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

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

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

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 au-

thenticity 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 notions for online AEs: (i) 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.
   (ii) 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). 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.
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 also study the relations/differences among popular AEAD notions to compare them with OAE-RUP and to argue its relevance. Due to lack of space, we provide the detailed comparison analysis in App. D.

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. For completeness, we provide a detailed explanation of Table 1 by revising all the properties and describing how the checkmarks are derived in App. B.

2 Preliminaries

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

For a string X of ℓ bits, we denote by $X[i]$ the i^{th} bit of X for $i = 0,\ldots,\ell-1$ (counting from left to right) and define $X[i\ldots j] = X[i]\|X[i+1]\|\ldots\|X[j]$ for $0 \leq i < j < \ell$. We let $\mathsf{left}_\ell(X) = X[0\ldots(\ell-1)]$ denote the ℓ leftmost bits of X and $\mathsf{right}_r(X) = X[(|X|-r)\ldots(|X|-1)]$ the r rightmost bits of X, such that $X = \mathsf{left}_\chi(X)\|\mathsf{right}_{|X|-\chi}(X)$ for all $0 \leq \chi \leq |X|$. We let $(L,R) = \mathsf{lsplit}_n(X)$ denote splitting a string $X \in \{0,1\}^*$ into two parts such that $L = \mathsf{left}_{\min(|X|,n)}(X)$ and

$R = \mathsf{right}_{|X|-|L|}(X)$. In particular, for $n \geq |X|$ we have $(X, \varepsilon) = \mathsf{lsplit}_n(X)$. Given an integer (possibly implicit) $n > 0$ and an $X \in \{0,1\}^*$, we use $X\|10^*$ to denote $X\|10^{n-(|X| \bmod n)-1}$ for simplicity. Further, for the same integer n, we define $\mathsf{pad10}(X) = X\|10^*$ that returns X if $|X| \equiv 0 \pmod n$ and $X\|10^*$ otherwise.

String partitioning. We fix an arbitrary integer n for this work and call it the block size. For any string X, We let $|X|_n = \lceil |X|/n \rceil$ and use $X_1, \ldots, X_x, X_* \xleftarrow{n} X$ to define the partitioning of X into n-bit blocks, such that $X = X_1\| \ldots \|X_x\|X_*$ with $|X_i| = n$ for $i = 1, \ldots, x$ and $0 < |X_*| \leq n$. Hence, $x = |X|_n - 1$.

Blocks. We use B_n to denote the set of all n-bit strings (or blocks) i.e., $\{0,1\}^n$. We define $\mathsf{B}_n^* = \{\varepsilon\} \cup \bigcup_{i=1}^{\infty} \mathsf{B}_n^i$ where ε denotes the empty string with length 0. We say a string X is "n-aligned" iff $X \in \mathsf{B}_n^*$. We let X_i denote the i^{th} n-bit block of an n-aligned string X. For two distinct and n-aligned strings $X, Y \in \mathsf{B}_n^*$ with $|X|_n \leq |Y|_n$ w.l.o.g, we let $\mathsf{llcp}_n(X, Y) = \max\{1 \leq i \leq |X|_n \mid X_j = Y_j \text{ for } 1 \leq j \leq i\}$ denote the length of the longest common prefix (in n-bit blocks) of X and Y.

Miscellaneous. We let $X \leftarrow\!\!\$\ \mathcal{X}$ denote the sampling of an element X from a finite set $\mathcal{X}$ under the uniform distribution. We let $(p)_q$ denote the falling factorial $p \cdot (p-1) \cdot (p-2) \cdot \ldots \cdot (p-q+1)$ where $(p)_0 = 1$. We define a predicate $\mathsf{P}(x)$ as $\mathsf{P}(x) = 1$ if it is true and $\mathsf{P}(x) = 0$ if it is false. We use lexicographic comparison for integer tuples (to exemplify, $(i', j') < (i, j)$ iff $i' < i$ or $i' = i$ and $j' < j$). The symbol $\perp$ denotes an undefined value or an error. Let $\mathcal{A}$ be an adversary (algorithm) who wants to distinguish a world $\mathcal{O}_{re}$ (or game G_{re}) from a world $\mathcal{O}_{id}$ (or game G_{id}, respectively). We denote by $\mathcal{A}^{\mathcal{O}_x}$ (or $\mathcal{A}^{\mathsf{G}_x}$) the event that $\mathcal{A}$ 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 AEAD scheme under the RUP setting is a tuple $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{V})$. The key space $\mathcal{K}$ is a finite set. The deterministic encryption algorithm $\mathcal{E}\colon \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{M} \to \mathcal{C}$ maps a secret key K, a nonce N, an associated data A and a message M to a ciphertext $C = \mathcal{E}(K, N, A, M)$. The nonce, associated data, and message domains are all subsets of $\{0,1\}^*$. The deterministic decryption algorithm $\mathcal{D}\colon \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{C} \to \mathcal{M}$ takes a tuple (K, N, A, C) and returns a message $M \in \mathcal{M}$. The deterministic verification algorithm $\mathcal{V}\colon \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{C} \to \{\top, \perp\}$ takes a tuple (K, N, A, C) and either returns the distinguished symbol $\top$ to indicate a successful authentication or $\perp$ to indicate an authentication error.

We require that for every $M \in \mathcal{M}$, we have $\{0,1\}^{|M|} \subseteq \mathcal{M}$ (i.e., for any integer m, either all or no strings of length m belong to $\mathcal{M}$) and that for all $(K, N, A, M) \in \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{M}$ we have $|\mathcal{E}(K, N, A, M)| = |M| + \theta$ for some non-negative integer θ called the stretch of Π. For correctness of Π, we require that for all $(K, N, A, M) \in \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{M}$ we have $M = \mathcal{D}(K, N, A, \mathcal{E}(K, N, A, M))$ and $\top = \mathcal{V}(K, N, A, \mathcal{E}(K, N, A, M))$. We let $\mathcal{E}_K(N, A, M) = \mathcal{E}(K, N, A, M), \mathcal{D}_K(N, A, C) = \mathcal{D}(K, N, A, C)$ 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, $\mathbf{soprpf\text{-}real}_\Pi$ and $\mathbf{soprpf\text{-}ideal}_\Pi$. 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 $\mathbf{soprpf\text{-}real}_\Pi$, the oracles use the encryption and decryption algorithms of Π with a randomly picked secret key. On the other hand, in the game $\mathbf{soprpf\text{-}ideal}_\Pi$, 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 $|M_R| \neq 0$, $P_{N,A} \leftarrow\!\$\ \mathrm{OPerm}(n)$ is a random online permutation for each pair (N, A), $\pi_{N,A,M_L,\lfloor |M_R|/n \rfloor} \leftarrow\!\$\ \mathrm{Perm}(n)$ 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 $P_{N,A}^{-1}(C_L) \| \pi^{-1}_{N,A,P^{-1}_{N,A}(C_L),\lfloor |C_T|/n \rfloor}(C_R)$ where $P_{N,A}^{-1}$ 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 $\pi_{N,A,P_{N,A}^{-1}(C_L),\lfloor |C_T|/n \rfloor}$, respectively. The **soprpf** advantage of an adversary $\mathcal{A}$ against $\Pi = \mathrm{SAEF}[\mathsf{F}]$ is defined as $\mathbf{Adv}_{\Pi}^{\mathbf{soprpf}}(\mathcal{A}) = \Pr[\mathcal{A}^{\mathbf{soprpf\text{-}real}_{\Pi}}] - \Pr[\mathcal{A}^{\mathbf{soprpf\text{-}ideal}_{\Pi}}]$.

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. We also note that syntax-wise sOPRPF may not apply to all types of online AE schemes and therefore we additionally propose a weaker version of RUP confidentiality that is simple, intuitive, and applicable to all types of online AE schemes called CR-RUP in App. C.

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*

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

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*

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

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 follow the formalism of forkcipher provided by Andreeva et al. [39]. Informally, a forkcipher F is a tweakable symmetric primitive that maps a secret key K, a tweak T and an n-bit input block M to two n-bit ciphertext blocks C_0 and C_1, such that C_0 and C_1 are two independent permutations of M.

Syntax. A forkcipher is formally defined by a pair of deterministic algorithms, the encryption algorithm $\mathsf{F}\colon \{0,1\}^k \times \mathcal{T} \times \{0,1\}^n \times \{0,1,\mathsf{b}\} \to \{0,1\}^n \cup \{0,1\}^n \times \{0,1\}^n$ and the inversion algorithm $\mathsf{F}^{-1}\colon \{0,1\}^k \times \mathcal{T} \times \{0,1\}^n \times \{0,1\} \times \{\mathsf{i},\mathsf{o},\mathsf{b}\} \to \{0,1\}^n \cup \{0,1\}^n \times \{0,1\}^n$.

The encryption algorithm takes a key K, a tweak $\mathsf{T} \in \mathcal{T}$, a plaintext block M and an output selector s, and outputs the left n-bit ciphertext block C_0 if $s = 0$, the right n-bit ciphertext block C_1 if $s = 1$, and both the blocks C_0, C_1 if $s = \mathsf{b}$. We use $\mathsf{F}(K,\mathsf{T},M,s) = \mathsf{F}_K(\mathsf{T},M,s) = \mathsf{F}^{\mathsf{T}}_K(M,s) = \mathsf{F}^{\mathsf{T},s}_K(M)$ interchangeably.

Similarly, the inversion algorithm takes a key K, a tweak T, a ciphertext block C (either the left or right half of the output block), an indicator b to indicate whether the fed block should be treated as the left or the right ciphertext block and an output selector s, and outputs the plaintext block M if $s = \mathsf{i}$, the other ciphertext block C' if $s = \mathsf{o}$, and both blocks M, C' if $s = \mathsf{b}$. We use $\mathsf{F}^{-1}(K,\mathsf{T},M,b,s) = {\mathsf{F}^{-1}}_K(\mathsf{T},M,b,s) = {\mathsf{F}^{-1}}^{\mathsf{T}}_K(M,b,s) = {\mathsf{F}^{-1}}^{\mathsf{T},b,s}_K(M)$ interchangeably.

[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).

We say that a tweakable forkcipher F is correct if, for each pair of key and tweak, the forkcipher applies two independent permutations on the input to produce the corresponding two output blocks. Formally, for every tuple (K, T, M, β) with $K \in \{0,1\}^k, \mathsf{T} \in \mathcal{T}, M \in \{0,1\}^n$ and $\beta \in \{0,1\}$, the forkcipher must satisfy the following conditions: (i) $\mathsf{F}^{-1}(K, \mathsf{T}, \mathsf{F}(K, \mathsf{T}, M, \beta), \beta, \mathsf{i}) = M$, and (ii) $\mathsf{F}^{-1}(K, \mathsf{T}, \mathsf{F}(K, \mathsf{T}, M, \beta), \beta, \mathsf{o}) = \mathsf{F}(K, \mathsf{T}, M, \beta \oplus 1)$, and (iii) $(\mathsf{F}(K, \mathsf{T}, M, 0), \ \mathsf{F}(K, \mathsf{T}, M, 1)) = \mathsf{F}(K, \mathsf{T}, M, \mathsf{b})$, and (iv) $\big(\mathsf{F}^{-1}(K, \mathsf{T}, C, \beta, \mathsf{i}), \mathsf{F}^{-1}(K, \mathsf{T}, C, \beta, \mathsf{o})\big) = \mathsf{F}^{-1}(K, \mathsf{T}, C, \beta, \mathsf{b})$.

For the rest of the paper, we assume that $\mathcal{T} = \{0,1\}^t$ for some positive t. We call k, n, and t the keysize, blocksize, and tweaksize of F, respectively.

Forkcipher Security. The security of a forkcipher F is defined as the indistinguishability between the real $\mathbf{prtfp\text{-}real}_{\mathsf{F}}$ and ideal $\mathbf{prtfp\text{-}ideal}_{\mathsf{F}}$ worlds when adversary accesses either worlds in a chosen ciphertext fashion. In the real world, the forkcipher oracle implements the true F algorithm faithfully, whereas in the latter world, the oracle replaces F by two tweakable random permutations $\pi_{\mathsf{T},0}, \pi_{\mathsf{T},1} \leftarrow\!\!\$\ \mathrm{Perm}(n)$ for $\mathsf{T} \in \mathcal{T}$. We then define the advantage of $\mathcal{A}$ as:

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

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 $\mathcal{T}_{good}$ and $\mathcal{T}_{bad}$. Also, assume there exist $\epsilon_1, \epsilon_2 \geq 0$ such that for any transcript $\tau \in \mathcal{T}_{good}$, we have $\frac{\Pr[\Theta_{\mathsf{real}} = \tau]}{\Pr[\Theta_{\mathsf{ideal}} = \tau]} \geq 1 - \epsilon_1$, and $\Pr[\Theta_{\mathsf{ideal}} \in \mathcal{T}_{bad}] \leq \epsilon_2$. Then, for all adversaries $\mathcal{A}$, it holds that*

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

3 SAEF and its OAE-RUP Security

SAEF (short for Sequential AE from a Forkcipher) is a nonce-based AEAD scheme that uses a tweakable forkcipher F (as defined in Section 2.3) as an underlying primitive with $\mathcal{T} = \{0,1\}^t$ for a positive $t \leq n$. SAEF[F] $= (\mathcal{K}, \mathcal{E}, \mathcal{D}, \mathcal{V})$ has a key space $\mathcal{K} = \{0,1\}^k$, nonce space $\mathcal{N} = \{0,1\}^{t-4}$, and the associated data (AD) and message spaces are both $\{0,1\}^*$. The ciphertext expansion of SAEF is n bits. The encryption, decryption, and verification algorithms are given in Figure 1 and the encryption algorithm is illustrated in Figure 2 (App. A). Note that the earlier SAEF representation [7] did not have an explicit decryption and verification functionality separation. The present syntax splits explicitly these functionalities and introduces no change to the actual input and output behavior of the SAEF algorithm.

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).

3.1 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.

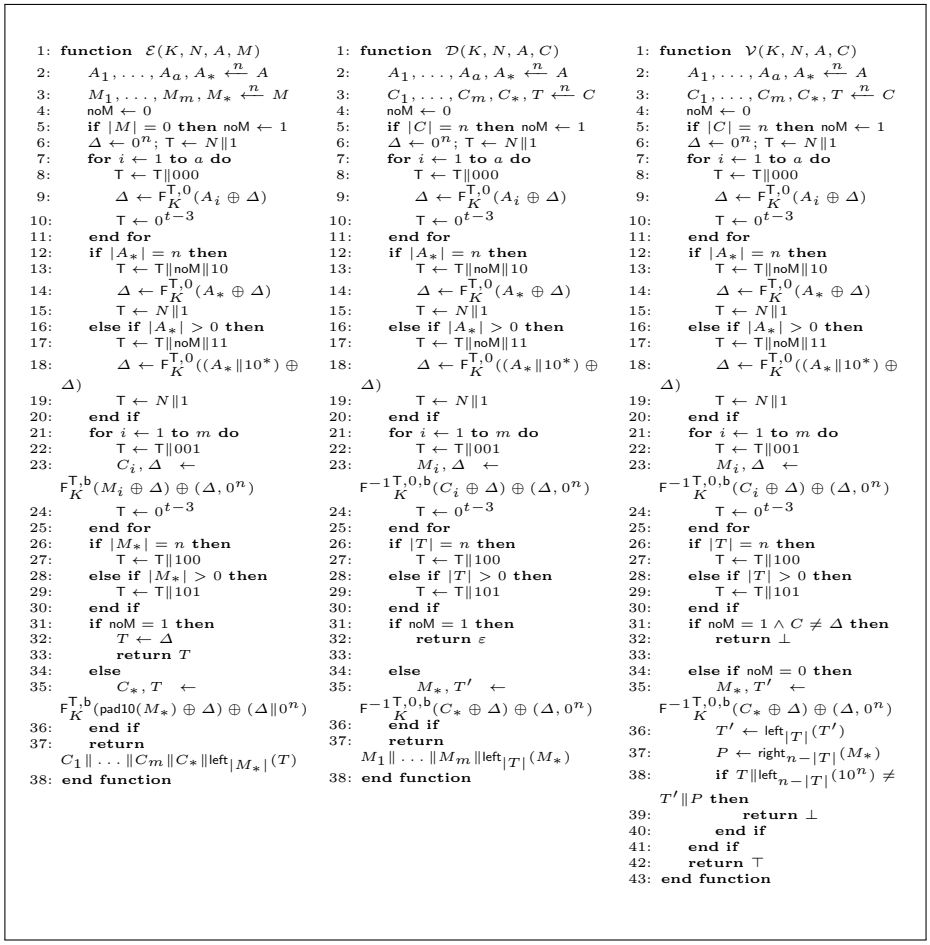

Fig. 1: The SAEF[F] AEAD scheme.

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.

The proof of Theorem 1 follows in Sections 3.2 and 3.3.

We use a similar proof approach and case analysis for SAEF RUP integrity as provided by Andreeva et al. in [4] for SAEF integrity under Nonce-Misuse. Our proof and analysis derive the claimed security bound by utilizing the key properties of SAEF that are the results of its sequential structure. One such property is the preservation of common length prefixes over queries (encryption or decryption). This can better be understood by an example. Consider two encryption queries (w.l.o.g.) with same nonce N, same associated data of two blocks $A_1\|A_2$ but different messages (of two blocks) $M_1\|M_2$ and $M_0\|M_2$ where $M_1 \neq M_0$. One can follow the encryption algorithm (as shown in Figure 1) for these two queries and can notice that the values of the F tweak strings and of Δ masks used to process the input data upto block A_2 are same for both the queries. However, these equalities of tweak strings and the Δ masks for A_2, when combined with the inequality $M_1 \neq M_0$, imply that the next input blocks in these encryption queries will necessarily differ, i.e., $\Delta \oplus M_1 \neq \Delta \oplus M_0$. This will randomize the Δ masks that are used to process the next input blocks. In short, the internal variables of SAEF's encryption algorithm preserve a common prefix length over queries and get randomized after that. We use this property of SAEF to prove its RUP integrity.

3.2 Integrity

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 $\mathbf{Adv}^{\mathbf{intrup}}_{\mathrm{SAEF}[\mathsf{F}]}(\mathcal{A}) \leq \mathbf{Adv}^{\mathbf{prtfp}}_{\mathsf{F}}(\mathcal{B}) + \mathbf{Adv}^{\mathbf{intrup}}_{\mathrm{SAEF}[(\pi_0,\pi_1)]}(\mathcal{A})$.

Now, the adversary is left to distinguish between the games $\mathbf{intrup\text{-}real}_{\mathrm{SAEF}[(\pi_0,\pi_1)]}$ and $\mathbf{intrup\text{-}ideal}_{\mathrm{SAEF}[(\pi_0,\pi_1)]}$. For simplicity, we denote these games by "real-int world" and "ideal-int world", respectively. Hence, we want to bound $\mathbf{Adv}^{\mathbf{intrup}}_{\mathrm{SAEF}[(\pi_0,\pi_1)]}(\mathcal{A}) = \Pr[\mathcal{A}^{\mathbf{intrup\text{-}real}_{\mathrm{SAEF}[(\pi_0,\pi_1)]}}] - \Pr[\mathcal{A}^{\mathbf{intrup\text{-}ideal}_{\mathrm{SAEF}[(\pi_0,\pi_1)]}}]$.

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

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

For the i^{th} query to the encryption oracle with input (N^i, A^i, M^i) and output C^i, SAEF internally processes A^i, M^i and C^i in blocks $A^i_1, \ldots, A^i_{a^i}, A^i_*$, and $M^i_1, \ldots, M^i_{m^i}, M^i_*$, and $C^i_1, \ldots, C^i_{m^i}, C^i_*, T^i$, respectively (defined as per the encryption algorithm of SAEF, Figure 1). Here, a^i and m^i represent the length of A^i and M^i in n-bit blocks, respectively, i.e., $a^i = |A^i|_n - 1$ and $m^i = |M^i|_n - 1$. SAEF also processes and uses the internal chaining values as the whitening masks for encryption which we denote here by a sequence of Δs. The whitening masks used to process $A^i_1, \ldots, A^i_{a^i}, A^i_*$ are denoted by $\Delta^i_1, \ldots, \Delta^i_{a^i+1}$, respectively, and

the whitening masks used to process the blocks $M^i_1, \ldots, M^i_{m^i}, M^i_*$ are denoted by $\Delta^i_{a^i+2}, \ldots, \Delta^i_{a^i+m^i+2}$, respectively.

Similarly, for the i^{th} query to the decryption oracle with input $(\bar{N}^i, \bar{A}^i, \bar{C}^i)$ and output $\bar{M}^i$, SAEF internally processes $\bar{A}^i, \bar{C}^i$ and $\bar{M}^i$ in blocks $\bar{A}^i_1, \ldots, \bar{A}^i_{\bar{a}^i}, \bar{A}^i_*$, and $\bar{C}^i_1, \ldots, \bar{C}^i_{\bar{m}^i}, \bar{C}^i_*, \bar{T}^i$, and $\bar{M}^i_1, \ldots, \bar{M}^i_{\bar{m}^i}, \bar{M}^i_*$, respectively (defined as per the decryption algorithm of SAEF, Figure 1). Here, $\bar{a}^i$ and $\bar{m}^i$ represent the length of $\bar{A}^i$ and $\bar{C}^i$ (excluding tag block from the count) in n-bit blocks, respectively, i.e., $\bar{a}^i = |\bar{A}^i|_n - 1$ and $\bar{m}^i = |\bar{C}^i|_n - 2$. SAEF also processes and uses the internal chaining values as the whitening masks for decryption which we denote here by a sequence of $\bar{\Delta}$s. The whitening masks used to process $\bar{A}^i_1, \ldots, \bar{A}^i_{\bar{a}^i}, \bar{A}^i_*$ are denoted by $\bar{\Delta}^i_1, \ldots, \bar{\Delta}^i_{\bar{a}^i+1}$, respectively, and the whitening masks used to process the blocks $\bar{C}^i_1, \ldots, \bar{C}^i_{\bar{m}^i}, \bar{C}^i_*$ are denoted by $\bar{\Delta}^i_{\bar{a}^i+2}, \ldots, \bar{\Delta}^i_{\bar{a}^i+\bar{m}^i+2}$, respectively.

Similarly, for the i^{th} query to the verification oracle with input $(\tilde{N}^i, \tilde{A}^i, \tilde{C}^i)$ and output $b^i \in \{\top, \bot\}$, SAEF internally processes $\tilde{A}^i$ and $\tilde{C}^i$ in blocks, denoted as $\tilde{A}^i_1, \ldots, \tilde{A}^i_{\tilde{a}^i}, \tilde{A}^i_*$ and $\tilde{C}^i_1, \ldots, \tilde{C}^i_{\tilde{m}^i}, \tilde{C}^i_*, \tilde{T}^i$, where $\tilde{a}^i$ and $\tilde{m}^i$ are respectively equal to the length of $\tilde{A}^i$ and the length of $\tilde{C}^i$ in n-bit blocks (excluding tag block from the count), i.e., $\tilde{a}^i = |\tilde{A}^i|_n - 1$ and $\tilde{m}^i = |\tilde{C}^i|_n - 2$. Additionally, SAEF internally computes the plaintext blocks $\tilde{M}^i_1, \ldots, \tilde{M}^i_{\tilde{m}^i}, \tilde{M}^i_*$ as well as $\tilde{\Delta}^i_1, \ldots, \tilde{\Delta}^i_{\tilde{a}^i+1}$, the whitening masks used to process $\tilde{A}^i_1, \ldots, \tilde{A}^i_{\tilde{a}^i}, \tilde{A}^i_*$ respectively, and $\tilde{\Delta}^i_{\tilde{a}^i+2}, \ldots, \tilde{\Delta}^i_{\tilde{a}^i+\tilde{m}^i+2}$,the whitening masks used to process the blocks $\tilde{C}^i_1, \ldots, \tilde{C}^i_{\tilde{m}^i}, \tilde{C}^i_*$ respectively.

Additional information. To make the proof analysis simple, we additionally provide the adversary with all the whitening masks (encryption masks Δ^i_j, decryption masks $\bar{\Delta}^i_j$ and verification masks $\tilde{\Delta}^i_j$), and internally computed plaintexts $\tilde{M}^i_j$ when it has made all its queries and only the final response is pending.

In the real-int world, all these variables are internally computed by oracles that faithfully evaluate SAEF. In the ideal-int world also, the encryption and decryption oracles evaluate SAEF, hence the Δ^i_j and $\bar{\Delta}^i_j$ masks are defined. However, the verification oracle of the ideal-int world does not make any computations, and hence $\tilde{\Delta}^i_j$ and $\tilde{M}^i_j$ are not defined. We therefore have to define the sampling of these variables which will be done at the end of the experiment (and thus do not affect the adversarial queries).

We fix $\tilde{\Delta}^i_1 = 0^n$ for $1 \le i \le q_v$ and sample each of the remaining masks $\tilde{\Delta}^i_j$ uniformly and independently at random, except when such a mask is trivially defined due to a "common prefix" (defined shortly) with a previous query (encryption, decryption or verification). Once these masks are sampled, we use the SAEF decryption algorithm with π_0 and these masks to compute $\tilde{M}^i_j$. This giveaway of additional information can only help the adversary by increasing its advantage and hence can be considered here for upper bounding the targeted (above-mentioned) adversarial advantage.

Block-tuple representation. To minimize notation and ease the analysis, we switch to an equivalent representation (called *block-tuple* representation)

by defining the i^{th} encryption query as $(\mathsf{T}^i_j, \Delta^i_j, X^i_j, Y^i_j)^{\ell^i}_{j=1}, T^i$, such that $\ell^i = a^i + m^i + 2$. The j^{th} quadruple (out of these ℓ^i) represents the processing done in the j^{th} forkcipher call in the query, with the string T^i_j used as forkcipher tweak, the corresponding whitening mask Δ^i_j, the (possibly padded) associated data/plaintext block X^i_j and the empty/ciphertext block Y^i_j. With formal details:

- For the very first block, we always have $\mathsf{T}^i_1 = N\|1\|F$ for a flag $F \in \{0,1\}^3$ and $\Delta^i_1 = 0^n$. For blocks with $j > 1$ we have $\mathsf{T}^i_j = 0^{t-3}\|F$ for an $F \in \{0,1\}^3$.
- If $|A| > 0$, for $1 \leq j \leq a^i$ 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 defined in Figure 1.
- If $|M| > 0$, for $a^i + 2 \leq j < \ell^i$ 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 defined in Figure 1.
- If $A = M = \varepsilon$, we have $j = \ell^i = 1$, $X^i_j = \mathsf{pad10}(\varepsilon), Y^i_j = \varepsilon$ and $F = 111$.

With similar definition, we define the block-tuple representation for decryption queries as $(\bar{\mathsf{T}}^i_j, \bar{\Delta}^i_j, \bar{X}^i_j, \bar{Y}^i_j)^{\bar{\ell}^i}_{j=1}, \bar{T}^i$ with $\bar{\ell}^i = \bar{m}^i + \bar{a}^i + 2$, and for verification queries as $(\tilde{\mathsf{T}}^i_j, \tilde{\Delta}^i_j, \tilde{X}^i_j, \tilde{Y}^i_j)^{\tilde{\ell}^i}_{j=1}, \tilde{T}^i, b^i$ with $\tilde{\ell}^i = \tilde{m}^i + \tilde{a}^i + 2$. We now simplify the notation by re-indexing the decryption queries from $q_e + 1$ to $q_e + q_d$ and the verification queries from $q_e + q_d + 1$ to $q_e + q_d + q_v$. Further, with this new indexing, we drop the bars and tildes from the variables. The decryption queries and verification queries are thus denoted as $(\mathsf{T}^i_j, \Delta^i_j, X^i_j, Y^i_j)^{\ell^i}_{j=1}, T^i$ for $q_e + 1 \leq i \leq q_e + q_d$ and $(\mathsf{T}^i_j, \Delta^i_j, X^i_j, Y^i_j)^{\ell^i}_{j=1}, T^i, b^i$ for $q_e + q_d + 1 \leq i \leq q_e + q_d + q_v$, respectively.

We recall that the transcript of $\mathcal{A}$ is the set of all its query-response tuples which is uniquely defined at the end of its interaction (when q_e, q_d, and q_v are finalized). This means that the transcript for $\mathcal{A}$ is invariant of its query order, i.e., it remains the same even when $\mathcal{A}$ makes the same encryption, decryption, and verification queries but in a different order, and therefore, this re-indexing can be seen as a valid step which simplifies the notation in the transcripts.

The block-tuple representation for SAEF mode was originally introduced in [4] with an equivalence proof showing how to reconstruct the original representation from it. In this work, we use the block-tuple representation with the original notation and refer the reader to [4, App. A] for the full proof of equivalence between the two representations.

Blockwise common prefix of queries. The block-tuple notation allows us to introduce the following natural definition of the longest common blockwise prefix between any two *queries*. We define the longest common prefix between the i^{th} and i'^{th} query with $\ell^i \leq \ell^{i'}$ w.l.o.g. as

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

Note that this definition covers common blockwise prefixes between all types of query pairs (for example, between two encryption queries or between an encryption and a decryption query, or between a decryption and a verification query,

etc). Informally, $\mathsf{llcp}_n(i,i')$ represents the number of internal chaining values Δs that are trivially equal between the i^{th} and i'^{th} query. To exemplify, if the nonces N^i and $N^{i'}$ are different then we have $\mathsf{llcp}_n(i,i') = 0$. If we have two queries with $N^i = N^{i'}$ but the i'^{th} query has AD $A^{i'} = A^i \| M_1^i$ i.e., equal to the AD of the i^{th} query appended with its first message block, 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 $\mathsf{llcp}_n(i) = \max_{1<i'<i} \mathsf{llcp}_n(i,i')$. One should note here that for a verification query, all the encryption and decryption queries are always taken into account (as per the convention of query indexing).

Sampling of Δ masks. Since the block-tuple notation and common prefix are defined, we can now use them to formally define the sampling of Δ_j^i masks for the verification queries (i.e., for $q_e + q_d < i \leq q_e + q_d + q_v$) of the ideal-int world.

For the i^{th} verification query with $1 \leq j \leq \mathsf{llcp}_n(i) + 1$, we let $\Delta_j^i = \Delta_j^{i'}$ for the smallest $i' < i$ such that 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.

Extended transcripts. Using the block-tuple notation, we can now re-define the extended transcripts as

$$\tau = \left\langle \left(\left(\mathsf{T}_j^i, \Delta_j^i, X_j^i, Y_j^i \right)_{j=1}^{\ell^i}, T^i \right)_{i=1}^{q_e+q_d}, \left(\left(\mathsf{T}_j^i, \Delta_j^i, X_j^i, Y_j^i \right)_{j=1}^{\ell^i}, T^i, b^i \right)_{i=q_e+q_d+1}^{q_e+q_d+q_v} \right\rangle .$$

Note that the terms q_e, q_d, q_v, a and m here are themselves random variables and hence can vary for different *attainable* transcripts. However, due to the assumption that the adversary can only make at most σ many block queries, we always have $\sum_{i=1}^{q_e+q_d+q_v}(a^i + m^i + 2) = \sigma$. Also, note that it is impossible for two distinct transcripts $\tau = \langle (N^i, A^i, M^i, C^i)_{i=1}^{q_e}, (\bar{N}^i, \bar{A}^i, \bar{C}^i, \bar{M}^i)_{i=1}^{q_d}, (\tilde{N}^i, \tilde{A}^i, \tilde{C}^i, b^i)_{i=1}^{q_v} \rangle$ and $\tau' = \langle (N'^i, A'^i, M'^i, C'^i)_{i=1}^{q'_e}, (\bar{N}'^i, \bar{A}'^i, \bar{C}'^i, \bar{M}'^i)_{i=1}^{q'_d}, (\tilde{N}'^i, \tilde{A}'^i, \tilde{C}'^i, b'^i)_{i=1}^{q'_v} \rangle$ to have the same block-tuple representation (for the proof of this claim, we refer the reader to [4], Proposition 1).

Coefficients H. Let us represent the distribution of the transcript in the real-int world and the ideal-int world by Θ_{rein} and Θ_{idin}, respectively.

The proof relies on the fundamental lemma of the coefficients H technique as defined in Lemma 1 above. We represent the j^{th} block call of the i^{th} query in a transcript by the index tuple (i,j). We say that an attainable transcript τ is bad if one of the following conditions occurs:

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

$\mathsf{BadT_2}$ a.k.a. "Mask Collision": There exists $(i',j') < (i,j)$ such that $1 \leq i \leq q_e + q_d + q_v$, $\mathsf{llcp}_n(i) < j < \ell^i$ (not lies in the longest common prefix), and both

the block calls have the same tweaks $\mathsf{T}^i_j = \mathsf{T}^{i'}_{j'}$ and different inputs $X^i_j \oplus \Delta^i_j \neq X^{i'}_{j'} \oplus \Delta^{i'}_{j'}$ but the following masks $\Delta^i_{j+1} = \Delta^{i'}_{j'+1}$ collide. (although such a collision cannot occur in the real-int world where the masks are generated using permutation, it can still occur in the ideal-int world).

$\mathsf{BadT_3}$ a.k.a. "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^i_j is 0 and $\pi_{\mathsf{T}^i_j,1}(X^i_j \oplus \Delta^i_j) = T^i$.

Case 2. The last bit of T^i_j is 1, $\mathsf{right}_{n-|T^i|}(X^i_j) = 10^{n-|T^i|-1}$ and $\mathsf{left}_{|T^i|}(\pi_{\mathsf{T}^i_j,1}(X^i_j \oplus \Delta^i_j)) = T^i$.

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

We highlight that case 3 (of the forgery bad event $\mathsf{BadT_3}$) above is not needed in the existing integrity analyses [4, 7] of SAEF, however, it is required under RUP setting where an adversary can observe the unverified plaintexts. In simple words, it captures the scenario where an adversary makes decryption queries (with incomplete last blocks) that only differ in the last complete ciphertext blocks (i.e., $\bar{C}_*$s) to find an unverified padding that matches with 10^{w-1} for some integer $1 \leq w \leq n-1$. Once found, the adversary uses that same ciphertext with its tag part (of size $n-w$ bits) replaced with random strings as its forgeries.

We denote by $\mathcal{T}_{\text{bad}}$, the set of "bad" transcripts that is defined as the subset of attainable transcripts for which the transcript predicate $\mathsf{BadT}(\tau) = (\mathsf{BadT_1}(\tau) \vee \mathsf{BadT_2}(\tau) \vee \mathsf{BadT_3}(\tau)) = 1$. We denote by $\mathcal{T}_{\text{good}}$, the set of attainable transcripts which are not in the set $\mathcal{T}_{\text{bad}}$ (and are therefore called good transcripts).

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

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

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

Given the well-defined bad events, both of these lemmas can be easily proved using standard probability analysis. We defer the proof of Lemma 2 and 3 to App. E to save space. Combining the results of Lemma 2 and 3 (taking $\epsilon_1 = 0$) into Lemma 1, we obtain the upper bound $\mathbf{Adv}^{\mathbf{intrup}}_{\text{SAEF}[(\pi_0,\pi_1)]}(\mathcal{A}) \leq \frac{\sigma^2}{2^n} + \frac{4 \cdot q_d q_v}{2^n}$ and hence the result of the Theorem 1.

3.3 Confidentiality

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

With this replaced "F" scenario, the adversary is left to distinguish between $\mathbf{soprpf\text{-}real}_{\mathrm{SAEF}[(\pi_0,\pi_1)]}$ (called the "real-conf world") and $\mathbf{soprpf\text{-}ideal}_{\mathrm{SAEF}[(\pi_0,\pi_1)]}$ (called the "ideal-conf world"). We now need to bound $\mathbf{Adv}^{\mathbf{soprpf}}_{\mathrm{SAEF}[(\pi_0,\pi_1)]}(\mathcal{A}) = \Pr[\mathcal{A}^{\mathbf{soprpf\text{-}real}_{\mathrm{SAEF}[(\pi_0,\pi_1)]}}] - \Pr[\mathcal{A}^{\mathbf{soprpf\text{-}ideal}_{\mathrm{SAEF}[(\pi_0,\pi_1)]}}]$.

Transcripts and additional information. As before, we record the interaction of $\mathcal{A}$ with its oracle in a transcript containing the encryption and decryption queries and their responses as $\tau = \langle (N^i, A^i, M^i, C^i)_{i=1}^{q_e+q_d} \rangle$ such that the AD A, message M, and ciphertext C are further partitioned in blocks as described in the integrity proof. Note that as before, q_e, q_d, a and m here are themselves random variables and thus can vary for distinct *attainable* transcripts. We also assume that the adversary can make at most $\sigma' \leq \sigma$ block queries. Thus, we always have $\sum_{i=1}^{q_e+q_d}(a^i + m^i + 2) \leq \sigma$.
To simplify the proofs, we again provide the adversary with additional information: **(i)** Δ_j^i masks for $1 \leq i \leq q_e + q_d$ and for $1 \leq j \leq \ell^i = a^i + m^i + 2$ that are internally computed by the encryption and decryption algorithms of SAEF. **(ii)** Tag bits that would normally be discarded by truncation (if any), i.e., we now have $|T^i| = n$ for $1 \leq i \leq q_e$.

This information is given to the adversary after it has made all its queries. These masks, as well the bits extending the tags to n bits, are well-defined in the real-conf world, which faithfully implements SAEF encryption and decryption algorithms. In the ideal-conf world, however, they are not defined as they are directly computed by an online permutation (or its inverse), a random permutation (or its inverse), and a random function, and the tags are directly sampled with the desired length. Hence, for the ideal world, we sample the masks uniformly at random while maintaining consistency with SAEF's prefix preservation (defined in detail below), and each authentication tag is simply extended by 0 to $n-1$ uniform bits as necessary.

Block-tuple representation. As before, we use the block-tuple representation. We represent the i^{th} query (encryption or decryption) as $(\mathsf{T}_j^i, \Delta_j^i, X_j^i, Y_j^i)_{j=1}^{\ell^i}, T^i$, with $\ell^i = a^i + m^i + 2$ and each of the ℓ^i quadruples consisting of the string T_j^i used as forkcipher tweak, the j^{th} whitening mask Δ_j^i, the (possibly padded) associated data/plaintext block X_j^i and the empty/ciphertext block Y_j^i.

We reuse the same definition of the length of the longest common blockwise prefix between the i^{th} and the i'^{th} query $\mathsf{llcp}_n(i, i')$ and between the i^{th} query and all preceding queries $\mathsf{llcp}_n(i)$.

Sampling of Δ masks. Using the block-tuple representation, we now detail the sampling of Δ masks in the ideal world. For each $1 \leq i \leq q_e + q_d$, we let $\Delta_j^i = \Delta_j^{i'}$ for $1 \leq j \leq \mathsf{llcp}_n(i)+1$ with the smallest $i' < i$ such that i'^{th} query has $\mathsf{llcp}_n(i) = \mathsf{llcp}_n(i, i')$. For $\mathsf{llcp}_n(i)+1 < j \leq \ell^i$ we sample the mask Δ_j^i uniformly at random. Note that this sampling of Δ masks is not specific to the sequential indexing of these queries, i.e., the same definition of Δ masks holds for the case of an adaptive adversary with encryption and decryption queries in any order.

Extended transcripts. The extended transcripts available to the adversary at the decision-making time are denoted as $\tau = \langle((\mathsf{T}^i_j, \Delta^i_j, X^i_j, Y^i_j)^{\ell^i}_{j=1}, T^i)^{q_e+q_d}_{i=1}\rangle$. Also, by convention, none of these transcripts contain duplicate queries.

Coefficients H. Let Θ_{reco} and Θ_{idco} represent 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 defined as Lemma 1 above. We represent the j^{th} block call of the i^{th} query in a transcript by the index tuple (i, j). We say an attainable transcript τ is bad if one of the following conditions occur:

BadT'_1 a.k.a. "Input Collision": There exists $(i', j') < (i, j)$ (the block indexed by (i', j') precedes (i, j)) such that $1 \leq i \leq q_e + q_d$, $\mathsf{llcp}_n(i) < j \leq \ell^i$ is not in the longest common prefix of the i^{th} query, and the (i, j) block call has 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 a.k.a. "Output Collision": There exists $(i', j') < (i, j)$ such that $1 \leq i \leq q_e + q_d$, $\mathsf{llcp}_n(i) < j < \ell^i$ (not in the longest common prefix), and both the block calls have the same tweaks $\mathsf{T}^i_j = \mathsf{T}^{i'}_{j'}$ and different inputs $X^i_j \oplus \Delta^i_j \neq X^{i'}_{j'} \oplus \Delta^{i'}_{j'}$, however, one of the outputs collide i.e., one of the followings are 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 $1 \leq i \leq q_e$, $j = \ell^i$ 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 $1 \leq i \leq q_e$, $j = \ell^i$ and $j' = \ell^{i'}$.

(Note that such a collision cannot occur in the real-conf world where the masks and the tags are generated with a permutation).

One should also note that these bad events are not specific to the case of sequential indexing of these queries as that indexing is only used to make the proof simpler to understand. The same definition of these bad events holds for the case of adaptive adversaries with encryption and decryption queries in any order.

We let $\mathcal{T}'_{\text{bad}}$ be the set of "bad" transcripts defined as the subset of attainable transcripts for which the transcript predicate $\mathsf{BadT}'(\tau) = (\mathsf{BadT}'_1(\tau) \vee \mathsf{BadT}'_2(\tau))$ evaluates true. We define $\mathcal{T}'_{\text{good}}$ as the set of attainable transcripts which are not in the set $\mathcal{T}'_{\text{bad}}$ (and therefore from now on called good transcripts).

Lemma 4. *For $\mathcal{T}'_{bad}$ as defined above, we have* $\Pr[\Theta_{idco} \in \mathcal{T}'_{bad}] \leq \frac{3 \cdot \sigma^2}{2^{n+1}}$.

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

Given the well-defined bad events, both of these lemmas can be easily proved using standard probability analysis. We defer the proof of Lemma 4 and 5 to App. E to save space. Combining the results of Lemma 4 and 5 (taking $\epsilon_1 = 0$) into Lemma 1, we obtain the upper bound $\mathbf{Adv}^{\mathbf{soprpf}}_{\text{SAEF}[(\pi_0,\pi_1)]}(\mathcal{A}) \leq \frac{3 \cdot \sigma^2}{2^{n+1}}$ and thus the confidentiality result of the Theorem 1.

4 Conclusion

We propose two RUP confidentiality notions for online AE schemes named sOPRPF and CR-RUP. sOPRPF 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. CR-RUP is a weaker version of RUP confidentiality that says the confidentiality of plaintexts that are not subject to leakage and nonce-misuse is preserved under RUP. 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 prove that SAEF provides OAE-RUP security as long as the total amount of data processed with a single key is $\ll 2^{n/2}$ blocks, with n being the blocksize of the underlying forkcipher. This concludes that SAEF continues to provide reasonable security even when the unverified plaintext is released. More specifically, the integrity of SAEF remains intact whereas the confidentiality is degraded but preserved up to the longest common prefix. Our newly proven security properties on the original SAEF constructions are of high relevance to many resource-constrained applications of lightweight cryptography where the constrained devices are forced or required to leak portions of unverified plaintext in decryption and may also suffer from accidental or forced nonce-repetitions.

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.

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 ANY-DAE. 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 SAEF: Pictorial Diagram

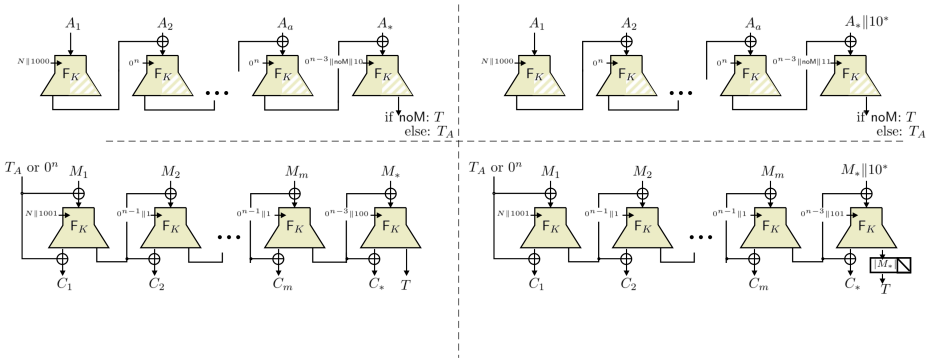

Fig. 2: The encryption algorithm of SAEF[F] mode. The bit $\mathsf{noM} = 1$ iff $|M| = 0$. The picture illustrates the processing of AD when length of AD is a multiple of n (**top left**) and when the length of AD is not a multiple of n (**top right**), and the processing of the message when length of the message is a multiple of n (**bottom left**) and when the length of message is not a multiple of n (**bottom right**). The white hatching denotes that an output block is not computed.

B 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 [4] (OAE) Security: OAE ensures that the AE mode can be implemented in an online manner with reasonable security guarantees, i.e., the mode is secure against blockwise (hence also nonce-misusing) adversaries.

2. Nonce-Misuse Resilience [8] (NMR) Security: NMR ensures that the mode provides reasonable security for a query even when the nonce is repeated in "other" queries, i.e., security against a specific set of nonce-misusing adversaries.

3. Misuse-Resistant AE [41] (MRAE) Security: MRAE is a stronger version of NMR. This property ensures that the AE mode provides reasonable security guarantees against nonce-misusing adversaries.

Note that MRAE is a more robust security definition when compared with OAE (which is more robust than NMR) but an MRAE-secure scheme requires

at least two-pass over plaintext data to achieve full dependence from every bit of plaintext to every bit of ciphertext and thus cannot be implemented in an online fashion (which allows only one-pass over data). This makes OAE security the optimal choice in applications with online requirements.

4. Integrity under RUP [5] (INT-RUP): INT-RUP ensures that the AE mode is secure w.r.t. integrity even when the release of unverified plaintexts is allowed, i.e., integrity against adversaries seeing unverified decrypted plaintexts.

5. Plaintext Awareness [5] (PA): PA, when combined with IND-CPA, ensures that the AE mode is secure w.r.t. confidentiality even when the release of unverified plaintexts is allowed, i.e., confidentiality against adversaries seeing unverified decrypted plaintexts.

Note that PA is a very strong notion of security that just like MRAE requires at least two passes over plaintext data for encryption (where ideally, the reason for the first pass is to generate a plaintext-dependent random and secret IV on which the adversary has no control. It can then be used with later passes for encryption). Hence, no OAE-secure scheme (as it allows only one pass over data) can achieve it without weakening its existing definition.

6. sOPRPF (Section 2.2): sOPRPF security ensures that the AE scheme provides confidentiality (up to the leakage of the longest common prefix) even when the release of unverified plaintexts is allowed and the nonce is repeated. We keep sOPRPF and INT-RUP separated in Table 1 instead of using OAE-RUP as some schemes only come with INT-RUP security results.

7. One-pass over data for encryption and decryption: These properties ensure that the mode only requires one pass over the data throughout its execution, i.e., the mode supports online encryption and online decryption, respectively. Again, note that OAE-RUP implies single-pass encryption and decryption but the opposite implication is not always true.

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.

C 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, $\mathsf{CR\text{-}RUP\text{-}real}_{\Pi}$ and $\mathsf{CR\text{-}RUP\text{-}ideal}_{\Pi}$. In both games, the adversary is given access to an encryption, a decryption, and a verification oracle. In the game $\mathsf{CR\text{-}RUP\text{-}real}_{\Pi}$, all three oracles faithfully implement the corresponding algorithms of Π using the same randomly sampled secret key. In the game $\mathsf{CR\text{-}RUP\text{-}ideal}_{\Pi}$, the decryption and verification oracles are the same as in $\mathsf{CR\text{-}RUP\text{-}real}_{\Pi}$ 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* $\mathsf{CR\text{-}RUP\text{-}real}_{\Pi}$ *and* $\mathsf{CR\text{-}RUP\text{-}ideal}_{\Pi}$ *be two games as defined above. The* CR-RUP *advantage of $\mathcal{A}$ against Π is then defined as*

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

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.

D Relations Among AEAD Notions

In this section, we discuss the relations/differences among popular AEAD security notions. For the comparison, we consider two key parameters that defines a security notion - the powers given to the adversary and the achieved security goal.

For simplicity, we denote an AEAD security notion that achieves X-type of security (i.e., confidentiality and integrity of the processed messages and integrity of the processed associated data) against Y-type adversaries by (X, Y)-notion. In order to keep the comparison fair for various input-order sensitive notions such as OAE, we also assume that the AEAD schemes targeted by any of these notions processes the inputs in the ordered form (N, A, M) where N, A and M are the nonce, associated data and the message.

We provide two tables consisting X and Y types and a plot over these types to compare the strengths of the popular AEAD notions in Fig. 3. 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], $\mathsf{RAE}_{\mathsf{sim}}$ [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).

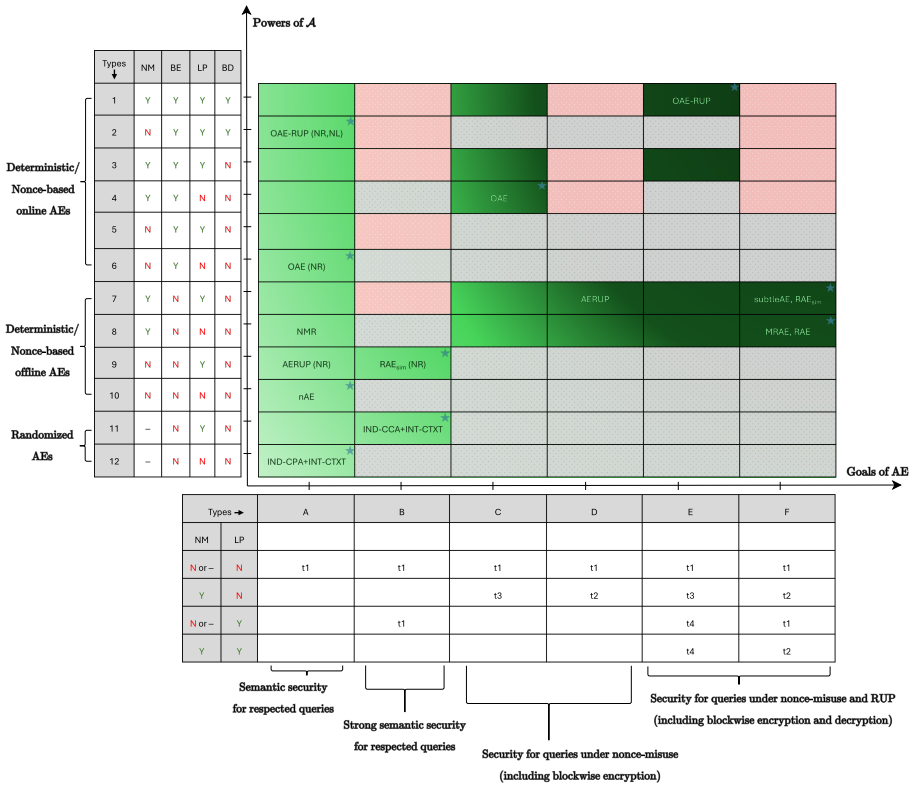

Fig. 3: Relations among popular AEAD notions.

Abbreviations. In Fig. 3, Y types varies from 1 to 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 (NP) 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., $Y = 4$ 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. 3.

We consider four different types of well-defined security goals dubbed t1, t2, t3 and 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) t1 - confidentiality with only leakage of the length of 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., $X =$ 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. 3, 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. 3. With all the abbreviations defined, (X, Y)-notion can now easily be understood. The position of an (X, Y)-notion in Fig. 3 is simply defined by the target adversary setting and the captured goals of it. Let us consider the following example - $\text{RAE}_{\mathsf{sim}}$ [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 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. 3 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 App. C), OAE-RUP (NR,NL) can be defined as CR-RUP+INT-RUP.

E Omitted Lemma Proofs

E.1 Proof of Lemma 2

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

Note that for all $i' < i$ and $j = j' = \mathsf{llcp}_n(i) + 1$, we have $\Delta^i_j = \Delta^{i'}_{j'}$ but $X^i_j \neq X^{i'}_{j'}$ and thus for all such cases the probability that the above equality occurs is 0. Contrastingly, for all $i' \leq i$ and $j' \neq j$ or $j \neq \mathsf{llcp}_n(i) + 1$, the two masks has marginal probability $1/2^n$ to collide in Θ_{idin}. Since there are total σ possible values of (i, j) in a transcript, each having no more than σ possible values of (i', j'), we get $\Pr[\mathsf{BadT}_1(\Theta_{idin}) = 1] \leq \frac{\sigma^2}{2} \cdot \max\left\{0, \frac{1}{2^n}\right\} = \frac{\sigma^2}{2^{n+1}}$.

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

Note that from the definition of the predicate BadT_2, we have $j + 1 \neq \mathsf{llcp}_n(i) + 1$. This means that the marginal probability of Δ^i_{j+1} being equal to $\Delta^{i'}_{j'+1}$ is $1/2^n$. Since there are total σ possible values of (i, j) in a transcript, each with no more than σ possible values of (i', j'), we get $\Pr[\mathsf{BadT}_2(\Theta_{idin}) = 1] \leq \frac{\sigma^2}{2^{n+1}}$.

BadT_3. Now, for any transcript in $\mathcal{T}_{\text{bad}}$ with BadT_3 set to 1 and BadT_1 set to 0, we know that 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 $j = j' = \mathsf{llcp}_n(i)$. Clearly, in such a case $\Delta^i_j = \Delta^{i'}_{j'}$, $X^i_j = X^{i'}_{j'}$ but $T^i \neq T^{i'}$. Since $T^{i'}$ is the correct tag for the given ciphertext, $T^i \neq T^{i'}$ cannot trigger BadT_3, and yields 0 probability.
2. For some $i' \leq q_e + q_d$, $j = \ell^i$ and $j' = \ell^{i'}$, we have $j = j' = \mathsf{llcp}_n(i) + 1$. We have $\Delta^i_j = \Delta^{i'}_{j'}$ but $X^i_j \neq X^{i'}_{j'}$ and thus 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 there is produced with a tweak used at most once per encryption query, corresponding to a probability $1/(2^n - q_e) \leq 2/2^n$. For the second condition, we can upper bound the product of the probabilities of having the correct padding in the block X^i_j (at most $2^{|T^i|}/(2^n - q_e - q_d)$), and of having the correct truncated tag (at most $2^{n-|T^i|}/(2^n - q_e)$) by $4/2^n$. For the third condition, with any choice of $q_e + 1 \leq i_d \leq q_e + q_d$ such that $|T^i| = |T^{i_d}|$, we can upper bound the product of the probabilities of having the correct padding in the block $X^{i_d}_{\ell^{i_d}}$ (at most $2^{|T^i|}/(2^n - q_e - q_d)$), and of having the correct truncated tag for the verification query (at most $2^{n-|T^i|}/(2^n - q_e)$) by $4/2^n$. Now, since there are a total q_d many possible choices for i_d, the total probability of the third condition is upper bounded by $4q_d/2^n$.
3. For all $i' \leq q_e + q_d$ when we have $j > \mathsf{llcp}_n(i, i') + 1$. We know that the Δ^i_j 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 $1/2^n$. For the second condition, the correct padding is found with probability $1/2^{n-|T^i|}$ (using the randomness of Δ^i_j), and the correct tag is found with probability at most $2^{n-|T^i|}/(2^n - q_e)$, thanks to freshness of $X^i_j \oplus \Delta^i_j$, relying on $\mathsf{BadT}_1(\Theta_{idin}) = 0$ w.l.o.g., yielding a probability of

at most $2/2^n$. For the third condition, with similar reasoning the correct padding is found with probability $q_d/2^{n-|T^i|}$ (using the randomness of Δ^i_j), and the correct tag is found with probability at most $2^{n-|T^i|}/(2^n - q_e)$, providing a probability of at most $2q_d/2^n$.

Since there are total q_v possible verification queries, we get $\Pr[\mathsf{BadT}_3(\Theta_{idin}) = 1|\mathsf{BadT}_1(\Theta_{idin}) = 0] \leq q_v \cdot \max\left\{0, \frac{4q_d}{2^n}, \frac{2q_d}{2^n}\right\} = \frac{4 \cdot q_d q_v}{2^n}$ and we obtain by the union bound that $\Pr[\Theta_{idin} \in \mathcal{T}_{\text{bad}}] \leq \frac{\sigma^2}{2^n} + \frac{4 \cdot q_d q_v}{2^n}$. □

E.2 Proof of Lemma 3

Proof. Note that a good transcript has the following 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 same tweaks (i.e., $\mathsf{T}^i_j = \mathsf{T}^{i'}_{j'}$) then both calls will always have different inputs and different outputs. 2. **(ii)** For each query to the verification oracle i.e., $1 \leq i \leq q_v$, the transcript contains $b^i = \perp$ in the verification result i.e., 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 two parts of a transcript τ consisting respectively encryption-decryption and verification queries, so that $\tau = \langle \tau_{ed}, \tau_v \rangle$. With a slight abuse of notation, 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 $\Pr[\Theta_{idin} = \tau] = \Pr[\Theta_{idin,ed} = \tau_{ed}] \cdot \Pr[\Theta_{idin,v} = \tau_v | \Theta_{idin,ed} = \tau_{ed}]$ and consequently

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

This is true because the encryption and decryption oracles in the real-int world and in the ideal-int world are identical, and so $\Pr[\Theta_{rein,ed} = \tau_{ed}] = \Pr[\Theta_{idin,ed} = \tau_{ed}]$. Further abusing notation, we let $\tau_{v,\Delta}$ denote the marginal event of all Δ masks in the verification queries (as variables) being equal to 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 with random permutations with the input-output pairs fixed from the encryption-decryption parts in both worlds. Further, using this equality, we get

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

Let us now consider that there are δ many Δs in τ that are fixed/predefined due to all internal common prefixes. Clearly, one can write that $\delta = \sum_{i=1}^{q_e+q_d+q_v} (\mathsf{llcp}_n(i) + 1)$ (the extra 1 represents the Δ^i_1 as it is always fixed to 0). In the ideal-int world, since the Δs corresponding to the remaining $(\sigma - \delta)$ unique block calls are sampled uniformly and independently and all verification oracle results are $\perp$, one has $\Pr[\Theta_{idin,v,\Delta} = \tau_{v,\Delta} | \Theta_{idin,ed} = \tau_{ed}] = \frac{1}{(2^n)^{\sigma-\delta}}$. In the real-int world, these $(\sigma - \delta)$ Δs are no longer uniformly distributed

but are instead 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 the tweak 0^n and at least $g_2 = \sum_{i=1}^{q_e+q_d+q_v}(m^i - 1)$ block calls with the tweak $0^{n-1}\|1$. Thus, one has $\Pr[\Theta_{rein,v,\Delta} = \tau_{v,\Delta} | \Theta_{rein,ed} = \tau_{ed}] \geq \frac{1}{(2^n)_{g_1}(2^n)_{g_2}(2^n)^{\sigma-\delta-g_1-g_2}}$.
One should note here that the above expression is not an equality and only provides an upper bound on the targeted probability because there exist more permutation calls that can have tweak collisions (for example, the first block calls of any set of queries will have same tweaks if they all have same nonce). Now, from the above expressions, we get

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

□

E.3 Proof of Lemma 4

Proof. **BadT'_1.** For any transcript in $\mathcal{T}'_{\text{bad}}$ with BadT'_1 set to 1, we know that there exists at least one pair of (i, j) and (i', j') such that $\mathsf{llcp}_n(i) < j \leq \ell^i$, $(i', j') < (i, j)$ and $\Delta^i_j \oplus \Delta^{i'}_{j'} = X^i_j \oplus X^{i'}_{j'}$.

Note that for all $i' < i$ and $j = j' = \mathsf{llcp}_n(i) + 1$, we have $\Delta^i_j = \Delta^{i'}_{j'}$ but $X^i_j \neq X^{i'}_{j'}$ and thus for such cases the probability that the above equality occurs is 0 (one should notice that the case when we have only the nonce collision is also covered in here. This is because for all $i' < i$ if $N^i = N^{i'}$ then we have $j = j' = 1$ and $\mathsf{llcp}_n(i) = 0$ which implies that $\Delta^i_1 = \Delta^{i'}_1 = 0$ and $X^i_1 \neq X^{i'}_1$). On the other hand, 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}. Since there are total $\sigma' \leq \sigma$ possible values of (i, j) in a transcript, each having no more than $\sigma' \leq \sigma$ possible values of (i', j'), we get $\Pr[\mathsf{BadT}'_1(\Theta_{idco}) = 1] \leq \frac{\sigma^2}{2} \cdot \max\left\{0, \frac{1}{2^n}\right\} = \frac{\sigma^2}{2^{n+1}}$.

BadT'_2. Similarly, for any transcript in $\mathcal{T}'_{\text{bad}}$ with BadT'_2 set to 1, we know that there exists at least one pair of blocks $(i', j') < (i, j)$ such that $\mathsf{llcp}_n(i) < j < \ell^i$ and one of the followings is true (with appropriate values of i and j as defined in the bad events above)

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

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

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

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

E.4 Proof of Lemma 5

Proof. Note that 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 same tweaks (i.e., $\mathsf{T}^i_j = \mathsf{T}^{i'}_{j'}$) then both blocks will always 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 $\Delta^i_{j+1} \neq \Delta^{i'}_{j'+1}$ and if $1 \leq i \leq q_e$ then different tags (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 the ideal-conf worlds can now be computed. With a slight abuse of notation, we let τ_Δ denote the marginal event of all Δ masks in the queries (as variables) being equal to the values in the transcript. With these notations, 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 whereas the right side of this inequality corresponds to mappings of a random online permutation with input size of $\geq n$-bits. Similarly, for the tags (fixed and unique), the left side of this inequality corresponds to a random permutation whereas the right side of this inequality corresponds to a random function with same input size (n-bits).

Let us now consider that τ has in total δ' Δs that are fixed/predefined due to all internal common prefixes. Clearly, one can write that $\delta' = \sum_{i=1}^{q_e+q_d} (\mathsf{llcp}_n(i)+1)$ (the extra 1 here stands for the Δ^i_1 which is always fixed to 0). In the ideal-conf world, since the Δs corresponding to these $\sigma' - \delta'$ unique block calls are sampled uniformly and independently, one has $\Pr[\Theta_{idco,\Delta} = \tau_\Delta] = \frac{1}{(2^n)^{\sigma'-\delta'}}$ (Note that the sampling of the tags is already covered with the inequality defined above and is independent of the sampling of Δs here).

In the real-conf world, these $\sigma' - \delta'$ Δs are no longer uniformly distributed but are defined 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 the tweak 0^n and at least $g'_2 = \sum_{i=1}^{q_e+q_d} (m^i - 1)$ block calls with the tweak $0^{n-1}\|1$. Thus, one has

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

Now, from the above three expressions, we get

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

□

F Block-Tuple and Extended Transcript Representation Equivalence Proof [4]

We prove the equivalence by going from a block-tuple-represented query to the SAEF input and output arguments. This is possible because of the sequence of tweak strings $\mathsf{T}^i_1, \ldots, \mathsf{T}^i_{\ell^i}$, that can uniquely identify the types and sizes of their corresponding input/output blocks i.e., if they were AD or message blocks, and whether the last block of AD or message was n-bit long or not. Given a block-tuple represented query $((\mathsf{T}j, \Delta_j, X_j, Y_j)j = 1^\ell, T)$, we can get back (N, A, M, C) as follows (where $\mathsf{unpad10}(X)$ removes 10^* from the end of string X) by iterating over the block tuples with the current tuple as $(\mathsf{T}, \Delta, X, Y)$:

1. Parse T as $N\|1\|F$. If $F \in \{110, 111, 100, 101\}$ then necessarily $\ell = 1$, and if
   $F = 110$**:** Set $A = X, M = \varepsilon$ and $C = T$ and stop.
   $F = 111$**:** Set $A = \mathsf{unpad10}(X), M = \varepsilon$ and $C = T$ and stop.
   $F = 100$**:** Set $A = \varepsilon, M = X$ and $C = Y\|T$ and stop.
   $F = 101$**:** Set $A = \varepsilon, M = \mathsf{unpad10}(X)$ and $C = Y\|T$ and stop.
   If $F \in \{010, 011\}$ then set $A = X_1$ respectively $A = \mathsf{unpad10}(X_1)$ and $M = C = \varepsilon$ (no more AD blocks are expected). If $F = 001$ set $A = \varepsilon$, $M = X_1$, and $C = Y_1$ (no more AD blocks are expected). Finally, if $F = 000$ set $A = X_1$ and $M = C = \varepsilon$ (more AD blocks are expected). Set $F_{last} \leftarrow F$ and go to the next tuple.
2. Parse T as $S\|F$. If $S \neq 0^{n-3}$ abort. If $F \neq 000$ go to the next step. If $F_{last} \neq 000$ abort. Set $A = A\|X$ and $F_{last} = 000$. Go to the next tuple and repeat this step.
3. Parse T as $S\|F$. If $S \neq 0^{n-3}$ abort. If $F \notin \{010, 011, 110, 111\}$ go to the next step. If $F_{last} \neq 000$ abort. If
   $F = 110$**:** Set $A = A\|X, M = \varepsilon$ and $C = T$ and stop.
   $F = 111$**:** Set $A = A\|\mathsf{unpad10}(X), M = \varepsilon$ and $C = T$ and stop.
   $F = 010$**:** Set $A = A\|X$.
   $F = 011$**:** Set $A\|\mathsf{unpad10}(X)$ and stop.
   Set $F_{last} = F$. Go to the next tuple.
4. Parse T as $S\|F$. If $S \neq 0^{n-3}$ abort. If $F \neq 001$ go to the next step. If $F_{last} \notin \{010, 011, 110, 111\}$ abort. Set $M = M\|X$, $C = C\|Y$ and $F_{last} = 001$. Go to the next tuple and repeat this step.
5. Parse T as $S\|F$. If $S \neq 0^{n-3}$ abort. If $F \notin \{100, 101\}$ or if $F_{last} \notin \{010, 011, 001\}$ abort. If
   $F = 100$**:** Set $M = M\|X$.
   $F = 101$**:** Set $M = M\|\mathsf{unpad10}(X)$.
   Set $C = C\|Y\|T$ and stop.