Parallelized Authenticated Encryption with Tag Combiners

Christoph Dobraunig[1][0000−0002−3816−0187] and Charlotte Lefevre[2][0000−0003−0860−0097]

[1] Intel, Hillsboro, USA
christoph.dobraunig@intel.com
[2] Univ Rennes, Inria, CNRS, IRISA, Rennes, France
charlotte.lefevre@irisa.fr

Abstract. When looking at authenticated encryption schemes, we have schemes that process the input data by having serial calls to their underlying building blocks, like duplex-based constructions, and schemes that allow for parallel calls to their underlying building blocks, like the Galois Counter Mode (GCM). Naturally, one can parallelize a serial scheme by distributing the data to encrypt over different calls to the serial scheme. However, there are many different choices to be made, like how to choose the nonce for the different instances, or if and how to combine the multiple tags into a single one. In this paper, we investigate different possible choices providing proofs for their security. Interestingly, we see a huge variance in the provable properties and hence, the security in making a serial scheme parallel. Or, motivating the problem more generally, we are investigating tag combiners, where the single tags to be combined are secret to the adversary.

Keywords: authenticated encryption, parallelization, provable security

1 Introduction

One property that differentiates nonce-based authenticated encryption schemes is how they process the input data, either in a serial or parallel fashion. Typically, this refers to how the underlying building block, e.g., a blockcipher or cryptographic permutation, is used. Take for example AES-GCM [22]. Here, the blockcipher AES [14] is used in counter mode allowing to encrypt each block of ciphertext in parallel. The following authentication is done using a universal hash function [20, 27] that also allows for parallel processing of ciphertext blocks. In contrast, we have authenticated encryption schemes like Ascon [15], which is based on the duplex construction [10]. In this case, the input data is split into blocks that are processed one after each other so that upcoming blocks are influenced by previous blocks.

However, we can also use a serial scheme to process data in parallel. This can be done by splitting the data and processing it in parallel with different

instances of the serial authenticated encryption scheme. However, in doing so, we have different options in choosing the nonce of these instances and we must also tackle the questions on what to do with the tags. For example, if we split the data across x instances, we need to handle x nonces and x tags. To save space when storing or bandwidth when transmitting, it would be favorable to compress the x nonces and x tags to a single nonce and a single tag. There are many ways to do this, which all have potential impact on the achievable security.

In this paper, we explore numerous ways of handling the nonces and tags when parallelizing serial nonce-based authenticated encryption schemes. In particular, we put our focus on nonce-based authentication schemes that also recompute pseudo-random tags during verification and compare it with the provided tag, as detailed in Section 3. For these schemes, we explore different ways of initializing them, by either putting no restriction on the nonce, splitting the nonce into a nonce and a counter, or including the degree of parallelism as part of the associated data. The transformations that we consider are defined in detail in Section 4. On the tag side, we consider two major cases, either the single tag is just the XOR of the individual x tags, or the x tags are just concatenated, plus various post-processing of those major cases with, e.g., hash functions, or pseudorandom functions.

To the best of our knowledge, this is the first extensive study on different ways to parallelize serial nonce-based authenticated encryption schemes that comes with proofs on their security properties.

The idea of taking a serial authenticated encryption scheme and making it parallel appears at some places throughout the literature. Specially in the case of the duplex construction [10], we see remarks that they can be parallelized in a similar fashion as tree hashing [9]. The work in [24] further specifies ways to use the duplex construction in a parallel fashion, but lacks security proofs. During the CAESAR competition [12], several duplex-based parallel authenticated encryption algorithms were submitted, including NORX [3], the parallel mode of ICEPOLE [23], and π-Cipher [17]. A subsequent analysis provided security proofs covering NORX and ICEPOLE [18, 19]. In contrast to previous works, which are more or less dedicated schemes, our work is an in-depth exploration on various ways of making serial authenticated encryption schemes parallel.

2 Preliminaries

Let $a, b \in \mathbb{N}$ with $a \leq b$. $[\![a, b]\!]$ denotes the set $\{a, \ldots, b\}$. Given $x, \ell \in \mathbb{N}$ such that $\lceil \log_2(x) \rceil < \log_2(\ell)$, $\langle x \rangle_\ell$ represents the encoding of x over $\log_2(\ell)$ bits. We denote by $\{0,1\}^a$ the set of a-bit strings and $\{0,1\}^*$ the set of arbitrarily long strings, including the empty string. For $X, Y \in \{0,1\}^*$, $X \| Y$ denotes their concatenation, and if $|X| = |Y|$, we write their bitwise exclusive or by $X + Y$. Finally, given a finite set $\mathcal{S}$, $x \xleftarrow{\$} \mathcal{S}$ denotes that x is sampled uniformly at random in $\mathcal{S}$.

Adversaries and Distinguishing Advantages. An adversary $\mathcal{A}$ is an algorithm that is given access to a collection of oracles. If $\mathcal{A}$ is a distinguishing adversary,

after its interaction with the different oracles, it returns a decision bit $b \in \{0,1\}$. We denote this as "$\mathcal{A}[W] \to b$". For two collections of randomized oracles W_1 and W_2, we denote the advantage of an adversary $\mathcal{A}$ in distinguishing between them by

$$\Delta_{\mathcal{A}}[W_1, W_2] = |\mathbf{Pr}\left(\mathcal{A}[W_1] \to 1\right) - \mathbf{Pr}\left(\mathcal{A}[W_2] \to 1\right)| \ .$$

We will focus on information-theoretic adversaries, meaning that their complexities are only measured by the number and type of oracle queries.

Nonce-Based Message Authentication Code. A nonce-based message authentication code (NBMAC) $\mathcal{NBM}$ comprises two algorithms, indexed by a key $K \in \mathcal{K}$: one for generating tags $\mathcal{M}_K : \mathcal{N} \times \mathcal{M} \to \mathcal{T}$, and one verification algorithm $\mathcal{V}_K : \mathcal{N} \times \mathcal{M} \times \mathcal{T} \to \{\top, \bot\}$. In the associated security game, the key K is sampled uniformly from $\mathcal{K}$, and the adversary has oracle access to $\mathcal{M}_K$ and $\mathcal{V}_K$. The adversary is never allowed to query $\mathcal{V}_K$ with an input (N, M, T) that corresponds to a past query $\mathcal{M}_K(N, M) = T$. Moreover, we will be in the nonce-respecting setting, meaning that the adversary is not allowed to repeat a nonce N in its queries to $\mathcal{M}_K$. The adversary is said to produce a forgery if it makes a query to $\mathcal{V}_K$ that returns $\top$. The advantage of the adversary is defined as follows:

$$\mathbf{Adv}^{\mathrm{NBMAC}}_{\mathcal{NBM}}(\mathcal{A}) = \mathbf{Pr}\left(\mathcal{A}\left[\mathcal{M}_K\right] \text{ forges}\right) \ ,$$

where $K \xleftarrow{\$} \mathcal{K}$. Furthermore, $\mathbf{Adv}^{\mathrm{NBMAC}}_{\mathcal{NBM}}(q_M, q_F)$ denotes the maximum of $\mathbf{Adv}^{\mathrm{NBMAC}}_{\mathcal{NBM}}(\mathcal{A})$, over all *nonce-respecting* adversaries allowed to make q_M queries to $\mathcal{M}_K$ and q_F queries to $\mathcal{V}_K$.

Multicollisions. To upper bound the expected value of multicollisions, we will use the following result of [13], later slightly improved in [21].

Lemma 1. *Let $n, R \in \mathbb{N}$. Consider the experiment of throwing uniformly at random n balls in R bins. Let* $\mathrm{mCol}(n, R)$ *denote the expected highest number of balls in a single bin (max bin load). We have*

$$\mathrm{mCol}(n, R) \leq \frac{2n}{R} + 3\ln(R) + 4 \, .$$

The proof of Lemma 1 can be found in [21].

3 Authenticated Encryption with Associated Data

3.1 Definition

An AEAD scheme based on a primitive p is a tuple of algorithms $\mathtt{AEAD}^p = (\mathtt{AEenc}^p, \mathtt{AEdec}^p)$. The encryption algorithm takes as input a key $K \in \mathcal{K}$, a nonce $N \in \mathcal{N}$, associated data $A \in \mathcal{A}$, and a plaintext $P \in \mathcal{P}$. It outputs a ciphertext $C \in \mathcal{C}$ and a tag $T \in \mathcal{T}$. The decryption algorithm takes as input

a key $K \in \mathcal{K}$, a nonce $N \in \mathcal{N}$, associated data $A \in \mathcal{A}$, ciphertext $C \in \mathcal{C}$, and a tag $T \in \mathcal{T}$. It outputs either the corresponding plaintext $P \in \mathcal{P}$ or a failure symbol $\perp$. For convenience of notation, we will use the key as a subscript to the algorithms. Thus, for any $K \in \mathcal{K}$, we have

$$\begin{aligned} \texttt{AEenc}_K^p : \mathcal{N} \times \mathcal{A} \times \mathcal{P} &\longrightarrow \mathcal{C} \times \mathcal{T} \\ (N, A, P) &\longrightarrow (C, T)\,, \\ \texttt{AEdec}_K^p : \mathcal{N} \times \mathcal{A} \times \mathcal{C} \times \mathcal{T} &\longrightarrow \mathcal{P} \cup \{\perp\} \\ (N, A, C, T) &\longrightarrow P/\perp\,. \end{aligned}$$

We will focus on the binary case, so that $\mathcal{K} := \{0,1\}^k, \mathcal{N} := \{0,1\}^n, \mathcal{T} := \{0,1\}^t$, and $\mathcal{P}, \mathcal{C}, \mathcal{A} := \{0,1\}^*$. Another point of focus is that the AEAD has a constant expansion. More precisely, there exists $w \geq 0$ such that if $\texttt{AEenc}_K^p(N, A, P) = (C, T)$, then $|C| = |P| + w$. Note that with our notation, the ciphertext does not include the tag. Thus, in particular, when $w > 0$, this means that the scheme adds some redundancy to the plaintext, such as appending to the plaintext a string of w 0 bits. This concept was formalized in [8], and has recently been used to generically make AEADs more robust regarding key-commitment [1, 25].

We are interested in a subset of AEADs that allows to retrieve the plaintext without requiring the tag[3]. More formally, these schemes include a sub-algorithm

$$\texttt{D}_K^p : \mathcal{N} \times \mathcal{A} \times \mathcal{C} \to (\mathcal{P} \cup \{\perp\}) \times \mathcal{T}\,,$$

such that $\texttt{AEdec}_K^p$, on input (N, A, C, T), operates as follows:

- $(P, T^*) \leftarrow \texttt{D}_K^p(N, A, C)$;
- if $T = T^*$ return P. Else, return $\perp$.

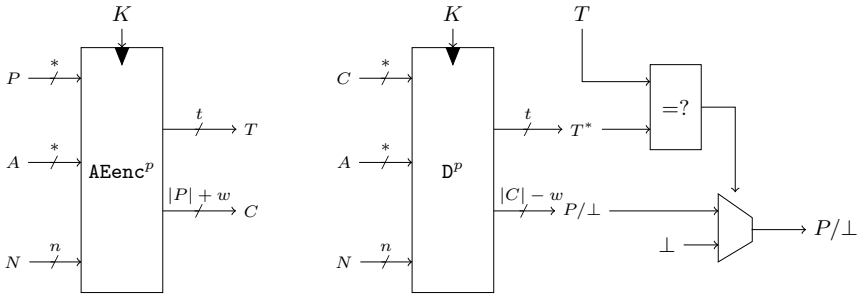
D handles the w bits of plaintext redundancy internally, which may result in returning $\perp$. The structure is illustrated in Fig. 1.

3.2 Security Model

We will consider schemes whose security proofs have been made in the ideal primitive model: at the beginning of the game, the primitive underlying the AEAD p is sampled uniformly at random from the appropriate set, denoted by $p \xleftarrow{\$} \text{Prim}$.[4] In our analysis, we consider the multi-user setting, where the scheme is used simultaneously by μ users. Let $(\$_m)_{m=1}^{\mu}$ be μ independent instances of a random function. For each new input $(N, A, P) \in \{0,1\}^n \times \{0,1\}^* \times \{0,1\}^*$, $\$_m$ returns a random string (C, T) with $|T| = t$ and $|C| = |P| + w$. The adversary $\mathcal{A}$ is never allowed to repeat a decryption query for which it already knows the

[3] The tag combiners we use may make the resulting scheme based on AEADs that do not have this property, e.g., APE [2], nonfunctional

[4] This definition also captures AEAD schemes based on standard-model assumptions by setting Prim to be a singleton. Our general formulation allows to additionally capture, among others, permutation-based constructions.



(a) Encryption (b) Decryption

Fig. 1: Structure of an AEAD scheme with plaintext retrieval.

answer thanks to a prior encryption query. $\mathcal{A}$ is called *nonce-respecting* if it never makes two ditinct encryption queries with the same nonce to the same user. The multi-user AE security of $\mathtt{AEAD}$ against an adversary $\mathcal{A}$ is defined as

$$\mathbf{Adv}^{\mathrm{AE}}_{\mathtt{AEAD}}(\mathcal{A}) = \Delta_{\mathcal{A}}\left(\left[\left(\mathtt{AEAD}^{p}_{K_m}\right)_{m=1}^{\mu}, p\right]; [(\$_m, \bot)_{m=1}^{\mu}, p]\right),$$

where $K_1, \ldots, K_\mu \xleftarrow{\$} \{0,1\}^k$, and $p \xleftarrow{\$}$ Prim, and $\bot$ denotes the function that always returns $\bot$.

Without loss of generality we focus on a specific class of adversaries called *orderly* [4,7]. These adversaries make decryption queries only after all encryption queries have been made, and the decryption queries do not depend on previous failed decryption attempts, although they may depend on earlier encryption queries. In our setting where the ciphertext has a constant expansion, it has been shown that for every adversary $\mathcal{A}$, there exists an orderly adversary $\mathcal{A}'$ with at most the same complexity as $\mathcal{A}$, such that [4, 7, 11]

$$\mathbf{Adv}^{\mathrm{AE}}_{\mathtt{AEAD}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{AE}}_{\mathtt{AEAD}}(\mathcal{A}') + \frac{q_D}{2^{t+w}},$$

where q_D denotes the number of decryption queries made by $\mathcal{A}$. Moreover, if $\mathcal{A}$ is nonce-respecting, so is $\mathcal{A}'$. The proof of this reduction can be found in the full version of [4] ([5, Appendix E]).

It will be convenient to split the AE security into confidentiality and authenticity, which are respectively defined as follows:

$$\mathbf{Adv}^{\mathrm{conf}}_{\mathtt{AEAD}}(\mathcal{A}) := \Delta_{\mathcal{A}}\left(\left[\left(\mathtt{AEenc}^{p}_{K_m}\right)_{m=1}^{\mu}, p\right]; [(\$_m)_{m=1}^{\mu}, p]\right), \tag{1}$$

$$\mathbf{Adv}^{\mathrm{auth}}_{\mathtt{AEAD}}(\mathcal{A}) := \mathbf{Pr}\left(\mathcal{A}\left[\left(\mathtt{AEAD}^{p}_{K_m}\right)_{m=1}^{\mu}, p\right] \text{ forges}\right), \tag{2}$$

where "$\mathcal{A}$ forges" means that it makes a query to the decryption oracle that does not return $\bot$ and that is not the result of a previous encryption query. We have that [6]:

$$\mathbf{Adv}^{\mathrm{AE}}_{\mathtt{AEAD}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{conf}}_{\mathtt{AEAD}}(\mathcal{A}') + \mathbf{Adv}^{\mathrm{auth}}_{\mathtt{AEAD}}(\mathcal{A}''), \tag{3}$$

where $\mathcal{A}'$ and $\mathcal{A}''$ have the same query complexity as $\mathcal{A}$, and preserve nonce-respecting.

4 Transformation

Let $\mathtt{AEAD} = (\mathtt{AEenc}, \mathtt{AEdec})$ be an AEAD that complies to the aforementioned conditions, with $\mathtt{AEdec}$ using $\mathtt{D}$. Let $t' \in \mathbb{N}^*$, $\mathcal{T}' = \{0,1\}^{t'}$, and let $\ell \in \mathbb{N}$. The transformation operates on top of a (potentially keyed and/or nonce-based) finalization function $F : \mathcal{N}' \times \mathcal{K}' \times \bigcup_{l=1}^{\ell} (\mathcal{T})^l \to \mathcal{T}'$. The transformation can process up to $L \in [\![1, \ell]\!]$ chunks of data in parallel. Our AEAD combiners can have further requirements on structuring the nonces or absorbing information on L. We introduce three variants:

- General case:

$$\mathfrak{T}_{F,K'}(\mathtt{AEAD}_K^p) = (\mathfrak{T}_{F,K'}(\mathtt{AEenc}_K^p), \mathfrak{T}_{F,K'}(\mathtt{D}_K^p)) \ ;$$

- Single nonce (requires that $n > \lceil \log_2(\ell) \rceil$):

$$\mathfrak{T}^{\aleph}_{F,K'}(\mathtt{AEAD}_K^p) = \left(\mathfrak{T}^{\aleph}_{F,K'}(\mathtt{AEenc}_K^p), \mathfrak{T}^{\aleph}_{F,K'}(\mathtt{D}_K^p)\right) \ ;$$

- Single nonce and split (requires that $n > \lceil \log_2(\ell) \rceil$):

$$\mathfrak{T}^{\aleph L}_{F,K'}(\mathtt{AEAD}_K^p) = \left(\mathfrak{T}^{\aleph L}_{F,K'}(\mathtt{AEenc}_K^p), \mathfrak{T}^{\aleph L}_{F,K'}(\mathtt{D}_K^p)\right) \ .$$

The three transformations have key space $\mathcal{K} \times \mathcal{K}'$, message space $\bigcup_{l=1}^{\ell} (\mathcal{P})^l$, associated data space $\bigcup_{l=1}^{\ell} (\mathcal{A})^l$, ciphertext space $\bigcup_{l=1}^{\ell} (\mathcal{C})^l$, and tag space $\mathcal{T}'$. The number of elements in the plaintext/ciphertext tuples must match the number of elements in the associated data tuples.

General Case. $\mathfrak{T}_{F,K'}(\mathtt{AEAD}_K^p)$ has a nonce space of $\bigcup_{l=1}^{\ell} (\mathcal{N})^l$, and the number of elements in the nonce tuple must be equal to the number of elements in the plaintext/ciphertext tuples. In this case, the nonce space of the finalization function is either $\mathcal{N}' = \emptyset$ or $\mathcal{N}' = \bigcup_{l=1}^{\ell} (\mathcal{N})^l$. It is defined as follows:

- Encryption: on input $(N_1, \ldots, N_L), (A_1, \ldots, A_L), (P_1, \ldots, P_L)$ with keys (K, K'):
  - Step 1: for $l = 1, \ldots, L$, compute $(C_l, T_l) \leftarrow \mathtt{AEenc}_K^p(N_l, A_l, P_l)$
  - Step 2: if $\mathcal{N}' = \emptyset$, let $N' = \emptyset$; otherwise, let $N' = (N_1, \ldots, N_L)$, then compute $T = F(N', K', (T_1, \ldots, T_L))$
  - Output: $((C_1, \ldots, C_L), T)$
- Decryption: on input $(N_1, \ldots, N_L), (A_1, \ldots, A_L), (C_1, \ldots, C_L), T$ with keys (K, K'):
  - Step 1: for $l = 1, \ldots, L$, compute $(P_l, T_l^*) \leftarrow \mathtt{D}_K^p(N_l, A_l, C_l)$
  - Step 2: if one of the P_l equals to $\perp$, return $\perp$. Else, proceed.
  - Step 3: if $\mathcal{N}' = \emptyset$, let $N' = \emptyset$; otherwise, let $N' = (N_1, \ldots, N_L)$, then compute $T^* = F(N', K', (T_1^*, \ldots, T_L^*))$
  - Output: if $T = T^*$, return $(P_1, \ldots, P_L)$ else return $\perp$

Single Nonce. For $\mathfrak{T}^{\aleph}_{F,K'}(\mathtt{AEAD}^p_K)$, we have further restrictions on the nonce space; we essentially create a tuple $(N_1, \ldots, N_L)$ from a single main nonce $\aleph \in \{0,1\}^{n-\lceil \log_2(\ell) \rceil}$ concatenated with a counter l. In this case, the nonce space of the finalization function is either $\mathcal{N}' = \emptyset$ or $\mathcal{N}' = \{0,1\}^{n-\lceil \log_2(\ell) \rceil}$. It is defined as follows:

- Encryption: on input $\aleph, (A_1, \ldots, A_L), (P_1, \ldots, P_L)$ with keys (K, K'):
  - Step 1: for $l = 1, \ldots, L$, compute $(C_l, T_l) \leftarrow \mathtt{AEnc}^p_K(\aleph \| \langle l-1 \rangle_\ell, A_l, P_l)$
  - Step 2: if $\mathcal{N}' = \emptyset$, let $\aleph' = \emptyset$; otherwise, let $\aleph' = \aleph$, then compute $T = F(\aleph', K', (T_1, \ldots, T_L))$
  - Output: $((C_1, \ldots, C_L), T)$
- Decryption: on input $\aleph, (A_1, \ldots, A_L), (C_1, \ldots, C_L), T$ with keys (K, K'):
  - Step 1: for $l = 1, \ldots, L$, compute $(P_l, T^*_l) \leftarrow \mathtt{D}^p_K(\aleph \| \langle l-1 \rangle_\ell, A_l, C_l)$
  - Step 2: if one of the P_l equals to $\perp$, return $\perp$. Else, proceed.
  - Step 3: if $\mathcal{N}' = \emptyset$, let $\aleph' = \emptyset$; otherwise, let $\aleph' = \aleph$, then compute $T^* = F(\aleph', K', (T^*_1, \ldots, T^*_L))$
  - Output: if $T = T^*$, return $(P_1, \ldots, P_L)$ else return $\perp$

Single Nonce plus Split. For $\mathfrak{T}^{\aleph L}_{F,K'}(\mathtt{AEAD}^p_K)$, we require additionally that L is concatenated with the associated data. It is defined as follows:

- Encryption: on input $\aleph, (A_1, \ldots, A_L), (P_1, \ldots, P_L)$ with keys (K, K'):
  - Step 1: for $l = 1, \ldots, L$, compute $(C_l, T_l) \leftarrow \mathtt{AEnc}^p_K(\aleph \| \langle l-1 \rangle_\ell, \langle L-1 \rangle_\ell \| A_l, P_l)$
  - Step 2: if $\mathcal{N}' = \emptyset$, let $\aleph' = \emptyset$; otherwise, let $\aleph' = \aleph$, then compute $T = F(\aleph', K', (T_1, \ldots, T_L))$
  - Output: $((C_1, \ldots, C_L), T)$
- Decryption: on input $\aleph, (A_1, \ldots, A_L), (C_1, \ldots, C_L), T$ with keys (K, K'):
  - Step 1: for $l = 1, \ldots, L$, compute $(P_l, T^*_l) \leftarrow \mathtt{D}^p_K(\aleph \| \langle l-1 \rangle_\ell, \langle L-1 \rangle_\ell \| A_l, C_l)$
  - Step 2: if one of the P_l equals to $\perp$, return $\perp$. Else, proceed.
  - Step 3: if $\mathcal{N}' = \emptyset$, let $\aleph' = \emptyset$; otherwise, let $\aleph' = \aleph$, then compute $T^* = F(\aleph', K', (T^*_1, \ldots, T^*_L))$
  - Output: if $T = T^*$, return $(P_1, \ldots, P_L)$ else return $\perp$

Instances for F. We will consider different functions to process the tags. The two base combiners are the concatenation $id_\|(T_1, \ldots, T_L) = T_1 \| \cdots \| T_L$ and the XOR $id_+(T_1, \ldots, T_L) = T_1 + \cdots + T_L$. Both of these combiners can be composed with a post-processing function: a pseudo-random function (PRF), a truncated random oracle ($\mathcal{RO}$), or an NBMAC. For a post-processing function $F \in \{\mathrm{PRF}, \mathcal{RO}, \mathcal{NBM}\}$ and a combiner $c \in \{\|, +\}$, we write F_c for the function that first combines the tags with id_c and then applies F, i.e., $F_c(T_1, \ldots, T_L) = F\big(id_c(T_1, \ldots, T_L)\big)$. We will also consider applying a block cipher directly after id_+, which we denote $\mathcal{BC}_+$, i.e., $\mathcal{BC}_+(T_1, \ldots, T_L) = \mathcal{BC}(T_1 + \cdots + T_L)$. In this analysis, only NBMACs take a nonce as input, and they use the same nonce that is provided to $\mathtt{AEAD}$. In the general case, the nonce is then $N_1 \| N_2 \| \ldots \| N_L$; in all other cases, it is $\aleph$. This is because most MACs that do not behave like PRFs are universal-hash based, like Wegman-Carter [27], whose security relies on nonce-respecting security. This ensures a relatively wide coverage.

On Encoding Data Into Tuples. Our scheme is somewhat agnostic on how associated data and plaintext/ciphertext is encoded (split) into the required associated data and plaintext/ciphertext tuples. The encoding just has to be understood in the same way by the sender and receiver. One possible encoding is to split d-bit data into, e.g., $\lceil d/128 \rceil$ blocks with a possible last block that is incomplete and distribute them as evenly as possible among the tuples. For example, consider $L = 2$. We use Ascon-AEAD128 [26] and want to process 120-bit associated data and 448-bit plaintext. Following the proposed encoding, we get the tuple (A_1, A_2) with length 120 bits and 0 bit and the tuple (P_1, P_2) with length 256 bits and 192 bits. So, the first Ascon instance makes 2 calls to p^{12}, 1 call to p^8 due to the existing associated data and 2 calls to p^8 due to the plaintext (due to $10*$ padding of the plaintext). The second Ascon instance makes 2 calls to p^{12}, 0 calls to p^8 since no associated data is present and 1 call to p^8 due to the plaintext.

Another encoding option is to not look at blocks, but just distribute the data as evenly as possible on a byte level. So, first data byte goes to the first instance, second data byte to the second and so on. Let us consider the example of $L = 2$, Ascon-AEAD128, 120-bit associated data, and 448-bit plaintext again. Then our tuple (A_1, A_2) has a length of 64 bits and 56 bits and the tuple (P_1, P_2) has a lengths 224 bits and 224 bits. So, the first Ascon instance makes 2 calls to p^{12}, 1 call to p^8 due to the associated data and 1 call to p^8 due to the plaintext. The second Ascon instance makes 2 calls to p^{12}, 1 call to p^8 due to the associated data and 1 call to p^8 due to the plaintext.

So, we see that different encodings lead to different utilization of the instances, but also to the overall calls to the underlying primitives. In the worst case, encodings that interact badly with the underlying specifics of the used authenticated encryption scheme create a lot of "empty" calls to the underlying primitives. Hence, one should consider the specifics of the used authenticated encryption scheme when deciding on which encoding to use.

5 Security Analysis

Let $s \in \{\text{“”}, \aleph, \aleph L\}$. We split the analysis of $\mathfrak{T}^s$ into confidentiality and authenticity as in (3). In the following, a *chunk* corresponds to one tuple $B_l = (N_l, A_l, M_l)$ that is processed by one call to $\texttt{AEenc}_K^p$ or $\texttt{D}_K^p$.

Nonce-Respecting Adversaries. When $s \in \{\aleph, \aleph L\}$, a nonce-respecting adversary $\mathcal{A}$ is also nonce-respecting in the calls to `AEAD` (i.e., the calls to $\texttt{AEenc}_K^p$ do not repeat the same nonce). However, in the context of the transformation $\mathfrak{T}_{F,K'}$, $\mathcal{A}$ being nonce-respecting does not translate to a nonce-respecting setting in the underlying AEAD calls. For instance, a nonce-respecting adversary may make two queries with input nonces respectively (N_1, N_2) and (N_1', N_2), and in that case, the AEAD call at the second position repeats a nonce. We therefore call an adversary *chunkwise nonce-respecting* that never repeats individual nonces.

Table 1: Summary of authenticity security bounds for different combinations of the transformation and the finalization function. "NR" stands for nonce-respecting, "NM" for nonce-misuse. $^{\mathtt{T\$}}\mathtt{AEAD}$ is defined and discussed in Section 5.2. Proofs for the $\cdot_+$ family can be found in Section 6 and for the $\cdot_\|$ family in Section 7.

<table>
<tr><th>Transform / F</th><th>$\mathfrak{T}_F(^{\mathtt{T\$}}\mathtt{AEAD})$</th><th>$\mathfrak{T}^{\aleph}_F(^{\mathtt{T\$}}\mathtt{AEAD})$</th><th>$\mathfrak{T}^{\aleph L}_F(^{\mathtt{T\$}}\mathtt{AEAD})$</th></tr>
<tr><td>id_+ (NR)</td><td rowspan="8">Insecure</td><td colspan="2">$\frac{q_D}{2^t}$</td></tr>
<tr><td>id_+ (NM)</td><td colspan="2">Insecure</td></tr>
<tr><td>$\mathcal{BC}_+$ (NR)</td><td colspan="2">$\frac{q_D}{2^t}$</td></tr>
<tr><td>$\mathcal{BC}_+$ (NM)</td><td colspan="2">$\frac{2q_D}{2^t - q_t} + \frac{q_t(q_t+1)}{2^{t+1}}$</td></tr>
<tr><td>PRF_+ (NR)</td><td colspan="2">$\frac{q_D}{2^{t'}} + \frac{2q_Dq_E}{2^{t+t'}} + \frac{(3t'+4)q_D}{2^t}$ $(*_{\mathrm{PRF}})$</td></tr>
<tr><td>PRF_+ (NM)</td><td colspan="2">$\frac{q_t(q_t+1)}{2^{t+1}} + (*_{\mathrm{PRF}})$</td></tr>
<tr><td>$\mathcal{RO}_+$ (NR)</td><td colspan="2">$\frac{q_D}{2^{t'}} + \frac{2q_D(q_E+q_O)}{2^{t+t'}} + \frac{(3t'+4)q_D}{2^t}$ $(*_{\mathcal{RO}})$</td></tr>
<tr><td>$\mathcal{RO}_+$ (NM)</td><td colspan="2">$\frac{q_tq_O}{2^t} + \frac{q_t(q_t+1)}{2^{t+1}} + (*_{\mathcal{RO}})$</td></tr>
<tr><td>$\mathcal{NBM}_+$ (NR)</td><td colspan="3">$\frac{q_D}{2^t} + \mathbf{Adv}^{\mathrm{NBMAC}}_{\mathcal{NBM}}(q_E, q_D)$ $(*_{\mathcal{NBM}})$</td></tr>
<tr><td>$id_\|$ (NR)</td><td rowspan="2" colspan="2">Insecure</td><td>$\frac{q_D}{2^t}$</td></tr>
<tr><td>$id_\|$ (NM)</td><td>Insecure</td></tr>
<tr><td>$\mathrm{PRF}_\|$ (NR)</td><td>$\frac{q_E(q_E-1)}{2^{2t+1}} + (*_{\mathrm{PRF}})$</td><td colspan="2">$(*_{\mathrm{PRF}})$</td></tr>
<tr><td>$\mathrm{PRF}_\|$ (NM)</td><td>$\frac{\sigma_E(\sigma_E-1)}{2^{t+1}} + (*_{\mathrm{PRF}})$</td><td>$\frac{q_t(q_t+1)}{2^t} + \frac{q_E(q_E-1)}{2^{2t+1}} + (*_{\mathrm{PRF}})$</td><td>$\frac{q_t(q_t+1)}{2^t} + (*_{\mathrm{PRF}})$</td></tr>
<tr><td>$\mathcal{RO}_\|$ (NR)</td><td>$\frac{q_Eq_O}{2^t} + \frac{q_E(q_E-1)}{2^{2t+1}} + (*_{\mathcal{RO}})$</td><td>$\frac{q_Eq_O}{2^{2t}} + (*_{\mathcal{RO}})$</td><td>$(*_{\mathcal{RO}})$</td></tr>
<tr><td>$\mathcal{RO}_\|$(NM)</td><td>$\frac{q_Eq_O}{2^t} + \frac{\sigma_E(\sigma_E-1)}{2^{t+1}} + (*_{\mathcal{RO}})$</td><td>$\frac{q_tq_O}{2^t} + \frac{q_Eq_O}{2^{2t}} + \frac{q_t(q_t+1)}{2^{t+1}} + \frac{q_E(q_E-1)}{2^{2t+1}} + (*_{\mathcal{RO}})$</td><td>$\min\left(\frac{2q_Eq_O}{2^{2t}}, \frac{q_tq_O}{2^t}\right) + \frac{q_t(q_t+1)}{2^t} + (*_{\mathcal{RO}})$</td></tr>
<tr><td>$\mathcal{NBM}_\|$ (NR)</td><td colspan="3">$(*_{\mathcal{NBM}})$</td></tr>
</table>

Nonce-Misuse Adversaries. We will consider authenticity both in the nonce-respecting and the nonce-misuse setting. However, our bounds in the nonce-misuse setting are only preservation results: if $\mathtt{AEAD}$ is already misuse-resistant, our transformations may preserve nonce-misuse authenticity up to the specified bounds and the conditions specified in Section 5.2. However, they *cannot* generically repair an AEAD that is insecure under nonce reuse; doing so would require a dedicated analysis.

Metrics for Queries. The considered adversary $\mathcal{A}$ makes q_E (resp., q_D) encryption (resp., decryption) queries with a total number of encryption (resp., decryption) chunks equal to σ_E (resp., σ_D) to $\mathfrak{T}^s_{F,K'}(\mathtt{AEAD}^p_K)$. Note we have $\sigma_X \leq \ell \times q_X$. With a nonce-misuse adversary, we denote by σ_t the maximum number of chunks where a nonce is being reused and q_t as the maximum number of queries where a nonce is reused. Moreover, when the finalization function considered is a random oracle, we denote by q_O the number of queries made by $\mathcal{A}$ to the random oracle.

5.1 Confidentiality

Note that in our setting, confidentiality can be achieved only if the adversary is (chunkwise) nonce-respecting. Let therefore $\mathcal{A}$ be a (chunkwise) nonce-respecting adversary. Let $(\Gamma_m)^\mu_{m=1}$ be μ independent idealized version of the transformation with the same input/output interface, returning independent uniform outputs for every fresh query. Moreover, let $(\$_m)^\mu_{m=1}$ be μ independent random functions as defined in Section 3.2. We decompose the confidentiality advantage as follows:

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{conf}}_{\mathfrak{T}^s_{F,K'}(\mathtt{AEAD})}(\mathcal{A}) &= \Delta_{\mathcal{A}}\left(\left[\left(\mathfrak{T}^s_{F,K'_m}(\mathtt{AEenc}^p_{K_m})\right)^\mu_{m=1}, p\right]; [(\Gamma_m)^\mu_{m=1}, p]\right) \\
&\leq \Delta_{\mathcal{A}}\left(\left[\left(\mathfrak{T}^s_{F,K'_m}(\mathtt{AEenc}^p_{K_m})\right)^\mu_{m=1}, p\right]; \left[\left(\mathfrak{T}^s_{F,K'_m}(\$_m)\right)^\mu_{m=1}, p\right]\right) \\
&\quad + \Delta_{\mathcal{A}}\left(\left[\left(\mathfrak{T}^s_{F,K'_m}(\$_m)\right)^\mu_{m=1}, p\right]; [(\Gamma_m)^\mu_{m=1}, p]\right) \\
&\leq \mathbf{Adv}^{\mathrm{conf}}_{\mathtt{AEAD}}(\mathcal{A}') + \Delta_{\mathcal{A}}\left(\left[\left(\mathfrak{T}^s_{F,K'_m}(\$_m)\right)^\mu_{m=1}, p\right]; [(\Gamma_m)^\mu_{m=1}, p]\right),
\end{aligned}$$

where $\mathcal{A}'$ is a nonce-respecting adversary that makes at most σ_E encryption queries to $\mathtt{AEenc}^p_K$. The remaining distance depends on the behavior of the finalization function when applied to random inputs.

If $F \in \{id_+, id_\|, \mathcal{BC}_+\}$, then the finalization function preserves uniformity and independence of the tags, so that

$$\Delta_{\mathcal{A}}\left(\left[\left(\mathfrak{T}^s_{F,K'_m}(\$_m)\right)^\mu_{m=1}, p\right]; [(\Gamma_m)^\mu_{m=1}, p]\right) = 0\,.$$

If $F \in \{\mathcal{RO}_\|, \mathcal{RO}_+, \mathrm{PRF}_\|, \mathrm{PRF}_+\}$, then the finalization function introduces an additional source of collisions at its input, and we have

$$\Delta_{\mathcal{A}}\left(\left[\left(\mathfrak{T}^s_{F,K'_m}(\$_m)\right)^\mu_{m=1}, p\right]; [(\Gamma_m)^\mu_{m=1}, p]\right) \leq \frac{\sigma_E(\sigma_E - 1)}{2^{t+1}}\,.$$

Finally, if F is an NBMAC construction, then nothing guarantees that the resulting tags remain indistinguishable from random, so it may not achieve confidentiality.

Therefore, confidentiality depends on the finalization function, and the corresponding term quantifies the loss of tag pseudo-randomness induced by the chosen finalization function.

5.2 Authenticity

Security Requirements on Tag Generation. For authenticity, we require that the tag-generation mechanism used by $\mathtt{AEenc}$ and $\mathtt{D}$ produces tags that are indistinguishable from uniform random values. To formalize this, we define from $\mathtt{AEAD}$ an auxiliary construction ${}^{\mathtt{T\$}}\mathtt{AEAD} = ({}^{\mathtt{T\$}}\mathtt{AEenc}, {}^{\mathtt{T\$}}\mathtt{AEdec})$, which follows the same structure defined in Section 3.1, with plaintext-retrieval construction ${}^{\mathtt{T\$}}\mathtt{D}$. Informally, ${}^{\mathtt{T\$}}\mathtt{AEAD}^p_K$ behaves identically to $\mathtt{AEAD}^p_K$, except that the tags are replaced by uniform random values for every new input. The exact definition of ${}^{\mathtt{T\$}}\mathtt{AEenc}^p_K$ and ${}^{\mathtt{T\$}}\mathtt{D}^p_K$ are provided in Algorithm 1. We require that the distinguishing advantage between $\mathtt{AEAD}$ and ${}^{\mathtt{T\$}}\mathtt{AEAD}$ is negligible even under nonce-misuse, that is

$$\Delta_{\mathcal{A}}\left(\left[\left(\mathtt{AEAD}^p_{K_m}\right)^{\mu}_{m=1}, p\right]; \left[\left({}^{\mathtt{T\$}}\mathtt{AEAD}^p_{K_m}\right)^{\mu}_{m=1}, p\right]\right), \tag{4}$$

must be negligible. This property is implied by confidentiality in the nonce-misuse setting. However, schemes that are secure in the nonce-respecting setting but do not achieve nonce-misuse AE security may still satisfy this property (e.g., duplex-based schemes).

Our notion is closely related to the PR-tag notion introduced by [16], which was defined in the context of online misuse-resistant authenticated encryption (OAE) and captures that tags produced on encryption queries *only* should be indistinguishable from random. We discuss the relation between our notion and PR-tag security in Supplementary Material A.

Security Reduction. Remember that we consider an orderly adversary. We also forbid the adversary to make a decryption query that corresponds to a previous encryption query. We have:

$$\begin{aligned}
&\mathbf{Pr}\left(\mathcal{A}\left[\left(\mathfrak{T}^s_{F,K'_m}(\mathtt{AEAD}^p_{K_m})\right)^{\mu}_{m=1}, p\right] \text{ forges}\right) = \\
&\mathbf{Pr}\left(\mathcal{A}\left[\left(\mathfrak{T}^s_{F,K'_m}(\mathtt{AEAD}^p_{K_m})\right)^{\mu}_{m=1}, p\right] \text{ forges}\right) \\
&\quad - \mathbf{Pr}\left(\mathcal{A}\left[\left(\mathfrak{T}^s_{F,K'_m}({}^{\mathtt{T\$}}\mathtt{AEAD}^p_{K_m})\right)^{\mu}_{m=1}, p\right] \text{ forges}\right) \\
&\quad + \mathbf{Pr}\left(\mathcal{A}\left[\left(\mathfrak{T}^s_{F,K'_m}({}^{\mathtt{T\$}}\mathtt{AEAD}^p_{K_m})\right)^{\mu}_{m=1}, p\right] \text{ forges}\right) \leq \\
&\Delta_{\mathcal{A}''}\left(\left[\left(\mathtt{AEAD}^p_{K_m}\right)^{\mu}_{m=1}, p\right]; \left[\left({}^{\mathtt{T\$}}\mathtt{AEAD}^p_{K_m}\right)^{\mu}_{m=1}, p\right]\right) + \mathbf{Adv}^{\mathrm{auth}}_{\mathfrak{T}^s_{F,K'}({}^{\mathtt{T\$}}\mathtt{AEAD}_{K_m})}(\mathcal{A}),
\end{aligned}$$

Algorithm 1: Definition of ${}^{\mathtt{T\$}}\mathtt{AEnc}^p_K$ and ${}^{\mathtt{T\$}}\mathtt{D}^p_K$. Inputs include $N \in \{0,1\}^n$, $A, P, C \in \{0,1\}^*$.

Function `Init()`:
  $\mathtt{DataBase} \leftarrow \emptyset$;
Function ${}^{\mathtt{T\$}}\mathtt{AEnc}^p_K(N, A, P)$:
  if $\exists C, T$ *such that* $(N, A, P, C, T) \in \mathtt{DataBase}$ **then**
    return (C, T);
  end
  $C, _ \leftarrow \mathtt{AEnc}^p_K(N, A, P)$;
  $T \xleftarrow{\$} \{0,1\}^{t'}$;
  $\mathtt{DataBase} \leftarrow \mathtt{DataBase} \cup \{(N, A, P, C, T)\}$;
  return (C, T);
Function ${}^{\mathtt{T\$}}\mathtt{D}^p_K(N, A, C)$:
  if $\exists P, T$ *such that* $(N, A, P, C, T) \in \mathtt{DataBase}$ **then**
    return (P, T);
  end
  $P, _ \leftarrow \mathtt{D}^p_K(N, A, C)$;
  $T \xleftarrow{\$} \{0,1\}^{t'}$;
  $\mathtt{DataBase} \leftarrow \mathtt{DataBase} \cup \{(N, A, P, C, T)\}$;
  return (P, T);

where $\mathcal{A}''$ is a distinguisher that makes at most σ_E (resp., σ_D) encryption (resp., decryption) queries. The leftmost distance corresponds to (4). The remainder of this work is dedicated to the analysis of

$$\mathbf{Adv}^{\mathrm{auth}}_{\mathfrak{T}^s_{F,K'}({}^{\mathtt{T\$}}\mathtt{AEAD}_{K_m})}(\mathcal{A}) := \mathbf{Pr}\left(\mathcal{A}\left[\left(\mathfrak{T}^s_{F,K'_m}({}^{\mathtt{T\$}}\mathtt{AEAD}^p_{K_m})\right)^{\mu}_{m=1}, p\right] \text{ forges}\right) .$$

Table 1 summarizes our findings.

6 Authenticity Bounds for the $\cdot_+$ family

We assess authenticity, and consider for simplicity non 0-padded schemes (i.e., $w = 0$), so that authenticity just comes from the tags. Let us define some further terminology.

Assume that the adversary makes an encryption query

$$N, (A_1, \ldots, A_L), (P_1, \ldots, P_L) \rightarrow (C_1, \ldots, C_L), T\,.$$

Moreover, let $(N_1, \ldots, N_L)$ be the ordered list of nonces fed into the AEAD. A subsequent decryption query is called "truncated" if there exists $l \in [\![1, L-1]\!]$ such that the decryption query has input $N', (A_1, \ldots, A_l), (C_1, \ldots, C_l), T'$, where the ordered list of nonces fed into the AEAD equals $(N_1, \ldots, N_l)$. Moreover, with the general transformation $\mathfrak{T}_F$, a decryption query is "re-ordered" if it has input

$$(N_{\pi(1)}, \ldots, N_{\pi(L)}), (A_{\pi(1)}, \ldots, A_{\pi(L)})(C_{\pi(1)}, \ldots, C_{\pi(L)}), T'\,,$$

where π is a permutation over $[\![1, L]\!]$. A *partial intermediate* tag is a tag T_l produced by a call to `AEenc` or `D`. An *intermediate* tag is the input to the finalization function. Remark that, when the finalization function does not take a nonce as input, $\mathfrak{T}^{\aleph L}_{F,K'}$ can be simulated using $\mathfrak{T}^{\aleph}_{F,K'}$, which can be itself simulated using $\mathfrak{T}_{F,K'}$. Therefore, when $F \notin \{\mathcal{NBM}_{\|}, \mathcal{NBM}_{+}\}$, we have

$$\mathbf{Adv}^{\mathrm{auth}}_{\mathfrak{T}_{F,K'}(\tau\$\mathtt{AEAD})}(\mathcal{A}) \geq \mathbf{Adv}^{\mathrm{auth}}_{\mathfrak{T}^{\aleph}_{F,K'}(\tau\$\mathtt{AEAD})}(\mathcal{A}) \geq \mathbf{Adv}^{\mathrm{auth}}_{\mathfrak{T}^{\aleph L}_{F,K'}(\tau\$\mathtt{AEAD})}(\mathcal{A})\,. \quad (5)$$

Finally, the intermediate tags have size t, while the finalization function maps to tags of size t' bits.

6.1 Nonce-Respecting Setting

General Case, Chunkwise Nonce-Respecting. For $F \in \{id_+, \mathcal{BC}_+, \mathcal{RO}_+, \mathrm{PRF}_+\}$, this is insecure due to the possible reordering of $((N_l)_{l=1}^{\ell}, (A_l)_{l=1}^{\ell}, (C_l)_{l=1}^{\ell})$ without influencing tag value. For example, $((N_1, N_2), (A_1, A_2), (C_1, C_2))$ and $((N_2, N_1), (A_2, A_1), (C_2, C_1))$ lead to the same tag value.

$\mathcal{NBM}_+$. We set **Bad** if there exists a decryption query with a nonce $(N_1, \ldots, N_L)$ such that its intermediate tag collides with that of a (necessarily unique) earlier encryption query with the same nonce $(N_1, \ldots, N_L)$. We have $\mathbf{Pr}\,(\mathbf{Bad}) \leq \frac{q_D}{2^t}$.

Assuming $\neg$**Bad**, the intermediate tag of every decryption query, along with the nonce used, forms an input to the NBMAC that has never occurred in an earlier encryption query. Hence, the adversary must forge a tag for a new input. Therefore,

$$\mathbf{Adv}^{\mathrm{auth}}_{\mathfrak{T}_{\mathcal{NBM}_+}(\tau\$\mathtt{AEAD})}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{NBMAC}}_{\mathcal{NBM}}(q_E, q_D) + \frac{q_D}{2^t}\,. \quad (*_{\mathcal{NBM}})$$

Single Nonce. In the following, we introduce a game $\mathbb{G}$, useful for analyzing both nonce-respecting and nonce-misuse cases. In this game, for each new encryption or decryption query, if the intermediate tag is not a linear combination of earlier intermediate tags, we replace the sampling-then-sum step with a direct uniform sampling of an intermediate tag of t bits. Otherwise, if it is a linear combination, the intermediate tag generated corresponds to that combination. Consider for instance the following query history (in the nonce-misuse case):

- Encryption query with chunks (B_1, B_2): generates T_1, T_2, intermediate tag is $T_1 + T_2 \rightarrow$ replaced with $T^{(1)}$;
- Encryption query with chunks (B_1, B'_2): generates T'_2, intermediate tag is $T_1 + T'_2 \rightarrow$ linearly independent of earlier queries $\rightarrow$ replaced with $T^{(2)}$;
- Encryption query with chunks (B'_1, B_2): generates T'_1, intermediate tag is $T'_1 + T_2 \rightarrow$ linearly independent of earlier queries $\rightarrow$ replaced with $T^{(3)}$;
- Encryption or decryption query with chunk (B'_1, B'_2): intermediate tag is $T'_1 + T'_2 \rightarrow$ dependent on prior queries $\rightarrow$ replaced with $T^{(1)} + T^{(2)} + T^{(3)}$.

This game hop replaces the uniform sampling of a vector X and multiplication by an invertible matrix A to get $Y = AX$ with directly sampling Y uniformly. Thus, the distance between the original game and $\mathbb{G}$ is zero in both nonce-respecting and nonce-misuse settings.

Remember that the adversary considered is orderly, which implies that a decryption query can only be dependent on past encryption queries. Thus, in the nonce-respecting setting with game $\mathbb{G}$, a decryption query has its intermediate tag sampled uniformly at random at the time of the query. We can now argue security for each family member in game $\mathbb{G}$, and will assume in the following that the encryption queries give the intermediate tags to the adversary.

id_+ and $\mathcal{BC}_+$. In the case of $\mathcal{BC}_+$, we can, out of generosity, assume the adversary is given the key. This reduces the scenario to applying a public permutation, making it equivalent to the id_+ case. The probability to forge a randomly generated tag is upper bounded by $\frac{q_D}{2^t}$.

$\mathcal{NBM}_+$. The reasoning is similar to the one adopted for the general case. Indeed, a decryption query has its intermediate tag either colliding with the (necessarily unique) decryption query that has the same nonce, or forms a new input to the NBMAC. Therefore,

$$\mathbf{Adv}^{\mathrm{auth}}_{\widetilde{\Upsilon}^{\mathcal{N}}_{\mathcal{NBM}_+}[\tau\$\mathtt{AEAD}]}(\mathcal{A}) \leq (*_{\mathcal{NBM}})\,.$$

$\mathcal{RO}_+$. The case of $\mathcal{RO}_+$ differs slightly due to the adversary's ability to exploit multicollisions across random oracle calls. Let $\mathcal{Q}_{\mathcal{RO}}$ be the set of distinct inputs provided to the random oracle, either through direct adversarial queries or encryption queries. Define the random variable $\Theta_{\max}$ to represent the size of the largest multicollision observed among the random oracle calls. More formally,

$$\Theta_{\max} = \max_{Z \in \{0,1\}^{t'}} \#\left\{X \in \mathcal{Q}_{\mathcal{RO}} \mid \mathcal{RO}(X) = Z\right\}\,.$$

This corresponds exactly to the bin-and-ball experiment described in Lemma 1, allowing us to state:

$$\mathsf{E}\left(\Theta_{\max}\right) \leq \mathrm{mCol}(q_E + q_O, 2^{t'}) \leq \frac{2\left(q_E + q_O\right)}{2^{t'}} + 3t' + 4\,.$$

Now, consider a decryption query. Its intermediate tag T is sampled uniformly at random. We have two possibilities:

- If T does not collide with any prior random oracle input, then $\mathcal{RO}(T)$ is generated uniformly at random at the time of the decryption query. The probability that this case occurs and that the adversary successfully forges is of at most $\frac{1}{2^{t'}}$.
- If T collides with a previous random oracle input, then the set of possible candidates for forgeries consists of all values $\mathcal{RO}(X)$, for $X \in \mathcal{Q}_{\mathcal{RO}}$. Conditioned on $\Theta_{\max} = \theta_{\max}$, at most $\theta_{\max}$ inputs map any given output of the random oracle. Therefore, the probability that this case occurs and that the adversary successfully forges is of at most $\frac{\theta_{\max}}{2^t}$.

Therefore,

$$\mathbf{Adv}^{\mathrm{auth}}_{\mathfrak{T}^{\aleph}_{\mathcal{RO}_+}(\tau\$\mathtt{AEAD})}(\mathcal{A}) = \sum_{\theta_{\max}=1}^{q_E+q_O} \mathbf{Pr}\left(\mathcal{A} \text{ forges} \mid \Theta_{\max} = \theta_{\max}\right) \mathbf{Pr}\left(\Theta_{\max} = \theta_{\max}\right)$$
$$\leq \frac{q_D}{2^{t'}} + \frac{q_D \mathsf{E}\left(\Theta_{\max}\right)}{2^t} \leq \frac{q_D}{2^{t'}} + \frac{2q_D(q_E+q_O)}{2^{t+t'}} + \frac{(3t'+4)q_D}{2^t}\,. \qquad (*_{\mathcal{RO}})$$

PRF_+. The reasoning for PRF+ follows the same principle, with the key distinction being that the adversary does not have oracle access to the PRF. Nevertheless, the adversary can still exploit multicollisions among the q_E encryption queries. As a result, we obtain

$$\mathbf{Adv}^{\mathrm{auth}}_{\mathfrak{T}^{\aleph}_{\mathrm{PRF}_+}(\tau\$\mathtt{AEAD})}(\mathcal{A}) \leq \frac{q_D}{2^{t'}} + \frac{2q_Dq_E}{2^{t+t'}} + \frac{(3t'+4)q_D}{2^t}\,. \qquad (*_{\mathrm{PRF}})$$

Remark 1. The same upper bounds can be derived for the $\cdot_{\|}$ family members, provided that at least t bits of the intermediate tags of the decryption query are uniformly generated.

Remark 2. If instead of a PRF, a nonce-based PRF is used, the reasoning for $\mathcal{NBM}_+$ applies, so that the multicollision term disappears. In that case, when considering the nonce-misuse setting, the term q_E appearing due to multicollisions would be replaced by q_t, the number of queries reusing a nonce.

Single Nonce plus Split. For all the finalization functions, security follows from security in the single nonce and (5).

6.2 Nonce-Misuse Setting

We denote by q_t the number of queries where a nonce is reused, and by σ_t the total number of chunks associated to nonce-misuse queries.

General Case. As for the nonce-respecting setting, this is still insecure.

Single Nonce.

id_+. This scheme is insecure. Consider the following attack:

- Make an encryption query with input $(\aleph, (A_1, A_2), (P_1, P_2)) \rightarrow$ obtain ciphertexts (C_1, C_2), tag $T_1 + T_2$, denote by $T^{(1)}$ the obtained tag;
- Make an encryption query with input $(\aleph, (A_1, \bar{A}_2), (P_1, \bar{P}_2)) \rightarrow$ obtain ciphertexts $(C_1, \bar{C}_2)$, tag $T_1 + \bar{T}_2$, denote by $T^{(2)}$ the obtained tag;
- Make an encryption query with input $(\aleph, (\bar{A}_1, A_2), (\bar{P}_1, P_2)) \rightarrow$ obtain ciphertexts $(\bar{C}_1, C_2)$, tag $\bar{T}_1 + T_2$, denote by $T^{(3)}$ the obtained tag;
- We can forge for $(\aleph, (\bar{A}_1, \bar{A}_2), (\bar{C}_1, \bar{C}_2))$ with a tag $T^{(1)} + T^{(2)} + T^{(3)}$.

Note that this attack applies as well to the single nonce plus split case.

PRF_+. We again use the sampling technique from $\mathbb{G}$ (cf., beginning of Section 6.1), so that the intermediate tags have form $T^{(i)}$. They are either directly sampled at random, or sum of past queries. We define $\mathbf{ColM}_+$ as follows:

- Either there exists four queries, two of them with a nonce $\aleph$, with intermediate tags $T^{(i_1)}, T^{(i_2)}$ two others with another nonce $\aleph'$, with intermediate tags $T^{(j_1)}, T^{(j_2)}$, such that $T^{(i_1)} = T^{(j_1)}$ and $T^{(i_2)} = T^{(j_2)}$.
- Or there exist two queries with the same nonce $\aleph$, with intermediate tags $T^{(i)}, T^{(j)}$ such that $T^{(i)} = T^{(j)}$.

This bad event is set if the adversary observes collisions with intermediate tags that correspond, at the moment of the triggering query, to nonce-reuse queries. Moreover, even if the input is a linear combination of past prior queries, it remains uniformly random and hidden from the adversary. Therefore,

$$\mathbf{Pr}\left(\mathbf{ColM}_+\right) \leq \frac{q_t(q_t+1)}{2^{t+1}}\,.$$

Consider a decryption query, and assume that $\neg\mathbf{ColM}_+$ holds. We do a case distinction:

- If the game $\mathbb{G}$ would sample a new intermediate tag uniformly at random (i.e., the decryption query uses a new nonce, is a truncated version of a prior nonce-respecting query, or reuses a nonce from a past encryption query, but $\mathbb{G}$ would still sample a new intermediate tag): the analysis boils down to the nonce-respecting setting;
- Otherwise, we are left with the case where the decryption query reuses old chunks from a prior nonce-repeating query. It is thus of the form $T^{(i_1)} + \cdots + T^{(i_D)}$, where $D \geq 2$, and all $T^{(i)}$'s corresponds to the same nonce. In that case, apart from guessing the random values, the only way for the adversary to forge is to exploit collisions. For instance, assume that it made the following encryption queries:
  - One with input $B_1 \rightarrow$ intermediate tag $T^{(1)}$;
  - One with input $B_1' \rightarrow$ intermediate tag $T^{(2)}$;
  - One with input $(B_1, B_2) \rightarrow$ intermediate tag $T^{(3)}$;
  - One with input $(B_1, B_2') \rightarrow$ intermediate tag $T^{(4)}$;

  where $T^{(3)} = T^{(4)}$. In that case, the adversary can make a query with input (B_1', B_2) get the tag $T = \mathrm{PRF}(T^{(1)} + T^{(2)} + T^{(3)})$, and deduce a forgery for (B_1', B_2') with the tag T. When the linear combinations are more complex, it might be that the collisions include linear combinations of intermediate tags, but in any case the adversary must have made the specific associated queries to identify collisions. This possibility is prevented by $\neg\mathbf{ColM}_+$. Therefore, the input and output of the decryption query are uniformly random and hidden from the adversary.

Therefore,

$$\mathbf{Adv}^{\mathrm{auth}}_{\mathfrak{T}^{\aleph}_{\mathrm{PRF}_+}[\mathtt{T\$AEAD}]}(\mathcal{A}) \leq \frac{q_t(q_t+1)}{2^{t+1}} + (*_{\mathrm{PRF}})\,.$$

$\mathcal{RO}_+$. We will reuse the bad event $\mathbf{ColM}_+$ from the prior case. On top of this, we define **GuessM** as follows: there exists a RO query that collides with an intermediate tag, itself coming from a query known to be nonce-misuse.[5] We have

$$\mathbf{Pr}\left(\mathbf{GuessM} \vee \mathbf{ColM}_+\right) \leq \frac{q_t q_O}{2^t} + \frac{q_t(q_t+1)}{2^{t+1}}\,.$$

Now, consider a decryption query and assume $\neg\mathbf{GuessM} \wedge \neg\mathbf{ColM}_+$. We do a case distinction, as before:

- If the game $\mathbb{G}$ samples a new intermediate tag at this point, the analysis boils down to the nonce-respecting setting;
- Otherwise, we are left with the case where the decryption query reuses old chunks from a prior nonce-repeating query. It is thus of form $T^{(i_1)} + \cdots + T^{(i_D)}$, where $D \geq 2$, and all $T^{(i)}$'s corresponds to the same nonce. Thanks to $\neg\mathbf{GuessM}$, the adversary knows via RO calls at most one of the partial intermediate tags. The remaining values remain uniformly random and hidden from the adversary, thus this is also the case of the sum $T^{(i_1)} + \cdots + T^{(i_D)}$. Therefore, the reasoning that we derived for PRF_+, single nonce still holds.

Therefore,

$$\mathbf{Adv}^{\mathrm{auth}}_{\mathfrak{T}^{\aleph}_{\mathcal{RO}_+}(\tau\$\mathtt{AEAD})}(\mathcal{A}) \leq \frac{q_t q_O}{2^t} + \frac{q_t(q_t+1)}{2^{t+1}} + (*_{\mathcal{RO}})\,.$$

$\mathcal{BC}_+$. The analysis is similar to the one of PRF_+. The only difference is that, in the case where the decryption query is a linear combination from prior nonce-misuse queries, and assuming $\neg\mathbf{ColM}_+$, the input is not known to collide with a past nonce-misuse encryption query. Therefore, the output of each decryption query is sampled uniformly at random in a set of size at least $2^t - q_t$, so that

$$\mathbf{Adv}^{\mathrm{auth}}_{\mathfrak{T}^{\aleph}_{\mathcal{BC}_+}(\tau\$\mathtt{AEAD})}(\mathcal{A}) \leq \frac{2q_D}{2^t - q_t} + \frac{q_t(q_t+1)}{2^{t+1}}\,.$$

Single Nonce plus Split. id_+ is also insecure because of the aforementioned attack. Security of the other schemes follows from the single nonce case.

7 Authenticity Bounds for the $\cdot_{\|}$ Family

7.1 Nonce-Respecting Setting

General Case, Chunkwise Nonce-Respecting. We focus on the chunkwise nonce-respecting case.

[5] To be clear, if the adversary guesses a certain input of a nonce-respecting query with nonce $\aleph$, but later $\aleph$ is repeated, this does not set **GuessM**.

$id_{\|}$. It is insecure, due to re-ordering or truncation.

$\mathcal{NBM}_{\|}$. The reasoning is similar to the one adopted for $\mathcal{NBM}_{+}$, so that

$$\mathbf{Adv}^{\text{auth}}_{\mathfrak{T}_{\mathcal{NBM}_{\|}}[\tau\$\,\mathtt{AEAD}]}(\mathcal{A}) \leq (*_{\mathcal{NBM}})\,.$$

$\text{PRF}_{\|}$. With the function $\text{PRF}_{\|}$, an adversary can only obtain information on the values of the intermediate tags by observing collisions on the tag T. Let us now at the beginning consider a restricted case, where we only allow at least two chunk queries, meaning that $l \geq 2$. Let us then define the bad event **Col2** that we observe a collision in T, because the concatenation of the partial intermediate tags T_i collided. Since we have $l \geq 2$, we get

$$\mathbf{Pr}\,(\mathbf{Col2}) \leq \frac{q_E(q_E-1)}{2^{2t+1}}\,.$$

Note that this bad event can be directly exploited in an attack. If the adversary finds a collision[6] between the intermediate tags of two queries with respective chunks (B_1, B_2) and (B'_1, B'_2), it can swap two chunks (e.g., replace B_1 with B'_1) while obtaining the same tag.

If we now assume $\neg$**Col2**, all encryption query inputs to the PRF are randomly drawn, and unknown to the adversary. A decryption query may reuse only old chunks, but these are kept hidden from the adversary. Hence, each decryption query an attacker makes is a combination of unknown random values and thus, we get

$$\mathbf{Adv}^{\text{auth}}_{\mathfrak{T}_{\text{PRF}_{\|}}[\tau\$\,\mathtt{AEAD}]|l\geq 2}(\mathcal{A}) \leq \frac{q_E(q_E-1)}{2^{2t+1}} + (*_{\text{PRF}})\,.$$

Now let us allow single-chunk queries and especially, assume that such encryption queries exist with internal collisions on the partial intermediate tag T_1. Assuming an adversary makes single-chunk decryption queries, they have to be different from encryption by definition and knowledge from previous encryption queries play no role. Using information of single-chunk encryption queries in multi-chunk decryption queries does not provide an advantage for the attacker, since still, the partial intermediate tags T_i of multi-chunk encryption queries are random and unknown. Also single-chunk encryption queries cannot be reused in other multi-chunk encryption queries since we are in the (chunkwise) nonce-respecting setting. Hence, we get

$$\mathbf{Adv}^{\text{auth}}_{\mathfrak{T}_{\text{PRF}_{\|}}[\tau\$\,\mathtt{AEAD}]}(\mathcal{A}) \leq \frac{q_E(q_E-1)}{2^{2t+1}} + (*_{\text{PRF}})\,.$$

$\mathcal{RO}_{\|}$. Contrarily to the $\mathcal{RO}_{+}$ setting where we could reduce to a scenario in which decryption queries generate uniformly random tags, this is not the case here. The adversary might guess RO inputs with single-chunk queries to create a forgery. Consider the following attack:

[6] Assuming t' is not too small compared to t, false positives can be filtered out.

– Make q_O queries to the random oracle with random inputs;
– Make q_E encryption queries, each with a single-chunk, and pairwise distinct nonces;
– With probability $\approx \left(\frac{q_E q_O}{2^t}\right)^2$, there exist two distinct RO queries that each collide with an encryption query, with respective chunks B_1, B_2 and intermediate tags T_1, T_2. The adversary can observe this collision[7], recover B_1, B_2, T_1 and T_2, and then forge for the input (B_1, B_2) with tag $\mathcal{RO}(T_1 \| T_2)$.

Therefore, besides the bad event **Col2** from the $\mathrm{PRF}_{\|}$ case, we define **Guess** as follows: there exists a random oracle query that collides with an intermediate tag. We have

$$\mathbf{Pr}\left(\mathbf{Guess} \vee \mathbf{Col2}\right) \leq \frac{q_E q_O}{2^t} + \frac{q_E(q_E-1)}{2^{2t+1}}\,.$$

Assuming $\neg\mathbf{Guess} \wedge \neg\mathbf{Col2}$, the direct queries to the random oracle from the adversary do not leak any information about the partial intermediate tags, and the adversary cannot exploit collisions between the encryption queries to forge, as discussed for the $\mathrm{PRF}_{\|}$ case. Therefore,

$$\mathbf{Adv}^{\mathrm{auth}}_{\mathfrak{T}_{\mathcal{RO}_{\|}}[\mathtt{T\$AEAD}]}(\mathcal{A}) \leq \frac{q_E q_O}{2^t} + \frac{q_E(q_E-1)}{2^{2t+1}} + (*_{\mathcal{RO}})\,.$$

Single Nonce. A single nonce adversary is necessarily (chunkwise) nonce-respecting. $id_{\|}$ is still insecure due to truncation.

$\mathcal{NBM}_{\|}$. Again, the reasoning is similar to the one adopted for $\mathcal{NBM}_{+}$, and we have

$$\mathbf{Adv}^{\mathrm{auth}}_{\mathfrak{T}^{\mathrm{N}}_{\mathcal{NBM}_{\|}}[\mathtt{T\$AEAD}]}(\mathcal{A}) \leq (*_{\mathcal{NBM}})\,.$$

$\mathrm{PRF}_{\|}$ *and* $\mathcal{RO}_{\|}$. For both $\mathcal{RO}_{\|}$ and $\mathrm{PRF}_{\|}$ we can refine the analysis compared to the general case. Indeed, the attacks for $\mathcal{RO}_{\|}$ and $\mathrm{PRF}_{\|}$ in the general case rely on exploiting collisions or guessing encryption queries with different nonces, and then combining or swapping some chunks in a subsequent decryption query, which cannot be done in the single nonce, nonce-respecting setting.

$\mathrm{PRF}_{\|}$. We have several possibilities for a decryption query:

– If the decryption query repeats the nonce of a prior query, and is truncated: it is of the form $(B_1, \ldots, B_l)$ with partial intermediate tags $T_1, \ldots, T_l$, and there exists a (necessarily unique) prior encryption query of form $(B_1, \ldots, B_L)$ with partial intermediate tags $T_1, \ldots, T_L$ and $L > l$. In that case, the only information on the partial intermediate tags that the adversary can infer is whether $T_1 \| \cdots \| T_L$ collides with another (unknown) $T'_1 \| \cdots \| T'_L$, associated with chunks $(B'_1, \ldots, B'_L)$, but this information is not helpful. Without

[7] Assuming t' is not too small compared to t, false positives can be filtered out.

access to the intermediate tags nor the PRF, the values of $T_1\|\cdots\|T_l$ and $\mathrm{PRF}(T_1\|\cdots\|T_l)$ remain random and unknown to the adversary, and we can repeat the analysis made with PRF_+, single nonce setting;
- If the decryption query repeats the nonce of a prior query and contains at least one new chunk: at least t bits of the intermediate tag is generated uniformly at random at the moment of the decryption query. Therefore, this case also reduces to PRF_+, single nonce setting;
- Otherwise: necessarily the nonce is new, all the intermediate tags are sampled uniformly at random, thus this case also reduces to PRF_+, single nonce setting.

Therefore,

$$\mathbf{Adv}^{\mathrm{auth}}_{\mathfrak{T}^{\mathsf{N}}_{\mathrm{PRF}_\|}[\tau\$\mathtt{AEAD}]}(\mathcal{A}) \leq (*_{\mathrm{PRF}})\,.$$

$\mathcal{RO}_\|$. The attack for $\mathcal{RO}_\|$, general case with success probability $\approx \left(\frac{q_E q_O}{2^t}\right)^2$ involves guessing two independent single-chunk encryption queries, and combine them later in a decryption query. However, this strategy fails in the nonce-respecting and single nonce setting. Indeed, if the adversary guesses a single-chunk query, then any new decryption query reusing that chunk has the others partial intermediate tags sampled uniformly at random at the moment of the decryption query. Therefore, in order to exploit guessed inputs via RO calls, the adversary must guess a RO input from encryption queries with at least two chunks. Denote this bad event by **Guess2**. Since the chunks for each encryption query are generated uniformly at random, we have

$$\mathbf{Pr}\,(\mathbf{Guess2}) \leq \frac{q_E q_O}{2^{2t}}\,.$$

Assuming $\neg$**Guess2**, we have several possibilities for a decryption query:

- If the decryption query repeats the nonce of a prior query, and is truncated: it is of the form $(B_1,\ldots,B_l)$, with partial intermediate tags $T_1,\ldots,T_l$, and there exists a (necessarily unique) prior encryption query of form $(B_1,\ldots,B_L)$ with partial intermediate tags $T_1,\ldots,T_L$ and $L > l$. In that case, by $\neg$**Guess2**, the adversary has not made the query $\mathcal{RO}(T_1\|\cdots\|T_L)$, which therefore means that the adversary does not know the value of $T_1\|\cdots\|T_l$. Even if it has queried $\mathcal{RO}(T_1\|\cdots\|T_l)$, it does not know this. Thus, $T_1\|\cdots\|T_l$, as well as $\mathcal{RO}(T_1\|\cdots\|T_l)$ remain hidden from the adversary, and we can conduct the same analysis as for $\mathrm{PRF}_\|$, single nonce setting;
- If the decryption query repeats the nonce of a prior query and contains at least one new chunk: at least t bits of the intermediate tag is generated uniformly at random at the moment of the decryption query. Therefore, this case reduces to $\mathcal{RO}_+$, single nonce setting;
- Otherwise: necessarily the nonce is new, all the intermediate tags are sampled uniformly at random, thus this case reduces to $\mathcal{RO}_+$, single nonce setting.

Therefore,

$$\mathbf{Adv}^{\mathrm{auth}}_{\mathfrak{T}^{\mathsf{N}}_{\mathcal{RO}_\|}[\tau\$\mathtt{AEAD}]}(\mathcal{A}) \leq \frac{q_E q_O}{2^{2t}} + (*_{\mathcal{RO}})\,.$$

Single Nonce plus Split. For encryption queries, we are in the nonce-respecting setting and hence, every encryption query leads to a new randomly drawn $T_1\|\dots\|T_L$. For decryption queries, those have to be different from encryption queries. Since we are in the single nonce setting, chunks cannot be swapped while leaving the nonce intact. A change in one chunk l leads to a new randomly drawn T_l. Since L is included in the associated data, also truncating the number of chunks leads to a change in all T_l's. Therefore, all the partial intermediate tags are generated uniformly at random at the moment of the decryption query, and we can apply the same analysis as for the $\cdot_+$ family, single nonce setting. Therefore,

$$\mathbf{Adv}^{\mathrm{auth}}_{\mathfrak{T}^{\mathbb{N}L}_{id_\|}[\tau\$\mathtt{AEAD}]}(\mathcal{A}) \le \frac{q_D}{2^t}\,, \qquad \mathbf{Adv}^{\mathrm{auth}}_{\mathfrak{T}^{\mathbb{N}L}_{\mathcal{RO}_\|}[\tau\$\mathtt{AEAD}]}(\mathcal{A}) \le (*_{\mathcal{RO}})\,,$$
$$\mathbf{Adv}^{\mathrm{auth}}_{\mathfrak{T}^{\mathbb{N}L}_{\mathrm{PRF}_\|}[\tau\$\mathtt{AEAD}]}(\mathcal{A}) \le (*_{\mathrm{PRF}})\,, \qquad \mathbf{Adv}^{\mathrm{auth}}_{\mathfrak{T}^{\mathbb{N}L}_{\mathcal{NBM}_\|}[\tau\$\mathtt{AEAD}]}(\mathcal{A}) \le (*_{\mathcal{NBM}})\,.$$

7.2 Nonce-Misuse Setting

General Case.

$id_\|$. It is insecure, due to re-ordering or truncation.

$\mathrm{PRF}_\|$. The security of $\mathrm{PRF}_\|$ drastically degrades as soon as one single query repeating a nonce is allowed. Indeed, the adversary can first make nonce-respecting queries with chunks $\left(B_1^{(i)}\right)_i$, until it observes a collision $T^{(i)} = T^{(j)}$. Assume that this collision occurs between the (partial) intermediate tags. Then the adversary can add a supplementary chunk B_2 to $B_1^{(i)}$, make an encryption query with input $(B_1^{(i)}, B_2)$, and deduce a forgery for $(B_1^{(j)}, B_2)$ using the tag obtained from the earlier query.

Denote by **Col** the event that two partial intermediate tags, *no matter their position* collide together. We have

$$\mathbf{Pr}\left(\mathbf{Col}\right) \le \frac{\sigma_E(\sigma_E-1)}{2^{t+1}}\,.$$

Now, consider a decryption query, and assume $\neg$**Col**:

- If at least one of the partial intermediate tags is freshly generated, the analysis reduces to the nonce-respecting setting;
- Otherwise, then necessarily all chunks come from past encryption queries, thus the input is truncated, or is a mixture of previous encryption queries all having different partial intermediate tags. In all cases, the input presented to the PRF is new.

Therefore,

$$\mathbf{Adv}^{\mathrm{auth}}_{\mathfrak{T}_{\mathrm{PRF}_\|}[\tau\$\mathtt{AEAD}]}(\mathcal{A}) \le \frac{\sigma_E(\sigma_E-1)}{2^{t+1}} + (*_{\mathrm{PRF}})\,.$$

$\mathcal{RO}_{\|}$. Besides **Col**, we use the bad event **Guess** from the chunkwise respecting case of $\mathcal{RO}_{\|}$. We have

$$\mathbf{Pr}\left(\mathbf{Guess} \vee \mathbf{Col}\right) \leq \frac{q_E q_O}{2^t} + \frac{\sigma_E(\sigma_E - 1)}{2^{t+1}}\,.$$

Assuming $\neg$**Guess**, the random oracle queries made by the adversary are not helpful to them, so that the partial intermediate tags remain secret to the adversary. Moreover, as discussed in the previous $\mathrm{PRF}_{\|}$ case, the adversary cannot exploit collisions between encryption queries to forge, so that

$$\mathbf{Adv}^{\mathrm{auth}}_{\mathfrak{T}_{\mathcal{RO}_{\|}}[^{\mathrm{T}\$}\mathtt{AEAD}]}(\mathcal{A}) \leq \frac{q_E q_O}{2^t} + \frac{\sigma_E(\sigma_E - 1)}{2^{t+1}} + (*_{\mathcal{RO}})\,.$$

Single Nonce.

$id_{\|}$. Is still insecure, due to truncation.

$\mathrm{PRF}_{\|}$. In the general case, a collision between two different encryption queries was sufficient to produce forgeries. Not anymore. Indeed, in the single nonce setting, an encryption query is bound to a fixed nonce prefix, limiting substitution possibilities. Let us extend the reasoning that was used in the nonce-respecting setting. Assuming $l \geq 2$, we will upper bound the probability of the bad event **Col2** (from the nonce-respecting case of $\mathrm{PRF}_{\|}$). This bad event can occur in two scenarios:

- The two queries triggering this event use different nonces. Even though nonces are allowed to repeat this time, the intermediate tags are random and stay hidden from the adversary, so the reasoning adopted does not change compared to the nonce-respecting case;
- The two queries involved in the event share the same nonce. Then it might be that chunks repeat across the queries. Still, at least one chunk has to be new so that the adversary only needs to find a collision on one partial intermediate tag.

Therefore,

$$\mathbf{Pr}\left(\mathbf{Col2} \mid l \geq 2\right) \leq \frac{q_t(q_t + 1)}{2^{t+1}} + \frac{q_E(q_E - 1)}{2^{2t+1}}\,.$$

Now, assuming $\neg$**Col2**, once again the adversary is not aware of collisions, thus the intermediate tags are hidden from the adversary. Therefore,

$$\mathbf{Adv}^{\mathrm{auth}}_{\mathfrak{T}^{\mathfrak{N}}_{\mathrm{PRF}_{\|}}[^{\mathrm{T}\$}\mathtt{AEAD}]|l\geq 2}(\mathcal{A}) \leq \frac{q_t(q_t + 1)}{2^{t+1}} + \frac{q_E(q_E - 1)}{2^{2t+1}} + (*_{\mathrm{PRF}})\,.$$

If we additionally allow queries with one chunk, we need to consider additional bad collisions. Notably, only collisions between queries of the same length matter, as these are the only collisions observable by the adversary. We have two possibilities for 1-chunk collisions:

– Collisions between two chunks that are repeating a nonce: this is captured in the bound $\frac{q_t(q_t+1)}{2^{t+1}}$;
– Collisions between two chunks, with different nonces: it does not allow the adversary to later substitute them. If it wants to make the collision useful later, it needs to stretch the collision, but this is prevented by $\neg$**Col2**.

Therefore,

$$\mathbf{Adv}^{\mathrm{auth}}_{\widetilde{\mathfrak{T}}^{\mathrm{N}}_{\mathrm{PRF}_{\|}}(\tau\$\,\mathtt{AEAD})}(\mathcal{A}) \leq \frac{q_t(q_t+1)}{2^t} + \frac{q_E(q_E-1)}{2^{2t+1}} + (*_{\mathrm{PRF}})\,.$$

$\mathcal{RO}_{\|}$. Besides the bad event **Col2**, let us reuse the bad events **Guess2** and **GuessM** from the single nonce case of respectively $\mathcal{RO}_{\|}$ (nonce-respecting setting) and $\mathcal{RO}_{+}$ (nonce-misuse setting). **Guess2**$\vee$**GuessM** is set whenever either (i) the adversary guesses the intermediate tag of a multiple chunk query that is nonce-respecting at the moment when this event is set, or (ii) the adversary guesses an intermediate tag of a query known to be nonce-misuse at the moment when this event is set. An adversary that guesses a 1-chunk nonce-respecting query cannot use this guess to deduce larger queries, unless it reuses the nonce from a previous query to obtain more intermediate tags and guesses them, which would itself trigger **GuessM**. We have

$$\mathbf{Pr}\left(\mathbf{Guess2} \vee \mathbf{GuessM} \vee \mathbf{Col2}\right) \leq \frac{q_t q_O}{2^t} + \frac{q_E q_O}{2^{2t}} + \frac{q_t(q_t+1)}{2^{t+1}} + \frac{q_E(q_E-1)}{2^{2t+1}}\,.$$

Now, consider a decryption query and assume $\neg$**Guess2** $\wedge$ $\neg$**GuessM** $\wedge$ $\neg$**Col2**. We have several possibilities:

– If the decryption query combines chunks from at least two different past queries: then the partial intermediate tag of at least one of them was generated during a query known to be misusing the nonce when it was made. Since $\neg$**GuessM** holds, the associated partial intermediate tag remains hidden from the adversary. As a result, we can apply the same reasoning as $\mathrm{PRF}_{\|}$;
– Else if the decryption query is a truncated version of a prior query: then the reasoning is the same as the nonce-respecting case of $\mathcal{RO}_{\|}$, single nonce setting;
– Otherwise, at least one partial intermediate tag is generated uniformly at the moment of the decryption query, so that this reduces to the nonce-respecting setting.

Therefore,

$$\mathbf{Adv}^{\mathrm{auth}}_{\widetilde{\mathfrak{T}}^{\mathrm{N}}_{\mathcal{RO}_{\|}}(\tau\$\,\mathtt{AEAD})}(\mathcal{A}) \leq \frac{q_t q_O}{2^t} + \frac{q_E q_O}{2^{2t}} + \frac{q_t(q_t+1)}{2^{t+1}} + \frac{q_E(q_E-1)}{2^{2t+1}} + (*_{\mathcal{RO}})\,.$$

Single Nonce plus Split. $id_{\|}$ is insecure, since nonce-reuse allows to substitute queries between them.

$\mathrm{PRF}_{\|}$. Define **ColM** as: there exists a collision between two partial intermediate tags, both known to be misusing the nonce at the moment when the query triggering the event is made. We have

$$\mathbf{Pr}\left(\mathbf{ColM}\right) \leq \frac{q_t(q_t+1)}{2^t}\,.$$

Consider a decryption query, assume $\neg\mathbf{ColM}$ and that the decryption query is using only chunks from past encryption queries (otherwise, security follows). Since the number of chunks L is included in the associated data, this decryption query contains at least two chunks that come from at least two past queries with the same nonce $\aleph$, and past encryption queries with one chunk do not bring any useful information to the adversary. In this scenario, because the intermediate tags remain hidden from the adversary and $\neg\mathbf{ColM}$ holds, the adversary cannot identify a past encryption query whose intermediate tag collides with the one from the decryption query. Therefore,

$$\mathbf{Adv}^{\mathrm{auth}}_{\mathfrak{T}^{\aleph L}_{\mathrm{PRF}_{\|}}[\mathtt{T\$AEAD}]}(\mathcal{A}) \leq \frac{q_t(q_t+1)}{2^t} + (*_{\mathrm{PRF}})\,.$$

$\mathcal{RO}_{\|}$. We use the bad event **ColM** from the case $\mathrm{PRF}_{\|}$. Moreover, define the bad event **Guess2M** as follows: there exists a RO query that collides with a multi-chunk query known to be misusing a nonce at the moment when the event is triggered. In order to set **Guess2M**, the adversary needs to: i) guess the intermediate tag of a query with at least two chunks and ii) guess at least one partial intermediate tag of a query known to misuse a nonce. Therefore,

$$\mathbf{Pr}\left(\mathbf{Guess2M} \vee \mathbf{ColM}\right) \leq \min\left(\frac{2q_Eq_O}{2^{2t}}, \frac{q_tq_O}{2^t}\right) + \frac{q_t(q_t+1)}{2^t}\,.$$

Consider a decryption query, assuming $\neg\mathbf{ColM} \wedge \neg\mathbf{Guess2M}$, where the query has length $l \geq 2$ and uses only chunks from previous encryption queries (otherwise, security follows). Thus, at least one chunk comes from a query of the same length that was known to misuse a nonce when it was made. By $\neg\mathbf{Guess2M}$, the associated partial intermediate tag stays hidden from the adversary, so that we can apply the same reasoning as $\mathrm{PRF}_{\|}$. Therefore,

$$\mathbf{Adv}^{\mathrm{auth}}_{\mathfrak{T}^{\aleph L}_{\mathcal{RO}_{\|}}[\mathtt{T\$AEAD}]}(\mathcal{A}) \leq \min\left(\frac{2q_Eq_O}{2^{2t}}, \frac{q_tq_O}{2^t}\right) + \frac{q_t(q_t+1)}{2^t} + (*_{\mathcal{RO}})\,.$$

8 Conclusion

In this paper, we have analyzed the security of many different ways to parallelize serial authenticated encryption. We were able to show that already embedding a counter in the nonce and summing the tags leads to a secure construction in a nonce-respecting scenario. Just processing the sum of tags further, e.g., with a PRF already gives a secure solution in the nonce-misuse scenario.

Overall, in terms of security benefits, there are only limited scenarios where concatenation is preferable over summing. This occurs either in scenarios where there are no nonce restrictions, or under a very specific set of conditions that allow for a slight security improvement. We think that these observations could draw interesting parallels to the use of authenticated encryption at the protocol level.

Acknowledgements. We would like to thank the anonymous reviewers of ISC as well as Bart Mennink for their valuable comments that helped improve this work. Charlotte Lefevre is supported by the France 2030 program under grant agreement No. ANR-22-PECY-0010.

References

1. Albertini, A., Duong, T., Gueron, S., Kölbl, S., Luykx, A., Schmieg, S.: How to Abuse and Fix Authenticated Encryption Without Key Commitment. In: Butler, K.R.B., Thomas, K. (eds.) 31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022. pp. 3291–3308. USENIX Association (2022), https://www.usenix.org/conference/usenixsecurity22/presentation/albertini
2. Andreeva, E., Bilgin, B., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., Yasuda, K.: APE: authenticated permutation-based encryption for lightweight cryptography. In: Cid, C., Rechberger, C. (eds.) Fast Software Encryption - 21st International Workshop, FSE 2014, London, UK, March 3-5, 2014. Revised Selected Papers. pp. 168–186. Lecture Notes in Computer Science, Springer (2014). https://doi.org/10.1007/978-3-662-46706-0_9, https://doi.org/10.1007/978-3-662-46706-0_9
3. Aumasson, J., Jovanovic, P., Neves, S.: NORX v3.0. Submission to CAESAR Competition (2016)
4. Bellare, M., Hoang, V.T.: Efficient Schemes for Committing Authenticated Encryption. In: Dunkelman, O., Dziembowski, S. (eds.) Advances in Cryptology - EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30 - June 3, 2022, Proceedings, Part II. Lecture Notes in Computer Science, vol. 13276, pp. 845–875. Springer (2022). https://doi.org/10.1007/978-3-031-07085-3_29, https://doi.org/10.1007/978-3-031-07085-3_29
5. Bellare, M., Hoang, V.T.: Efficient Schemes for Committing Authenticated Encryption. IACR Cryptol. ePrint Arch. p. 268 (2022), https://eprint.iacr.org/2022/268
6. Bellare, M., Namprempre, C.: Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. J. Cryptol. **21**(4), 469–491 (2008). https://doi.org/10.1007/S00145-008-9026-X, https://doi.org/10.1007/s00145-008-9026-x
7. Bellare, M., Ng, R., Tackmann, B.: Nonces Are Noticed: AEAD Revisited. In: Boldyreva, A., Micciancio, D. (eds.) Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I. Lecture Notes in Computer Science, vol. 11692, pp. 235–265. Springer (2019). https://doi.org/10.1007/978-3-030-26948-7_9, https://doi.org/10.1007/978-3-030-26948-7_9

8. Bellare, M., Rogaway, P.: Encode-Then-Encipher Encryption: How to Exploit Nonces or Redundancy in Plaintexts for Efficient Cryptography. In: Okamoto, T. (ed.) Advances in Cryptology - ASIACRYPT 2000, 6th International Conference on the Theory and Application of Cryptology and Information Security, Kyoto, Japan, December 3-7, 2000, Proceedings. Lecture Notes in Computer Science, vol. 1976, pp. 317–330. Springer (2000). https://doi.org/10.1007/3-540-44448-3_24, https://doi.org/10.1007/3-540-44448-3_24
9. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Cryptographic sponge functions (2011), https://keccak.team/files/CSF-0.1.pdf, https://keccak.team/files/CSF-0.1.pdf
10. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications. In: Miri, A., Vaudenay, S. (eds.) Selected Areas in Cryptography - 18th International Workshop, SAC 2011, Toronto, ON, Canada, August 11-12, 2011, Revised Selected Papers. Lecture Notes in Computer Science, vol. 7118, pp. 320–337. Springer (2011). https://doi.org/10.1007/978-3-642-28496-0_19, https://doi.org/10.1007/978-3-642-28496-0_19
11. Bose, P., Hoang, V.T., Tessaro, S.: Revisiting AES-GCM-SIV: Multi-user Security, Faster Key Derivation, and Better Bounds. In: Nielsen, J.B., Rijmen, V. (eds.) Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part I. Lecture Notes in Computer Science, vol. 10820, pp. 468–499. Springer (2018). https://doi.org/10.1007/978-3-319-78381-9_18, https://doi.org/10.1007/978-3-319-78381-9_18
12. CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness (May 2014), http://competitions.cr.yp.to/caesar.html
13. Choi, W., Lee, B., Lee, J.: Indifferentiability of Truncated Random Permutations. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part I. Lecture Notes in Computer Science, vol. 11921, pp. 175–195. Springer (2019). https://doi.org/10.1007/978-3-030-34578-5_7, https://doi.org/10.1007/978-3-030-34578-5_7
14. Daemen, J., Rijmen, V.: The Design of Rijndael - The Advanced Encryption Standard (AES), Second Edition. Information Security and Cryptography, Springer (2020). https://doi.org/10.1007/978-3-662-60769-5, https://doi.org/10.1007/978-3-662-60769-5
15. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Ascon v1.2: Lightweight Authenticated Encryption and Hashing. J. Cryptol. **34**(3), 33 (2021). https://doi.org/10.1007/S00145-021-09398-9, https://doi.org/10.1007/s00145-021-09398-9
16. Endignoux, G., Vizár, D.: Linking Online Misuse-Resistant Authenticated Encryption and Blockwise Attack Models. IACR Trans. Symmetric Cryptol. **2016**(2), 125–144 (2016). https://doi.org/10.13154/TOSC.V2016.I2.125-144, https://doi.org/10.13154/tosc.v2016.i2.125-144
17. Gligoroski, D., Mihajloska, H., Samardjiska, S., Jacobsen, H., El-Hadedy, M., Jensen, R.E., Otte, D.: π-Cipher v2.0. Submission to CAESAR Competition (2015)
18. Jovanovic, P., Luykx, A., Mennink, B.: Beyond 2 c/2 Security in Sponge-Based Authenticated Encryption Modes. In: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the The-

ory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I. Lecture Notes in Computer Science, vol. 8873, pp. 85–104. Springer (2014). https://doi.org/10.1007/978-3-662-45611-8_5, https://doi.org/10.1007/978-3-662-45611-8_5
19. Jovanovic, P., Luykx, A., Mennink, B., Sasaki, Y., Yasuda, K.: Beyond Conventional Security in Sponge-Based Authenticated Encryption Modes. J. Cryptol. **32**(3), 895–940 (2019). https://doi.org/10.1007/S00145-018-9299-7, https://doi.org/10.1007/s00145-018-9299-7
20. Krawczyk, H.: LFSR-based Hashing and Authentication. In: Desmedt, Y. (ed.) Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings. Lecture Notes in Computer Science, vol. 839, pp. 129–139. Springer (1994). https://doi.org/10.1007/3-540-48658-5_15, https://doi.org/10.1007/3-540-48658-5_15
21. Lefevre, C., Mennink, B.: Generic Security of the Ascon Mode: On the Power of Key Blinding. In: Maria Eichlseder, S.G. (ed.) Selected Areas in Cryptography - SAC 2024 - 31th International Conference, Montreal, Canada, August 28-30, 2024, Revised Selected Papers. Lecture Notes in Computer Science, Springer (2024)
22. McGrew, D.A., Viega, J.: The Security and Performance of the Galois/Counter Mode (GCM) of Operation. In: Canteaut, A., Viswanathan, K. (eds.) Progress in Cryptology - INDOCRYPT 2004, 5th International Conference on Cryptology in India, Chennai, India, December 20-22, 2004, Proceedings. Lecture Notes in Computer Science, vol. 3348, pp. 343–355. Springer (2004). https://doi.org/10.1007/978-3-540-30556-9_27, https://doi.org/10.1007/978-3-540-30556-9_27
23. Morawiecki, P., Gaj, K., Homsirikamol, E., Matusiewicz, K., Pieprzyk, J., Rogawski, M., Srebrny, M., Wójcik, M.: ICEPOLE v2.0. Submission to CAESAR Competition (2015)
24. Morawiecki, P., Pieprzyk, J.: Parallel authenticated encryption with the duplex construction. Cryptology ePrint Archive (2013)
25. Naito, Y., Sasaki, Y., Sugawara, T.: Committing Security of Ascon: Cryptanalysis on Primitive and Proof on Mode. IACR Trans. Symmetric Cryptol. **2023**(4), 420–451 (2023). https://doi.org/10.46586/TOSC.V2023.I4.420-451, https://doi.org/10.46586/tosc.v2023.i4.420-451
26. Sönmez Turan, M., McKay, K., Kang, J., Kelsey, J., Chang, D.: Ascon-based lightweight cryptography standards for constrained devices: Authenticated encryption, hash, and extendable output functions. NIST Special Publication NIST SP 800-232, National Institute of Standards and Technology (NIST) (August 2025). https://doi.org/10.6028/NIST.SP.800-232, https://doi.org/10.6028/NIST.SP.800-232
27. Wegman, M.N., Carter, L.: New Hash Functions and Their Use in Authentication and Set Equality. J. Comput. Syst. Sci. **22**(3), 265–279 (1981). https://doi.org/10.1016/0022-0000(81)90033-7, https://doi.org/10.1016/0022-0000(81)90033-7

Supplementary Material

A Relation Between Our Requirement and PR-Tag Security

We now study the relation between the notion defined in (4) and PR-tag security. Let $\mathcal{A}$ be an adversary that makes q_D decryption queries and never repeats a decryption query for which it already knows the answer thanks to a prior encryption query. We have

$$\begin{aligned}
(4) &= \Delta_{\mathcal{A}}\left(\left[\left(\mathtt{AEAD}^p_{K_m}\right)_{m=1}^{\mu}, p\right]; \left[\left(\mathtt{T\$}\,\mathtt{AEAD}^p_{K_m}\right)_{m=1}^{\mu}, p\right]\right) \\
&\le \Delta_{\mathcal{A}}\left(\left[\left(\mathtt{AEAD}^p_{K_m}\right)_{m=1}^{\mu}, p\right]; \left[\left(\mathtt{AEenc}^p_{K_m}, \bot\right)_{m=1}^{\mu}, p\right]\right) + \\
&\quad \Delta_{\mathcal{A}}\left(\left[\left(\mathtt{AEenc}^p_{K_m}, \bot\right)_{m=1}^{\mu}, p\right]; \left[\left(\mathtt{T\$}\,\mathtt{AEenc}^p_{K_m}, \bot\right)_{m=1}^{\mu}, p\right]\right) + \\
&\quad \Delta_{\mathcal{A}}\left(\left[\left(\mathtt{T\$}\,\mathtt{AEenc}^p_{K_m}, \bot\right)_{m=1}^{\mu}, p\right]; \left[\left(\mathtt{T\$}\,\mathtt{AEAD}^p_{K_m}\right)_{m=1}^{\mu}, p\right]\right) \\
&\le \Delta_{\mathcal{A}}\left(\left[\left(\mathtt{AEAD}^p_{K_m}\right)_{m=1}^{\mu}, p\right]; \left[\left(\mathtt{AEenc}^p_{K_m}, \bot\right)_{m=1}^{\mu}, p\right]\right) + \\
&\quad \Delta_{\mathcal{A}}\left(\left[\left(\mathtt{AEenc}^p_{K_m}, \bot\right)_{m=1}^{\mu}, p\right]; \left[\left(\mathtt{T\$}\,\mathtt{AEenc}^p_{K_m}, \bot\right)_{m=1}^{\mu}, p\right]\right) + \frac{q_D}{2^t}\,,
\end{aligned}$$

where we used that the third distance is upper-bounded by the probability of successfully guessing a random tag. The first distance corresponds to nonce-misuse authenticity, while the second corresponds to PR-tag.

PR-tag security alone is insufficient to guarantee nonce-misuse authenticity when using our combiners. As a toy example, consider an AEAD that behaves ideally for all inputs except when the plaintext is the key: we set $\mathtt{AEenc}^p_K(N, A, K) = (0^{|K|}, T_0 \oplus T_1)$, where T_X is the tag produced by $\mathtt{AEenc}^p_K(N, A, X)$. This AEAD is secure in the nonce-respecting setting and satisfies PR-tag security, yet it is completely insecure when used with our combiners.