Nonce-Misuse Security of the SAEF Authenticated Encryption mode⋆

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

[1] Alpen-Adria University, Klagenfurt, Austria
[2] imec-COSIC, KU Leuven, Belgium
[3] CSEM, Switzerland.
elena.andreeva@aau.at, amitsingh.bhati@esat.kuleuven.be, damian.vizar@csem.ch

Abstract. ForkAE is a NIST lightweight cryptography candidate that uses the forkcipher primitive in two modes of operation – SAEF and PAEF – optimized for authenticated encryption of the shortest messages. SAEF is a sequential and *online* AEAD that minimizes the memory footprint compared to its alternative parallel mode PAEF, catering to the most constrained devices. SAEF was proven AE secure against nonce-respecting adversaries.

Due to their more acute and direct exposure to device misuse and mishandling, in most use cases of lightweight cryptography, nonce reuse presents a very realistic attack vector. Furthermore, many lightweight applications mandate security for their online AEAD schemes against block-wise adversaries. Surprisingly, a very few NIST lightweight AEAD candidates come with provable guarantees against these security threats.

In this work we investigate the provable security guarantees of SAEF when nonces are repeated under a refined version of the notion of online authenticated encryption OAE given by Fleischmann et al. in 2012. Using the coefficient H technique we show that, with no modifications, SAEF is OAE secure up to the birthday security bound, i.e., up to $2^{n/2}$ processed blocks of data, where n is the block size of the forkcipher. The implications of our work is that SAEF is safe to use in a block-wise fashion, and that if nonces get repeated, this has no impact on ciphertext integrity and confidentiality only degrades by a limited extent up to repetitions of common message prefixes.

Keywords: Authenticated encryption, forkcipher, lightweight cryptography, short messages, online, provable security, nonce misuse.

1 Introduction

An authenticated encryption (AE) algorithm aims to achieve message confidentiality and integrity (authentication). The majority of AE schemes nowadays process two additional inputs: associated data AD and a nonce. The associated data is a piece of information, such as a network packet header, that does not require the provision of confidentiality but it does require authentication. The nonce input to an AE(AD) scheme is also practically motivated. A nonce is a unique value that enables the simplification of randomized or stateful AE algorithm to a deterministic one. This removes the need for random values generation or storing a state across distinct encryptions. The formalisation of nonce-based authenticated encryption was introduced in 2002 by Rogaway [25].

Two of the most prominent AEAD schemes research, development and standardization efforts in recent years have been the CAESAR competition (2014–2018) [8] and the ongoing NIST lightweight cryptography standardization process (2018–). The CAESAR competition produced

---

⋆ This is an updated version of the original work appeared in SAC 2020 [2].

a portfolio of algorithms for recommended use in three categories: 1. Lightweight (resource constrained environments): Ascon [10] and ACORN [29]; 2. High-performance: AEGIS-128 [31] and OCB [20]; 3. Defense in depth (stronger security guarantees): COLM [1], Deoxys II [19], and MORUS [30]. The CAESAR winners come with various advantages over the present standards GCM (NIST SP 800-38D) and CCM (IEEE 802.11i, IPsec ESP and IKEv2) and are expected to be adopted by new applications and standards.

The main focus of the defense in depth CAESAR category is nonce misuse resistance (NMR), a security target motivated by attacks which exploit implementation flaws leading to a repeated use of the same nonce by an application. More precisely, authenticity preservation despite nonce misuse is stated as *critical* and a limited privacy damage from nonce misuse as desirable. Nonce-misuse resistant AE (MRAE) was introduced by Rogaway and Shrimpton [27]. Later works dealt with online nonce misuse-resistant AE (OAE) [12] and online AE(OAE2) [16]. Nonce misuse purportedly presents a greater risk for small devices, such as IoT, where nonces could repeat due to various memory or space constraints, or remote usage. Maintaining the correct use of nonces is also especially challenging in distributed systems where nodes and connections can fail at any time. Recent attacks have illustrated the severity of nonce misuse in practice. In 2016 Böck et al. [9] investigated the nonce misuse security of the AES-GCM [11] AE mode and managed to completely break the authenticity of those connections where servers repeated the nonce. Their Internet-wide survey identified 184 such HTTPS servers. They showed how this vulnerability can be utilized to inject seemingly valid content into encrypted sessions. The next year Vanhoef and Piesens [28] introduced the key reinstallation attack which forces nonce repetitions and breaks the WPA2 wireless protocol.

The MRAE [26] and the RAE [15] security notions capture the "best possible" security of nonce based AEAD (the latter for an extended syntax) in face of nonce repetitions but unfortunately can not be satisfied by any online AEAD scheme. These are AE schemes which parse the plaintext as a sequence of smaller, usually fixed-size blocks during encryption, and produce the ciphertext as a sequence of such blocks, such that i^{th} ciphertext block can be immediately computed after having seen the first i plaintext blocks. Importantly, online encryption is better fitted for lightweight applications, where it is often critical to compute the ciphertext blocks on the fly with constant latency (e.g. streaming), or where a constant memory implementation is required. Such constraints can be found most recently in the consumer grade IoT applications, which come with stringent cost constraints and often get inadequate security as a consequence [21,22], and which would greatly benefit from a lightweight AEAD scheme with robust security.

In 2012 Fleischmann et al. [12] proposed a weaker NMR security notion called OAE for online schemes that retains the same level of integrity and a well-defined, albeit lower level of confidentiality in face of nonce misuse. Hoang et al. [16] pointed out inconsistencies in this definition, reformulated it, demonstrated that in certain situations the confidentiality afforded by OAE secure schemes is not sufficient, and critiqued several aspects of the OAE definition. Addressing the latter, Hoang et al. proposed an alternative to OAE dubbed OAE2 which is a stronger security with more versatile syntax. However, the change in syntax also means that the notion is inapplicable to designs adhering to the most common nonce-based AEAD syntax, and as shown by Hoang et al., the possible exploits of the decay in confidentiality are unavoidable in any kind of online schemes, OAE2-secure ones included. Despite this controversy, OAE can afford pragmatic protection against nonce-reuse with a sufficiently large block size and an appropriate use by the higher-level protocol/security primitive/etc.

In 2016 Endignoux and Vizár [14] showed that the notion of OAE is also meaningful in the context of block-wise adaptive attacks. The authors proved that OAE is equivalent with an adapted version of the blockwise-adaptive AE notions from [13]. The equivalence result shows that OAE-secure schemes are safe to use in settings where block-wise adaptive attackers exist. The main observation in [13] was that if an application, such as a smart card, outputs a ciphertext block each time it is fed a plaintext block, then a potential attacker gets more power by allowing him to adaptively construct the queries block-by-block. Such attacks pose a genuine risk in the lightweight setting where small devices are for example not equipped with sufficient memory and work block-wise. The OAE notion has been adopted by several AEAD designs, amongst which the COLM CAESAR winner in the defense in depth category (satisfying the authenticity and limited privacy damage against nonce misuse).

Surprisingly, among the 32 second round candidates, only a handful come with a provable form of nonce misuse security. The SP1 mode in the Spook [7] submission achieves the nonce misuse resilience notion proposed by Ashur et al. [4]. Tiny JAMBU [17] provides authentication security when nonce is misused by adjusting the associated data to also play the role of nonce. When the nonce is repeated but AD is different, the security of the cipher would not be affected by the repeated nonce, yet when both nonce and AD repeat, the scheme does not offer strong security guarantees in the sense of OAE. Romulus-M [18] is an MRAE [26] mode which is secure against misuse (repetition) of nonces in encryption queries but prevents the scheme from being implemented in an online fashion.

Our Contributions. In this work we investigate the OAE security of the ForkAE [3] second round NIST candidate which is optimized to be efficient for short messages. We focus on ForkAE and more precisely on its SAEF mode for three reasons. First, SAEF comes with existing provable security guarantees in the nonce-respecting setting, hence its (in)security in presence of nonce misuse is an interesting open question. Secondly, SAEF is an online scheme and the OAE security as refined by Hoang et al. [16] is a natural target for the analysis of its nonce-reuse security. Finally, an analysis of SAEF sheds more light on the potential of its relatively novel and less researched underlying forkcipher primitive [3] Our results show that the SAEF mode of the ForkAE NIST second round submission is provably OAE secure without the need of applying any design modifications.

The corrected variant of the OAE definition by Hoang et al. [16] (labeled as OAE1 in the publication) was defined to handle plaintext whose length is a multiple of the underlying blocksize n. Hence we extended the formalism to deal with messages that are not n-bits aligned and target the resulting notion in our analysis. Additionally, we opt to use two separate requirements for confidentiality and authenticity of OAE schemes because this allows us to take slightly different approach for analysis for each property in favor of brevity and simplicity of the proofs. We use Coefficient H technique [23] as the main vehicle for the analysis and prove that SAEF is OAE-secure up to $2^{n/2}$ blocks of processed data in total, where n is the block size of the underling forkcipher.

2   Preliminaries

Strings. All strings are binary strings. The set of all strings of length n (for a positive integer n) is denoted $\{0,1\}^n$ and the set of all strings of all possible lengths is denoted $\{0,1\}^*$. 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 $\{0,1\}^n$. We

denote by $\mathrm{Func}(m, n)$ the set of all functions with domain $\{0,1\}^m$ and range $\{0,1\}^n$, and we let $\mathrm{Inj}(m, n) \subset \mathrm{Func}(m, n)$ denote the set of all injective functions with the same signature.

For a string X of ℓ bits, we let $X[i]$ denote the ith bit of X for $i = 0, \ldots, \ell-1$ (starting from the left) and $X[i \ldots j] = X[i] \| X[i+1] \| \ldots \| X[j]$ for $0 \le i < j < \ell$. 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 any $0 \le \chi \le |X|$. Given a (possibly implicit) positive integer n and an $X \in \{0,1\}^*$, we let $X \| 10^*$ denote $X \| 10^{n-(|X| \bmod n)-1}$ for simplicity. Given an implicit block length n, we let $\mathsf{pad10}(X) = X \| 10^*$ return X if $|X| \equiv 0 \pmod{n}$ and $X \| 10^*$ otherwise.

String partitioning. For the rest of the section, we fix an arbitrary integer n and call it the block size. Given a string X, we let $X_1, \ldots, X_x, X_* \xleftarrow{n} X$ denote partitioning X into n-bit blocks, such that $|X_i| = n$ for $i = 1, \ldots, x$, $0 \le |X_*| \le n$ and $X = X_1 \| \ldots \| X_x \| X_*$, so $x = \max(0, \lceil X/n \rceil - 1)$. We let $|X|_n = \lceil X/n \rceil$. We let $(M', M_*) = \mathsf{msplit}_n(M)$ (as in message split) denote a splitting of a string $M \in \{0,1\}^*$ into two parts $M' \| M_* = M$, such that $|M_*| \equiv |M| \pmod{n}$ and $0 \le |M_*| \le n$, where $|M_*| = 0$ if and only if $|M| = 0$. We let $(C', C_*, T) = \mathsf{csplit}_n(C)$ (as in ciphertext split) denote splitting a string C of at least n bits into three parts $C' \| C_* \| T = C$, such that $|C_*| = n$, $|T| \equiv |C| \pmod{n}$, and $0 \le |T| \le n$, where $|T| = 0$ if and only if $|C| = n$. Finally, we let $C'_1, \ldots, C'_m, C_*, T \leftarrow \mathsf{csplit\text{-}b}_n(C)$ (as in csplit to blocks) denote the result of $\mathsf{csplit}_n(C)$ followed by partitioning of C' into $|C'|_n$ blocks of n bits, such that $C' = C'_1 \| \ldots \| C'_m$.

Blocks. We let $\mathsf{B}_n = \{0,1\}^n$ denote the set of all n-bit strings (or else n-bit blocks), and define $\mathsf{B}_n^* = \{\varepsilon\} \cup \bigcup_{i=1}^{\infty} \mathsf{B}_n^i$ with ε denoting the empty string of length 0. We call a string $X \in \mathsf{B}_n^*$ "n-aligned". For an n-aligned string X, X_i denotes its ith n-bit block. For two distinct n-aligned strings $X, Y \in \mathsf{B}_n^*$ such that $|X|_n \le |Y|_n$ w.l.o.g, we let $\mathsf{llcp}_n(X, Y) = \max\{1 \le i \le |X|_n | X_j = Y_j \text{ for } 1 \le j \le i\}$ denote the length of the longest common prefix of X and Y in n-bit blocks.

Miscellaneous. The symbol $\bot$ denotes an error signal, or an undefined value. We denote by $X \leftarrow_{\$} \mathcal{X}$ sampling an element X from a finite set $\mathcal{X}$ following the uniform distribution. We let $(n)_q$ denote the falling factorial $n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot (n-q+1)$ with $(n)_0 = 1$. For a predicate $\mathsf{P}(x)$, we equate $\mathsf{P}(x) = \text{true}$ with $\mathsf{P}(x) = 1$ and $\mathsf{P}(x) = \text{false}$ with $\mathsf{P}(x) = 0$. We use lexicographic comparison of tuples of integers; i.e. $(i', j') < (i, j)$ iff $i' < i$ or $i' = i$ and $j' < j$.

2.1   Coefficient H Technique

The coefficient H technique is a simple but powerful proof technique provided by Patarin in [23]. The coefficient H technique is used to prove indistinguishability of a construction from an idealized object in face of an information-theoretic adversary using the concept of "transcripts". A transcript here is defined as complete record of the interaction of an adversary $\mathcal{A}$ with its oracles in the indistinguishability experiment. To exemplify, if (M_i, C_i) denotes the input and output of the i-th query of $\mathcal{A}$ to its oracle over q queries then the corresponding transcript (denote it τ) is $\tau = \langle (M_1, C_1), \ldots, (M_q, C_q) \rangle$. The task given to $\mathcal{A}$ is to distinguish the real world $\mathcal{O}_{\mathsf{real}}$ from the ideal world $\mathcal{O}_{\mathsf{ideal}}$. Let us denote the distribution of transcripts in the real and the ideal world by Θ_{real} and Θ_{ideal}, respectively. We say a transcript τ is *attainable* if the probability of achieving τ in the ideal world is non-zero. We also assume w.l.o.g. that $\mathcal{A}$ does not make duplicate or prohibited queries. The fundamental Lemma of coefficient H technique can now be stated.

Lemma 1 (Fundamental Lemma of the coefficient H Technique [23]). *Consider 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 $\frac{\Pr[\Theta_{real} = \tau]}{\Pr[\Theta_{ideal} = \tau]} \geq 1 - \epsilon_1$, and $\Pr[\Theta_{ideal} \in \mathcal{T}_{bad}] \leq \epsilon_2$. Then, for all adversaries $\mathcal{A}$, it holds that $|\Pr[\mathcal{A}^{\mathcal{O}_{real}}] - \Pr[\mathcal{A}^{\mathcal{O}_{ideal}}]| \leq \epsilon_1 + \epsilon_2$.*

2.2 Syntax of AEAD

Our targeted mode SAEF follows the AEAD syntax by Rogaway [25]. A nonce-based AEAD scheme is a triplet $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. The key space $\mathcal{K}$ is a finite set. The deterministic encryption algorithm $\mathcal{E} : \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{M} \rightarrow \mathcal{C}$ 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, AD and message domains are all subsets of $\{0,1\}^*$. The deterministic decryption algorithm $\mathcal{D} : \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{C} \rightarrow \mathcal{M} \cup \{\perp\}$ takes a tuple (K, N, A, C) and either returns a message $M \in \mathcal{M}$, or a distinguished symbol $\perp$ 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 $K, N, A, M \in \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{M}$ 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 $K, N, A, M \in \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{M}$ we have $M = \mathcal{D}(K, N, A, \mathcal{E}(K, N, A, M))$. We let $\mathcal{E}_K(N, A, M) = \mathcal{E}(K, N, A, M)$ and $\mathcal{D}_K(N, A, C) = \mathcal{D}(K, N, A, C)$.

2.3 OAE Security

Our targeted notion of security is the definition of online AE (OAE) by Fleischmann et al. [12]. We use the variant of the notion described by Hoang et al. [16], extended as necessary to deal with messages that are not n-aligned. We opt for the two-requirement flavor of the notion, separating confidentiality and authenticity.

OAE Confidentiality. An online permutation [6] is a function $\pi : \mathsf{B}_n^* \rightarrow \mathsf{B}_n^*$ such that (i) π is a length preserving permutation; i.e., for any integer $m \geq 0$, the function π restricted to mn-bit inputs $\pi(K, T, \cdot) : \mathsf{B}_n^m \rightarrow \mathsf{B}_n^m$ is a permutation (with a slight notation abuse); (ii) π preserves the length of blockwise prefix; i.e., for each $M, M' \in \mathsf{B}_n^*$, we have that $\mathsf{llcp}_n(M, M') = \mathsf{llcp}_n(\pi(M), \pi(M'))$.

We denote the set of all such permutations $\mathrm{OPerm}(n)$. We can define the distribution of a "random" online permutation[4] that is useful for experiments with a finite number of queries. We observe that each $\pi \in \mathrm{OPerm}(n)$ can be equivalently defined as a collection $(\pi_M)_{M \in \mathsf{B}_n^*}$, such that for any $M_1 \| M_2 \| \dots \| M_r \in \mathsf{B}_n^r$ we define $\pi(M_1 \| M_2 \| \dots \| M_r)$ as $\pi_\varepsilon(M_1) \| \pi_{M_1}(M_2) \| \dots \| \pi_{M_1 \| \dots \| M_{r-1}}(M_r)$, and that there is in fact a bijection between $\mathrm{OPerm}(n)$ and the set of all such permutation collections [6]. The expression $\pi \leftarrow_\$ \mathrm{OPerm}(n)$ should thus be understood as lazy sampling of those permutations in the collection $(\pi_M)_{M \in \mathsf{B}_n^*}$ that are necessary to process adversarial queries.

We define OAE confidentiality of an AEAD scheme Π with two games, **oprpf-real**$_\Pi$ and **oprpf-ideal**$_\Pi$. In both games $\mathcal{A}$ can make arbitrary chosen plaintext queries to a black box encryption oracle, in particular $\mathcal{A}$ is allowed to repeat the nonces. In the game **oprpf-real**$_\Pi$,

---

[4] $\mathrm{OPerm}(n)$ is a countably infinite set, so the notion of a uniform online permutation is not defined.

the encryption oracle faithfully implements the encryption algorithm of Π using a randomly sampled secret key. In the game **oprpf-ideal**$_\Pi$, upon an encryption query N, A, M the oracle returns $\pi_{N,A}(M_L)\|f_{N,A,M}$ where $M_L \leftarrow \mathsf{left}_{\lfloor |M/n| \rfloor}(M)$ is the longest n-aligned prefix of M, $\pi_{N,A} \leftarrow_\$ \mathrm{OPerm}(n)$ is a random online permutation for each $N, A \in \mathcal{N} \times \mathcal{A}$, and $f_{N,A,M} \leftarrow_\$ \{0,1\}^{n+(|M| \bmod n)}$ is a random string for each $N, A, M \in \mathcal{N} \times \mathcal{A} \times \mathcal{M}$ (with a length corresponding to the n-bit authentication tag and no more than $n-1$ bit suffix of M remaining after the maximal n-aligned prefix M_L). The **oprpf** advantage of an adversary $\mathcal{A}$ against $\Pi = \mathrm{SAEF}[\mathsf{F}, \nu]$ can now be defined as $\mathbf{Adv}_\Pi^{\mathbf{oprpf}}(\mathcal{A}) = \Pr[\mathcal{A}^{\mathbf{oprpf\text{-}real}_\Pi}] - \Pr[\mathcal{A}^{\mathbf{oprpf\text{-}ideal}_\Pi}]$.

OAE Authenticity. The OAE authenticity notion coincides with the definition of ciphertext integrity where the adversary is allowed to repeat nonces, as first defined by Rogaway and Shrimpton [26]. A nonce-misusing chosen ciphertext attack of an adversary $\mathcal{A}$ against the integrity of a nonce-based AE scheme Π is modeled through the security game **auth**. The adversary is given access to a pair of blackbox oracles. It can make arbitrary chosen plaintext queries to the encryption oracle, including queries with repeated nonces. $\mathcal{A}$ can further make arbitrary chosen ciphertext queries to the decryption oracle with the goal of finding a forgery: a tuple that decrypts correctly but is not trivially known to be correct from the encryption queries. We define the advantage of $\mathcal{A}$ in breaking the authenticity of Π as $\mathbf{Adv}_\Pi^{\mathbf{auth}}(\mathcal{A}) = \Pr[\mathcal{A}^{\mathbf{auth}_\Pi} \text{forges}]$ where "$\mathcal{A}$ forges" denotes a decryption query that returns a value $\neq \bot$. We assume w.l.o.g. that $\mathcal{A}$ does not make duplicate queries.

2.4   Forkcipher

We follow the formalism by Andreeva et al. [24]. Informally, a forkcipher F is a tweakable symmetric-key primitive which maps a secret key K, a tweak T and an input block M of n bits to *two* ciphertext blocks C_0 and C_1, such that C_0 and C_1 are each an (independent) permutation of M.

Syntax. Formally, a forkcipher is a pair of deterministic algorithms, the encryption algorithm: $\mathsf{F} : \{0,1\}^k \times \mathcal{T} \times \{0,1\}^n \times \{0, 1, \mathsf{b}\} \to \{0,1\}^n \cup \{0,1\}^n \times \{0,1\}^n$ and the inversion algorithm: $\mathsf{F}^{-1}\{0,1\}^k \times \mathcal{T} \times \{0,1\}^n \times \{0,1\} \times \{\mathsf{i}, \mathsf{o}, \mathsf{b}\} \to \{0,1\}^n \cup \{0,1\}^n \times \{0,1\}^n$. 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 write $\mathsf{F}(K, \mathsf{T}, M, s) = \mathsf{F}_K(\mathsf{T}, M, s) = \mathsf{F}_K^\mathsf{T}(M, s) = \mathsf{F}_K^{\mathsf{T},s}(M)$ interchangeably. The inversion algorithm takes a key K, a tweak T, a ciphertext block C (left/right half of output block), an indicator b of whether this 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 write $\mathsf{F}^{-1}(K, \mathsf{T}, M, b, s) = \mathsf{F}^{-1}{}_K(\mathsf{T}, M, b, s) = \mathsf{F}^{-1}{}_K^\mathsf{T}(M, b, s) = \mathsf{F}^{-1}{}_K^{\mathsf{T},b,s}(M)$ interchangeably. We call k, n and $\mathcal{T}$ the keysize, blocksize and tweak space of F, respectively.

A tweakable forkcipher F is correct if for each pair of key and tweak, the forkcipher applies two independent permutations to the input to produce the two output blocks. Formally, for each $K \in \{0,1\}^k, \mathsf{T} \in \mathcal{T}, M \in \{0,1\}^n$ and $\beta \in \{0,1\}$ the following conditions must be met: (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) $\big(\mathsf{F}^{-1}(K, \mathsf{T}, C, \beta, \mathsf{i}), \mathsf{F}^{-1}(K, \mathsf{T}, C, \beta, \mathsf{o})\big) = \mathsf{F}^{-1}(K, \mathsf{T}, C, \beta, \mathsf{b})$. In the rest of the paper, we assume that $\mathcal{T} = \{0,1\}^t$ for some positive t.

Security. The security of a correct forkcipher F is defined through indistinguishability of the games **prtfp-real$_\mathsf{F}$** (implementing F algorithms faithfully) and **prtfp-ideal$_\mathsf{F}$** which replaces F by two tweakable random permutations $\pi_{\mathsf{T},0}, \pi_{\mathsf{T},1} \leftarrow_\$ \mathrm{Perm}(n)$ for $\mathsf{T} \in \mathcal{T}$ in a natural way, in a chosen ciphertext attack. We define the advantage of $\mathcal{A}$ as $\mathbf{Adv}_\mathsf{F}^{\mathrm{prtfp}}(\mathcal{A}) = \Pr[\mathcal{A}^{\mathbf{prtfp\text{-}real}_\mathsf{F}} \Rightarrow 1] - \Pr[\mathcal{A}^{\mathbf{prtfp\text{-}ideal}_\mathsf{F}} \Rightarrow 1]$ with the games defined in.

3 SAEF and its OAE Security

SAEF (short for Sequential AE from a Forkcipher) is a nonce-based AEAD scheme. It takes as a parameter a tweakable forkcipher F (as defined in Section 2.4) with $\mathcal{T} = \{0,1\}^t$ for a positive $t \leq n$. SAEF[F] $= (\mathcal{K}, \mathcal{E}, \mathcal{D})$ has a key space $\mathcal{K} = \{0,1\}^k$, nonce space $\mathcal{N} = \{0,1\}^{t-4}$, and the AD and message spaces are both $\{0,1\}^*$. The ciphertext expansion of SAEF[F] is n bits. The encryption and decryption algorithms are given in Figure 2 and the encryption algorithm is depicted in Figure 1.

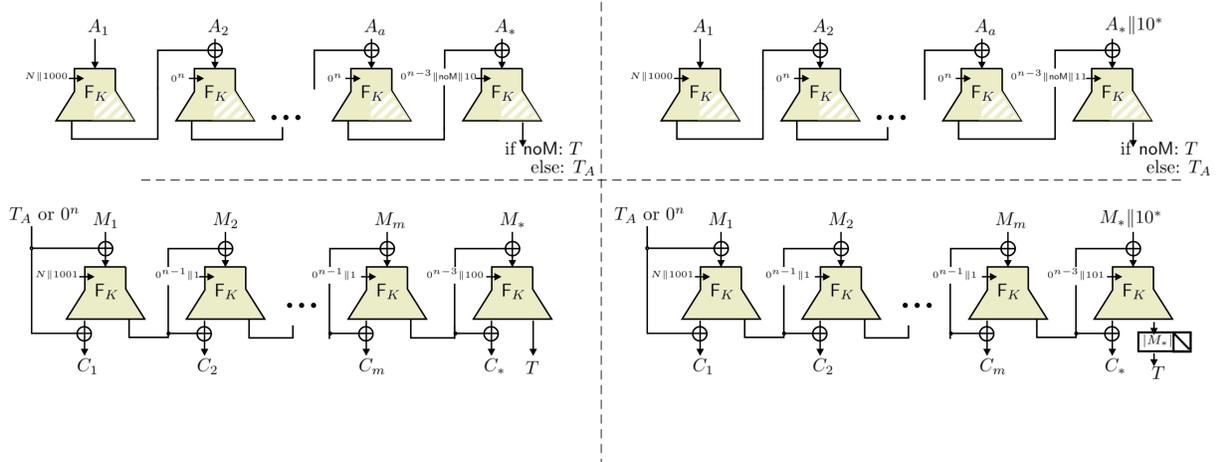Fig. 1: The encryption algorithm of SAEF[F] mode. The bit $\mathsf{noM} = 1$ iff $|M| = 0$. The picture illustrates the processing of AD when length of AD is a multiple of n (**top left**) and when the length of AD is not a multiple of n (**top right**), and the processing of the message when length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

An encryption query is processed in blocks of n bits, first AD and then the message, using a single call to the forkcipher per block. The calls are tweaked by composing: (1) the nonce followed by a 1-bit *in the initial* F *call of the query*, and the string 0^{t-3} otherwise, (2) three-bit flag f. The flag f is used to ensure proper domain separation for different "types" of blocks in the encryption algorithm. The values $f \in \{000, 010, 011, 110, 111, 001, 100, 101\}$ are used, respectively, 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.

One output block of every F call is used as a chaining value, masking either the input (in AD processing) or both the input and the output (in message processing) of the following F call. The very first F call in each query is unmasked (but has the nonce in the tweak). The tag is the,

```
 1: function  E(K, N, A, M)                    1: function  D(K, N, A, C)
 2:     A_1,...,A_a,A_* ←ⁿ A                    2:     A_1,...,A_a,A_* ←ⁿ A
 3:     M_1,...,M_m,M_* ←ⁿ M                    3:     C_1,...,C_m,C_*,T ← csplit-b_n C
 4:     noM ← 0                                 4:     noM ← 0
 5:     if |M| = 0 then noM ← 1                 5:     if |C| = n then noM ← 1
 6:     Δ ← 0ⁿ; T ← N‖0^{t-4}‖1                 6:     Δ ← 0ⁿ; T ← N‖0^{t-4}‖1
 7:     for i ← 1 to a do                       7:     for i ← 1 to a do
 8:         T ← T‖000                           8:         T ← T‖000
 9:         Δ ← F_K^{T,0}(A_i ⊕ Δ)              9:         Δ ← F_K^{T,0}(A_i ⊕ Δ)
10:         T ← 0^{t-3}                        10:         T ← 0^{t-3}
11:     end for                                11:     end for
12:     if |A_*| = n then                      12:     if |A_*| = n then
13:         T ← T‖noM‖10                        13:         T ← T‖noM‖10
14:         Δ ← F_K^{T,0}(A_* ⊕ Δ)             14:         Δ ← F_K^{T,0}(A_* ⊕ Δ)
15:         T ← 0^{t-3}                        15:         T ← 0^{t-3}
16:     else if |A_*| > 0 or |M| = 0 then      16:     else if |A_*| > 0 or |T| = 0 then
17:         T ← T‖noM‖11                        17:         T ← T‖noM‖11
18:         Δ ← F_K^{T,0}((A_*‖10*) ⊕ Δ)       18:         Δ ← F_K^{T,0}((A_*‖10*) ⊕ Δ)
19:         T ← 0^{t-3}                        19:         T ← 0^{t-3}
20:     end if        ▷ Do nothing if A=ε, M≠ε 20:     end if        ▷ Do nothing if A=ε, M≠ε
21:     for i ← 1 to m do                      21:     for i ← 1 to m do
22:         T ← T‖001                          22:         T ← T‖001
23:         C_i,Δ ← F_K^{T,b}(M_i ⊕ Δ)⊕(Δ,0ⁿ) 23:         M_i,Δ ← F^{-1}_K^{T,0,b}(C_i ⊕ Δ)⊕(Δ,0ⁿ)
24:         T ← 0^{t-3}                        24:         T ← 0^{t-3}
25:     end for                                25:     end for
26:     if |M_*| = n then                      26:     if |T| = n then
27:         T ← T‖100                          27:         T ← T‖100
28:     else if |M_*| > 0 then                 28:     else if |T| > 0 then
29:         T ← T‖101                          29:         T ← T‖101
30:     else                                   30:     else
31:         return Δ                           31:         if C_* ≠ Δ then return ⊥
32:     end if                                 32:         return ε
33:     C_*,T ← F_K^{T,b}(pad10(M_*)⊕Δ)⊕(Δ‖0ⁿ) 33:     end if
34:     return C_1‖...‖C_m‖C_*‖left_{|M_*|}(T) 34:     M_*,T' ← F^{-1}_K^{T,0,b}(C_* ⊕ Δ)⊕(Δ,0ⁿ)
35: end function                               35:     T' ← left_{|T|}(T'); P ← right_{n-|T|}(M_*)
                                               36:     if T' ≠ T return ⊥
                                               37:     if P ≠ left_{n-|T|}(10^{n-1}) return ⊥
                                               38:     return M_1‖...‖M_m‖left_{|T|}(M_*)
                                               39: end function
```
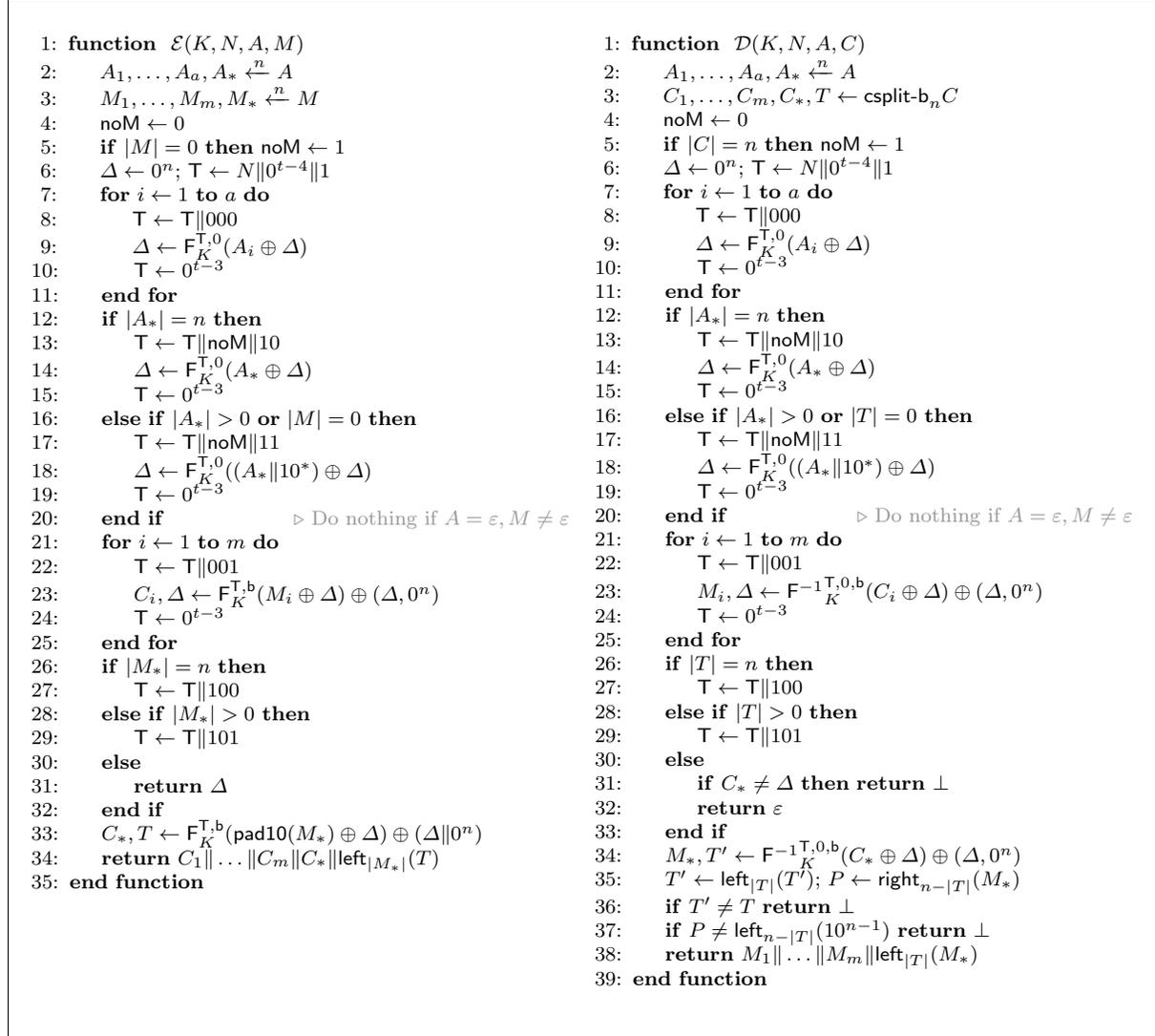
Fig. 2: The SAEF[F] AEAD scheme.

possibly truncated, last "right" output of F produced in the query (in case of truncation message padding is used for partial integrity check). In a decryption query, the processing is similar to the encryption, with the plaintext blocks and the chaining values in the message processing part being computed with the inverse F algorithm.

3.1 Security of SAEF

Andreeva et al. proved that if nonce-uniqueness is guaranteed, SAEF achieves the standard AEAD confidentiality and integrity up to the birthday bound. There have been no investigations into the security of SAEF under nonce misuse (i.e., if nonces accidentally repeat). We state the formal claim about confidentiality and integrity of SAEF under nonce-misuse in Theorem 1.

Theorem 1. *Let F be a tweakable forkcipher with $\mathcal{T} = \{0,1\}^t$. Then for any nonce-misuse adversary $\mathcal{A}$ who makes at most $q_e \leq 2^{n-1}$ encryption queries, at most q_v decryption queries*

such that the total number of forkcipher calls induced by all the queries is at most σ, we have

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

$$\mathbf{Adv}^{\mathbf{auth}}_{\mathrm{SAEF[F]}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathbf{prtfp}}_{\mathsf{F}}(\mathcal{C}) + \frac{\sigma^2 + 4 \cdot q_v}{2^n}$$

for some adversaries $\mathcal{B}$ and $\mathcal{C}$, each making at most 2σ queries, and running in 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.

Main ingredients. The crux of the proof lies in adapting the analysis approach to the properties of SAEF arising from its sequential structure. These properties are best illustrated by an example. In two queries using the same nonce N, the same associated data $A_1 \| A_2$ of two blocks and two messages $M_1^1 \| M_2^1 \| M_3^1$ and $M_1^2 \| M_2^2 \| M_3^2$ of three blocks each and *differing in the first message block*, following the encryption algorithm in Figure 2, the values of the F tweak string and of Δ mask used to process each block of A would be identical between the two queries, as would be the tweak (equal to $0^{n-1}1$) and the Δ mask used to process the first message block M_1^1 and M_1^2 respectively. However, this equality of tweaks and Δ masks together with the non-equality $M_1^1 \neq M_1^2$ imply that the F input blocks $\Delta \oplus M_1^1$ and $\Delta \oplus M_1^2$ will necessarily differ, which will randomize the Δ masks used to process the next of each message.

Similarly, in another case with executing two same queries but with extending the AD used in the first query to $A_1 \| A_2 \| 110^{n-2}$ and for the second query to $A_1 \| A_2 \| 1$, the processing will again be identical for A_1 and A_2 and for the third block, the Δ masks and the F inputs $\Delta \oplus 110^{n-2} = \Delta \oplus \mathsf{pad10}(1)$ will be identical but the tweaks will differ. What emerges is that the internal variables of SAEF's encryption algorithm preserve a certain "common prefix length" between the queries, and get randomized just past it. This observation is at the center of our proofs. Giving a formal definition of a common prefix between AE queries is also concise but is not straightforward, and requires a different representation of queries, which is the second main ingredient of our proofs.

3.2 Integrity

We switch to an alternative definition of AEAD integrity through indistinguishability, which is equivalent with the notion introduced in Section 3. We define two games, $\mathbf{auth\text{-}real}_\Pi$ and $\mathbf{auth\text{-}ideal}_\Pi$. In both games $\mathcal{A}$ is given access to an encryption and a decryption oracle. In the game $\mathbf{auth\text{-}real}_\Pi$, both oracles faithfully implement the respective algorithms of SAEF using the same randomly sampled secret key, except that the decryption oracle returns $\top$ in case of a successful forgery, and $\bot$ otherwise. In the game $\mathbf{auth\text{-}ideal}_\Pi$, the encryption oracle is the same as in $\mathbf{auth\text{-}real}_\Pi$ but the decryption oracle always returns $\bot$. We claim that $\mathbf{Adv}^{\mathbf{auth}}_\Pi(\mathcal{A}) = \Pr[\mathcal{A}^{\mathbf{auth\text{-}real}_\Pi}] - \Pr[\mathcal{A}^{\mathbf{auth\text{-}ideal}_\Pi}]$.

It is easy to see that the equality holds, by establishing inequalities in both directions. An adversary $\mathcal{A}$ playing the game $\mathbf{auth}_\Pi$ can be used to construct a distinguishing adversary $\mathcal{B}$ which forwards $\mathcal{A}$'s queries and outputs 1 iff $\mathcal{A}$ forges, achieving the same advantage as $\mathcal{A}$. In the other direction, we can construct an adversary $\mathcal{A}$ for game $\mathbf{auth}_\Pi$ from an indistinguishability adversary $\mathcal{B}$, which forwards $\mathcal{B}$'s queries and automatically wins if $\mathcal{B}$ produces a valid forgery. $\mathcal{A}$ achieves the same advantage as $\mathcal{B}$, because if no forgery occurs, the games $\mathbf{auth\text{-}real}_\Pi$ and $\mathbf{auth\text{-}ideal}_\Pi$ are indistinguishable.

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

The adversary is thus left with the goal of distinguishing between the games $\mathbf{auth\text{-}real}_{\mathrm{SAEF}[(\pi_0, \pi_1)]}$ (called the "real-int world") and $\mathbf{auth\text{-}ideal}_{\mathrm{SAEF}[(\pi_0, \pi_1)]}$ (called the "ideal-int world"). Hence, we want to bound $\mathbf{Adv}_{\mathrm{SAEF}[(\pi_0, \pi_1)]}^{\mathbf{auth}}(\mathcal{A}) = \Pr[\mathcal{A}^{\mathbf{auth\text{-}real}_{\mathrm{SAEF}[(\pi_0, \pi_1)]}}] - \Pr[\mathcal{A}^{\mathbf{auth\text{-}ideal}_{\mathrm{SAEF}[(\pi_0, \pi_1)]}}]$.

Transcripts. Following the coefficients H technique [23], we characterize the interactions of $\mathcal{A}$ with its oracles in a *transcript*:

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

In the ith query to the encryption oracle (N^i, A^i, M^i), returning the ciphertext C^i, SAEF internally processes A^i, M^i and C^i in blocks $A_1^i, \ldots, A_{a^i}^i, A_*^i$, and $M_1^i, \ldots, M_{m^i}^i, M_*^i$, and $C_1^i, \ldots, C_{m^i}^i, C_*^i, T^i$ respectively (defined according to the encryption algorithm in Figure 2). We have a^i and m^i equal to, respectively, the length of A^i and M^i in n-bit. SAEF additionally uses whitening masks, denoted here $\Delta_1^i, \ldots, \Delta_{a^i+1}^i$ for the whitening masks used to process $A_1^i, \ldots, A_{a^i}^i, A_*^i$ respectively, and $\Delta_{a^i+2}^i, \ldots, \Delta_{a^i+m^i+2}^i$ for the whitening masks used to process the blocks $M_1^i, \ldots, M_{m^i}^i, M_*^i$ respectively.

Similarly, in the ith decryption query $(\bar{N}^i, \bar{A}^i, \bar{C}^i)$, returning $b^i \in \{\top, \bot\}$ SAEF internally processes $\bar{A}^i$ and $\bar{C}^i$ in blocks, denoted as $\bar{A}_1^i, \ldots, \bar{A}_{\bar{a}^i}^i, \bar{A}_*^i$ and $\bar{C}_1^i, \ldots, \bar{C}_{\bar{m}^i}^i, \bar{C}_*^i, \bar{T}^i$, where $\bar{a}^i$ and $\bar{m}^i$ are respectively equal to the length of $\bar{A}^i$ and the length of $\bar{C}^i$ in n-bit blocks (excluding the tag from the count). Additionally, SAEF internally computes the plaintext blocks $\bar{M}_1^i, \ldots, \bar{M}_{\bar{m}^i}^i, \bar{M}_*^i$ as well as $\bar{\Delta}_1^i, \ldots, \bar{\Delta}_{\bar{a}^i+1}^i$, the whitening masks used to process $\bar{A}_1^i, \ldots, \bar{A}_{\bar{a}^i}^i, \bar{A}_*^i$ respectively, and $\bar{\Delta}_{\bar{a}^i+2}^i, \ldots, \bar{\Delta}_{\bar{a}^i+\bar{m}^i+2}^i$, the whitening masks used to process the blocks $\bar{C}_1^i, \ldots, \bar{C}_{\bar{m}^i}^i, \bar{C}_*^i$ respectively.

Additional information. To simplify the proofs, we additionally provide the adversary with all the encryption masks Δ_j^i, all the decryption masks $\bar{\Delta}_j^i$ and internally computed plaintexts $\bar{M}_j^i$ 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 the oracles faithfully evaluating SAEF. In the ideal-int world, the encryption also evaluates SAEF, so Δ_j^i masks are defined, but the decryption oracle does not make any computations, hence $\bar{\Delta}_j^i$ and $\bar{M}_j^i$ are not defined. We therefore have to define their sampling, to be done *at the end of the experiment* (thus having no influence of adversarial queries).

For $j = 1$, we have $\bar{\Delta}_1^i = 0^n$ for $1 \leq i \leq q_v$. We sample each of the remaining $\bar{\Delta}_j^i$ mask uniformly, except when its value is trivially determined due to a "common prefix" with an encryption, or decryption query (defined shortly). Once the masks are sampled, we run the SAEF decryption algorithm using π_0 and these masks to compute $\bar{M}_j^i$. Clearly, this give away of additional information can only increase the adversary's advantage and thus can be considered here for upper bounding the above mentioned adversarial advantage.

Block-tuple representation. The ith encryption query can be equivalently represented as $(\mathsf{T}_j^i, \Delta_j^i, X_j^i, Y_j^i)_{j=1}^{\ell^i}, T^i$, such that $\ell^i = a^i + m^i + 2$ and each of the ℓ^i quadruples represents the processing done in j^{th} call to the forkcipher in the query, consisting of the string T_j^i used as fork-

cipher tweak, the current whitening mask Δ_j^i, the (possibly padded) associated data/plaintext block X_j^i and the empty/ciphertex block Y_j^i. In more detail:

- In the first block, we always have $\mathsf{T}_1^i = N\|1\|F$ for a flag $F \in \{0,1\}^3$ and $\Delta_1^i = 0^n$. For blocks with $j > 1$ we have $\mathsf{T}_j^i = 0^{t-3}\|F$ for an $F \in \{0,1\}^3$.
- If $|A| > 0$, for $1 \leq j \leq a^i$ we have $X_j^i = A_j^i$, $Y_j^i = \varepsilon$ and $F = 000$. For $j = a^i + 1$ we have $X_j^i = \mathsf{pad10}(A_*^i)$, $Y_j^i = \varepsilon$ and $F \in \{0,1\}^3$ as defined in Figure 2.
- If $|M| > 0$, for $a^i + 2 \leq j < \ell^i$ we have $X_j^i = M_j^i$, $Y_j^i = C_j^i$ and $F = 001$. For $j = \ell^i$ we have $X_j^i = \mathsf{pad10}(M_*^i)$, $Y_j^i = C_*^i$ and $F \in \{0,1\}^3$ as defined in Figure 2.
- If $A = M = \varepsilon$, we have $j = \ell^i = 1$, $X_j^i = \mathsf{pad10}(\varepsilon)$, $Y_j^i = \varepsilon$ and $F = 111$.

We call this the *block-tuple* representation.

We similarly define the block-tuple representation $(\bar{\mathsf{T}}_j^i, \bar{\Delta}_j^i, \bar{X}_j^i, \bar{Y}_j^i)_{j=1}^{\bar{\ell}^i}, \bar{T}^i, b^i$ for decryption queries with $\bar{\ell}^i = \bar{m}^i + \bar{a}^i + 2$. We further streamline the notation by re-indexing the decryption queries from $q_e + 1$ to $q_e + q_v$, and dispensing of the bar above the variables. The decryption queries are thus denoted as $(\mathsf{T}_j^i, \Delta_j^i, X_j^i, Y_j^i)_{j=1}^{\ell^i}, T^i, b^i$ for $q_e + 1 \leq i \leq q_e + q_v$.

Proposition 1. *The transformation of an extended transcript from the default to the block-tuple representation is injective.*

The proof of Proposition 1 is given in Appendix A.

Blockwise common prefix of queries This notation allows for the following natural definition of longest common blockwise prefix between two *queries*. We define the longest common prefix of the ith and i'^{th} query (in the block-tuple notation) with $\ell^i \leq \ell^{i'}$ w.l.o.g. as

$$\mathsf{llcp}_n(i, i') = \max\{1 \leq u \leq \ell^i | (\mathsf{T}_j^i, \Delta_j^i, X_j^i, Y_j^i) = (\mathsf{T}_j^{i'}, \Delta_j^{i'}, X_j^{i'}, Y_j^{i'}) \text{ for } 1 \leq j \leq u\}.$$

Note that this definition covers common blockwise prefix between two encryption queries (with $i, i' \leq q_e$), two decryption queries (with $i, i' > q_e$)) and between an encryption and a decryption query (with $1 \leq i' \leq q_e < i \leq q_e + q_v$ w.l.o.g.). Informally, $\mathsf{llcp}_n(i, i')$ counts how many internal chaining values (the Δ masks) are trivially equal between the ith and i'^{th} encryption query. For example, if the nonces N^i and $N^{i'}$ are distinct we have $\mathsf{llcp}_n(i, i') = 0$. If we have two queries with $N^i = N^{i'}$ and one having AD $A^{i'} = A^i\|M_1^i$ equal to the AD of the other appended with its first message block, we will still have $\mathsf{llcp}_n(i, i') = a^i + 1$, thanks to including the tweak string in the block tuples. We finally define the length of the longest common blockwise prefix of a query *with all previous queries* as $\mathsf{llcp}_n(i) = \max_{1 < i' < i} \mathsf{llcp}_n(i, i')$. Note that for a decryption query, all the encryption queries are always taken into account (due to our convention about query indexing).

Sampling of Δ masks. Using this notation, we now precisely define the sampling of Δ_j^i masks in decryption queries (i.e., for $q_e < i \leq q_e + q_v$) of the ideal-int world. In the ith decryption query, for $1 \leq j \leq \mathsf{llcp}_n(i) + 1$, 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 $\mathsf{llcp}_n(i) + 1 < j \leq \ell^i$, Δ_j^i is sampled uniformly at random. In other words, Δ masks that are trivially known to be equal to some previous mask due to the iterative nature of SAEF are simply set to that value, and the rest is sampled uniformly.

Extended transcripts. With the new notation in mind, the extended transcripts can be re-defined as

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

Note that q_e, q_v, a and m here are themselves random variables and thus can vary for distinct *attainable* transcripts. However, due to the assumption that the adversary can make at most σ block queries, we always have $\sum_{i=1}^{q_e+q_v}(a^i + m^i + 2) = \sigma$.

The following corollary shows that switching to block-tuple representation does not have an impact of the result of the analysis and in particular on the bounds derived using the block-tuple representation.

Corollary 1. *Due to Proposition 1, it is impossible for two distinct transcripts $\tau = \langle (N^i, A^i, M^i, C^i)_{i=1}^{q_e}, (\bar{N}^i, \bar{A}^i, \bar{C}^i, b^i)_{i=1}^{q_v} \rangle$ and $\tau' = \langle (N'^i, A'^i, M'^i, C'^i)_{i=1}^{q'_e}, (\bar{N}'^i, \bar{A}'^i, \bar{C}'^i, b'^i)_{i=1}^{q'_v} \rangle$ to have the same block-tuple representation.*

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

The proof relies on the fundamental lemma of the coefficient H technique defined as Lemma 1 above. We say an attainable transcript τ is bad if one of the following conditions occurs:

BadT_1 a.k.a. "Input Collision": There exists $(i', j') < (i, j)$ (the block indexed by (i', j') precedes (i, j)) such that $1 \leq i \leq q_e + q_v$, $\mathsf{llcp}_n(i) < j \leq \ell^i$ is not in the longest common prefix of the ith 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 $X_j^i \oplus \Delta_j^i = X_{j'}^{i'} \oplus \Delta_{j'}^{i'}$.

BadT_2 a.k.a. "Mask Collision": There exists $(i', j') < (i, j)$ such that $1 \leq i \leq q_e + q_v$, $\mathsf{llcp}_n(i) < j < \ell^i$ (not in the longest common prefix), and both the block calls have the same tweaks $\mathsf{T}_j^i = \mathsf{T}_{j'}^{i'}$ and different inputs $X_j^i \oplus \Delta_j^i \neq X_{j'}^{i'} \oplus \Delta_{j'}^{i'}$, however, the following masks $\Delta_{j+1}^i = \Delta_{j'+1}^{i'}$ collide. (Note that such a collision cannot occur in the real-int world where the masks are generated with permutation).

BadT_3 a.k.a. "Forgery": There exists $q_e + 1 \leq i \leq q_e + q_v$ such that for $j = \ell^i$ we have any of the following:

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

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

We let $\mathcal{T}_{\text{bad}}$ be the set of "bad" transcripts defined as the subset of attainable transcripts for which the transcript predicate $\mathsf{BadT}(\tau) = (\mathsf{BadT}_1(\tau) \vee \mathsf{BadT}_2(\tau) \vee \mathsf{BadT}_3(\tau))$ 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 as good transcripts).

Lemma 2. *For $\mathcal{T}_{bad}$ above and $q_e \leq 2^{n-1}$, we have $\Pr[\Theta_{idin} \in \mathcal{T}_{bad}] \leq \frac{\sigma^2}{2^n} + \frac{4 \cdot q_v}{2^n}$.*

Proof. $\mathbf{BadT}_1$**.** For any transcript in $\mathcal{T}_{\text{bad}}$ with BadT_1 set to 1, we know that there exists at least one pair of block indices $(i', j') < (i, j)$ such that $\mathsf{llcp}_n(i) < j \leq \ell^i$ and $\Delta_j^i \oplus \Delta_{j'}^{i'} = X_j^i \oplus X_{j'}^{i'}$.

Note that for all $i' < i$ and $j = j' = \mathsf{llcp}_n(i) + 1$, we have $\Delta_j^i = \Delta_{j'}^{i'}$ but $X_j^i \neq X_{j'}^{i'}$ and thus for such cases the probability that the above equality occurs is 0. On the other hand, for all $i' \leq i$ and $j' \neq j$ or $j \neq \mathsf{llcp}_n(i) + 1$, the two masks has marginal probability $1/2^n$ of being same 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 $\Pr[\mathsf{BadT}_1(\Theta_{idin}) = 1] \leq \frac{\sigma^2}{2} \cdot \max\{0, \frac{1}{2^n}\} = \frac{\sigma^2}{2^{n+1}}$.

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 $\text{llcp}_n(i) < j < \ell^i$ and $\Delta^i_{j+1} \oplus \Delta^{i'}_{j'+1} = 0$.

Note that from the definition of the predicate BadT_2 we have $j + 1 \neq \text{llcp}_n(i) + 1$. 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 $\Pr[\text{BadT}_2(\Theta_{idin}) = 1] \leq \frac{\sigma^2}{2^{n+1}}$.

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 for Θ_{idin} one of the following can happen:

For some $i' \leq q_e$, $j = \ell^i$ and $j' = \ell^{i'}$, we have $j = j' = \text{llcp}_n(i)$. Clearly, in such a case $\Delta^i_j = \Delta^{i'}_{j'}$, $X^i_j = X^{i'}_{j'}$ but $T^i \neq T^{i'}$. $T^{i'}$ being the correct tag for the given ciphertext, $T^i \neq T^{i'}$ cannot trigger BadT_3, yielding 0 probability.

For some $i' \leq q_e$, $j = \ell^i$ and $j' = \ell^{i'}$, we have $j = j' = \text{llcp}_n(i) + 1$. We have $\Delta^i_j = \Delta^{i'}_{j'}$ but $X^i_j \neq X^{i'}_{j'}$ and thus the probability of any of the two conditions of BadT_3 occurring for a given query is at most $4/2^n$ assuming $q_e \leq 2^{n-1}$. For the first condition, this holds as every tag there is produced with a tweak used at most once per query, corresponding to a probability $1/(2^n - q_e) \leq 2/2^n$. For the second condition, we upper bound the product of the probabilities of having the correct padding in the block X^i_j (at most $2^{|T^i|}/(2^n - q_e)$), and of having the correct truncated tag (at most $2^{n-|T^i|}/(2^n - q_e)$).

For all $i' \leq q_e$, we have $j > \text{llcp}_n(i, i') + 1$. We know that the Δ^i_j is not inherited from an encryption 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 $\text{BadT}_1(\Theta_{idin}) = 0$ w.l.o.g., yielding a probability of at most $2/2^n$.

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

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

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. $\text{llcp}_n(i, i') < j < \ell^i$ 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 decryption oracle i.e. $1 \leq i \leq q_v$, the transcript contains $b^i = \perp$ in the decryption result i.e. the conditions for a successful verification are not met.

The probability to obtain a good transcript τ in the real-int and the ideal-int worlds can now be computed. Let τ_e and τ_d denote the two parts of a transcript τ consisting respectively encryption and decryption queries, so that $\tau = \langle \tau_e, \tau_d \rangle$. With a slight abuse of notation, we have $\Pr[\Theta_{rein} = \tau] = \Pr[\Theta_{rein,e} = \tau_e] \cdot \Pr[\Theta_{rein,d} = \tau_d | \Theta_{rein,e} = \tau_e]$ and $\Pr[\Theta_{idin} = \tau] = \Pr[\Theta_{idin,e} = \tau_e] \cdot \Pr[\Theta_{idin,d} = \tau_d | \Theta_{idin,e} = \tau_e]$ and consequently

$$\frac{\Pr[\Theta_{rein,e} = \tau_e] \cdot \Pr[\Theta_{rein,d} = \tau_d | \Theta_{rein,e} = \tau_e]}{\Pr[\Theta_{idin,e} = \tau_e] \cdot \Pr[\Theta_{idin,d} = \tau_d | \Theta_{idin,e} = \tau_e]} = \frac{\Pr[\Theta_{rein,d} = \tau_d | \Theta_{rein,e} = \tau_e]}{\Pr[\Theta_{idin,d} = \tau_d | \Theta_{idin,e} = \tau_e]} .$$

This is because the encryption oracles in the real-int world and in the ideal-int world are identical, and so $\Pr[\Theta_{rein,e} = \tau_e] = \Pr[\Theta_{idin,e} = \tau_e]$. Further abusing notation, we let $\tau_{d,\Delta}$ denote

the marginal event of all Δ masks in the decryption queries (as variables) being equal to the values in the transcript. We have $\Pr[\Theta_{rein,d} = \tau_d | \Theta_{rein,e} = \tau_e, \Theta_{rein,d,\Delta} = \tau_{d,\Delta}] = \Pr[\Theta_{idin,d} = \tau_d | \Theta_{idin,e} = \tau_e, \Theta_{idin,d,\Delta} = \tau_{d,\Delta}]$ because both sides of this equality correspond to mappings defined with random permutations with the input-output pairs fixed from the encryption part in both worlds. Further, using this equality, we get

$$\frac{\Pr[\Theta_{rein,d} = \tau_d | \Theta_{rein,e} = \tau_e]}{\Pr[\Theta_{idin,d} = \tau_d | \Theta_{idin,e} = \tau_e]} = \frac{\Pr[\Theta_{rein,d,\Delta} = \tau_{d,\Delta} | \Theta_{rein,e} = \tau_e]}{\Pr[\Theta_{idin,d,\Delta} = \tau_{d,\Delta} | \Theta_{idin,e} = \tau_e]} .$$

 Let us now consider that τ has in total δ many Δs that are fixed/predefined due to all internal common prefixes. Clearly, one can write that $\delta = \sum_{i=1}^{q_e+q_v}(\mathsf{llcp}_n(i) + 1)$ (the extra 1 here stands for the Δ_1^i which is always fixed to 0). In the ideal-int world, since the Δs corresponding to these $(\sigma - \delta)$ unique block calls are sampled uniformly and independently and all decryption oracle results are $\bot$, one has $\Pr[\Theta_{idin,d,\Delta} = \tau_{d,\Delta} | \Theta_{idin,e} = \tau_e] = \frac{1}{(2^n)^{\sigma-\delta}}$. In the real-int 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 $g_1 = \sum_{i=1}^{q_e+q_v}(a^i - 1)$ block calls with the tweak 0^n and at least $g_2 = \sum_{i=1}^{q_e+q_v}(m^i - 1)$ block calls with the tweak $0^{n-1}\|1$. Thus, one has $\Pr[\Theta_{rein,d,\Delta} = \tau_{d,\Delta} | \Theta_{rein,e} = \tau_e] \geq \frac{1}{(2^n)_{g_1}(2^n)_{g_2}(2^n)^{\sigma-\delta-g_1-g_2}}$.

Note that the above expression is not an equality and only gives an upper bound on the targeted probability as there are more permutation calls which can have tweak collisions (To exemplify, 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

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

Combining the results of Lemma 2 and 3 (taking $\epsilon_1 = 0$) into Lemma 1, we obtain the upper bound $\mathbf{Adv}_{\mathrm{SAEF}[(\pi_0,\pi_1)]}^{\mathbf{auth}}(\mathcal{A}) \leq \frac{\sigma^2}{2^n} + \frac{4 \cdot q_v}{2^n}$ and thus the integrity result of the Theorem 1.

3.3   Confidentiality

Replacing F. We replace F with a pair of independent random tweakable permutations $\pi_0 = (\pi_{\mathsf{T},0} \leftarrow_\$ \mathrm{Perm}(n))_{\mathsf{T} \in \{0,1\}^t}$ 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 $\mathbf{Adv}_{\mathrm{SAEF}[\mathsf{F}]}^{\mathbf{oprpf}}(\mathcal{A}) \leq \mathbf{Adv}_{\mathsf{F}}^{\mathbf{prtfp}}(\mathcal{B}) + \mathbf{Adv}_{\mathrm{SAEF}[(\pi_0,\pi_1)]}^{\mathbf{oprpf}}(\mathcal{A})$.

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

Transcripts and additional information. As before, we record the interaction of $\mathcal{A}$ with its oracle in a transcript containing the encryption queries and their responses as $\tau = \langle (N^i, A^i, M^i, C^i)_{i=1}^{q_e} \rangle$ 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, 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 $\sum_{i=1}^{q_e}(a^i + m^i + 2) \leq \sigma$. To simplify the proofs, we again provide the adversary with additional information: **(i)** Δ_j^i masks for $1 \leq i \leq q_e$ and for $1 \leq j \leq \ell^i = a^i + m^i + 2$ that are internally computed by the encryption

algorithm of SAEF. 1. **(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 real-conf world, which faithfully implements SAEF encryption algorithm. In the ideal-conf world, however, they are not defined (as ciphertexts are directly computed by an online permutation and a random function, and the tags are directly sampled with the desired length) and thus the masks are sampled uniformly at random while maintaining the 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 ith encryption query as $(\mathsf{T}^i_j, \Delta^i_j, X^i_j, Y^i_j)_{j=1}^{\ell^i}, T^i$, with $\ell^i = a^i + m^i + 2$ and each of the ℓ^i quadruples consisting of the string T^i_j used as forkcipher tweak, the jth whitening mask Δ^i_j, the (possibly padded) associated data/plaintext block X^i_j and the empty/ciphertex block Y^i_j.

We reuse the same definition of the length of longest common blockwise prefix between the ith and the i'^{th} query $\mathsf{llcp}_n(i, i')$ and between the ith 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 $1 \leq i \leq q_e$, we let $\Delta^i_j = \Delta^{i'}_j$ 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^i_j uniformly at random.

Extended transcripts. The extended transcripts available to the adversary at the decision-making time are denoted as $\tau = \langle ((\mathsf{T}^i_j, \Delta^i_j, X^i_j, Y^i_j)_{j=1}^{\ell^i}, T^i)_{i=1}^{q_e} \rangle$.

Coefficient 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 coefficient H technique defined as Lemma 1 above. We represent the jth block call of the ith 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 $1 \leq i \leq q_e$, $\mathsf{llcp}_n(i) < j \leq \ell^i$ is not in the longest common prefix of the ith 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 $1 \leq i \leq q_e$, $\mathsf{llcp}_n(i) < j < \ell^i$ (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 $j = \ell^i$ 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 $j = \ell^i$ 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).

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 as good transcripts).

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

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 $\text{llcp}_n(i) < j \leq \ell^i$, $(i',j') < (i,j)$ and $\Delta^i_j \oplus \Delta^{i'}_{j'} = X^i_j \oplus X^{i'}_{j'}$.

Note that for all $i' < i$ and $j = j' = \text{llcp}_n(i) + 1$, 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 $\text{llcp}_n(i) = 0$ 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 $j \neq \text{llcp}_n(i) + 1$, 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 $\Pr[\text{BadT}'_1(\Theta_{idco}) = 1] \leq \frac{\sigma^2}{2} \cdot \max\left\{0, \frac{1}{2^n}\right\} = \frac{\sigma^2}{2^{n+1}}$.

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 $\text{llcp}_n(i) < j < \ell^i$ and one of the followings is true (with appropriate values of j)

I. $j = \text{llcp}_n(i)+1$ 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 $\text{llcp}_n(i)$, $X^i_j \neq X^{i'}_{j'}$ implies that $Y^i_j \neq Y^{i'}_{j'}$)

II. $j > \text{llcp}_n(i)+1$ 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, $j + 1 \neq \text{llcp}_n(i) + 1$. 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 $j > \text{llcp}_n(i)$, 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 $\sigma - q_e$ valid values of (i,j) for a Δ collision, each with no more that $\sigma - q_e$ possible values of (i',j'), yielding no more than $(\sigma - q_e)^2/2$ 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 $\Pr[\text{BadT}'_2(\Theta_{idco}) = 1] \leq \frac{2 \cdot \sigma^2}{2^{n+1}}$.

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

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

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. $\text{llcp}_n(i,i') < j < \ell^i$ 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. $X^i_j \oplus \Delta^i_j \neq X^{i'}_{j'} \oplus \Delta^{i'}_{j'}$) and different outputs (i.e., $Y^i_j \oplus \Delta^i_j \neq Y^{i'}_{j'} \oplus \Delta^{i'}_{j'}$ and $\Delta^i_{j+1} \neq \Delta^{i'}_{j'+1}$ respectively $T^i \neq T^{i'}$).

The probability to obtain 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 (upto common prefix) input-output pairs (excluding the tags), the left side of this inequality corresponds to mappings of a random permutation with 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 both 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 $\delta' = \sum_{i=1}^{q_e}(\mathsf{llcp}_n(i) + 1)$ (the extra 1 here stands for the Δ_1^i 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 $g_1' = \sum_{i=1}^{q_e}(a^i - 1)$ block calls with the tweak 0^n and at least $g_2' = \sum_{i=1}^{q_e}(m^i - 1)$ 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

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

Combining the results of Lemma 4 and 5 (taking $\epsilon_1 = 0$) into Lemma 1, we obtain the upper bound $\mathbf{Adv}_{\mathrm{SAEF}[(\pi_0,\pi_1)]}^{\mathbf{oprpf}}(\mathcal{A}) \leq \frac{3\cdot\sigma^2}{2^{n+1}}$ and thus the confidentiality result of the Theorem 1.

4   Conclusion

We prove that SAEF is OAE-secure w.r.t. both confidentiality and integrity as long as the total amount of data processed with a single key is $\ll 2^{n/2}$ blocks, with n the blocksize of the underling forkcipher. This means that SAEF offers qualitatively stronger guarantees than what has been advertised in the original submission at the same quantitative security levels. Moreover, the newly discovered security properties of SAEF are highly relevant to many resource-constrained applications of lightweight cryptography, as discussed in Section 1. At the same time, SAEF can be implemented very efficiently with an efficient forkcipher instance such as ForkSKINNY [24], likely outperforming COLM (the de facto state of the art OAE-secure scheme) instantiated with SKINNY [5] tweakable blockcipher (which gives the most accurate comparison due to similarities with ForkSKINNY). [5] This makes SAEF a much more attractive implementation target than indicated by the previous results, especially for the most constrained devices.

An interesting avenue for future work is investigating the security of SAEF under the release of unverified plaintext (RUP). RUP security is also a natural target in the uses cases for lightweight cryptography, where the same constraints as discussed in Section 1 imply that constrained devices may be forced to leak portions of yet-unverified plaintext when decrypting long messages, complementing the blockcwise security of encryption implied by OAE. Our conjecture is that SAEF is INT-RUP secure without any modification.

---

[5] We acknowledge that the performance gain of ForkSKINNY comes with a more ambitious security goal for the construction. However the investigation of this trade-off is out of scope of this work.

5 Acknowledgments

References

1. Andreeva, E., Bogdanov, A., Datta, N., Luykx, A., Mennink, B., Nandi, M., Tischhauser, E., Yasuda, K.: COLM v1 (2014), `"https://competitions.cr.yp.to/round3/colmv1.pdf"`
2. Andreeva, E., Bhati, A.S., Vizár, D.: Nonce-Misuse Security of the SAEF Authenticated Encryption mode. In: Selected Areas in Cryptography (2020)
3. 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)
4. Ashur, T., Dunkelman, O., Luykx, A.: Boosting authenticated encryption robustness with minimal modifications. In: Annual International Cryptology Conference. pp. 3–33. Springer (2017)
5. Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: The SKINNY family of block ciphers and its low-latency variant MANTIS. In: Annual International Cryptology Conference. pp. 123–153. Springer (2016)
6. Bellare, M., Boldyreva, A., Knudsen, L.R., Namprempre, C.: Online Ciphers and the Hash-CBC Construction. In: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology. p. 292–309. CRYPTO '01, Springer-Verlag, Berlin, Heidelberg (2001)
7. 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**(S1), 295–349 (Jun 2020). https://doi.org/10.13154/tosc.v2020.iS1.295-349, `https://tosc.iacr.org/index.php/ToSC/article/view/8623`
8. Bernstein, D.J.: Cryptographic competitions: CAESAR. `http://competitions.cr.yp.to`
9. Böck, H., Zauner, A., Devlin, S., Somorovsky, J., Jovanovic, P.: Nonce-Disrespecting Adversaries: Practical Forgery Attacks on GCM in TLS. In: 10th USENIX Workshop on Offensive Technologies (2016)
10. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: ASCON v1.2 (2014), `"https://competitions.cr.yp.to/round3/asconv12.pdf"`
11. Dworkin, M.J.: SP 800-38D. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC (2007)
12. Fleischmann, E., Forler, C., Lucks, S.: McOE: A Family of Almost Foolproof On-Line Authenticated Encryption Schemes. In: Canteaut, A. (ed.) Fast Software Encryption - 19th International Workshop, FSE 2012, Washington, DC, USA, March 19-21, 2012. Revised Selected Papers. Lecture Notes in Computer Science, vol. 7549, pp. 196–215. Springer (2012). https://doi.org/10.1007/978-3-642-34047-5_12, `https://doi.org/10.1007/978-3-642-34047-5_12`
13. Fouque, P.A., Joux, A., Martinet, G., Valette, F.: Authenticated On-Line Encryption. In: Matsui, M., Zuccherato, R.J. (eds.) Selected Areas in Cryptography, 10th Annual International Workshop, SAC 2003. Lecture Notes in Computer Science, vol. 3006, pp. 145–159. Springer, Ottawa, Canada (2004). https://doi.org/10.1007/978-3-540-24654-1_11, `https://hal.inria.fr/inria-00563967`
14. Guillaume Endignoux, D.V.: Linking Online Misuse-Resistant Authenticated Encryption and Blockwise Attack Models. Cryptology ePrint Archive, Report 2017/184 (2017), `https://eprint.iacr.org/2017/184`
15. 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)
16. Hoang, V.T., Reyhanitabar, R., Rogaway, P., Vizar, D.: Online Authenticated-Encryption and its Nonce-Reuse Misuse-Resistance. vol. 9215, pp. 493–517. Gennaro, R, Springer Verlag (2015)
17. Hongjun, W., Tao, H.: TinyJAMBU: A Family of Lightweight Authenticated Encryption Algorithms (2019), `"https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/round-1/spec-doc/TinyJAMBU-spec.pdf"`

18. 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 (May 2020). https://doi.org/10.13154/tosc.v2020.i1.43-120, `https://tosc.iacr.org/index.php/ToSC/article/view/8560`
19. Jean, J., Nikolić, I., Peyrin, T., Seurin, Y.: Deoxys v1.41 v1 (2016), `"https://competitions.cr.yp.to/round3/deoxysv141.pdf"`
20. Krovetz, T., Rogaway, P.: OCB v1.1 (2014), `"https://competitions.cr.yp.to/round3/ocbv11.pdf"`
21. O'Donnell, L.: "2 Million IoT Devices Vulnerable to Complete Takeover". Threatpost (2019), `"https://threatpost.com/iot-devices-vulnerable-takeover/144167/"`
22. O'Donnell, L.: "Serious Security Flaws Found in Children's Connected Toys". Threatpost (2019), `"https://threatpost.com/serious-security-flaws-found-in-childrens-connected-toys/151020/"`
23. Patarin, J.: The "Coefficients H" Technique, p. 328–345. Springer-Verlag, Berlin, Heidelberg (2009), `https://doi.org/10.1007/978-3-642-04159-4_21`
24. Purnal, A., Andreeva, E., Roy, A., Vizár, D.: What the Fork: Implementation Aspects of a Forkcipher. In: NIST Lightweight Cryptography Workshop 2019 (2019)
25. Rogaway, P.: Authenticated-Encryption with Associated-Data. In: Proceedings of the 9th ACM conference on Computer and communications security. pp. 98–107 (2002)
26. 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)
27. Rogaway, P., Shrimpton, T.: Deterministic authenticated-encryption: A provable-security treatment of the key-wrap problem. IACR Cryptology ePrint Archive **2006**, 221 (2006)
28. Vanhoef, M., Piessens, F.: Key reinstallation attacks: Forcing nonce reuse in WPA2. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 1313–1328. ACM (2017)
29. Wu, H.: ACORN v3 (2014), `"https://competitions.cr.yp.to/round3/acornv3.pdf"`
30. Wu, H., Huang, T.: MORUS v2 (2014), `"https://competitions.cr.yp.to/round3/morusv2.pdf"`
31. Wu, H., Preneel, B.: AEGIS v1.1 (2014), `"https://competitions.cr.yp.to/round3/aegisv11.pdf"`

A   Appendix: Proof of Proposition 1

We prove the proposition by describing the transformation in the backward direction, i.e. reconstructing SAEF input and output arguments from a block-tuple represented query. This is possible thanks to the sequence of tweak strings $\mathsf{T}^i_1, \ldots, \mathsf{T}^i_{\ell^i}$, which allows to unambiguously determine the boundary between the AD and message blocks, as well as whether the final block of AD and message respectively were n-bit long or not. Given a block-tuple represented query $((\mathsf{T}_j, \Delta_j, X_j, Y_j)^\ell_{j=1}, T)$, we can reconstruct 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 current tuple denoted 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 next tuple.
2. Parse T as $S\|F$. If $S \neq 0^{n-3}$ abort. If $F \neq 000$ go to next step. If $F_{last} \neq 000$ abort. Set $A = A\|X$ and $F_{last} = 000$. Go to 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 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 next tuple.

4. Parse T as $S\|F$. If $S \neq 0^{n-3}$ abort. If $F \neq 001$ go to next step. If $F_{last} \notin \{010, 011, 110, 111\}$ abort. Set $M = M\|X$, $C = C\|Y$ and $F_{last} = 001$. Go to 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.