography Standardization Process. `https://csrc.nist.gov/Projects/Lightweight-Cryptography` (2018)
37. NIST: NIST Workshop on the Requirements for an Accordion Cipher Mode 2024 . `https://csrc.nist.gov/Events/2024/accordion-cipher-mode-workshop-2024` (2024)
38. Patarin, J.: The “coefficients H” technique. In: International Workshop on Selected Areas in Cryptography. pp. 328–345. Springer (2008)
39. Purnal, A., Andreeva, E., Roy, A., Vizár, D.: What the Fork: Implementation Aspects of a Forkcipher. In: NIST Lightweight Cryptography Workshop 2019 (2019)
40. Rogaway, P.: Authenticated-Encryption with Associated-Data. In: ACM conference on Computer and communications security. pp. 98–107 (2002)
41. Rogaway, P., Shrimpton, T.: A provable-security treatment of the key-wrap problem. In: Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 373–390. Springer (2006)

42. Vaudenay, S.: Security flaws induced by CBC padding—applications to SSL, IPSEC, WTLS... In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 534–545. Springer (2002)
43. Wu, H., Huang, T.: TinyJAMBU: A family of lightweight authenticated encryption algorithms. Submission to NIST LwC Standardization Process (2019)
44. Zhang, P., Wang, P., Hu, H., Cheng, C., Kuai, W.: INT-RUP security of checksum-based authenticated encryption. In: Provable Security: 11th International Conference, ProvSec 2017, Xi'an, China, October 23-25, 2017, Proceedings 11. pp. 147–166. Springer (2017)

A Table 1: Full Details

In this section, we first revise the (security) properties as described in Table 1 and then explain how each checkmark entry is derived.

1. Online AE (OAE) Security [4]: Ensures the AE mode can be implemented online with reasonable security, protecting against blockwise and nonce-misusing adversaries.

2. Nonce-Misuse Resilience (NMR) Security [8]: Provides security even when the nonce is repeated in other queries, safeguarding against specific nonce-misusing adversaries.

3. Misuse-Resistant AE (MRAE) Security [41]: A stronger version of NMR, ensuring security against nonce-misusing adversaries. MRAE is more robust than OAE and NMR but requires at least two passes over plaintext data, making it unsuitable for online implementations.

4. Integrity under RUP (INT-RUP) [5]: Ensures integrity even when unverified plaintexts are released, protecting against adversaries seeing unverified decrypted plaintexts.

5. Plaintext Awareness (PA) [5]: Combined with IND-CPA, it ensures confidentiality even when unverified plaintexts are released. PA requires at least two passes over plaintext data for encryption, making it incompatible with OAE security (and online implementations).

6. sOPRPF Security 2.2: Ensures confidentiality (up to the longest common prefix) even when unverified plaintexts are released and the nonce is repeated. sOPRPF is weaker than PA but is suitable for online implementations. It is kept separate from INT-RUP in Table 1 as some schemes only provide INT-RUP security.

7. One-pass Encryption and Decryption: Ensures the mode requires only one pass over the data, supporting online encryption and decryption. While OAE-RUP implies single-pass encryption and decryption, the reverse is not always true.

A.1 Results in Table 1

We now describe the results of Table 1. The MRAE security of ESTATE and Romulus-M is proven in [19] and [32], respectively which by definition implies the NMR security of them. Both of these modes require passing the message twice to encrypt it, i.e., they do not support one-pass encryption and OAE security. We note that for decryption, these modes only require passing the ciphertext once i.e. they have one-pass decryption and are proven INT-RUP secure in [19] and [32], respectively. We also note that in [32], Romulus-M is proven IND-CPA+PA1 using a general argument which says an SIV type MAC-then-Encrypt AEAD mode with MAC as a PRF and encryption as a PA1 scheme is IND-CPA+PA1. This implies that ESTATE which also follows the same composition idea with CBC-style MAC part (shown to be a PRF in [19]) and OFB encryption (can be similarly shown PA1 as CTR mode is shown in [5]) is also IND-CPA+PA1.

All the rest modes in Table 1 provide one-pass encryption and decryption and hence are neither MRAE nor IND-CPA+PA1 secure. The INT-RUP security of Oribatida, LOCUS/LOTUS, and TinyJAMBU is proven in [16], [18] and [24], respectively. The NMR security of Spook and TinyJAMBU is proven in [13] and [43], respectively and the OAE security of SAEF is proven in [4] which by definition implies its NMR security. Finally, the sOPRPF+INT-RUP (or jointly named as OAE-RUP) security of SAEF mode is proven in this work.

B CR-RUP Security: A Weaker Alternative to sOPRPF

We propose *confidentiality resilience under release of unverified plaintext* (CR-RUP), a basic security notion that targets full confidentiality for "unleaked plaintexts" i.e., plaintexts that are not subject to leakage, has unique nonces and has no common prefixes with the leaked plaintexts. More concretely, let us define two games, $\mathsf{CR\text{-}RUP}\textbf{-real}_\Pi$ and $\mathsf{CR\text{-}RUP}\textbf{-ideal}_\Pi$. In both games, the adversary is given access to an encryption, a decryption, and a verification oracle. In the game $\mathsf{CR\text{-}RUP}\textbf{-real}_\Pi$, all three oracles faithfully implement the corresponding algorithms of Π using the same randomly sampled secret key. In the game $\mathsf{CR\text{-}RUP}\textbf{-ideal}_\Pi$, the decryption and verification oracles are the same as in $\mathsf{CR\text{-}RUP}\textbf{-real}_\Pi$ but the encryption oracle is replaced with a uniform random function f (with the same input and output signature as in the real world).

Now, under the RUP setting, where the adversary is allowed to observe the unverified plaintext, the CR-RUP advantage can be defined as follows:

Definition 3 (CR-RUP Advantage). *Let $\mathcal{A}$ be a computationally bounded adversary with access to an encryption, a decryption, and a verification oracle namely $\mathcal{E}_K, \mathcal{D}_K$, and $\mathcal{V}_K$ for Π for some $K \xleftarrow{\$} \mathcal{K}$. Let $\mathcal{A}$ is not allowed to use the same nonce in both encryption and decryption queries, i.e., encryption responses are prefix-free from decryption queries. Also, let $\mathcal{A}$ is not allowed to repeat a nonce over encryption queries, i.e., $\mathcal{A}$ is nonce-respecting. Let $\mathsf{CR\text{-}RUP}\textbf{-real}_\Pi$ and $\mathsf{CR\text{-}RUP}\textbf{-ideal}_\Pi$ be two games as defined above. The* CR-RUP *advantage of*

$\mathcal{A}$ against Π is then defined as

$$\mathbf{Adv}_{\Pi}^{\mathsf{CR\text{-}RUP}}(\mathcal{A}) = \Pr[\mathcal{A}^{\mathsf{CR\text{-}RUP\text{-}}\mathbf{real}_{\Pi}}] - \Pr[\mathcal{A}^{\mathsf{CR\text{-}RUP\text{-}}\mathbf{ideal}_{\Pi}}].$$

We note that CR-RUP by definition is a strictly stronger notion than IND-CPA (under the nonce-respecting setting) and a strictly weaker notion than sOPRPF as it only claims full confidentiality of plaintexts that contain unique nonces and are not subject to decryption leakage. sOPRPF, on the other hand, additionally claims confidentiality for plaintexts that contain repeated nonces and/or are partially leaked under unverified decryption. More specifically, sOPRPF provides confidentiality up to leaking the length of the common prefix for plaintexts that share the same nonce and AD and up to leaking the common prefix for plaintexts that share prefixes with leaked unverified plaintexts.

C Omitted Lemma Proofs

C.1 Proof of Lemma 2

Proof. BadT_1**.** For any transcript in $\mathcal{T}_{\text{bad}}$ where BadT_1 is set to 1, there exists at least one pair of block indices $(i', j') < (i, j)$ such that $\mathsf{llcp}_n(i) < j \leq \ell^i$ and $\Delta^i_j \oplus \Delta^{i'}_{j'} = X^i_j \oplus X^{i'}_{j'}$.

For all $i' < i$ and $j = j' = \mathsf{llcp}_n(i) + 1$, we have $\Delta^i_j = \Delta^{i'}_{j'}$ but $X^i_j \neq X^{i'}_{j'}$, so the probability of the above equality occurring is 0. In contrast, for all $i' \leq i$ and $j' \neq j$ or $j \neq \mathsf{llcp}_n(i) + 1$, the probability that the two masks collide in Θ_{idin} is $1/2^n$. Given that there are σ possible values of (i, j) in a transcript, each with at most σ possible values of (i', j'), we find that $\Pr[\mathsf{BadT_1}(\Theta_{idin}) = 1] \leq \frac{\sigma^2}{2} \cdot \max\left\{0, \frac{1}{2^n}\right\} = \frac{\sigma^2}{2^{n+1}}$.

BadT_2**.** Similarly, for any transcript in $\mathcal{T}_{\text{bad}}$ where BadT_2 is set to 1, there exists at least one pair $(i', j') < (i, j)$ such that $\mathsf{llcp}_n(i) < j < \ell^i$ and $\Delta^i_{j+1} \oplus \Delta^{i'}_{j'+1} = 0$.

From the definition of the predicate BadT_2, we have $j + 1 \neq \mathsf{llcp}_n(i) + 1$, implying that the probability of Δ^i_{j+1} being equal to $\Delta^{i'}_{j'+1}$ is $1/2^n$. Given σ possible values of (i, j) in a transcript, each with no more than σ possible values of (i', j'), we find that $\Pr[\mathsf{BadT_2}(\Theta_{idin}) = 1] \leq \frac{\sigma^2}{2^{n+1}}$.

BadT_3**.** For any transcript in $\mathcal{T}_{\text{bad}}$ with BadT_3 set to 1 and BadT_1 set to 0, one of the following can happen for Θ_{idin}:

1. For some $i' \leq q_e$, $j = \ell^i$ and $j' = \ell^{i'}$, we have $j = j' = \mathsf{llcp}_n(i)$. Here, $\Delta^i_j = \Delta^{i'}_{j'}$, $X^i_j = X^{i'}_{j'}$, but $T^i \neq T^{i'}$. Since $T^{i'}$ is the correct tag for the given ciphertext, $T^i \neq T^{i'}$ yields a probability of 0.
2. For some $i' \leq q_e + q_d$, $j = \ell^i$ and $j' = \ell^{i'}$, we have $j = j' = \mathsf{llcp}_n(i) + 1$. Here, $\Delta^i_j = \Delta^{i'}_{j'}$ but $X^i_j \neq X^{i'}_{j'}$, and the probability of any of the three conditions of BadT_3 occurring for a given query is at most $4q_d/2^n$, assuming $q_e + q_d \leq 2^{n-1}$. For the first condition, this holds as every tag is produced with a tweak used at most once per encryption query, corresponding to a

probability of $1/(2^n - q_e) \leq 2/2^n$. For the second condition, the probability of having the correct padding in the block X_j^i (at most $2^{|T^i|}/(2^n - q_e - q_d)$), and the correct truncated tag (at most $2^{n-|T^i|}/(2^n - q_e)$) is at most $4/2^n$. For the third condition, with $q_e + 1 \leq i_d \leq q_e + q_d$ such that $|T^i| = |T^{i_d}|$, the probability of having the correct padding in the block $X_{\ell^{i_d}}^{i_d}$ (at most $2^{|T^i|}/(2^n - q_e - q_d)$), and the correct truncated tag for the verification query (at most $2^{n-|T^i|}/(2^n - q_e)$) is at most $4/2^n$. Since there are q_d possible choices for i_d, the total probability of the third condition is at most $4q_d/2^n$.

3. For all $i' \leq q_e + q_d$ when $j > \mathsf{llcp}_n(i, i') + 1$. We know that Δ_j^i is not inherited from an encryption or decryption query and is therefore sampled uniformly in Θ_{idin}. The first condition of BadT_3 thus occurs with a probability of $1/2^n$. For the second condition, the correct padding is found with a probability of $1/2^{n-|T^i|}$, and the correct tag is found with a probability of at most $2^{n-|T^i|}/(2^n - q_e)$, yielding a probability of at most $2/2^n$. For the third condition, with similar reasoning, the correct padding is found with a probability of $q_d/2^{n-|T^i|}$, and the correct tag is found with a probability of at most $2^{n-|T^i|}/(2^n - q_e)$, providing a probability of at most $2q_d/2^n$.

Since there are q_v possible verification queries, we get $\Pr[\mathsf{BadT_3}(\Theta_{idin}) = 1|\mathsf{BadT_1}(\Theta_{idin}) = 0] \leq q_v \cdot \max\left\{0, \frac{4q_d}{2^n}, \frac{2q_d}{2^n}\right\} = \frac{4 \cdot q_d q_v}{2^n}$. Using the union bound, we find that $\Pr[\Theta_{idin} \in \mathcal{T}_{\text{bad}}] \leq \frac{\sigma^2}{2^n} + \frac{4 \cdot q_d q_v}{2^n}$. □

C.2 Proof of Lemma 3

Proof. A good transcript satisfies two properties: 1. **(i)** For each $(i', j') < (i, j)$, if (i, j) is not in the longest common prefix of the two queries, i.e., $\mathsf{llcp}_n(i, i') < j < \ell^i$ and both π_0 calls have the same tweaks (i.e., $\mathsf{T}_j^i = \mathsf{T}_{j'}^{i'}$), then both calls will have different inputs and outputs. 2. **(ii)** For each verification query $1 \leq i \leq q_v$, the transcript contains $b^i = \perp$ in the verification result, meaning the conditions for a successful verification are not met.

The probability of obtaining a good transcript τ in the real-int and the ideal-int worlds can now be computed. Let τ_{ed} and τ_v denote the encryption-decryption and verification parts of a transcript τ, so that $\tau = \langle \tau_{ed}, \tau_v \rangle$. We have $\Pr[\Theta_{rein} = \tau] = \Pr[\Theta_{rein,ed} = \tau_{ed}] \cdot \Pr[\Theta_{rein,v} = \tau_v | \Theta_{rein,ed} = \tau_{ed}]$ and $\Pr[\Theta_{idin} = \tau] = \Pr[\Theta_{idin,ed} = \tau_{ed}] \cdot \Pr[\Theta_{idin,v} = \tau_v | \Theta_{idin,ed} = \tau_{ed}]$, leading to

$$\frac{\Pr[\Theta_{rein,ed} = \tau_{ed}] \cdot \Pr[\Theta_{rein,v} = \tau_v | \Theta_{rein,ed} = \tau_{ed}]}{\Pr[\Theta_{idin,ed} = \tau_{ed}] \cdot \Pr[\Theta_{idin,v} = \tau_v | \Theta_{idin,ed} = \tau_{ed}]} = \frac{\Pr[\Theta_{rein,v} = \tau_v | \Theta_{rein,ed} = \tau_{ed}]}{\Pr[\Theta_{idin,v} = \tau_v | \Theta_{idin,ed} = \tau_{ed}]}.$$

This equality holds because the encryption and decryption oracles in both the real-int and ideal-int worlds are identical, so $\Pr[\Theta_{rein,ed} = \tau_{ed}] = \Pr[\Theta_{idin,ed} = \tau_{ed}]$. Denote by $\tau_{v,\Delta}$ the event where all Δ masks in the verification queries match the values in the transcript. We have $\Pr[\Theta_{rein,v} = \tau_v | \Theta_{rein,ed} = \tau_{ed}, \Theta_{rein,v,\Delta} = \tau_{v,\Delta}] = \Pr[\Theta_{idin,v} = \tau_v | \Theta_{idin,ed} = \tau_{ed}, \Theta_{idin,v,\Delta} = \tau_{v,\Delta}]$ because both sides of this equality correspond to mappings defined by random permutations with input-output pairs fixed from the encryption-decryption parts in both worlds. Using this equality, we get

$$\frac{\Pr[\Theta_{rein,v} = \tau_v | \Theta_{rein,ed} = \tau_{ed}]}{\Pr[\Theta_{idin,v} = \tau_v | \Theta_{idin,ed} = \tau_{ed}]} = \frac{\Pr[\Theta_{rein,v,\Delta} = \tau_{v,\Delta} | \Theta_{rein,ed} = \tau_{ed}]}{\Pr[\Theta_{idin,v,\Delta} = \tau_{v,\Delta} | \Theta_{idin,ed} = \tau_{ed}]}.$$

There are δ many Δs in τ that are fixed due to internal common prefixes, so $\delta = \sum_{i=1}^{q_e+q_d+q_v} (\mathsf{llcp}_n(i) + 1)$ (the extra 1 represents Δ_1^i, which is always fixed to 0). In the ideal-int world, since the Δs for the remaining $(\sigma - \delta)$ unique block calls are sampled uniformly and independently and all verification oracle results are $\perp$, we have

$$\Pr[\Theta_{idin,v,\Delta} = \tau_{v,\Delta} | \Theta_{idin,ed} = \tau_{ed}] = \frac{1}{(2^n)^{\sigma-\delta}}.$$

In the real-int world, these $(\sigma-\delta)$ Δs are defined using the random tweakable permutation (π_0, π_1) with at least $g_1 = \sum_{i=1}^{q_e+q_d+q_v} (a^i - 1)$ block calls with tweak 0^n and at least $g_2 = \sum_{i=1}^{q_e+q_d+q_v} (m^i - 1)$ block calls with tweak $0^{n-1} \| 1$. Therefore,

$$\Pr[\Theta_{rein,v,\Delta} = \tau_{v,\Delta} | \Theta_{rein,ed} = \tau_{ed}] \geq \frac{1}{(2^n)_{g_1} (2^n)_{g_2} (2^n)^{\sigma-\delta-g_1-g_2}}.$$

Note that this expression provides an upper bound on the probability because more permutation calls can have tweak collisions (e.g., the first block calls of queries with the same nonce). From these expressions, we obtain

$$\frac{\Pr[\Theta_{rein} = \tau]}{\Pr[\Theta_{idin} = \tau]} \geq \frac{(2^n)^{\sigma-\delta}}{(2^n)_{g_1} (2^n)_{g_2} (2^n)^{\sigma-\delta-g_1-g_2}} = \frac{(2^n)^{g_1} (2^n)^{g_2}}{(2^n)_{g_1} (2^n)_{g_2}} \geq 1.$$

□

C.3 Proof of Lemma 4

Proof. $\mathsf{\textbf{BadT}}'_1$**.** For any transcript in $\mathcal{T}'_{\text{bad}}$ where $\mathsf{BadT'}_1$ is set to 1, there is at least one pair (i, j) and (i', j') such that $\mathsf{llcp}_n(i) < j \leq \ell^i$, $(i', j') < (i, j)$, and $\Delta_j^i \oplus \Delta_{j'}^{i'} = X_j^i \oplus X_{j'}^{i'}$.

For all $i' < i$ and $j = j' = \mathsf{llcp}_n(i) + 1$, we have $\Delta_j^i = \Delta_{j'}^{i'}$ but $X_j^i \neq X_{j'}^{i'}$, thus the probability of the equality above occurring is 0. This also addresses nonce collisions: if $N^i = N^{i'}$, then $j = j' = 1$ and $\mathsf{llcp}_n(i) = 0$, implying $\Delta_1^i = \Delta_1^{i'} = 0$ and $X_1^i \neq X_1^{i'}$. Conversely, for all $i' \leq i$ and $j' \neq j$ or $j \neq \mathsf{llcp}_n(i) + 1$, the two masks are sampled uniformly and independently in Θ_{idco}. Considering there are at most $\sigma' \leq \sigma$ possible values of (i, j) in a transcript, each with at most $\sigma' \leq \sigma$ possible values of (i', j'), we get $\Pr[\mathsf{BadT}'_1(\Theta_{idco}) = 1] \leq \frac{\sigma^2}{2} \cdot \max\left\{0, \frac{1}{2^n}\right\} = \frac{\sigma^2}{2^{n+1}}$.

$\mathsf{\textbf{BadT}}'_2$**.** Similarly, for any transcript in $\mathcal{T}'_{\text{bad}}$ where $\mathsf{BadT'}_2$ is set to 1, there is at least one pair $(i', j') < (i, j)$ such that $\mathsf{llcp}_n(i) < j < \ell^i$ and one of the following is true:

I. $j = \mathsf{llcp}_n(i) + 1$ and ($\Delta^i_{j+1} = \Delta^{i'}_{j'+1}$ or $T^i = T^{i'}$). (In this case, $Y^i_j + \Delta^i_j \neq Y^{i'}_{j'} + \Delta^{i'}_{j'}$ as $X^i_j \neq X^{i'}_{j'}$ implies $Y^i_j \neq Y^{i'}_{j'}$ by the definition of $\mathsf{llcp}_n(i)$.)

II. $j > \mathsf{llcp}_n(i) + 1$ and ($Y^i_j + \Delta^i_j = Y^{i'}_{j'} + \Delta^{i'}_{j'}$ or $\Delta^i_{j+1} = \Delta^{i'}_{j'+1}$ or $T^i = T^{i'}$).

From the definition of the predicate $\mathsf{BadT'}_2$, we know that $j+1 \neq \mathsf{llcp}_n(i)+1$. This means that Δ^i_{j+1} is uniformly and independently distributed from $\Delta^{i'}_{j'+1}$, with a collision probability of $1/2^n$. Each tag, generated as n uniform bits independent of all other tags, collides with a probability of $1/2^n$. For each $j > \mathsf{llcp}_n(i)$, Δ^i_j is uniformly and independently distributed from $\Delta^{i'}_{j'}$, making the masked ciphertexts collide with a probability of $1/2^n$.

There are at most σ possible values of (i, j) in a transcript that can cause a collision of masked ciphertexts, each with at most σ possible values of (i', j'), resulting in no more than $\sigma^2/2$ pairs. Additionally, there are no more than $\sigma - q_e - q_d$ valid values of (i, j) for a Δ collision, each with no more than $\sigma - q_e - q_d$ possible values of (i', j'), yielding no more than $(\sigma - q_e - q_d)^2/2$ pairs. Finally, there are q_e tags that can collide with one another, resulting in no more than $q_e^2/2$ pairs. Thus, we get $\Pr[\mathsf{BadT}'_2(\Theta_{idco}) = 1] \leq \frac{2\cdot\sigma^2}{2^{n+1}}$.

Therefore, using the union bound, we obtain $\Pr[\Theta_{idco} \in \mathcal{T}'_{\text{bad}}] \leq \frac{3\cdot\sigma^2}{2^{n+1}}$. □

C.4 Proof of Lemma 5

Proof. A good transcript has the following property: for each $(i', j') < (i, j)$, if (i, j) is not in the longest common prefix of the two queries, i.e., $\mathsf{llcp}_n(i, i') < j < \ell^i$ and both π_0 (resp. π_1) calls have the same tweaks (i.e., $\mathsf{T}^i_j = \mathsf{T}^{i'}_{j'}$), then both blocks will have different inputs (i.e., $X^i_j \oplus \Delta^i_j \neq X^{i'}_{j'} \oplus \Delta^{i'}_{j'}$), different outputs (i.e., $Y^i_j \oplus \Delta^i_j \neq Y^{i'}_{j'} \oplus \Delta^{i'}_{j'}$), and $\Delta^i_{j+1} \neq \Delta^{i'}_{j'+1}$. If $1 \leq i \leq q_e$, then the tags will also be different (i.e., $T^i \neq T^{i'}$ for any two encryption queries).

The probability of obtaining a good transcript τ in the real-conf and ideal-conf worlds can now be computed. Let τ_Δ denote the marginal event where all Δ masks in the queries match the values in the transcript. With this notation, we have

$$\Pr[\Theta_{reco} = \tau | \Theta_{reco,\Delta} = \tau_\Delta] \geq \Pr[\Theta_{idco} = \tau | \Theta_{idco,\Delta} = \tau_\Delta].$$

This is true because, for fixed and unique (up to common prefix) input-output pairs (excluding the tags), the left side of this inequality corresponds to mappings of a random permutation with an input size of n bits, while the right side corresponds to mappings of a random online permutation with an input size of at least n bits. For the tags (fixed and unique), the left side corresponds to a random permutation, whereas the right side corresponds to a random function with the same input size (n bits).

Consider that τ has δ' fixed/predefined Δs due to internal common prefixes. We can then write $\delta' = \sum_{i=1}^{q_e+q_d} (\mathsf{llcp}_n(i) + 1)$. Here the extra 1 accounts for Δ^i_1, which is fixed to 0.

In the ideal-conf world, since the Δs corresponding to these $\sigma' - \delta'$ unique block calls are sampled uniformly and independently, we have

$$\Pr[\Theta_{idco,\Delta} = \tau_\Delta] = \frac{1}{(2^n)^{\sigma' - \delta'}}.$$

In the real-conf world, these $\sigma' - \delta'$ Δs are not uniformly distributed but are computed using the random tweakable permutation (π_0, π_1) with at least $g'_1 = \sum_{i=1}^{q_e+q_d}(a^i - 1)$ block calls with tweak 0^n and at least $g'_2 = \sum_{i=1}^{q_e+q_d}(m^i - 1)$ block calls with tweak $0^{n-1}\|1$. Therefore,

$$\Pr[\Theta_{reco,\Delta} = \tau_\Delta] \geq \frac{1}{(2^n)_{g'_1}(2^n)_{g'_2}(2^n)^{\sigma' - \delta' - g'_1 - g'_2}}.$$

Now, combining the above expressions, we get

$$\frac{\Pr[\Theta_{reco} = \tau]}{\Pr[\Theta_{idco} = \tau]} \geq \frac{(2^n)^{\sigma' - \delta'}}{(2^n)_{g'_1}(2^n)_{g'_2}(2^n)^{\sigma' - \delta' - g'_1 - g'_2}} = \frac{(2^n)^{g'_1}(2^n)^{g'_2}}{(2^n)_{g'_1}(2^n)_{g'_2}} \geq 1.$$

$\square$