AsconAEAD128 Revisited in the Multi-user Setting

Bishwajit Chakraborty[1], Mridul Nandi[2], Soumit Pal[2],
Thomas Peyrin[1], Quan Quan Tan[1]

[1] Nanyang Technological University, Singapore
{bishwajit.chakrabort@,thomas.peyrin@,quaanquan001@e.}ntu.edu.sg
[2] Indian Statistical Institute, Kolkata, India
{mridul.nandi, soumitpal378 }@gmail.com

Abstract. After more than half a decade since its initiation, NIST declared Ascon as the winner of the LwC competition. In the first public draft of AsconAEAD128, NIST recognized that Ascon has limitations when used in multi-user applications. To mitigate this, NIST prescribed the use of a 256-bit key in multi-user applications and produced an instantiation on how to process this extra key size in the current AsconAEAD128 API. While doing so, they identified a limitation of this new scheme (which we refer to as mu-Ascon in this document): mu-Ascon is vulnerable to committing attack and hence cannot be used in cases where committing security is required. On the other hand, the full key-binding property in Ascon, which separated it from other sponge-type constructions, has been used to show that Ascon is much stronger in the sense that it presents a key recovery resistance even in the case where some intermediate state is recovered. We remark that the current mu-Ascon has the limitation that only a partial key is bound during initialization and finalization. In this work, we propose some alternative instantiations of AsconAEAD128 API for multi-user applications. In comparison with the current mu-Ascon proposal, our first construction Ascon-256.v2 guarantees CMT-4 committing security up to 64 bits, and our second construction Ascon-256.v3 leads to both CMT-4 committing security and full 256-bit key binding. Structurally, our instantiations use only an extra-permutation call to provide these extra security features compared to mu-Ascon, which has a negligible overhead in terms of performance (given the lightweight nature of the Ascon permutation).

Keywords: Ascon, Multi-user Security, 256-bit Key, AEAD , Tight Security, Lightweight Cryptography

1 Introduction

Authenticated Encryption with Associated Data (AEAD) schemes are a fundamental class of symmetric-key encryption schemes that have been extensively studied. Over the past two decades, the demand for AEAD schemes suitable for

lightweight devices without compromising security has increased dramatically. To address this need, in 2018, the National Institute of Standards and Technology (NIST) issued a call for proposals [18] (known as the NIST Lightweight Cryptography (LwC) competition) for a lightweight AEAD scheme suitable for standardization. The first round received approximately 56 submissions, and after nearly half a decade of rigorous research and evaluation, NIST selected the Ascon-128a protocol as the winner. In [22], NIST renamed the scheme to AsconAEAD128 and published an initial public draft of it for comment. Ascon's security has been extensively analyzed for over a decade since its participation in the CAESAR competition, where it also won in the resource-constrained use case.

1.1 Existing Analysis on Ascon

In the provable security domain, most studies on Ascon have focused on conventional AEAD security, namely privacy and authenticity. Initially, Ascon was primarily considered a Duplex-type construction. Chakraborty et al. [2] and Lefevre et al. [11] independently were among the first to treat Ascon AEAD as a dedicated mode when proving security. They demonstrated that the double key binding present in both the initialization and finalization states provides Ascon AEAD with significantly stronger security compared to the Duplex construction. The authors in [2] showed that these additional key bindings provide Ascon AEAD security up to $D \ll 2^c, T \ll 2^c$ in the ideal permutation model, where D and T represent the data and time complexity of the attacker, respectively, and c is the rate size. Independently, Lefevre et al. [11] showed that, unlike the general Duplex construction, Ascon maintains authenticity security even under state-recovery attacks; that is, recovering some intermediate state does not lead to forgery or key recovery. They derived a tight security bound against forgery under state-recovery in nonce-misuse settings. Furthermore, they provided bounds for Ascon in the multi-user setting, which were later improved by Chakraborty et al. [3]. A dominant term appearing in both [11] and [3] in the multi-user setting is of the form $\mu T/2^\kappa$, where μ is the number of users and κ is the key size. Since the key size of both Ascon-128 and Ascon-128a is 128 bits, achieving a security level of $T = 2^{112}$ does not allow for a large number of users (μ). To address this limitation, the authors in [3] proposed a 256-bit key variant of Ascon, named Ascon-256, and demonstrated its security even with a large number of users.

Besides conventional AEAD security, modern AEAD schemes offer additional security guarantees such as related-key security (RKA), key-dependent message security (KDM), and context-committing security (CMT). Farshim et al. [9] introduced the concept of key-committing security, where an adversary aims to find two keys $K \neq K'$ and a ciphertext-tag pair (C, T) such that (C, T) is valid under both keys. It was quickly shown that standard AEAD security does not imply key-committing security, and popular schemes like GCM [10,7], GCM-SIV [13], CCM [8,14], and ChaCha20-Poly1305 [10,17] were found to be vulnerable. These attacks often had significant practical implications, making

committing security a critical issue. Bellare and Hoang [1] proposed generalized security notions where the adversary's goal is to find two different contexts $(K, N, A, M) = (K', N', A', M')$ that produce the same ciphertext-tag pair. They defined these notions as CMT-1, CMT-3, and CMT-4, where CMT-1 represents conventional key-committing security, CMT-3 requires $(K, N, A) \neq (K', N, A')$, and CMT-4 requires $(K, N, A, M) \neq (K', N', A', M')$. They also showed that CMT-3 is equivalent to CMT-4 and is strictly stronger than CMT-1. Consequently, designing CMT-4 secure AEAD schemes has become a crucial research area. In fact, NIST's workshop on updating block-cipher modes [19] explicitly mentioned that commitment security will be a mandatory feature for updated block ciphers. Recently, Naito et al. [16] studied the context-committing security of AsconAEAD128 and proved that it achieves CMT-4 security up to the birthday bound in tag size. They also suggested a method to enhance this security by padding the message with zeros.

1.2 AsconAEAD128 in Multi-User Applications

Recently, in their first public draft [22] for AsconAEAD128, NIST acknowledged its limitations in multi-user settings due to the $\mu T/2^{\kappa}$ term in the security bound. To mitigate this, NIST proposed using a 256-bit key in multi-user scenarios, with the key processed according to the initialization procedure described by Dobraunig et al. [6].

In Section 5.2, we demonstrate that this multi-user instance of AsconAEAD128 (hereafter referred to as mu-Ascon) is susceptible to complete key recovery under state-recovery attacks in $T = 2^{\kappa/2}$ and hence the primary time/data complexity of this key-recovery attack is determined by the complexity of the intermediate state-recovery attack. The current key-recovery security guarantees against this attack for mu-Ascon rely solely on the time/data complexity restrictions specified in [18,22]. Our current attack does not violate these restrictions. However, if future advanced attacks reduce state-recovery attack complexities below these limits, complete key recovery will become a real concern. The "power of key-binding" (as termed in [11]) of mu-Ascon will then be limited by the strength of the lower half of the key.

Here we remark that the Ascon-256 construction is resistant to this attack due to the full 256-bit key binding. Unfortunately, the current Ascon-256 mode cannot be instantiated with the AsconAEAD128 mode API.

Another disadvantage of mu-Ascon, is that it is only suitable for applications where committing security is not required. This was acknowledged in the draft itself [22] showing a trivial committing attack. While committing security was not an initial requirement in the original NIST call [18], we must argue that it is a crucial security property for any modern and future-proof standardized AEAD scheme.

1.3 Our Contributions

Motivated by the newly introduced mu-Ascon with a 256-bit key, as described in the recent NIST public draft [22], this paper analyzes this new use case from a provable security perspective.

More specifically, in this draft, we deal with the two weaknesses exhibited by the current mu-Ascon instantiation. Namely,

1. It has only partial security against key-recovery under state recovery attacks.

2. It has no commitment security.

In this regard with some additional tweaks in the Ascon-256 construction by Chakraborty et al.[3], we generate two schemes which we call Ascon-256.v2 and Ascon-256.v3 for the multi-user settings, which are compatible with the AsconAEAD128 API. Ascon-256.v2 is able to resist CMT-4 attacks, while Ascon-256.v3 offers protection against both CMT-4 attacks and key-recovery under state recovery attacks. These improvements only come with a minimal performance cost: an additional Ascon permutation.

Table 1: Table comparing our provable security results in the ideal cipher model mu-Ascon, Ascon-256.v2 and Ascon-256.v3 use-cases in the nonce-respecting-multi-user settings. **Sr-Kr**:=Key-recovery under state recovery ; T is measured in the unit of number of permutation calls.

security-type	**Variant**	Adversarial-advantage
Sr-Kr	mu-Ascon/ Ascon-256.v2	$\geq \frac{T}{2^{128}}$ [section 5.2]
CMT-4	mu-Ascon	1 [22]
CMT-4	Ascon-256.v2	$\leq \frac{T^2}{2^{128}}$ [section 5.1].
CMT-4	Ascon-256.v3	$\leq \frac{T^2}{2^{128}}$ [section 5.1].

To prove the committing security, we reuse the committing security of AsconAEAD128 result by Naito et al. [16]. Given the lightweight nature, since our Ascon-256.v2 and Ascon-256.v3 use cases require only one extra permutation call per query, hence an efficiency difference between it and the current use-case is negligible compared to the amount of extra security guarantees it provides.

2 Preliminaries

2.1 Notations

Let $\{0,1\}^n$ represent the set of bit strings of length n, and $\{0,1\}^+$ denote the set of bit strings of arbitrary length. The empty string is denoted by λ, and we define $\{0,1\}^* = \{\lambda\} \cup \{0,1\}^+$. For any integers $a \leq b \in \mathbb{N}$, $[b]$ and $[a,b]$ denote the sets $\{1,2,\ldots,b\}$ and $\{a,a+1,\ldots,b\}$, respectively. For $n,k \in \mathbb{N}$ with $n \geq k$,

the falling factorial is defined as $(n)_k := n(n-1)\cdots(n-k+1)$. It's worth noting that $(n)_k \leq n^k$.

For any bit string $x = x_1x_2\cdots x_k \in \{0,1\}^k$ of length k, and for $n \leq k$, we use $\lceil x \rceil_n := x_1\cdots x_n$ (and $\lfloor x \rfloor_n := x_{k-n+1}\cdots x_k$) to denote the most (and least) significant n bits of x. The bit concatenation operation is denoted by $\|$. The notation $(x_1, \ldots, x_r)$ is also used to represent the bit concatenation operation $x_1\|\cdots\|x_r$, where $x_i \in \{0,1\}^*$. For instance, if $V := x\|z := (x,z) \in \{0,1\}^r \times \{0,1\}^c$, then $\lceil V \rceil_r = x$ and $\lfloor V \rfloor_c = z$. The bitwise XOR operation is denoted by $\oplus$.

For a finite set $\mathcal{X}$, $\mathsf{X} \xleftarrow{\$} \mathcal{X}$ denotes the uniform and random sampling of X from $\mathcal{X}$, and $\mathsf{X} \xleftarrow{\mathsf{wor}} \mathcal{X}$ denotes sampling without replacement of X from $\mathcal{X}$.

Padding and Parsing a Bit String. Let $r > 0$ be an integer and $X \in \{0,1\}^*$. Let $d = |X| \bmod r$ (the remainder while dividing $|X|$ by r).

$$\mathsf{pad}(X) = X\|1\|0^{r-1-d}.$$

Given $X \in \{0,1\}^*$, let $x = \lceil \frac{|X|+1}{r} \rceil$. We define $(X_1, \ldots, X_x) \xleftarrow{r}_* X$ as $X_1\|\cdots\|X_x = X$, $|X_1| = \cdots = |X_{x-1}| = r$ and

$$X_x = \begin{cases} \lambda & \text{if } |X| = r(x-1) \\ \lfloor X \rfloor_{|X|-r(x-1)} & \text{otherwise} \end{cases}.$$

For $N \geq 4,\ n = \log_2 N$, we define

$$\mathsf{mcoll}(q,N) = \begin{cases} 3 & \textit{if } 4 \leq q \leq \sqrt{N} \\ \frac{4\log_2 q}{\log_2 \log_2 q} & \textit{if } \sqrt{N} < q \leq N \\ 5n\left\lceil \frac{q}{nN} \right\rceil & \textit{if } N < q. \end{cases}$$

2.2 Authenticated Encryption with Associated Data: Definition and Security Model

An authenticated encryption scheme with associated data functionality, abbreviated as AEAD, is characterized by a tuple of algorithms $\mathsf{AE} = (\mathsf{Enc}, \mathsf{Dec})$. These algorithms, referred to as the encryption and decryption algorithms, operate over the *key space* $\mathcal{K}$, *nonce space* $\mathcal{N}$, *associated data space* $\mathcal{A}$, *message space* $\mathcal{M}$, *ciphertext space* $\mathcal{C}$, and *tag space* $\mathcal{T}$. The functionalities are defined as follows:

$$\mathsf{Enc} : \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{M} \to \mathcal{C} \times \mathcal{T} \quad \text{and} \quad \mathsf{Dec} : \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{C} \times \mathcal{T} \to \mathcal{M} \cup \{\mathsf{rej}\}.$$

Here, rej signifies that the tag-ciphertext pair is invalid and consequently rejected. Additionally, the correctness condition is imposed:

$$\mathsf{Dec}(K, N, A, \mathsf{Enc}(K, N, A, M)) = M \text{ for any } (K, N, A, M) \in \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{M}.$$

For a key $K \in \mathcal{K}$, we use $\mathsf{Enc}_K(\cdot)$ and $\mathsf{Dec}_K(\cdot)$ to denote $\mathsf{Enc}(K, \cdot)$ and $\mathsf{Dec}(K, \cdot)$, respectively. In this paper, we consider $\mathcal{K} = \{0,1\}^\kappa, \mathcal{N} = \{0,1\}^\nu, \mathcal{T} = \{0,1\}^\tau$, and $\mathcal{A}, \mathcal{M} = \mathcal{C} \subseteq \{0,1\}^*$.

AEAD Security in the Random Permutation Model.
Let $\mathsf{Perm}(b)$ denote the set of all permutations over $\{0,1\}^b$ and $\mathsf{Func}(\mathcal{N} \times \mathcal{A} \times \mathcal{M}, \mathcal{M} \times \mathcal{T})$ denote the set of all functions from (N, A, M) to (C, T) such that $|C| = |M|$. We consider the AEAD security in the multi-user (mu) setting, parameterized by the number of users μ. Let:

- $\Pi \xleftarrow{\$} \mathsf{Perm}(b)$ (we use the superscript $\pm$ to denote bidirectional access to Π),
- $\Gamma_1, \ldots, \Gamma_\mu \xleftarrow{\$} \mathsf{Func}(\mathcal{N} \times \mathcal{A} \times \mathcal{M}, \mathcal{M} \times \mathcal{T})$,
- rej denotes the degenerate function from $(\mathcal{N}, \mathcal{A}, \mathcal{M}, \mathcal{T})$ to $\{\mathsf{rej}\}$, and
- $K_1, \ldots, K_\mu \xleftarrow{\$} \mathcal{K}$.

We have the following definition:

Definition 1. *Let AE_Π be an AEAD scheme based on the random permutation Π, defined over $(\mathcal{K}, \mathcal{N}, \mathcal{A}, \mathcal{M}, \mathcal{T})$. The mu-AEAD advantage of an adversary $\mathscr{A}$ against AE_Π is defined as*

$$\mathbf{Adv}^{\mathsf{mu-aead}}_{\mathsf{AE}_\Pi}(\mathscr{A}) := \left| \Pr_{\substack{(\mathsf{K_i})_{i=1}^{\mu} \xleftarrow{\$} \mathcal{K} \\ \Pi^{\pm}}} \left[\mathscr{A}^{(\mathsf{Enc}_{\mathsf{K_i}}, \mathsf{Dec}_{\mathsf{K_i}})_{i=1}^{\mu}, \Pi^{\pm}} = 1 \right] - \Pr_{\substack{(\Gamma_i)_{i=1}^{\mu} \\ \Pi^{\pm}}} \left[\mathscr{A}^{(\Gamma_i)_{i=1}^{\mu}, \mathsf{rej}, \Pi^{\pm}} = 1 \right] \right|.$$

Here $\mathscr{A}^{\mathsf{Enc}_{\mathsf{K_i}}, \mathsf{Dec}_{\mathsf{K_i}}, \Pi^{\pm}}$ denotes $\mathscr{A}$'s response after its interaction with $\mathsf{Enc}_{\mathsf{K_i}}$, $\mathsf{Dec}_{\mathsf{K_i}}$, and $\Pi^{\pm}$ (i.e., both forward and backward queries to Π) respectively. Similarly, $\mathscr{A}^{\Gamma_i, \mathsf{rej}, \Pi^{\pm}}$ denotes $\mathscr{A}$'s response after its interaction with Γ_i, rej, and $\Pi^{\pm}$ respectively.

In this paper, we assume that the adversary is adaptive. This means that the adversary neither issues duplicate queries nor requests information for which the response is already known due to some previous query. Let q_e, q_d, and q_p represent the number of queries made across all $\mathsf{Enc}_{\mathsf{K_i}}$, all $\mathsf{Dec}_{\mathsf{K_i}}$, and $\Pi^{\pm}$, respectively. Furthermore, let σ_e and σ_d denote the sum of input lengths (including associated data and message) across all encryption and decryption queries, respectively. Additionally, let $\sigma := \sigma_e + \sigma_d$ represent the combined resources for construction queries.

Remark 1. Here σ corresponds to the online or data complexity, and q_p corresponds to the offline or time complexity of the adversary. An adversary adhering to the specified resource constraints is referred to as an $(q_p, \sigma_e, \sigma_d)$-adversary.

2.3 H-coefficient Technique

Consider an adversary $\mathscr{A}$, which is deterministic and computationally unbounded, attempting to distinguish between the real oracle, denoted as $\mathcal{O}_{\mathrm{re}}$, and the ideal oracle, denoted as $\mathcal{O}_{\mathrm{id}}$. The interaction of $\mathscr{A}$ with its oracle is captured by the query-response tuple denoted as ω. In certain scenarios, after the query-response phase of the game, the oracle may choose to reveal additional information to

the distinguisher. In such cases, the extended definition of the transcript may include that additional information. Let Θ_{re} (respectively, Θ_{id}) represent the random transcript variable when $\mathscr{A}$ interacts with $\mathcal{O}_{\mathrm{re}}$ (respectively, $\mathcal{O}_{\mathrm{id}}$). The probability of realizing a specific transcript ω in the security game with an oracle $\mathcal{O}$ is referred to as the *interpolation probability* of ω with respect to $\mathcal{O}$. Given the determinism of $\mathscr{A}$, this probability depends solely on the oracle $\mathcal{O}$ and the transcript ω. A transcript ω is considered *realizable* if $\Pr[\Theta_{\mathrm{id}} = \omega] > 0$. In this paper, $\mathcal{O}_{\mathrm{re}} = (\mathsf{Enc}_{\mathsf{K}}, \mathsf{Dec}_{\mathsf{K}}, \Pi^{\pm})$, $\mathcal{O}_{\mathrm{id}} = (\Gamma, \mathsf{rej}, \Pi^{\pm})$, and the adversary aims to distinguish $\mathcal{O}_{\mathrm{re}}$ from $\mathcal{O}_{\mathrm{id}}$ in an AEAD sense.

Proposition 1 (H-coefficient technique [20,21]). *Let Ω be the set of all realizable transcripts. For some $\epsilon_{\mathsf{bad}}, \epsilon_{\mathsf{ratio}} > 0$, suppose there is a set $\Omega_{\mathsf{bad}} \subseteq \Omega$ satisfying the following:*

- $\Pr[\Theta_{\mathrm{id}} \in \Omega_{\mathsf{bad}}] \leq \epsilon_{\mathsf{bad}}$;
- *For any $\omega \notin \Omega_{\mathsf{bad}}$,*

$$\frac{\Pr[\Theta_{\mathrm{re}} = \omega]}{\Pr[\Theta_{\mathrm{id}} = \omega]} \geq 1 - \epsilon_{\mathsf{ratio}}.$$

Then for any adversary $\mathscr{A}$, we have the following bound on its AEAD distinguishing advantage:

$$\mathbf{Adv}^{\mathsf{aead}}_{\mathcal{O}_{\mathrm{re}}, \mathcal{O}_{\mathrm{id}}}(\mathscr{A}) \leq \epsilon_{\mathsf{bad}} + \epsilon_{\mathsf{ratio}}.$$

A proof of Proposition 1 can be found in multiple papers including [21,5,15].

2.4 Expected Multicollision in a Uniform Random Sample

Let $S := (x_i)_{i \in I}$ be a tuple of elements from a set T. For any $x \in T$, we define $\mathsf{mc}_x(S) = |\{i \in I : x_i = x\}|$ (the number of times x appears in the tuple). Finally, we define multicollision of S as the $\mathsf{mc}(S) := \max_{x \in T} \mathsf{mc}_x(S)$. In this section, we revisit some multicollision results discussed in [2,4].

Lemma 1 ([4]). *Let $\mathcal{D}$ be a set of size $N \geq 4$, $n = \log_2 N$. Given random variables $\mathsf{X}_1, \ldots, \mathsf{X}_q \xleftarrow{\$} \mathcal{D}$, we have $\mathbb{E}[\mathsf{mc}(\mathsf{X}_1, \ldots, \mathsf{X}_q)] \leq \mathsf{mcoll}(q, N)$.*

Remark 2. Similar bounds as in the above Lemma 1 can be achieved in the case of non-uniform samplings. Let $\mathsf{Y}_1, \ldots, \mathsf{Y}_q \xleftarrow{\mathrm{wr}} \{0,1\}^b$ and define $X_i := [Y_i]_r$ for some $r < b$. If we take $N = 2^r$ for this truncated random sampling, then we have the same result as above for multicollisions among $\mathsf{X}_1, \ldots, \mathsf{X}_q$.

We also have the following general result:

Lemma 2 ([2]). *Let $\mathscr{A}$ be an adversary which makes queries to a b-bit random permutation $\Pi^{\pm}$ and τ-bit to τ-bit random function Γ. Let $(X_1, Y_1), \ldots, (X_{q_1}, Y_{q_1})$ and $(X_{q_1+1}, Y_{q_1+1}), \ldots, (X_{q_1+q_2}, Y_{q_1+q_2})$ be the tuples of input-output corresponding to Π and Γ respectively obtained by the $\mathscr{A}$. Let $q := q_1 + q_2 \leq 2^b$ and $\mathsf{Z}_i := \mathit{trunc}_\tau(X_i) \oplus \mathit{trunc}'_\tau(Y_i)$ for $i \in [q_1]$ and $\mathsf{Z}_i := (X_i \oplus Y_i)$ for $i \in [q_1 + 1, q]$, where trunc_τ and trunc'_τ represent some τ-bit truncations. For $\tau \geq 2$,*

$$\mathbb{E}[\mathsf{mc}(Z^q)] \leq \mathsf{mcoll}(q, 2^\tau).$$

2.5 Partial XOR-Function Graph

A *partial function* $\mathcal{L} : \{0,1\}^b \dashrightarrow \{0,1\}^c$ is a subset $\mathcal{L} = \{(p_1, q_1), \ldots, (p_t, q_t)\} \subseteq \{0,1\}^b \times \{0,1\}^c$ with distinct p_i values. An *injective partial function* has distinct q_i values. Define

$$\text{domain}(\mathcal{L}) = \{p_i : i \in [t]\}, \quad \text{range}(\mathcal{L}) = \{q_i : i \in [t]\}.$$

We write $\mathcal{L}(p_i) = q_i$ and for all $p \notin \text{domain}(\mathcal{L})$, $\mathcal{L}(p) = \perp$.

Consider a partial function $\mathcal{P} : \{0,1\}^b \dashrightarrow \{0,1\}^b$, $r \in [b-1]$. Define $\mathcal{P}^{\oplus} : \{0,1\}^b \times \{0,1\}^r \dashrightarrow \{0,1\}^b$ as

$$\mathcal{P}^{\oplus}(u, x) = \mathcal{P}((u' \oplus x) \| u''),$$

where $u = u' \| u''$ and $u' \in \{0,1\}^r$. Define $G^{\oplus} := G^{\mathcal{P}^{\oplus}}$ with labeled edges denoted as $u \xrightarrow{x}_{\oplus} v$. The details on Partial Function Graph are thoroughly discussed in Section A.

2.6 Revisiting Committing Security of Ascon Modes

In this section we informally revisit the commitment security of Ascon mode as proved by Naito et al [16].

for any non-negative integer z, with the initialization/finalization permutation P_1 and intermediate permutation P_2 define

$$\mathsf{AsconAEAD128}_{ZP}^{[P_1,P_2]}.\mathsf{Enc}(K, N, A, M) = \mathsf{AsconAEAD128}^{[P_1,P_2]}.\mathsf{Enc}(K, N, A, M \| 0^z).$$

Theorem 1. *[16] Let P_1 and P_2 be independent random permutations. For any CMT-4 adversary making a total of T queries to P_1,P_1^{-1}, P_2, or P_2^{-1}, we have*

$$\mathbf{Adv}^{\text{cmt}-4}_{\mathsf{Ascon}_{ZP}} \leq \left(\frac{11T^2}{2^c} + \frac{5T^2}{2^{n-\nu}} + \frac{0.5T^2}{2^{\tau+z}} + \frac{0.5T^2}{2^{n+\tau-\kappa-\nu}}\right) \cdot \left(1 - \frac{0.5T^2}{2^n}\right).$$

where, ν, κ, τ, c, n denote the nonce-size, key-size, tag-size , capacity-size and permutation state-size respectively.

Assuming $T < 2^{n/2}$, the term $\left(1 - \frac{0.5T^2}{2^n}\right)$ is $\mathcal{O}(1)$. Then assuming $\kappa \geq \tau$, the above bound shows that Ascon_{ZP} is CMT-4 secure as long as $T \ll \min\{2^{\frac{c}{2}}, 2^{\frac{\tau+z}{2}}, 2^{\frac{n+\tau-k-\nu}{2}}\}$, ensuring $\min\{\frac{c}{2}, \frac{\tau+z}{2}, \frac{n+\tau-k-\nu}{2}\}$ bit CMT-4 security.

Corollary 1. *In the random permutation model,*

$$\mathbf{Adv}^{\text{cmt}-4}_{\mathsf{AsconAEAD128}} = \mathcal{O}\left(\frac{T^2}{2^{128}}\right).$$

Proof. The CMT-4 security of AsconAEAD128 follows from theorem 1 by plugging in the actual parameter sizes and the observation that AsconAEAD128 is $\mathsf{AsconAEAD128}_{ZP}$ with $z = 0$.

3 Existing Ascon Proposals in the Multi-User Settings

Consider the AsconAEAD128 design in the multi-user settings. due to the term $\frac{\mu T}{2^\kappa}$ appearing in the security bound [11,3], the standard key size of 128-bits doesn't leave room for a large μ (number of users). In this section, we revisit existing Ascon-based constructions, which are tailor-made to deal with large numbers of users. More specifically, we revisit the modes of operations defined in [22,3].

3.1 mu-Ascon

In the recently published public draft [22], NIST acknowledged the issue of having a small key size for the use case where there are a large number of users. To tackle this, they used the masked nonce initialization in the Duplex paradigm introduced by Dobraunig et al. [6]. More specifically in the multi-user setting each user has a 256-bit key and given a encryption tuple (N, A, M), an user uses its key $K := K_1 \| K_2$ in the AsconAEAD128 construction to get the ciphertext-tag pair as follows:

$$\mathsf{mu\text{-}Ascon}(K_1 \| K_2, N, A, M) := \mathsf{AsconAEAD128.Enc}(K_1, N \oplus K_2, A, M)$$

The advantages of this specific use instance are that it can be instantiated directly with theAsconAEAD128 API and does not compromise the performance.

The disadvantage of this construction is that although it does not violet the security requirement prescribed in [18], as shown in section 5.2, this use case is prone to key-recovery under state recovery attack and in no case provide > 192-bit key-recovery security.

Furthermore, it does not provide any committing security. For any two pair $(K, N) \neq (K', N')$ such that $K \oplus 0^{128} \| N = K' \oplus 0^{128} \| N'$ and any (A, M) this outputs same (C, T) pair. Here, we argue that, although the committing security was not put in as an initial requirement in [18], it is a must-have security for a future-ready standardized AEAD scheme.

3.2 Ascon-256

Another construction that is tailor-made for multi-user settings with a large number of users was introduced by Chakraborty et al. [3] called the Ascon-256 construction. On the high level the construction is as follows:
INITIALIZATION : $V_0 = P(K \| IV) \oplus 0^{64} \| K$.
DATA PROCESSING : given nonce N and associated data A, and message M define $A' := N \| A$, Process A', M using the AsconAEAD128 associated data and message processing protocols.
FINALIZATION : in the finalization process xor $K \| 0^{64}$ to the input of the permutation and add $\lfloor K \rfloor_\tau$ o the output of the permutation to generate the tag.

The multi-user settings construction security follows the bounds achieved in [3].

The disadvantage of Ascon-256 is that in its present form; it cannot be directly instantiated using the AsconAEAD128 API due to the addition of full 256 bit K binding to the output (resp. input) of the initialization (resp. finalization) permutation.

Remark 3. The commitment security of Ascon-256 is still an open problem, but with a quick read of the proof in [16], we think it should follow in the same line as AsconAEAD128. Since our objective is to come up with a construction which is compatible with AsconAEAD128 API, we declare it as out of the scope of this paper.

4 Ascon-256.v2, Ascon-256.v3: Alternate Proposals for Multi-User Settings

In this section we modify the Ascon-256 construction so as to make it compatible with the AsconAEAD128 API. Our main objective is to construct use-cases of AsconAEAD128 which can resists against the weaknesses of mu-Ascon, i.e., they should be resistant to

- CMT-4 attack
- Key recovery against state-recovery attack.

In this respect we define two new use-instances which we call Ascon-256.v2 and Ascon-256.v3 constructions depending on how the AsconAEAD128 API is processed. More specifically,

- We define Ascon-256.v2 for the use-case when the user can only pre-processes her input before feeding it in the AsconAEAD128.
- We define Ascon-256.v3 for the use-case when the user can also process the AsconAEAD128 output before releasing it.

We present our constructions using the AsconAEAD128 construction as a black-box. We start by defining two functions,

$$F_{pre} : \{0,1\}^{256} \times \{0,1\}^{128} \times \{0,1\}^* \to \{0,1\}^{128} \times \{0,1\}^{128} \times \{0,1\}^*$$

$$(K_1 \| K_2, N, A) \mapsto (K_2, K_1, A_1)$$

where $A_1 := (N \oplus \lceil K_1 \rceil_{128}) \| A$.

$$F_{post} : \{0,1\}^{128} \times (\{0,1\}^* \cup \perp) \to (\{0,1\}^* \cup \perp)$$

$$(K, \perp) \mapsto \perp$$

$$(K, M) \mapsto (M \oplus 0^{|M|-128} \| K) \text{ if } |M| \geq 128$$

$$(K, M) \mapsto ((M \oplus \lceil K \rceil_{|M|})) \text{ if } |M| \leq 128$$

Definition 2. *(Ascon-256.v2)*

$$\textsf{Ascon-256.v2.Enc}(K_1\|K_2, N, A, M) = \textsf{AsconAEAD128.Enc}(F_{pre}(K_1\|K_2, N, A), M)$$

$$\textsf{Ascon-256.v2.Dec}(K_1\|K_2, N, A, C, T) = \textsf{AsconAEAD128.Dec}(F_{pre}(K_1\|K_2, N, A), C, T)$$

Definition 3. *(Ascon-256.v3)*

$$\textsf{Ascon-256.v3.Enc}(K_1\|K_2, N, A, M) = (F_{post}(K_1, C), T)$$

where $(C, T) = \textsf{AsconAEAD128.Enc}(F_{pre}(K_1\|K_2, N, A), F_{post}(K_1, M))$.

$$\textsf{Ascon-256.v3.Enc}(K_1\|K_2, N, A, M) = (F_{post}(K_1, C), T)$$

where $(C, T) = \textsf{AsconAEAD128.Enc}(F_{pre}(K_1\|K_2, N, A), F_{post}(K_1, M))$.

$$\textsf{Ascon-256.v3.Dec}(K_1\|K_2, N, A, C, T) = F_{post}(K_1, M)$$

where $M = \textsf{AsconAEAD128.Dec}(F_{pre}(K_1\|K_2, N, A), F_{post}(K_1, C), T)$

In the following section we compare different security notions for AsconAEAD128, Ascon-256 and our new variants Ascon-256.v2, Ascon-256.v3.

5 Comparative Security Analysis of mu-Ascon, Ascon-256.v2 and Ascon-256.v3

In this section, we first try to do a comparative security analysis of mu-Ascon, Ascon-256.v2 and Ascon-256.v3 in the nonce-respecting multi-user settings under different security notions. More specifically the CMT-4 security, key-recovery under state recovery security of these constructions. To the best of our knowledge the AEAD security of mu-Ascon has never been formally explored. In their draft NIST [22] justify the AEAD security of mu-Ascon with reference to the results in [6,3]. In this respect we will argue that Ascon-256 in [3], has a full key-binding rather than the partial key binding in mu-Ascon and the analysis in [6] was explored on the Duplex rather than Ascon mode. Hence, for the sake of completeness later in section 7 we do a revisit of the proofs in [3] with the necessary adjustment needed in the analysis of mu-Ascon, Ascon-256.v2 and Ascon-256.v3.

5.1 CMT-4 Security of Ascon-256.v2 and Ascon-256.v3

In this section, we show that unlike mu-Ascon, our variants namely Ascon-256.v2 and Ascon-256.v3 both enjoy commitment security.

Theorem 2. *For any CMT-4 adversary $\mathscr{A}$,*

$$\mathbf{Adv}^{\text{cmt}-4}_{\textsf{Ascon-256.v2}}(\mathscr{A}) \leq \mathbf{Adv}^{\text{cmt}-4}_{\textsf{AsconAEAD128}}(\mathscr{A}).$$

Proof. Suppose there exists an adversary $\mathscr{A}$ who can break the CMT-4 security of Ascon-256.v2. Let $\mathscr{A}$ outputs $(K_1\|K_2, N, A, M) \neq (K_1'\|K_2', N', A', M')$ such that

$$\mathsf{Ascon\text{-}256.v2.Enc}(K_1\|K_2, N, A, M) = \mathsf{Ascon\text{-}256.v2.Enc}(K_1'\|K_2', N', A', M').$$

Note that F_{pre} is an injective function. Hence

$$(F_{pre}(K_1\|K_2, N, A), M) \neq (F_{pre}(K_1'\|K_2', N', A'), M')$$

but

$$\mathsf{AsconAEAD128.Enc}((F_{pre}(K_1\|K_2, N, A), M)) = \mathsf{AsconAEAD128.Enc}((F_{pre}(K_1'\|K_2', N', A'), M')).$$

Hence the adversary $\mathscr{A}$ breaks the CMT-4 security of AsconAEAD128 by outputting $(F_{pre}(K_1\|K_2, N, A), M) \neq (F_{pre}(K_1'\|K_2', N', A'), M')$.

Corollary 2. *In the random permutation model,*

$$\mathbf{Adv}^{\mathrm{cmt-4}}_{\mathit{Ascon\text{-}256.v2}} = \mathcal{O}\left(\frac{T^2}{2^{128}}\right).$$

Proof. The corollary follows from theorems 1 and 2.

Theorem 3. *For any CMT-4 adversary $\mathscr{A}$ in the random permutation model,*

$$\mathbf{Adv}^{\mathrm{cmt-4}}_{\mathit{Ascon\text{-}256.v3}} = \mathcal{O}\left(\frac{T^2}{2^{128}}\right).$$

Proof. The proof of this theorem can be proved with a similar approach as taken by Naito et al. to prove the CMT-4 security of AsconAEAD128. We provide the proof details in section 6.

5.2 Key-Recovery from State-Recovery Attack on mu-Ascon and Ascon-256.v2 in Nonce-Respecting Multi-User Settings

We start by revisiting the security model defined by Lefevre et al. [12] which they call the authenticity under state recovery. The authors formally define the security model as follows.

Definition 4. *[12] Consider an adversary $\mathscr{A}$ with access to two learning oracles $\mathcal{LE}$ and $\mathcal{LD}$, which are defined as $\mathcal{E}$ and $\mathcal{D}$ but that additionally leak all input/output values of the evaluations of the inner permutations. We say that $\mathscr{A}$ wins if it ever makes a query to one of its learning decryption oracles that is successful and that is not the result of an earlier encryption query. This leads to the following model:*

$$\mathbf{Adv}^{\mathrm{sr-Auth}}_{\mathit{Ascon}}(\mathscr{A}) = \Pr\left[\mathscr{A}^{\mathcal{LE},\mathcal{LD}} \textit{ forges}\right].$$

In this section we extend the security model defined by Lefevre et al. [12] and call it key-recovery under state recovery. The formal definition of the security model in the multi-user settings is a simple extension of definition 4 and is as follows.

Definition 5. *Consider an adversary $\mathscr{A}$ with access to two learning oracles $\mathcal{LE}_u$ and $\mathcal{LD}_u$ which are defined as $\mathcal{E}_u$ and $\mathcal{D}_u$ for each of the users u such that for some user u, they additionally leak all input/output values of the evaluations of their inner permutations. We say that $\mathscr{A}$ wins if it can recover the entire secret-key for one such user u:*

$$\mathbf{Adv}^{\mathrm{mu-sr-kr}}_{\textit{mu-Ascon},\textit{Ascon-256.v2},\textit{Ascon-256}}(\mathscr{A}) = \Pr\left[\mathscr{A}^{\mathcal{LE},\mathcal{LD}} \textit{recovers the key for some user } u\right].$$

Now we explore the Key-recovery under state-recovery attack on mu-Ascon and Ascon-256.v2. As the definition suggests the objective of the adversary is to recover the entire secret-key of one of the users given an intermediate state has been recovered for some encryption query made to the user. Note that, since the intermediate states are processed using the duplex construction hence if for some query with σ_e blocks of data for an user any of the intermediate state is recovered then the state after the initialization and state-before the finalization can be recovered in time complexity σ_e.

Theorem 4. *Given some and thus all intermediate states of some mu-Ascon encryption query by one of the users is know. Then there exists a nonce-respecting adversary $\mathscr{A}$ who can recover the full 256-bit key of that user in $q << 2^{129}$ queries. i.e.*

$$\mathbf{Adv}^{\mathrm{mu-sr-kr}}_{\textit{mu-Ascon}}(\mathscr{A}) \geq \frac{q}{2^{129}}$$

Proof. The attack is straight forward. But for the sake of completeness we define how such an adversary $\mathscr{A}$ can work.

- $\mathscr{A}$knows the X_1, Y_l, and T where X_1 is the full intermediate state after the initialization, Y_l is the last intermediate state before the finalization and T is the tag .
- $\mathscr{A}$makes 2^{128} many permutation queries of the form $P2(Y_l \oplus 0^{128} \| K_{i,1} \| 0^{64})_{i \in [2^{128}]}$ and receives responses of the form Z_i.
- For each Z_i, $\mathscr{A}$checks if $\lfloor Z_i \rfloor_{128} \oplus K_{i,1} = T$.
- For all Z_i such the $\mathscr{A}$gets a matching, it makes one inverse permutation query of the form $(P2)^{-1}(X_l \oplus 0^{192} \| K_{i,2})_{i \in [2^{128}]}$ and receives responses of the form W_i and checks if $\lfloor W_i \rfloor_{192} = IV \| K_{i,2}$.
- If the above check is successful the adversary define $K_{i,1} = N \oplus \lceil W_i \rceil_{128}$.
- $\mathscr{A}$then verifies if $K_{i,1} \| K_{i,2}$ is the key for the user by making an empty data query to the user and matching the tag received with the tag computed by herself using $K_{i,1} \| K_{i,2}$ as the key.

If the number of $K_{i,2}$ such that $\lfloor Z_i \rfloor_{128} \oplus K_{i,1} = T$ is t. Then the number of total permutation and inverse permutation queries required is at most $2^{128}+t \ll 2^{129}$.

Theorem 5. *Given some and thus all intermediate states of some* Ascon-256.v2 *encryption query by one of the users is know. Then for any nonce-respecting adversary can recover the full* 256*-bit key of that user in* $q << 2^{129}$ *queries. i.e.*

$$\mathbf{Adv}^{\mathrm{mu-sr-kr}}_{\textsf{Ascon-256.v2}}(\mathscr{A}) \geq \frac{q}{2^{129}}$$

Proof. Consider an adversary $\mathscr{A}$ which recovers the intermediate states for some user. For that user,

- $\mathscr{A}$ knows the X_1 and Y_l and τ where X_1 is the full intermediate state after the initialization, Y_l is the last intermediate state before the finalization and T is the tag .
- $\mathscr{A}$ makes T many permutation queries of the form $P2(Y_l \oplus 0^{128} \| K_{i,2} \| 0^{64})_{i \in [2^{128}]}$ and receives responses of the form Z_i.
- For each Z_i, $\mathscr{A}$checks if $\lfloor Z_i \rfloor_{128} \oplus K_{i,1} = \tau$.
- Once a $K_{i,2}$ is identified it makes T inverse permutation queries of the form $(P2)^{-1}(X_l \oplus (N \oplus K_{i,2}) \| 0^{64} \| K_{i,2})_{i \in [2^{128}]}$ and receives responses of the form W_i and checks if $W_i = K_{i,1} \| IV \| K_{i,2}$.
- If the above check is successful, then $\mathscr{A}$ verifies if $K_{i,1} \| K_{i,2}$ is the key for the user by making an empty data query to the user and matching the tag received with the tag computed by herself using $K_{i,1} \| K_{i,2}$ as the key.

If we assume $P2$ to be ideal random permutation then the expected number of $K_{i,2}$ such that $\lfloor Z_i \rfloor_{128} \oplus K_{i,1} = T$ is 1. Then the number of total permutation and inverse permutation queries required is at most 2^{129}.

6 Proof of Theorem 3

Suppose the adversary makes q permutation queries (forward/backward) to the ideal world oracles $\mathcal{O}$. The oracle $\mathcal{O}$ chooses a random permutations P. Each query (forward/backward) is responded by the oracle $\mathcal{O}$ using this random permutation. Let $\mathcal{F} := \{(X_i, Y_i) \mid P(X_i) = Y_i; i \in [q]\}$ denotes the list generated by the adversarial queries. We define the ordering $\prec$ on $\mathcal{F}$ such that given any pair $\{(X_i, Y_i) \neq (X_j, Y_j)\} \in \mathcal{F}$, $(X_i, Y_i) \prec (X_j, Y_j)$ if and only if the query corresponding to X_i, Y_i occurred before the query corresponding to (X_j, Y_j). We start by defining a Bad event generated due to $\mathcal{F}$.

We say the adversary forms a "full sequence" $S = \{(X_i, Y_i) \mid i \in [0, t_S]\} \subseteq \mathcal{F}$ if it represents the input/output states of the underlying permutation calls of some valid decryption query

$$\textsf{Ascon-256.v3.Dec}((K_1 \| K_2, N, A, C, T)) \neq \perp$$

Given any sequence S define;

$$K_S := \lceil X_0 \rceil_{128}; T_S := \lceil Y_t \rceil_{128} \oplus K_S;$$

It is easy to note that for any full sequence S, the following equations holds.

(1) $\lfloor X_0 \rfloor_{64} = IV$.
(2) $\forall i \in [1, t_S - 1]$, $\lceil Y_i \rceil_{192} \oplus \lceil X_{i+1} \rceil_{192} \in \{0^{192}, 0^{191} \| 1\}$.
(3) $\lceil Y_0 \rceil_{192} \oplus \lceil X_1 \rceil_{192} = 0^{64} \| K_S$.
(4) $\lceil Y_{T_S - 1} \rceil_{192} \oplus \lceil X_{T_S} \rceil_{192} \in \{K_S \| 0^{64}, K_S \| 0^{63} \| 1\}$.

Proposition 2. *Suppose an adversary breaks the CMT-4 security game against Ascon-256.v3. Then there exists two distinct full sequences S, S' such that $T_S = T_{S'} = T$ and they both correspond to the same cipher text.*

Given any full sequence $S = \{(X_i, Y_i) \mid i \in [0, t_S]\} \in \mathcal{F}$, we call $S_{in} = \{(X_i, Y_i) \mid i \in [0, t_S - 1]\}$ the internal sequence of S. We define some bad events while generating some internal sequence S_{in} using $\mathcal{F}$. For any $(X_i, Y_i) \in \mathcal{F}$, define

$$\Delta_i := \{0^{192}, 0^{64} \| \lceil X_i \rceil_{128}, 0^{191} \| 1, \}.$$

`Fcon` : There exists $(X_i, Y_i) \prec (X_j, Y_j) \in \mathcal{F}$, such that (X_j, Y_j) is a forward query and $\lceil Y_j \rceil_{192} \oplus \lceil X_i \rceil_{192} \in \Delta_j$.
`Bcon` : There exists $(X_i, Y_i) \prec (X_j, Y_j) \in \mathcal{F}$ such that (X_j, Y_j) is a backward query and $\lceil Y_i \rceil_{192} \oplus \lceil X_j \rceil_{192} \in \Delta_i$.
`Bend` : There exists $(X_i, Y_i) \prec (X_j, Y_j) \in \mathcal{F}$, both backward queries $\lfloor X_j \rfloor_{64} = IV$ and $\lceil X_j \rceil_{128} = \lceil Y_j \oplus X_i \rceil_{128}$.

Next consider the following events due to two distinct internal sequences.

`Scoll`: There exists two internal sequence $S_{in} \neq S'_{in}$ in $\mathcal{F}$ and integers $i \in [0, t_S - 2], j \in [0, T_{S'} - 2]$ such that $S_{in}[i] \neq S'_{in}[j]$ but $S_{in}[i+1] = S'_{in}[j+1]$.

Define `SBAD` := `Fconnect` $\cup$ `Bconnect` $\cup$ `Bend` $\cup$ `Scoll`

Lemma 3.

$$\Pr[\texttt{SBAD}] \leq \frac{7q^2}{2^{193}}.$$

Proof. Proof of the lemma follows from the observation that $\mathcal{F}$ is generated using a random permutation Π.

Proposition 3. *If* `SBAD` *doesn't occur, then the following holds in $\mathcal{F}$.*

– *for all internal sequences S_{in} in $\mathcal{F}$*

$$S_{in}[0] \prec \cdots \prec S_{in}[t_S - 1]$$

– *For any two internal sequences* $S_{in} \neq S'_{in}$ *in* $\mathcal{F}$

$$S_{in}[t_S - 1] \neq S'_{in}[t_{S'} - 1]$$

Corollary 3. *If* `SBAD` *doesn't occur in* $(\mathcal{F})$*, then there exists at most* q *distinct internal sequences in* $\mathcal{F}$*.*

Lemma 4. *If* `SBAD` *doesn't in* $\mathcal{F}$*. Then,*

$$\mathbf{Adv}^{\text{cmt-4}}_{\textit{Ascon-256.v3}} \leq \frac{q^2}{2^{129}} + \frac{3q^2}{2^{193}} + \frac{q^4}{2^{514}} + \frac{q^3}{2^{322}}.$$

Proof. Let S, S' denote the two full sequences in $\mathcal{F}$ generated by the CMT-4 adversary. Let S_{in}, S'_{in} denote the internal sequences of S, S'. First suppose $S_{in} = S'_{in}$ and C, T denote the corresponding ciphertext. Then if $|C| = d \mod 128$, from the construction of Ascon-256.v3 we must have,

$$\lfloor X_{t_S} \oplus X_{t_{S'}} \rfloor_{128} := \lfloor K'_S \oplus K'_{S'} \rfloor_d \| 0^{128-d}$$

where $S_{in}[0] = IV \| K'_S \| K_S$ and $S'_{in}[0] = IV \| K'_{S'} \| K_{S'}$ are fixed given any two S_{in}, S'_{in}.

With these observations we consider the following cases.

`CASE1`: $S_{in} = S'_{in}$. But then we have

$$X_{t_S} = X_{t_{S'}}$$

and hence $S = S'$.

`CASE2`: $S_{in} \neq S'_{in}$, $(X_{t_S}, Y_{t_S}) \prec (X_{t_S - 1}, Y_{t_S - 1})$. In this case without loss of generality we consider the following sub-cases.

- $(X_{t_S}, Y_{t_S}) \prec (X_0, Y_0)$. Note that this implies the case that given any X, Y in $\mathcal{F}$ one constructs a sequence S_{in} such that $\lceil Y_{t_S - 1} \rceil_{192} \oplus 0^{64} \| \lceil X_0 \rceil_{128} = \lceil X \rceil_q$. Further since $X_{t_S - 1}, Y_{t_S - 1}$ must be a forward query, Hence probability this happens is bounded by $\frac{q}{2^{192}}$. varying over all possible S_{in} we have probability of this case is bounded by $\frac{q^2}{2^{193}}$.
- $(X_0, Y_0) \prec (X_{t_S}, Y_{t_S}) \prec (X_{t_S - 1}, Y_{t_S - 1})$. Note that in this case given (X_0, Y_0) there exists a unique forward query of the form $X_{t_S - 1}, Y_{t_S - 1}$. Further since $(X_{t_S}, Y_{t_S}) \prec (X_{t_S - 1}, Y_{t_S - 1})$ hence $\lceil X_{t_S} \rceil_{192}$ is fixed. Hence in this case probability that $\lceil Y_{t_S - 1} \rceil_{192} \oplus 0^{64} \| \lceil X_0 \rceil_{128} = \lceil X_{t_S} \rceil_q$ is again bounded by $\frac{1}{2^{192}}$ varying over all $(X_0, Y_0) \prec (X_{t_S}, Y_{t_S})$ we have probability of this case is bounded by $\frac{q^2}{2^{193}}$.

`CASE3`: $S_{in} \neq S'_{in}$, $(X_{t_S - 1}, Y_{t_S - 1}) \prec (X_{t_S}, Y_{t_S})$ and $(X_{t_{S'} - 1}, Y_{t_{S'} - 1}) \prec (X_{t_{S'}}, Y_{t_{S'}})$. In this case without loss of generality suppose $(X_{t_S}, Y_{t_S}) \prec (X_{t_{S'}}, Y_{t_{S'}})$. Note that in this case $K_S, K'_S, K_{S'}, K'_{S'}$ are fixed. We divide the case into the following sub-cases.

- Both are backward queries. in this case the adversary has full control over T. Hence varying over all S, S',probability that this case occurs is bounded by

$$\frac{q^2}{2} \times \sum_{(X,Y)\neq(X',Y')} \Pr\left[\begin{array}{c} \lceil X\rceil_{192}\oplus 0^{64} \| K_S = \lceil Y_{T_S-1}\rceil_{192} \\ \wedge \\ \lceil X'\rceil_{192}\oplus 0^{64} \| K_{S'} = \lceil Y_{T_{S'}-1}\rceil_{192} \\ \wedge \\ \lfloor X\oplus X'\rfloor_{128} := \lfloor K'_S \oplus K'_{S'}\rfloor_d \| 0^{128-d} \end{array}\right] \leq \frac{q^4}{2^{514}}.$$

- (X_{t_S}, Y_{t_S}) forward query and $(X_{t_{S'}}, Y_{t_{S'}})$ backward query. In this case T is fixed. Hence the probability of this case is bounded by

$$\frac{q^2}{2} \times \sum_{(X,Y)\neq(X',Y')} \Pr\left[\begin{array}{c} \lceil X_{S'}\rceil_{192}\oplus 0^{64} \| K_{S'} = \lceil Y_{T_{S'}-1}\rceil_{192} \\ \wedge \\ \lfloor X_{t_S}\oplus X_{t_{S'}}\rfloor_{128} := \lfloor K'_S \oplus K'_{S'}\rfloor_d \| 0^{128-d} \end{array}\right] \leq \frac{q^3}{2^{322}}.$$

- (X_{t_S}, Y_{t_S}) backward query and $(X_{t_{S'}}, Y_{t_{S'}})$ forward query. Probability that this case occurs is bounded by

$$\frac{q^2}{2} \times \sum_{(X,Y)\neq(X',Y')} \Pr\left[\begin{array}{c} \lceil X_S\rceil_{192}\oplus 0^{64} \| K_S = \lceil Y_{T_S-1}\rceil_{192} \\ \wedge \\ \lceil Y_{t_S}\rceil_{128} \oplus \lceil Y_{t_{S'}}\rceil_{128} := K_S \oplus K_{S'} \end{array}\right] \leq \frac{q^3}{2^{322}}.$$

- Both are forward queries. This case is bounded by

$$\frac{q^2}{2} \times \Pr\left[\lceil Y_{t_S}\rceil_{128} \oplus \lceil Y_{t_{S'}}\rceil_{128} := K_S \oplus K_{S'}\right] \leq \frac{q^2}{2^{129}}.$$

7 AEAD Security of Ascon-256.v2 and Ascon-256.v3 in the Nonce-Respecting Multi-User Settings

In this section we analyse the AEAD security of our newly proposed schemes namely Ascon-256.v2 and Ascon-256.v3. In this regard we would like to remark that as constructions which use AsconAEAD128 as an internal API, the security of these constructions can be derived directly from the security proofs shown in [2][3]. Nonetheless we provide a modular proof sketch which mostly describes the extra bad events that appear due to the tweaks in our constructions.

7.1 AEAD Security proof for Ascon-256.v2

Theorem 6 (Ascon-256.v2). *Consider a nonce-respecting AEAD adversary $\mathscr{A}$ making q_p permutation queries, q_e encryption queries with a total number of σ_e data blocks, q_d decryption queries with a total number of σ_d data blocks. Define $\sigma := \sigma_e + \sigma_d$. Then, the upper bound of the AEAD advantage of $\mathscr{A}$ against Ascon-256.v2 is the following:*

$$\begin{aligned}\mathbf{Adv}^{\mathsf{mu-AEAD}}_{\textit{Ascon-256.v2}}(\mathscr{A}) \leq\ & \frac{2q_d}{2^\tau} + \frac{\sigma_e^2}{2^b} + \frac{\sigma_d(q_p+\sigma_d)}{2^b} + \frac{\mathsf{mcoll}(\sigma_e, 2^r)\times(\sigma_d+q_p)}{2^c} \\ &+ \frac{\mathsf{mcoll}(q_e, 2^\tau)q_d}{2^c} + \frac{\mu^2}{2^\kappa} + \frac{\mu(q_p+\sigma)}{2^\kappa} \\ &+ \frac{q_d^2+q_e^2+q_eq_d+(q_e+q_d)(\sigma+q_p)}{2^b} \\ &+ \frac{\mathsf{mcoll}(q_e, 2^{b-\kappa_2})(\sigma+q_p)}{2^{\kappa_2}} + \frac{q_e(\sigma+q_p)}{2^b} \\ &+ \frac{\mathsf{mcoll}(\sigma+q_p, 2^\tau)q_d}{2^{\kappa_2}}\end{aligned}$$

Proof. The proof of this theorem directly follows from the works of Chakraborty et al. [2], [3]. Thus we omit the proof for this construction.

7.2 AEAD Security Proof for Ascon-256.v3

Theorem 7 (Ascon-256.v3). *Consider a nonce-respecting AEAD adversary $\mathscr{A}$ making q_p permutation queries, q_e encryption queries with a total number of σ_e data blocks, q_d decryption queries with a total number of σ_d data blocks. Define $\sigma := \sigma_e + \sigma_d$. Then, the upper bound of the AEAD advantage of $\mathscr{A}$ against Ascon-256.v3 is the following:*

$$\begin{aligned}\mathbf{Adv}^{\mathsf{mu-AEAD}}_{\textit{Ascon-256.v3}}(\mathscr{A}) \leq\ & \frac{2q_d}{2^\tau} + \frac{\sigma_e^2}{2^b} + \frac{\sigma_d(\sigma_d+q_p)}{2^b} + \frac{\mathsf{mcoll}(\sigma_e, 2^r)\times(\sigma_d+q_p)}{2^c} + \frac{\mu^2}{2^\kappa} \\ &+ \frac{\mu(q_p+\sigma)}{2^\kappa} + \frac{3q_d^2+3q_e^2+3q_eq_d+(q_e+q_d)(\sigma+q_p)}{2^b} \\ &+ \frac{\mathsf{mcoll}(\sigma+q_p, 2^r)\times(q_e+q_d)}{2^c} + \frac{\mathsf{mcoll}(\sigma+q_p, 2^\tau)q_d}{2^{\kappa_2}} \\ &+ \frac{\mathsf{mcoll}(q_e, 2^{b-\kappa_2})(\sigma+q_p)}{2^{\kappa_2}} + \frac{q_e(\sigma+q_p)}{2^b}\end{aligned}$$

Description of the Real World for Ascon-256.v3

The real world samples $\mathsf{K}_1, \ldots, \mathsf{K}_\mu \xleftarrow{\$} \{0,1\}^\kappa$ and a random permutation $\Pi \xleftarrow{\$} \mathsf{Perm}(b)$. All the queries then answered honestly following the Ascon-256.v3 as defined in Section **??** and the direct primitive queries to Π's are also answered honestly. After all queries have been made, all inputs-outputs used in Π for all encryption and decryption queries are included in the offline transcript. Let P represent the query responses for primitive queries (represented in terms of the partial function for Π), and let $\mathsf{P}_{\mathsf{fin}}$ denote the extended partial function. Let the online Transcript for all the queries be:

$$\Theta_{real,online} := \left((\mathsf{u}_i, \mathsf{N}_i, \mathsf{A}_i, \mathsf{M}_i, \mathsf{C}_i, \mathsf{T}_i)_{i\in[q_e]}, (\mathsf{u}'_i, \mathsf{N}'_i, \mathsf{A}'_i, \mathsf{C}'_i, \mathsf{T}'_i, \mathsf{M}'_i)_{i\in[q_d]}, \mathsf{P}\right)$$

Set the extended partial function to be $\mathsf{P}_{\mathsf{fin}}$ and set:

$$\Theta_{real} := \left((\mathsf{u}_i, \mathsf{N}_i, \mathsf{A}_i, \mathsf{M}_i, \mathsf{C}_i, \mathsf{T}_i)_{i\in[q_e]}, (\mathsf{u}'_i, \mathsf{N}'_i, \mathsf{A}'_i, \mathsf{C}'_i, \mathsf{T}'_i, \mathsf{M}'_i)_{i\in[q_d]}, \mathsf{P}_{\mathsf{fin}}\right)$$

For any real world realizable transcript

$$\theta := \left((u_i, N_i, A_i, M_i, C_i, T_i)_{i\in[q_e]}, (u'_i, N'_i, A'_i, C'_i, T'_i, M'_i)_{i\in[q_d]}, P_{\mathsf{fin}}\right)$$

we obtain that,

$$Pr(\Theta = \theta) = Pr(\mathsf{P}_{\mathsf{fin}} \subset \Pi) = \frac{1}{(2^b)_{|\mathsf{P}_{\mathsf{fin}}|}}$$

Description of the Ideal World for Ascon-256.v3

The ideal world sampling is divided in two phases, *Online Phase* and *Offline Phase*. We present the Ideal World Sampling along with the bad events by the help of 6 algorithms. We list the algorithms as follows.

Algorithm 1: It provides the Online Transcript obtained in the Ideal World and sets a bad event bad_1.

Algorithm 2: Sets the internal states of Encryption Queries in the Offline Phase, and sets a bad event bad_2.

Algorithm 3: Sets the internal states of the Decryption Queries in the Offline Phase and sets bad events bad_3 and bad_4. Thereby, extends the permutation table.

Algorithm 4: Samples the key and sets the initial states and sets bad events bad_5 and bad_6 and further extends the permutation table.

Algorithm 5: It sets the final two states of the construction in the Offline Phase and sets bad events bad_i for $i \in \{7, 8, 9, 10\}$.

Algorithm 6: At this step the tag consistency is checked and sets the bad events $\mathsf{bad}_{11}, \mathsf{bad}_{12}, \mathsf{bad}_{13}$. Finally, outputs the Ideal world Transcript along with the complete permutation table.

Algorithm 1: Ideal World: Online Transcript Ascon-256.v3

Online Phase
Input: q_e encryption, q_d decryption, q_p primitive queries
Output: Ideal world responses
for *encryption query* $i \in [q_e]$ *with* $(\mathsf{u}_i, \mathsf{N}_i, \mathsf{A}_i, \mathsf{M}_i)$ **do**
  $\mathsf{C}_i \xleftarrow{\$} \{0,1\}^{|\mathsf{M}_i|}$
  $\mathsf{T}_i \xleftarrow{\$} \{0,1\}^{\tau}$
  return $\boxed{(\mathsf{C}_i, \mathsf{T}_i)}$
end
for *decryption query* $i \in [q_d]$ *with* $(\mathsf{u}'_i, \mathsf{N}'_i, \mathsf{A}'_i, \mathsf{C}'_i, \mathsf{T}'_i)$ **do**
  return $\boxed{\mathsf{rej}}$
end
for *primitive query* $i \in [q_p]$ *with* $(\mathsf{Q}_i, \mathsf{dir}_i)$ **do**
  if $\mathsf{dir}_i = +1$ **then**
    $\mathsf{U}_i \leftarrow \mathsf{Q}_i$
    $\mathsf{V}_i \xleftarrow{\$} \{0,1\}^b \setminus \mathsf{range}(\mathsf{P})$
    $\mathsf{P} \leftarrow \mathsf{P} \cup \{(\mathsf{U}_i, \mathsf{V}_i)\}$
    return $\boxed{\mathsf{V}_i}$
  else
    $\mathsf{V}_i \leftarrow \mathsf{Q}_i$
    $\mathsf{U}_i \xleftarrow{\$} \{0,1\}^b \setminus \mathsf{domain}(\mathsf{P})$
    $\mathsf{P} \leftarrow \mathsf{P} \cup \{(\mathsf{U}_i, \mathsf{V}_i)\}$
    return $\boxed{\mathsf{U}_i}$
  end
end
return

$$\boxed{\Theta_{\text{ideal,online}} \leftarrow \big((\mathsf{u}_i, \mathsf{N}_i, \mathsf{A}_i, \mathsf{M}_i, \mathsf{C}_i, \mathsf{T}_i)_{i\in[q_e]}, (\mathsf{u}'_i, \mathsf{N}'_i, \mathsf{A}'_i, \mathsf{C}'_i, \mathsf{T}'_i, \mathsf{rej})_{i\in[q_d]}, \mathsf{P}\big)}$$

if $\exists (i,j) \in [q_e] \times [q_d]$ *with later* $i > j$ *and* $(\mathsf{u}_i, \mathsf{N}_i, \mathsf{A}_i, \mathsf{C}_i, \mathsf{T}_i) = (\mathsf{u}'_j, \mathsf{N}'_j, \mathsf{A}'_j, \mathsf{C}'_j, \mathsf{T}'_j)$ **then**
  $\boxed{\mathsf{bad}_1 \leftarrow 1}$
end

Algorithm 2: Ideal World (Offline Phase for Ascon-256.v3): Encryption Query Internal States

for $i \in [q_e]$ **do**
  $(\mathsf{A}_{i,1}, \ldots, \mathsf{A}_{i,a_i}) \xleftarrow{\text{r}} \mathsf{pad}_1(\mathsf{N}_i, \mathsf{A}_i)$; $(\mathsf{M}_{i,1}, \ldots, \mathsf{M}_{i,m_i}) \xleftarrow{\text{r}} \mathsf{M}_i$; $(\mathsf{C}_{i,1}, \ldots, \mathsf{C}_{i,m_i}) \xleftarrow{\text{r}} \mathsf{C}_i$
  $t_i = a_i + m_i$, $d_i = |\mathsf{M}_{i,m_i}|$
  $\mathsf{V}_{i,0}, \ldots, \mathsf{V}_{i,a_i-1} \xleftarrow{\$} \{0,1\}^b$, $\mathsf{Z}_{i,a_i+1}, \ldots, \mathsf{Z}_{i,t_i-1} \xleftarrow{\$} \{0,1\}^c$, $\delta_i^* \xleftarrow{\$} \{0,1\}^{r-d_i}$
  if $a_i > 0$ **then**
    for $j = 1$ **to** a_i **do**
      $\mathsf{U}_{i,j} = \mathsf{V}_{i,j-1} \oplus (\mathsf{A}_{i,j} \parallel 0^c)$
    end
    $\mathsf{V}_{i,a_i} = (\mathsf{C}_{i,1} \oplus \mathsf{M}_{i,1}) \parallel \mathsf{Z}_{i,a_i}$
  end
  if $m_i \geq 2$ **then**
    $\mathsf{U}_{i,a_i+1} = \mathsf{C}_{i,1} \parallel (\mathsf{Z}_{i,a_i} \oplus 0^{c-1}1)$
    for $j = 2$ **to** $m_i - 2$ **do**
      $\mathsf{U}_{i,a_i+j} = \mathsf{C}_{i,j} \parallel \mathsf{Z}_{i,a_i+j-1}$
    end
    for $j = 1$ **to** $m_i - 3$ **do**
      $\mathsf{V}_{i,a_i+j} = (\mathsf{C}_{i,j+1} \oplus \mathsf{M}_{i,j+1}) \parallel \mathsf{Z}_{i,a_i+j-1}$
    end
    $\mathsf{V}_{i,t_i-1} = (\mathsf{C}_{i,m_i} \oplus \mathsf{M}_{i,m_i}) \parallel \delta_i^* \parallel \mathsf{Z}_{i,t_i-1}$
  end
end
return $\boxed{\mathsf{P}_E = \{(\mathsf{U}_{i,j}, \mathsf{V}_{i,j}) : i \in [q_e], j \in [t_i - 1]\}}$
if $\exists (i,j) \neq (i',j')$ *with* $\mathsf{U}_{i,j} = \mathsf{U}_{i',j'}$ *or* $\mathsf{V}_{i,j} = \mathsf{V}_{i',j'}$ **then**
  $\boxed{\mathsf{bad}_2 \leftarrow 1}$
end

Algorithm 3: Ideal World (Offline Phase for Ascon-256.v3): Decryption Query Internal States (Extension of P)

for $i \in [q_d]$ **do**
  $\left(\mathsf{A}'_{i,1}, \ldots, \mathsf{A}'_{i,a_i}\right) \leftarrow \mathsf{pad}_1(\mathsf{N}'_i, \mathsf{A}'_i)$; $\left(\mathsf{C}'_{i,1}, \ldots, \mathsf{C}'_{i,c_i}\right) \leftarrow \mathsf{C}'_i$
  if $\nexists j \in [q_e]$ *with* $\mathsf{u}_j = \mathsf{u}'_i$ **then**
    $p_i \leftarrow -1$
  else if $\nexists j \in [q_e]$ *with* $(\mathsf{u}_j, \mathsf{N}_j) = (\mathsf{u}'_i, \mathsf{N}'_i)$ **then**
    $p_i \leftarrow 0$
  else
    $p_i \leftarrow$ LCP of $\left(\mathsf{A}'_{i,1}, \ldots, (\mathsf{A}'_{i,a'_i}, *), \mathsf{C}'_{i,1}, \ldots, \mathsf{C}'_{i,c_i-2}\right)$ and $\left(\mathsf{A}_{j,1}, \ldots, (\mathsf{A}_{j,a_j}, *), \mathsf{C}_{j,1}, \ldots, \mathsf{C}_{j,m_j-2}\right)$
  end
end
for $i \in [q_d]$ *with* $p_i = -1$ **do**
  if $(\mathsf{u}'_i, \mathsf{N}'_i) = (\mathsf{u}_j, \mathsf{N}_j)$ *for some* $j \in [i-1]$ **then**
    $\mathsf{V}'_{i,0} \leftarrow \mathsf{V}_{j,0}$
  else
    $\mathsf{V}'_{i,0} \xleftarrow{\$} \{0,1\}^b$
  end
  if $a'_i > 0$ **then**
    Run $\mathsf{xorRand\_Extn}^P(\mathsf{V}'_{i,0}, (\mathsf{A}'_{i,1}, \ldots, \mathsf{A}'_{i,a'_i}))$
  end
  if $c_i > 1$ **then**
    Run $\mathsf{Rand\_Extn}^P(\mathsf{V}'_{i,a'_i} \oplus 0^*1, \mathsf{C}'_{i,1} \| \ldots \| \mathsf{C}'_{i,c_i-2})$
  end
end
for $i \in [q_d]$ *with* $0 \leq p_i \leq a'_i$ **do**
  $\mathsf{V}'_{i,p_i} \leftarrow \mathsf{V}_{j,p_i}$ where $(\mathsf{u}'_i, \mathsf{N}'_i) = (\mathsf{u}_j, \mathsf{N}_j)$
  if $a'_i > p_i$ **then**
    Run $\mathsf{xorRand\_Extn}^P(\mathsf{V}'_{i,p_i}, (\mathsf{A}'_{i,p_i+1}, \ldots, \mathsf{A}'_{i,a'_i}))$
  end
  if $c_i > 1$ **then**
    Run $\mathsf{Rand\_Extn}^P(\mathsf{V}'_{i,a'_i} \oplus 0^*1, \mathsf{C}'_{i,1} \| \ldots \| \mathsf{C}'_{i,c_i-2})$
  end
end
for $i \in [q_d]$ *with* $a'_i < p_i < t_i - 1$ **do**
  $\mathsf{V}'_{i,p_i} \leftarrow \mathsf{V}_{j,p_i}$ where $(\mathsf{u}'_i, \mathsf{N}'_i) = (\mathsf{u}_j, \mathsf{N}_j)$
  if $p_i < t_i - 2$ **then**
    Run $\mathsf{Rand\_Extn}^P(\mathsf{V}'_{i,p_i} \oplus 0^*1, \mathsf{C}'_{i,p_i-a'_i+1} \| \ldots \| \mathsf{C}'_{i,c_i-2})$
  end
end
return $\boxed{\mathsf{P}_1 = \{(\mathsf{U}'_{i,j}, \mathsf{V}'_{i,j})\}_{i \in [q_d], j \in [t_i-1]}}$
if P_1 *not injective* **then**
  $\boxed{\mathsf{bad}_3 \leftarrow 1}$
end
if $\mathsf{domain}(\mathsf{P}_1) \cap \mathsf{domain}(\mathsf{P}_\mathsf{E}) \neq \emptyset \ \vee \ \mathsf{range}(\mathsf{P}_1) \cap \mathsf{range}(\mathsf{P}_\mathsf{E}) \neq \emptyset$ **then**
  $\boxed{\mathsf{bad}_4 \leftarrow 1}$
end
Extend: $\boxed{\mathsf{P}_2 \leftarrow \mathsf{P}_E \sqcup \mathsf{P}_1}$

Algorithm 4: Ideal World (Offline Phase for Ascon-256.v3): Key Sampling and Initialization

$\mathsf{K}_1, \ldots, \mathsf{K}_\mu \xleftarrow{\$} \{0,1\}^\kappa$ where $\mathsf{K}_i = \mathsf{K}_{i,1} \parallel \mathsf{K}_{i,2}$
$\mathcal{J} \leftarrow \{j \in [q_d] : \mathsf{u}'_j \neq \mathsf{u}_i \ \forall i\}$
for $i \in [q_e]$ **do**
  $\mathsf{I}_i \leftarrow IV \parallel K_{i,2} \parallel K_{i,1}$
  $\mathsf{O}_i \leftarrow \mathsf{V}_{i,0} \oplus (\mathsf{K}_{i,1} \parallel 0^{b-\kappa} \parallel K_{i,2})$
end
for $j \in \mathcal{J}$ **do**
  $\mathsf{I}'_j \leftarrow IV \parallel K_{j,2} \parallel K_{j,1}$
  $\mathsf{O}'_j \leftarrow \mathsf{V}'_{j,0} \oplus (\mathsf{K}_{i,1} \parallel 0^{b-\kappa} \parallel K_{i,2})$
end
for $j \in [q_d] \setminus \mathcal{J}$ **do**
  $\mathsf{I}'_j \leftarrow \mathsf{I}_i$
  $\mathsf{O}'_j \leftarrow \mathsf{O}_i$
end
return $\boxed{\mathsf{P}_{\mathsf{init}} \leftarrow \big((\mathsf{I}_i, \mathsf{O}_i)_{i \in [q_e]}, (\mathsf{I}'_j, \mathsf{O}'_j)_{j \in \mathcal{J}}\big)}$
if $\exists i \neq j \in [\mu]$ *with* $\mathsf{K}_i = \mathsf{K}_j$ **then**
  $\boxed{\mathsf{bad}_5 \leftarrow 1}$
end
if $\mathsf{domain}(\mathsf{P}_2) \cap \mathsf{domain}(\mathsf{P}_{\mathsf{init}}) \neq \emptyset \ \vee \ \mathsf{range}(\mathsf{P}_2) \cap \mathsf{range}(\mathsf{P}_{\mathsf{init}}) \neq \emptyset$ **then**
  $\boxed{\mathsf{bad}_6 \leftarrow 1}$
end
Extend : $\boxed{\mathsf{P}_3 \leftarrow \mathsf{P}_2 \sqcup \mathsf{P}_{\mathsf{init}}}$

Algorithm 5: Ideal World (Offline Phase for Ascon-256.v3): Finalization- Last Two Blocks

for *encryption query* $i \in [q_e]$ **do**
  $m_i \leftarrow \lceil \frac{|\mathsf{M}_i|+1}{r} \rceil$
  $d_i \leftarrow |\mathsf{M}_i| \bmod r$
  if $m_i \geq 2$ **then**
    $\mathsf{V}_{i,a_i+m_i-2} \leftarrow \left(\mathsf{C}_{i,m_i-1} \oplus \mathsf{M}_{i,m_i-1} \oplus (0^{d_i} \parallel \lceil \mathsf{K}_{i,1} \rceil_{r-d_i})\right) \parallel \mathsf{Z}_{a_i+m_i-2}$
    $\mathsf{F}_{i,0} \leftarrow \left(\mathsf{C}_{i,m_i-1} \oplus (0^{d_i} \parallel \lceil \mathsf{K}_{i,1} \rceil_{r-d_i})\right) \parallel \lfloor \mathsf{V}_{a_i+m_i-2} \rfloor_c$
    $\mathsf{V}_{i,a_i+m_i-1} \leftarrow (\mathsf{C}_{i,m_i} \oplus \mathsf{M}_{i,m_i} \oplus \lfloor \mathsf{K}_{i,1} \rfloor_{d_i}) \parallel \delta_i^* \parallel \mathsf{Z}_{a_i+m_i-1}$
    $\mathsf{F}_{i,1} \leftarrow (\mathsf{C}_{i,m_i} \oplus \lfloor \mathsf{K}_{i,1} \rfloor_{d_i}) \parallel \delta_i^* \parallel \lfloor \mathsf{V}_{a_i+m_i-1} \rfloor_c$
  end
  if $m_i = 1$ **then**
    $\mathsf{F}_{i,1} \leftarrow (\mathsf{C}_{i,1} \oplus \lfloor \mathsf{K}_{i,1} \rfloor_{d_i}) \parallel \delta_i^* \parallel (\mathsf{Z}_{i,a_i} \oplus 0^{c-1}1)$
    $\mathsf{F}_{i,0} \leftarrow \mathsf{U}_{i,a_i}$
  end
end
for *decryption query* $i \in [q_d]$ **do**
  if $c_i \geq 2$ **then**
    $\mathsf{F}'_{i,0} \leftarrow \left(\mathsf{C}'_{i,c_i-1} \oplus (0^{d_i} \parallel \lceil \mathsf{K}_{i,1} \rceil_{r-d_i})\right) \parallel \lfloor \mathsf{V}'_{a'_i+c_i-2} \rfloor_c$
    $\mathsf{V}'_{i,a'_i+c_i-1} \leftarrow \left(\mathsf{C}'_{i,c_i} \oplus \mathsf{M}'_{i,m'_i} \oplus \lfloor \mathsf{K}_{i,1} \rfloor_{d_i}\right) \parallel \delta_i'^* \parallel \mathsf{Z}'_{a'_i+c_i-1}$
    $\mathsf{F}'_{i,1} \leftarrow \left(\mathsf{C}'_{i,c_i} \oplus \lfloor \mathsf{K}_{i,1} \rfloor_{d_i}\right) \parallel 10^* \parallel \lfloor \mathsf{V}'_{a'_i+c_i-1} \rfloor_c$
  end
  if $c_i = 1$ **then**
    $\mathsf{F}'_{i,1} \leftarrow \left(\mathsf{C}'_{i,c_i} \oplus \lfloor \mathsf{K}_{i,1} \rfloor_{d_i}\right) \parallel 10^* \parallel \left(\lfloor \mathsf{V}'_{a'_i+c_i-1} \rfloor_c \oplus 0^{c-1}1\right)$
    $\mathsf{F}'_{i,0} \leftarrow \mathsf{U}'_{i,a_i}$
  end
end
return $\boxed{\mathsf{P}_F \leftarrow \{(\mathsf{F}_{i,0}, \mathsf{V}_{i,t_i-1})\}_{i\in[q_e]} \cup \{(\mathsf{F}'_{i,0}, \mathsf{V}'_{a'_i+c_i-1})\}_{i\in[q_d]}}$
if P_F *not injective* **then**
  $\boxed{\mathsf{bad}_7 \leftarrow 1}$
end
if $\mathsf{domain}(\mathsf{P}_F) \cap \mathsf{domain}(\mathsf{P}_3) \neq \emptyset \ \vee \ \mathsf{range}(\mathsf{P}_F) \cap \mathsf{range}(\mathsf{P}_3) \neq \emptyset$ **then**
  $\boxed{\mathsf{bad}_8 \leftarrow 1}$
end
if $\exists i \neq j \in [q_e]$ *with* $\mathsf{F}_{i,1} = \mathsf{F}_{j,1}$ ***or*** $\exists i \neq j \in [q_d]$ *with* $\mathsf{F}'_{i,1} = \mathsf{F}'_{j,1}$ **then**
  $\boxed{\mathsf{bad}_9 \leftarrow 1}$
end
if $\exists (i,j) \in [q_e] \times [q_d]$ *with* $(\mathsf{F}'_{i,1}, \mathsf{T}'_i) = (\mathsf{F}_{j,1}, \mathsf{T}_j)$ **then**
  $\boxed{\mathsf{bad}_{10} \leftarrow 1}$
end
Extend : $\boxed{\mathsf{P}_4 \leftarrow \mathsf{P}_3 \sqcup \mathsf{P}_F}$

Algorithm 6: Ideal World (Offline Phase for Ascon-256.v3): Finalization- Tag Consistency

for *encryption query* $i \in [q_e]$ **do**
  $\mathsf{X}_i \leftarrow \mathsf{F}_i \oplus 0^{b-\kappa_2} \parallel \mathsf{K}_{u_i,2}$
  $\mathsf{Y}_i \leftarrow \alpha_i \parallel (\mathsf{T}_i \oplus \mathsf{K}_{u_i,2})$ where $\alpha_i \xleftarrow{\$} \{0,1\}^{b-\tau}$
end
return $\boxed{\mathsf{P}_{\mathsf{tag}} \leftarrow (\mathsf{X}_i, \mathsf{Y}_i)_{i\in[q_e]}}$
if $\mathsf{domain}(\mathsf{P}_{\mathsf{tag}}) \cap \mathsf{domain}(\mathsf{P}_4) \neq \emptyset \ \vee \ \mathsf{range}(\mathsf{P}_{\mathsf{tag}}) \cap \mathsf{range}(\mathsf{P}_4) \neq \emptyset$ **then**
  $\boxed{\mathsf{bad}_{11} \leftarrow 1}$
end
Extend : $\boxed{\mathsf{P}_5 \leftarrow \mathsf{P}_4 \sqcup \mathsf{P}_{\mathsf{tag}}}$
for *decryption query* $i \in [q_d]$ **do**
  $\mathsf{X}'_i \leftarrow \mathsf{F}'_i \oplus 0^{b-\kappa_2} \parallel \mathsf{K}_{u'_i,2}$
  if $\mathsf{X}'_i \in \mathsf{domain}(\mathsf{P}_5)$ **then**
    $\mathsf{Y}'_i \leftarrow \mathsf{P}_4(\mathsf{X}'_i)$
    if $\lfloor \mathsf{P}_4(\mathsf{X}'_i) \rfloor_\tau \oplus \mathsf{K}_{u'_i,2} = \mathsf{T}'_i$ **then**
      $\boxed{\mathsf{bad}_{12} \leftarrow 1}$
    end
  else
    $\mathsf{Y}'_i \xleftarrow{\$} \{0,1\}^b$
    if $\lfloor \mathsf{Y}'_i \rfloor_\tau \oplus \mathsf{K}_{u'_i,2} = \mathsf{T}'_i$ **then**
      $\boxed{\mathsf{bad}_{13} \leftarrow 1}$
    end
  end
end
Extend : $\boxed{\mathsf{P}_{\mathsf{fin}} \leftarrow \mathsf{P}_5 \sqcup \{(\mathsf{X}'_i, \mathsf{Y}'_i)_{i\in[q_d]}\}}$
Ideal World Transcript :
$\boxed{\Theta_{\text{ideal}} \leftarrow \big((\mathsf{u}_i, \mathsf{N}_i, \mathsf{A}_i, \mathsf{M}_i, \mathsf{C}_i, \mathsf{T}_i)_{i\in[q_e]}, (\mathsf{u}'_i, \mathsf{N}'_i, \mathsf{A}'_i, \mathsf{C}'_i, \mathsf{T}'_i, \mathsf{M}'_i)_{i\in[q_d]}, \mathsf{P}_{\mathsf{fin}}\big)}$

Bad Analysis

Let us define,

$$\mathsf{bad}_* = \bigcup_{i=1}^{6} \mathsf{bad}_i \vee \bigcup_{j=10}^{13} \mathsf{bad}_j$$

Note that, bad_* is essentially same with the bad events defined in the earlier works of [2,3]. Thus, the similar bounds should hold true. We summarize the result in the following Lemma.

Lemma 5.

$$\begin{aligned}\Pr\left[\mathsf{bad}_* = 1\right] \leq \frac{q_d}{2^\tau} + \frac{\sigma_e^2}{2^b} + \frac{\sigma_d(\sigma_d + q_p)}{2^b} + \frac{\mathsf{mcoll}(\sigma_e, 2^r) \times (\sigma_d + q_p)}{2^c} + \frac{\mu^2}{2^\kappa} + \frac{\mu(q_p + \sigma)}{2^\kappa} \\ + \frac{q_d^2 + q_e^2 + q_e q_d + (q_e + q_d)(\sigma + q_p)}{2^b} + \frac{\mathsf{mcoll}(\sigma + q_p, 2^\tau) q_d}{2^{\kappa_2}} \\ + \frac{\mathsf{mcoll}(q_e, 2^{b-\kappa_2})(\sigma + q_p)}{2^{\kappa_2}} + \frac{q_e(\sigma + q_p)}{2^b} + \frac{\mathsf{mcoll}(\sigma + q_p, 2^\tau) q_d}{2^{\kappa_2}} + \frac{q_d}{2^\tau}\end{aligned}$$

Proof. Proof of this lemma directly follows from the works of Chakraborty et al. [2,3].

Hence we establish upper bounds on the probability of newly encountered bad events, i.e., $\mathsf{bad}_7, \mathsf{bad}_8$ and bad_9 (See Algorithm 5) dedicated to Ascon-256.v3.

Lemma 6. $\Pr\left[\mathsf{bad}_7 = 1\right] \leq \dfrac{(q_e + q_d)^2}{2^b}$

Proof. From the construction of P_F we have that $\mathsf{domain}(\mathsf{P}_F) \leq (q_e + q_d)$ and $\mathsf{range}(\mathsf{P}_F) \leq (q_e + q_d)$. For any two values $x, y \in \mathsf{domain}(\mathsf{P}_F)$ (or $x, y \in \mathsf{range}(\mathsf{P}_F)$) we have that $x = y$ with probability $\dfrac{1}{2^b}$. Hence, by union bound we derive the lemma.

Lemma 7. $\Pr\left[\mathsf{bad}_8 = 1\right] \leq \dfrac{\mathsf{mcoll}(\sigma + q_p, 2^r) \times (q_e + q_d)}{2^c}$

Proof. Let ρ_1 (and ρ_2) denote the multicollision on the values of $\lceil x \rceil_r$ for all $x \in \mathsf{domain}(\mathsf{P}_3)$ (and for all $x \in \mathsf{range}(\mathsf{P}_3)$, respectively). Then, by randomness of randomised extension process and xor randomised extension process we have that

$$\Pr(\mathsf{bad}_8 = 1 \mid \max\{\rho_1, \rho_2\} = \rho) \leq \frac{\rho \times (q_e + q_d)}{2^c}$$

Hence by using the expectation on ρ_1 we obtain,

$$\Pr(\mathsf{bad}_8 = 1) \leq \frac{\mathsf{mcoll}(\sigma + q_p, 2^r) \times (q_e + q_d)}{2^c}$$

Lemma 8. $\Pr\left[\mathsf{bad}_9 = 1\right] \leq \dfrac{q_e^2 + q_d^2}{2^b}$

Proof. The proof is straightforward.

Good Analysis

Let θ be a good transcript (no bad events occur). Note that we sample either inputs or outputs of $\mathsf{P}_{\mathsf{fin}} \setminus \mathsf{P}$ uniformly. Thus,

$$\Pr\left[\Theta_{ideal} = \theta\right] = \Pr\left[P \subseteq \Pi\right] \times 2^{-b(|\mathsf{P}_{\mathsf{fin}}|-|\mathsf{P}|)} \leq \frac{1}{(2^b)_{|\mathsf{P}_{\mathsf{fin}}|}} = \Pr\left[\Theta_{real} = \theta\right]$$

Thus by H-Coefficient Technique we conclude the proof of this theorem.

8 Conclusion

In this draft, we revisit the mu-Ascon instantiation proposed by NIST for multi-user applications, which only provides partial security against key-recovery under state recovery attacks and doesn't provide any commitment security. Dealing with these weaknesses, we generate two schemes which we call Ascon-256.v2 and Ascon-256.v3 for the multi-user settings, which are compatible with the AsconAEAD128 API. Due to the usage of a single extra Ascon permutation, Ascon-256.v2 and Ascon-256.v3 have a very negligible performance difference in comparison to that of mu-Ascon. On the other hand we show that in compensation to these negligible expense in performance, Ascon-256.v2 resists CMT-4 attacks and Ascon-256.v3 resist CMT-4 attacks and at the same time also enjoys the full key-binding property.

References

1. Mihir Bellare and Viet Tung Hoang. Efficient schemes for committing authenticated encryption. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022, Part II*, volume 13276 of *Lecture Notes in Computer Science*, pages 845–875, Trondheim, Norway, May 30 – June 3, 2022. Springer, Cham, Switzerland. doi:10.1007/978-3-031-07085-3_29.
2. Bishwajit Chakraborty, Chandranan Dhar, and Mridul Nandi. Exact security analysis of ASCON. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology – ASIACRYPT 2023, Part III*, volume 14440 of *Lecture Notes in Computer Science*, pages 346–369, Guangzhou, China, December 4–8, 2023. Springer, Singapore, Singapore. doi:10.1007/978-981-99-8727-6_12.
3. Bishwajit Chakraborty, Chandranan Dhar, and Mridul Nandi. Tight multi-user security of ascon and its large key extension. Lecture Notes in Computer Science, pages 57–76. Springer, Cham, Switzerland, December 1–3, 2024. doi:10.1007/978-981-97-5025-2_4.
4. Bishwajit Chakraborty, Ashwin Jha, and Mridul Nandi. On the security of sponge-type authenticated encryption modes. *IACR Transactions on Symmetric Cryptology*, pages 93–119, 07 2020. doi:10.46586/tosc.v2020.i2.93-119.
5. Shan Chen and John P. Steinberger. Tight security bounds for key-alternating ciphers. In *Advances in Cryptology - EUROCRYPT 2014. Proceedings*, pages 327–350, 2014.

6. Christoph Dobraunig and Bart Mennink. Generalized initialization of the duplex construction. In Christina Pöpper and Lejla Batina, editors, *ACNS 24: 22nd International Conference on Applied Cryptography and Network Security, Part II*, volume 14584 of *Lecture Notes in Computer Science*, pages 460–484, Abu Dhabi, UAE, March 5–8, 2024. Springer, Cham, Switzerland. doi:10.1007/978-3-031-54773-7_18.
7. Yevgeniy Dodis, Paul Grubbs, Thomas Ristenpart, and Joanne Woodage. Fast message franking: From invisible salamanders to encryptment. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 155–186, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Cham, Switzerland. doi:10.1007/978-3-319-96884-1_6.
8. Morris Dworkin. Nist special publication 800-38b recommendation for block.
9. Pooya Farshim, Claudio Orlandi, and Răzvan Roşie. Security of symmetric primitives under incorrect usage of keys. *IACR Transactions on Symmetric Cryptology*, 2017(1):449–473, 2017. doi:10.13154/tosc.v2017.i1.449-473.
10. Paul Grubbs, Jiahui Lu, and Thomas Ristenpart. Message franking via committing authenticated encryption. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part III*, volume 10403 of *Lecture Notes in Computer Science*, pages 66–97, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Cham, Switzerland. doi:10.1007/978-3-319-63697-9_3.
11. Charlotte Lefevre and Bart Mennink. Generic security of the ascon mode: On the power of key blinding. Cryptology ePrint Archive, Paper 2023/796, 2023. URL: https://eprint.iacr.org/2023/796.
12. Charlotte Lefevre and Bart Mennink. Generic security of the ascon mode: On the power of key blinding. Cryptology ePrint Archive, Report 2023/796, 2023. URL: https://eprint.iacr.org/2023/796.
13. Julia Len, Paul Grubbs, and Thomas Ristenpart. Partitioning oracle attacks. In *30th USENIX security symposium (USENIX Security 21)*, pages 195–212, 2021.
14. Sanketh Menda, Julia Len, Paul Grubbs, and Thomas Ristenpart. Context discovery and commitment attacks - how to break CCM, EAX, SIV, and more. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023, Part IV*, volume 14007 of *Lecture Notes in Computer Science*, pages 379–407, Lyon, France, April 23–27, 2023. Springer, Cham, Switzerland. doi:10.1007/978-3-031-30634-1_13.
15. Bart Mennink and Samuel Neves. Encrypted davies-meyer and its dual: Towards optimal security using mirror theory. In *Advances in Cryptology - CRYPTO 2017. Proceedings, Part III*, pages 556–583, 2017.
16. Yusuke Naito, Yu Sasaki, and Takeshi Sugawara. Committing security of ascon: Cryptanalysis on primitive and proof on mode. *IACR Transactions on Symmetric Cryptology*, 2023(4):420–451, 2023.
17. Yoav Nir and Adam Langley. Chacha20 and poly1305 for ietf protocols. Technical report, 2015.
18. NIST. Submission requirements and evaluation criteria for the Lightweight Cryptography Standardization Process, 2018. https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/final-lwc-submission-requirements-august2018.pdf.
19. NIST. the third NIST workshop on block cipher modes of operation, 2023. https://csrc.nist.gov/Events/2023/third-workshop-on-block-cipher-modes-of-operation.

20. Jacques Patarin. *Etude des Générateurs de Permutations Pseudo-aléatoires Basés sur le Schéma du DES*. PhD thesis, Université de Paris, 1991.
21. Jacques Patarin. The "coefficients H" technique. In *Selected Areas in Cryptography - SAC 2008. Revised Selected Papers*, pages 328–345, 2008.
22. Meltem Sönmez Turan, Kerry McKay, Donghoon Chang, Jinkeon Kang, and John Kelsey. Ascon-based lightweight cryptography standards for constrained devices: Authenticated encryption, hash, and extendable output functions. Technical report, National Institute of Standards and Technology, 2024.

A Graph Structures for Functions

The details of this section are thoroughly discussed in [2]. We provide a summary here as we will refer to the randomized extension algorithms in subsequent discussions.

A.1 Partial Function Graph

A *partial function* $\mathcal{L} : \{0,1\}^b \dashrightarrow \{0,1\}^c$ is a subset $\mathcal{L} = \{(p_1, q_1), \ldots, (p_t, q_t)\} \subseteq \{0,1\}^b \times \{0,1\}^c$ with distinct p_i values. An *injective partial function* has distinct q_i values. Define:

$$\mathsf{domain}(\mathcal{L}) = \{p_i : i \in [t]\}, \quad \mathsf{range}(\mathcal{L}) = \{q_i : i \in [t]\}$$

We write $\mathcal{L}(p_i) = q_i$ and for all $p \notin \mathsf{domain}(\mathcal{L}), \mathcal{L}(p) = \perp$.

For $f : \{0,1\}^b \dashrightarrow \{0,1\}^b$, $c \in [b-1]$, define $\lfloor f \rfloor_c : \{0,1\}^b \dashrightarrow \{0,1\}^c$ such that $\lfloor f \rfloor_c(x) = \lfloor f(x) \rfloor_c$ when $f(x) \neq \perp$.

Definition 6 (Partial Function Graph). *Let $\mathcal{L} : \{0,1\}^b \dashrightarrow \{0,1\}^c$ for $r := b - c > 0$. Define a labeled directed graph $G := G^{\mathcal{L}}$, called (labeled) partial function graph, over:*

$$V := \lfloor \mathsf{domain}(\mathcal{L}) \rfloor_c \cup \mathsf{range}(\mathcal{L}) \subseteq \{0,1\}^c$$

with labels $\{0,1\}^r$ and edges:

$$E(G) := \{u \xrightarrow{x} v \mid \mathcal{L}(x \| u) = v\}$$

We call it (labeled) function graph if $\mathcal{L}$ is known to be a function.

we write a walk,

$$u_0 \xrightarrow{x_1} u_1 \xrightarrow{x_2} \cdots \xrightarrow{x_{l-1}} u_{l-1} \xrightarrow{x_l} u_l$$

simply as $u_0 \xrightarrow{x^l} u_l$. If $u \xrightarrow{x} v_1$ and $u \xrightarrow{x} v_2$, then $v_1 = v_2$.

A.2 Sampling Process of a Labeled Walk

Let $f : \{0,1\}^b \dashrightarrow \{0,1\}^b$, $x := (x_1, \ldots, x_k)$ be a k-tuple label, $k \geq 0$, and $z_0 \in \{0,1\}^c$. Define a process $\mathsf{Rand\_Extn}^f(z_0, x^k)$ which extends f to complete the walk:

1. Initialize $f' = f$
2. For $j = 1$ to k:
   (a) $v_j = f'((x_j, z_{j-1}))$
   (b) If $v_j = \perp$:
      – $v_j \xleftarrow{\$} \{0,1\}^b$
      – $f' \leftarrow f' \cup \{(x_j \| z_{j-1}, v_j)\}$
   (c) $z_j = \lfloor v_j \rfloor_c$

Similarly, define $\mathsf{xorRand\_Extn}^{\mathcal{P}}(v_0, x^k)$ for $\mathcal{P}^{\oplus}$ where $v_0 \in \{0,1\}^b$ and $x_i \in \{0,1\}^r$:

1. Initialize $\mathcal{P}' = \mathcal{P}$
2. For $j = 1$ to k:
   (a) $v_j = \mathcal{P}'(v_{j-1} \oplus (x_j \| 0^c))$
   (b) If $v_j = \perp$:
      – $v_j \xleftarrow{\$} \{0,1\}^b$
      – $\mathcal{P}' \leftarrow \mathcal{P}' \cup \{(v_{j-1} \oplus (x_j \| 0^c), v_j)\}$

After this process obtain a modified partial function $\mathcal{P}' : \{0,1\}^b \dashrightarrow \{0,1\}^b$ with the walk,

$$v_0 \xrightarrow{x_1}_{\oplus} v_1 \xrightarrow{x_2}_{\oplus} v_2 \cdots \xrightarrow{x_{k-1}}_{\oplus} v_{k-1} \xrightarrow{x_k}_{\oplus} v_k$$