Tight Multi-user Security of Ascon and Its Large Key Extension *

Bishwajit Chakraborty[1], Chandranan Dhar[2], and Mridul Nandi[2]

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

Abstract. The Ascon cipher suite has recently become the preferred standard in the NIST Lightweight Cryptography standardization process. Despite its prominence, the initial dedicated security analysis for the Ascon mode was conducted quite recently. This analysis demonstrated that the Ascon AEAD mode offers superior security compared to the generic Duplex mode, but it was limited to a specific scenario: single-user nonce-respecting, with a capacity strictly larger than the key size. In this paper, we eliminate these constraints and provide a comprehensive security analysis of the Ascon AEAD mode in the multi-user setting, where the capacity need not be larger than the key size. Regarding data complexity D and time complexity T, our analysis reveals that Ascon achieves AEAD security when T is bounded by $\min\{2^\kappa/\mu, 2^c\}$ (where κ is the key size, and μ is the number of users), and DT is limited to 2^b (with b denoting the size of the underlying permutation, set at 320 for Ascon). Our results align with NIST requirements, showing that Ascon allows for a tag size as small as 64 bits while supporting a higher rate of 192 bits, provided the number of users remains within recommended limits. However, this security becomes compromised as the number of users increases significantly. To address this issue, we propose a variant of the Ascon mode called LK-Ascon, which enables doubling the key size. This adjustment allows for a greater number of users without sacrificing security, while possibly offering additional resilience against quantum key recovery attacks. We establish tight bounds for LK-Ascon, and furthermore show that both Ascon and LK-Ascon maintain authenticity security even when facing nonce-misuse adversaries.

1 Introduction

Authenticated Encryption (AE) serves as a fundamental component of symmetric cryptography, enabling the simultaneous encryption and authentication of a plaintext. Often, AE provides the capability to authenticate supplementary data, which, unlike the plaintext, is transmitted without encryption. In this context, AE is referred to as Authenticated Encryption with Associated Data (AEAD).

* This is the full version of the work accepted at ACISP 2024.

Extensive research has been dedicated in recent years to developing and scrutinizing efficient and secure AEAD algorithms. Notably, widely utilized AEAD mechanisms across the Internet include AES-GCM [MV04], which is used in TLS [SCM08], and Chacha20-Poly1305, XSalsa20-Poly1305 [Ber05,Ber08b,Ber08a].

A specific area of focus involves the application of AEAD schemes in lightweight cryptography, which has garnered significant attention in research over the past decade. This exploration has been primarily inspired by the CAESAR competition [Com14], emphasizing authenticated encryption design, and subsequently by the lightweight cryptography (LWC) competition [NIS18] hosted by the US National Institute of Standards and Technology (NIST).

Of particular interest in the realm of lightweight cryptography is the Ascon cipher suite, which has emerged as the winner of the CAESAR competition (in the lightweight applications category), and more recently in the NIST LWC competition. Initially proposed as a candidate in Round 1 of the CAESAR competition [Com14], subsequent iterations of Ascon (v1.1 and v1.2) incorporated minor modifications to the original design (version 1 [DEMS14]). The latest version, v1.2 [DEMS19], acclaimed as the victor of the NIST LWC competition, encompasses the Ascon-128 and Ascon-128a authenticated ciphers, alongside the Ascon-Hash hash function and the Ascon-Xof extendable output function. All the components within the suite ensure 128-bit security and utilize a shared 320-bit permutation internally, facilitating the implementation of both duplex-based AEAD and sponge-based extendable-output hashing using a single lightweight primitive.

1.1 Existing Security Analysis

The authenticated encryption mode of Ascon is based on the duplex construction [BDPA11], specifically the MonkeyDuplex construction [BDPA12]. However, in contrast to MonkeyDuplex, Ascon's mode integrates double-keyed initialization and double-keyed finalization to reinforce its resilience. For an elaborate depiction of the Ascon AEAD mode, please refer to Section 3.

Until recently, security analyses of Ascon predominantly regarded it as a variant of Duplex construction (as indicated in [DEMS19]). A common constraint in the existing analyses of Duplex constructions, is the condition $DT \ll 2^c$ (or comparable variations where D might be substituted by q_d), where D is the data complexity and T is the time complexity, c is the capacity of the underlying sponge and q_d is the number of decryption queries.

At Asiacrypt 2023, Chakraborty et al. [CDN23a] conducted the first dedicated security analysis of Ascon. They leveraged the double-keyed initialization and finalization of Ascon, demonstrating the removal of the term $DT/2^c$ for the Ascon AEAD. They achieved a bound of the order

$$\mathcal{O}\left(\frac{T}{2^\kappa} + \frac{D}{2^\tau} + \frac{DT}{2^b}\right).$$

The authors also demonstrated that their bound is tight. However, this bound was only attainable in the single-user nonce-respecting setting, where nonces

cannot be reused across encryption queries. Additionally, their analysis assumed that $\kappa < c$, i.e. the key size is strictly lesser than the capacity.[3]

In a concurrent work [ML23], Lefevre and Mennink also presented a dedicated security analysis of Ascon. While they focus on various settings (nonce-based confidentiality and authenticity, authenticity under nonce misuse and state recovery), they could only show the impact of strengthened initialization and finalization of Ascon in the case of authenticity under state recovery. However, in the case of conventional multi-user nonce-based authenticity, their bounds reduce to $\frac{q_d T}{2^c}$.

1.2 Our Contribution

In this work, we present a comprehensive analysis of the Ascon AEAD mode. Our first result establishes a tight AEAD security bound for Ascon in the multi-user nonce-respecting setting. Considering the number of users μ, tag size τ bits, key size κ bits, capacity c bits, and state size b bits, the derived bound is of the order

$$\mathcal{O}\left(\frac{\mu T}{2^\kappa} + \frac{D}{2^\tau} + \frac{DT}{2^b}\right).$$

Comparing this with the results of [CDN23a], we can see that although there is some multi-user security degradation, the term $DT/2^c$ can be overcome in this setting as well, thus improving over [ML23]. We also show that the achieved bound is tight. As an added bonus, we establish the aforementioned result for the $\kappa = c$ scenario as well, thus broadening the assumption to $\kappa \leq c$. This extension presents an added benefit: considering the NIST LWC requirements ($D \leq 2^{53}, T \leq 2^{112}$, $\kappa \geq 128$, $\tau \geq 64$), as long as the number of users does not become too large, our findings suggest that a capacity size of $c = 128$ (given $b = 320$) and $\tau = 64$ are adequate to ensure sufficient security for Ascon. This selection allows for a higher rate of 192 bits (constrained to 184 when $\kappa < c$), significantly enhancing efficiency while maintaining security within the parameters of the random permutation model.

In the nonce-misuse setting, where nonces can be reused for encryption queries, confidentiality cannot be guaranteed. However, our second result shows that as far as authenticity security is concerned, the bound is of the order

$$\mathcal{O}\left(\frac{\mu T}{2^\kappa} + \frac{D}{2^\tau} + \frac{DT}{2^b} + \frac{D^2}{2^c}\right).$$

This is also an improvement over [ML23], where the authors could not overcome the hurdle of $DT/2^c$ under any attack setting.

A significant drawback of the Ascon AEAD mode is its compromised security as the number of users increases, with the term $\mu T/2^\kappa$ becoming the

[3] In the original work [CDN23a], they initially assume $\kappa \leq c$ but later revise their assertions to $\kappa < c$ in the modified e-print version [CDN23b]. We delve into the intricacies of the case when $\kappa = c$ in Section 3.4.

dominant factor (due to the 128 bit size of the key). One simple solution to this limitation would be to increase the key size of Ascon. Moreover, a larger key size has the capability to enhance the resilience against key recovery attacks that utilize Grover's algorithm. However, key-size cannot be directly increased in the Ascon mode. For instance, if we opt for a nonce size of 128 bits, along with an extra 64-bit IV, the key-size becomes confined to 128 bits. The Ascon-80pq scheme was introduced as a component of the Ascon cipher suite, aiming to tackle this challenge. However, it should be noted that Ascon-80pq is only capable of accommodating a 160-bit key. As our final result, we introduce a novel AEAD mode, akin to Ascon, labeled as LK-Ascon (representing Large Key Ascon). This mode facilitates the doubling of the key size from 128 bits to 256 bits without requiring an increase in capacity, thus maintaining both security and efficiency. The resulting bound is of the order

$$\mathcal{O}\left(\frac{\mu T}{2^\kappa} + \frac{T}{2^c} + \frac{D}{2^{\min\{\tau,c\}}} + \frac{DT}{2^b}\right).$$

This bound is also tight. When nonces can be misused, the authnticity bound is again of the order

$$\mathcal{O}\left(\frac{\mu T}{2^\kappa} + \frac{D^2+T}{2^c} + \frac{D}{2^\tau} + \frac{DT}{2^b}\right).$$

A comparison among our results and the results of [CDN23a] and [ML23] can be found in Fig. 1.

Setting	**Security**	[CDN23a]	[ML23]	**This work**
su nr Ascon	AEAD	$\frac{T}{2^\kappa} + \frac{DT}{2^b}$	$\frac{T}{2^\kappa} + \frac{\sigma_d T}{2^c}$	$\frac{T}{2^\kappa} + \frac{DT}{2^b}$
mu nr Ascon	AEAD	-	$\frac{\mu T}{2^\kappa} + \frac{\sigma_d T}{2^c}$	$\frac{\mu T}{2^\kappa} + \frac{DT}{2^b}$
mu nm Ascon	Authenticity	-	$\frac{\mu T}{2^\kappa} + \frac{DT}{2^c}$	$\frac{\mu T}{2^\kappa} + \frac{DT}{2^b} + \frac{D^2}{2^c}$
mu nr LK-Ascon	AEAD	-	-	$\frac{\mu T}{2^\kappa} + \frac{DT}{2^b} + \frac{T}{2^c}$
mu nm LK-Ascon	Authenticity	-	-	$\frac{\mu T}{2^\kappa} + \frac{DT}{2^b} + \frac{D^2+T}{2^c}$

Fig. 1. Security Analysis Comparison. The expression $D/2^\tau$ is a common factor in all entries. "su" and "mu" represent single-user and multi-user, respectively. "nr" and "nm" denote nonce-respecting and nonce-misuse, respectively. The term σ_d refers to the data complexity of decryption queries.

1.3 Organization of the Paper

In section 2, we define the basic notations used in the paper. We give a brief description of the AEAD security in the random permutation model, and also briefly describe the H-coefficient technique. Moving forward, in S1ection 3, we present a detailed examination of the Ascon AEAD scheme. We present one of our two primary results, the security bound of Ascon, and establish its significance in relation to the NIST LWC criteria. To support our claims, we provide an interpretation of our findings within the context of the NIST guidelines, and discuss the tightness. Then, in Section 4, we present the authenticity security of Ascon in the nonce misuse setting. In Section 5, we define LK-Ascon, a variant of Ascon and state the security of LK-Ascon, along with a proof outline. Finally, in Section 6, we conclude the paper.

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).

$$\mathsf{pad}_1(X) = \begin{cases} \lambda & \text{if } |X| = 0 \\ X\|1\|0^{r-1-d} & \text{otherwise} \end{cases}$$

and

$$\mathsf{pad}_2(X) = X\|1\|0^{r-1-d}.$$

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

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

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 $\mathsf{AE} = (\mathsf{E}, \mathsf{D})$. 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:

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

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

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

For a key $K \in \mathcal{K}$, we use $\mathsf{E}_K(\cdot)$ and $\mathsf{D}_K(\cdot)$ to denote $\mathsf{E}(K, \cdot)$ and $\mathsf{D}(K, \cdot)$, 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*

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

Here $\mathscr{A}^{\mathsf{E}_{\mathsf{K}_i},\mathsf{D}_{\mathsf{K}_i},\Pi^{\pm}}$ denotes $\mathscr{A}$'s response after its interaction with $\mathsf{E}_{\mathsf{K}_i}$, $\mathsf{D}_{\mathsf{K}_i}$, 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 $\mathsf{E}_{\mathsf{K}_i}$, all $\mathsf{D}_{\mathsf{K}_i}$, 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.

Separation into Confidentiality and Authenticity For any AEAD scheme, the security can be separated into confidentiality and authenticity. In the mu setting, we have the following definitions:

Definition 2. *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-confidentiality advantage of an adversary $\mathscr{A}$ against AE_Π is defined as*

$$\mathbf{Adv}^{\mathsf{mu-conf}}_{\mathsf{AE}_\Pi}(\mathscr{A}) := \left| \Pr_{\substack{(\mathsf{K}_i)_{i=1}^{\mu} \xleftarrow{\$} \mathcal{K} \\ \Pi^{\pm}}} \left[\mathscr{A}^{(\mathsf{E}_{\mathsf{K}_i})_{i=1}^{\mu},\Pi^{\pm}} = 1\right] - \Pr_{\substack{(\Gamma_i)_{i=1}^{\mu} \\ \Pi^{\pm}}} \left[\mathscr{A}^{(\Gamma_i)_{i=1}^{\mu},\Pi^{\pm}} = 1\right] \right|,$$

and the mu-authenticity advantage of an adversary $\mathscr{A}$ against AE_Π is defined as

$$\mathbf{Adv}^{\mathsf{mu-auth}}_{\mathsf{AE}_\Pi}(\mathscr{A}) := \Pr_{\substack{(\mathsf{K}_i)_{i=1}^{\mu} \xleftarrow{\$} \mathcal{K} \\ \Pi^{\pm}}} \left[\mathscr{A}^{(\mathsf{E}_{\mathsf{K}_i},\mathsf{D}_{\mathsf{K}_i})_{i=1}^{\mu},\Pi^{\pm}} \text{ forges } \right].$$

In the context of authenticity, we use the term "$\mathscr{A}$ forges" to describe a situation where $\mathscr{A}$ successfully makes a query to one of its decryption oracles, and this query is not the result of a previous encryption query. By an easy reduction [DNT19], it can be shown that

$$\mathbf{Adv}^{\mathsf{mu-auth}}_{\mathsf{AE}_\Pi}(\mathscr{A}) \leq \left| \Pr_{\substack{(\mathsf{K}_i)_{i=1}^{\mu} \xleftarrow{\$} \mathcal{K} \\ \Pi^{\pm}}} \left[\mathscr{A}^{(\mathsf{E}_{\mathsf{K}_i},\mathsf{D}_{\mathsf{K}_i})_{i=1}^{\mu},\Pi^{\pm}} = 1\right] - \Pr_{\substack{(\Gamma_i)_{i=1}^{\mu} \\ \Pi^{\pm}}} \left[\mathscr{A}^{(\Gamma_i)_{i=1}^{\mu},\mathsf{rej},\Pi^{\pm}} = 1\right] \right|.$$

Proposition 1. *[BN08] There exist adversaries $\mathscr{A}$, $\mathscr{A}'$ and $\mathscr{A}''$ having same query complexities such that*

$$\mathbf{Adv}^{\mathsf{mu-aead}}_{\mathsf{AE}_\Pi}(\mathscr{A}) \leq \mathbf{Adv}^{\mathsf{mu-conf}}_{\mathsf{AE}_\Pi}(\mathscr{A}') + \mathbf{Adv}^{\mathsf{mu-auth}}_{\mathsf{AE}_\Pi}(\mathscr{A}'').$$

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, at the conclusion of 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, $\mathcal{O}_{\mathrm{re}} = (\mathsf{E_K}, \mathsf{D_K}, \Pi^{\pm})$, $\mathcal{O}_{\mathrm{id}} = (\Gamma, \mathsf{rej}, \Pi^{\pm})$, and the adversary aims to distinguish $\mathcal{O}_{\mathrm{re}}$ from $\mathcal{O}_{\mathrm{id}}$ in an AEAD sense.

Proposition 2 (H-coefficient technique [Pat91,Pat08]). *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:

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

A proof of Proposition 2 can be found in multiple papers including [Pat08,CS14,MN17].

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 [CJN20,CDN23a].

Lemma 1. *[CJN20] 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 $\mathsf{Y}_1, \ldots, \mathsf{Y}_q \xleftarrow{\mathsf{wor}} \{0,1\}^b$ and define $X_i := \lceil Y_i \rceil_r$ 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. *[CDN23a] 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 $(\mathsf{X}_1, \mathsf{Y}_1), \ldots, (\mathsf{X}_{q_1}, \mathsf{Y}_{q_1})$ and $(\mathsf{X}_{q_1+1}, \mathsf{Y}_{q_1+1}), \ldots, (\mathsf{X}_{q_1+q_2}, \mathsf{Y}_{q_1+q_2})$ 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 $\mathsf{Z}_i := \mathit{trunc}_\tau(\mathsf{X}_i) \oplus \mathit{trunc}'_\tau(\mathsf{Y}_i)$ for $i \in [q_1]$ and $\mathsf{Z}_i := (\mathsf{X}_i \oplus \mathsf{Y}_i)$ for $i \in [q_1 + 1, q]$, where trunc_τ and trunc'_τ represent some τ-bit truncations. For $\tau \geq 2$,*

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

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

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

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

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 $u \xrightarrow{x}_\oplus v$. A more detailed discussion on partial function graphs is given in Appendix A.

3 The Ascon AEAD mode

In this section, we introduce the Ascon AEAD [DEMS19] mode of operation, which is essentially a modified version of the Duplex construction. Let b represent the state size of the underlying permutation π, and consider $0 < r < b$ as the number of bits of associated data/message processed per permutation call. The term r is referred to as the rate of the Ascon construction, while $c = b - r$ is known as the capacity.

Let κ, ν, τ denote the key size, nonce size, and tag size, respectively, with the constraints: (i) $\tau \leq \kappa \leq c$, and (ii) $\kappa + \nu \leq b$. Note that, unlike [CDN23a], we assume $\kappa \leq c$, thus encompassing the $\kappa = c$ case.

We fix an $IV \in \{0,1\}^{b-\kappa-\nu}$. The AEAD utilizes a permutation π (the Ascon permutation), modeled as a random permutation for security analysis. Below, we provide a description of the encryption and decryption algorithms of the Ascon AEAD mode. For a visual representation of the encryption algorithm, refer to Fig. 2.

Encryption Algorithm. It receives an input of the form $(N, A, M) \in \{0,1\}^\nu \times \{0,1\}^* \times \{0,1\}^*$ and a key $K \in \{0,1\}^\kappa$. We divide the encryption algorithm into

three phases: (i) initialization, (ii) associated data and message processing, and (iii) tag generation, run sequentially.

Initialization. In this phase, we first apply the following function

$$\textsc{Init}^{\pi}(K,N) = \pi(IV\|K\|N) \oplus (0^{b-\kappa}\|K) := V_0.$$

Associated Data and Message Processing. We first parse the associated data and message:

$$(A_1,\ldots,A_a) \xleftarrow{r} \mathsf{pad}_1(A), \quad (M_1,\ldots,M_m) \xleftarrow{r} \mathsf{pad}_2(M).$$

Then, using the XOR-function graph corresponding to the function $\pi^{\oplus}$, we obtain a walk

$$V_0 \xrightarrow{A_1}_{\oplus} V_1 \xrightarrow{A_2}_{\oplus} \cdots \xrightarrow{A_a}_{\oplus} V_a, \quad V_a \oplus 0^*1 \xrightarrow{M_1}_{\oplus} V_{a+1} \cdots \xrightarrow{M_{m-1}}_{\oplus} V_{a+m-1}.$$

We define the ciphertext as follows:

$$C_i = \lceil V_{a+i-1} \rceil_r \oplus M_i, \quad \forall i \in [m], \quad C = \lceil C_1\|\cdots\|C_m \rceil_{|M|}.$$

We denote the above process as

$$\mathsf{AM\_Proc}^{\pi}(V_0, A, M) \to \big(C, F := V_{t-1} \oplus (M_m\|0^c)\big).$$

Tag Generation. Finally, we compute

$$T := \textsc{Tag}^{\pi}(K,F) = \lfloor \pi\big(F \oplus (0^r\|K\|0^{c-\kappa})\big) \rfloor_{\tau} \oplus \lfloor K \rfloor_{\tau}.$$

The Ascon AEAD returns (C,T).

Decryption Algorithm. The decryption algorithm consists of two steps. It first performs a verification algorithm to ensure the correctness of the ciphertext and tag pair. Upon successful verification, the algorithm advances to generate the corresponding message. Our emphasis is primarily on the verification process itself, rather than the specific steps involved in message computation.

On receiving an input of the form $(N,A,C,T) \in \{0,1\}^{\nu} \times \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^{\tau}$ and a key $K \in \{0,1\}^{\kappa}$, the steps of the verification process is outlined below:

1. $(A_1,\ldots,A_a) \xleftarrow{r} \mathsf{pad}_1(A)$ and $(C_1,\ldots,C_l) \xleftarrow{r} \mathsf{pad}_2(C)$.
2. Compute $V_0 := \textsc{Init}^{\pi}(K,N)$.
3. Compute the walk for the permutation π
$$V_0 \xrightarrow{A_1}_{\oplus} V_1 \xrightarrow{A_2}_{\oplus} \cdots \xrightarrow{A_a}_{\oplus} V_a.$$
4. Let $C_l = C_l'\|10^*$ for some C_l' (may be the empty string) and $|C_l'| = d$. Let $z_a = \lfloor V_a \oplus 0^*1 \rfloor_c$.
   - Case $l = 1$: Define $F = C_l' \parallel (\lfloor V_a \rfloor_{b-d} \oplus 10^*1)$.
   - Case $l \geq 2$: Compute
   $$z_a \xrightarrow{C_1} z_{a+1} \xrightarrow{C_2} \cdots \xrightarrow{C_{l-2}} z_{a+l-2}.$$
   Define $F = C_l' \parallel \lfloor \pi(C_{l-1}\|z_{a+l-2}) \oplus 10^* \rfloor_{b-d}$.
5. Reject if $T \neq \textsc{Tag}^{\pi}(K,F)$, otherwise, accept.

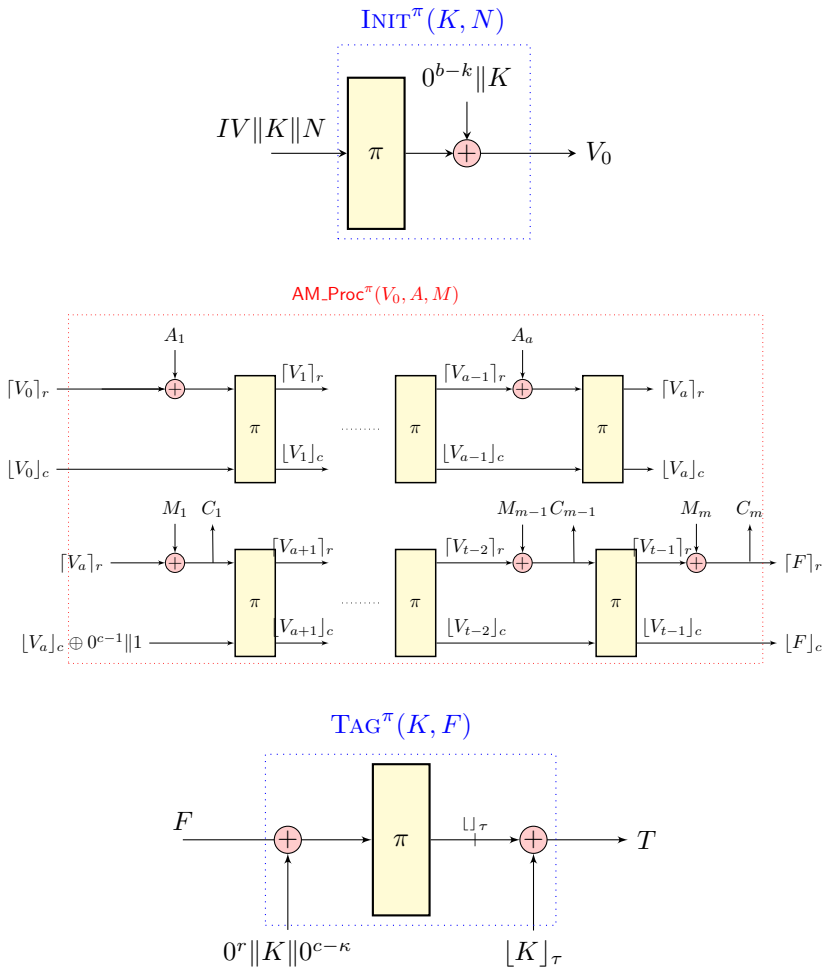

Fig. 2. Encryption in ASCON AEAD. The final ciphertext is $C = \lceil C_1\|\cdots\|C_m\rceil_{|M|}$, and tag is T. Here $t := a + m$.

3.1 Security bound of ASCON

Theorem 1. *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, and q_d decryption queries with a total number of σ_d data block. Define $\sigma := \sigma_e + \sigma_d$. Then, we can upper bound the mu-AEAD advantage of $\mathscr{A}$ against* ASCON *as*

follows:

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

3.2 Interpretation of Theorem 1

First, note that when $\mu = 1$ (corresponding to the su setting), the aforementioned bound aligns with the security bound established in [CDN23a]. In that work, the authors interpreted their bound in the context of the NIST LWC requirements, demonstrating the security of the Ascon mode even when $c = 136$ and $\tau = 64$, as opposed to the original description's requirement of $c \geq 192$ and $\tau = 128$.

As highlighted by the authors in [CDN23b], they were unable to reduce the capacity to 128 because their result was proven under the additional assumption that $\kappa < c$. Given that Ascon has a key size of 128 bits, the capacity needed to exceed 128 bits.

In this current work, we enhance the proof technique, and our proof encompasses the $\kappa = c$ scenario as well. Consequently, the Ascon mode remains secure even when $c = 128$. It is crucial to note that the design of Ascon prohibits $\kappa > c$, as the keys are XOR-ed in the capacity part.

Next, coming to the mu setting, note that the only extra terms in the security bound as compared to the su setting are $\frac{\mu^2}{2^\kappa}$ and $\frac{\mu(q_p+\sigma)}{2^\kappa}$. While the term $\frac{\mu^2}{2^\kappa}$ does not pose any threat, the term $\frac{\mu(q_p+\sigma)}{2^\kappa}$ reduces the security significantly as the number of users becomes large. For Ascon, the key size is 128, and according to NIST LWC specifications, q_p can be of the order 2^{112}. This does not leave room for a very large μ. For example, if the number of users is around 2^{15}, the advantage is less than half.

Hence, it is evident that in the mu setting, the security of the Ascon mode persists even when $c = 128$ and $\tau = 64$, provided the number of users does not reach excessively large values.

3.3 Tightness of the Bounds

We derive a bound of the following order:

$$\frac{\mu q_p}{2^\kappa} + \frac{\mu^2}{2^\kappa} + \frac{q_p}{2^c} + \frac{q_d}{2^c} + \frac{\sigma_e^2}{2^b} + \frac{\sigma_d^2}{2^b} + \frac{q_d}{2^\tau} + \frac{q_p\sigma_d}{2^b} + \frac{q_d}{2^\kappa}$$

As observed above, the only additional terms compared to the su setting, are $\mu q_p/2^\kappa$ and $\mu^2/2^\kappa$.

– The term $\frac{\mu q_p}{2^\kappa}$ corresponds to generic attacks which guesses one of the keys in primitive calls.
– The term $\frac{\mu^2}{2^\kappa}$ corresponds to generic attacks which guesses key collisions.

As attacks for the remaining terms were already shown in [CDN23a], our bounds are tight.

3.4 A Special Case: $\kappa = c$

Before delving into the proof of Theorem 1, let's first examine why the authors of [CDN23a] operated under the assumption $\kappa < c$. In the scenario where $\kappa = c$, in the special case of having only one message block and no associated data, the keys in the output of the initialization phase and the input of the finalization phase nullify each other. This interaction is illustrated in Fig. 3.

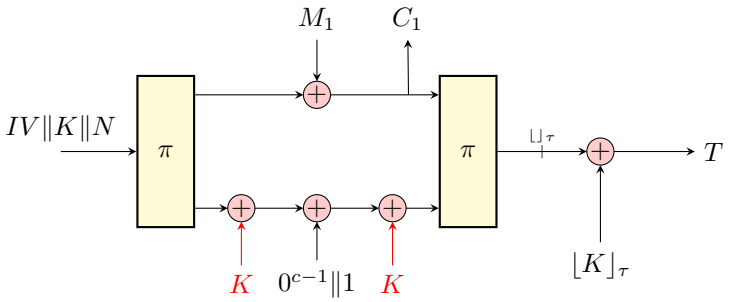

Fig. 3. Encryption in Ascon AEAD when $\kappa = c$, and we have only a single block message without any associated data. The two keys additions in red cancel each other out. The final ciphertext is $C = \lceil C_1 \rceil_{|M|}$, and tag is T.

We acknowledge this as a special case before initiating our proof to explain why the proof in [CDN23a] cannot be directly adapted to the multi-user setting with minor adjustments. Nevertheless, it is worth mentioning that our subsequent proof adheres to the general structure of [CDN23a], and some overlap is inevitable.

3.5 Proof Overview of Theorem 1

We employ the H-coefficient technique for our proof. In the real world, μ keys $K_1, \ldots, K_\mu$ and a random permutation Π are sampled independently. All queries are then responded to honestly. Extended transcript consists of:

– all inputs and outputs corresponding to encryption, decryption, and primitive queries,

– all inputs and outputs of the permutation calls corresponding to encryption and decryption queries.

The ideal world consists of an online phase and an offline phase. In the online phase:

– encryption queries responded to randomly,
– all decryption queries are rejected, and
– permutation queries are responded to faithfully.

The offline phase samples intermediate variables, and generates extended transcript. It proceeds in stages:

1. Start with permutation query transcript P.
2. Sample intermediate variables for encryption queries to obtain permutation input-output pairs P_E.
3. Randomly extend P to P_1 by setting input-outputs for decryption queries. Set $\mathsf{P}_2 := \mathsf{P}_1 \cup \mathsf{P}_\mathsf{E}$.
4. Finally, sample keys $K_1, \ldots, K_\mu$. Set input-output pairs for initialization first, and then move on to the finalization phase. Update P_2 twice to obtain $\mathsf{P}_{\mathsf{fin}}$.

In the offline phase of the ideal world, bad events occur when

– Variables sampled are not permutation-compatible.
– We have a correct forging.
– Decryption queries are not rejected.

Bounding the bad events conclude the proof. A detailed proof of the theorem is given in Appendix B.

4 Authenticity in the Nonce Misuse Setting

Up until now, we only considered the nonce-respecting setting, where no two encryption queries to the same user had the same nonce, although repetition of nonce across decryption queries was allowed. If the adversary reuses nonces for encryption queries, confidentiality cannot be guaranteed anymore, but we can still aim for authenticity. Considering the authenticity security of Ascon against adversaries that can possibly misuse nonces. We have the following result:

Theorem 2. *Consider a possibly nonce-misusing authentication adversary $\mathscr{A}$ making q_p permutation queries, q_e encryption queries with a total number of σ_e data blocks, and q_d decryption queries with a total number of σ_d data blocks. Define $\sigma := \sigma_e + \sigma_d$. Then, we can upper bound the mu-auth advantage of $\mathscr{A}$*

against Ascon *as follows:*

$$\begin{aligned}\mathbf{Adv}^{\mathrm{mu-auth}}_{\textsc{Ascon}}(\mathscr{A}) \le\ & \frac{\mu^2}{2^\kappa} + \frac{2q_d}{2^\tau} + \frac{\sigma_e^2}{2^b} + \frac{\sigma_d(q_p+\sigma_d)}{2^b} + \frac{\mathsf{mcoll}(\sigma_e, 2^r)(\sigma_d+q_p)}{2^c} \\ & + \frac{\mu(q_p+\sigma)}{2^\kappa} + \frac{\mathsf{mcoll}(q_e, 2^\tau)q_d}{2^c} + \frac{\mathsf{mcoll}(\sigma+q_p, 2^\tau)q_d}{2^\kappa} \\ & + \frac{q_e^2+q_d^2+q_eq_d+(q_e+q_d)(\sigma+q_p)}{2^b} + \frac{q_e(\sigma+q_p)}{2^b} \\ & + \frac{\mathsf{mcoll}(q_e, 2^{r+c-\kappa})(\sigma+q_p)}{2^\kappa} + \frac{q_d(\sigma+q_p)}{2^{c+\tau}}.\end{aligned}$$

This bound is the similar to that of Theorem 1, only the term $\sigma_e^2/2^b$ is replaced by $\sigma_e^2/2^c$. Note that in the lightweight setting, $\sigma_e^2 \ll 2^{128}$. This means Ascon maintains the authenticity security even under nonce misuse. The proof is very similar to that of the nonce-respecting setting, and is given in Appendix C. To show that the bound is tight, in Appendix F, we demonstrate a forgery that establishes the $\sigma_e^2/2^c$ bound.

5 Large Key Ascon

One of the major limitations of Ascon is its compromised security as the user count scales up. This issue is highlighted in Section 3.2, where the term $\frac{\mu q_p}{2^\kappa}$ emerges as the dominant factor in Theorem 1, thereby limiting the number of users. The most straightforward remedy for this constraint would involve augmenting the key size of Ascon. Additionally, a larger key size has the potential to fortify the resistance against key recovery attacks that leverage Grover's algorithm. It is essential to note that increasing the key size does not necessarily bolster security against quantum attacks in a generalized context, and the quantum security of any Ascon or related scheme must be assessed independently.

The key size cannot be directly increased in the original Ascon construction because of the constraints $\kappa + \nu \le b$ and $\kappa \le c$, where ν denotes the nonce size. For instance, if we opt for a nonce size of 128 bits, along with an extra 64-bit IV, the key size becomes confined to 128 bits. This limitation remains unchanged if we aim to permit a rate of 192 bits. Consequently, we introduce a novel AEAD mode akin to Ascon, known as the LK-Ascon mode, allowing for an arbitrary key size denoted as $\tau \le \kappa < b$, where τ signifies the tag size. To maintain consistency, we select an $IV \in \{0,1\}^{b-\kappa}$. The AEAD uses a permutation π (can be the same Ascon permutation), modeled to be the random permutation while we analyze its security.

Remark 3. While we define the mode for any $\tau \le \kappa < b$, we call this LK-Ascon (for Large Key Ascon) as this enables us to increase the key size from 128 bits to upto 320 bits. Of particular interest is the variant with key size 256 bits (allowing a 64-bit IV). We call this variant Ascon-256. Note that $\tau \le \kappa$ is necessary for masking the full tag.

Encryption Algorithm. It receives an input of the form $(N, A, M) \in \{0,1\}^{\nu} \times \{0,1\}^* \times \{0,1\}^*$ and a key $K \in \{0,1\}^{\kappa}$. We divide the encryption algorithm into the same three phases: (i) initialization, (ii) nonce, associated data and message processing, and (iii) tag generation, run sequentially.

Initialization. In this phase, we first apply the following function

$$\textsc{Init}^{\pi}(K) = \pi(IV \| K) \oplus (0^{b-\kappa} \| K) := V_0.$$

Note that the initialization process no longer takes the nonce N as an input. Here, it is processed with the associated data in the next step.

Nonce, Associated Data and Message Processing. We first parse them:

$$(A_1, \ldots, A_a) \xleftarrow{r} \mathsf{pad}_1(N, A), \quad (M_1, \ldots, M_m) \xleftarrow{r} \mathsf{pad}_2(M).$$

Note that a cannot be zero here, as even if there is no associated data, the nonce is parsed. As before, $m \geq 1$. Using the XOR-function graph corresponding to the function $\pi^{\oplus}$, we obtain a walk

$$V_0 \xrightarrow{A_1}_{\oplus} V_1 \xrightarrow{A_2}_{\oplus} \cdots \xrightarrow{A_a}_{\oplus} V_a, \quad V_a \oplus 0^*1 \xrightarrow{M_1}_{\oplus} V_{a+1} \cdots \xrightarrow{M_{m-1}}_{\oplus} V_{a+m-1}.$$

We define the ciphertext as follows:

$$C_i = \lceil V_{a+i-1} \rceil_r \oplus M_i, \quad \forall i \in [m], \quad C = \lceil C_1 \| \cdots \| C_m \rceil_{|M|}.$$

We denote the above process as

$$\mathsf{AM\_Proc}^{\pi}(V_0, N, A, M) \to \big(C, F := V_{t-1} \oplus (M_m \| 0^c)\big).$$

Tag Generation. Finally, we compute

$$T := \textsc{Tag}^{\pi}(K, F) = \lfloor \pi\big(F \oplus (K \| 0^{b-\kappa})\big) \rfloor_{\tau} \oplus \lfloor K \rfloor_{\tau}$$

The LK-Ascon AEAD returns (C, T). A pictorial description of the encryption algorithm of LK-Ascon can be found in Fig. 4.

Decryption Algorithm. As before, our focus lies primarily on the verification process itself, rather than the specific steps involved in message computation. On receiving an input of the form $(N, A, C, T) \in \{0,1\}^{\nu} \times \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^{\tau}$ and a key $K \in \{0,1\}^{\kappa}$, the steps of the verification process is outlined below: .

1. $(A_1, \ldots, A_a) \xleftarrow{r} \mathsf{pad}_1(N, A)$ and $(C_1, \ldots, C_l) \xleftarrow{r} \mathsf{pad}_2(C)$.
2. Compute $V_0 := \textsc{Init}^{\pi}(K)$.
3. Compute the walk for the permutation π

$$V_0 \xrightarrow{A_1}_{\oplus} V_1 \xrightarrow{A_2}_{\oplus} \cdots \xrightarrow{A_a}_{\oplus} V_a$$

4. Let $C_l = C_l' \| 10^*$ for some C_l' (may be the empty string) and $|C_l'| = d$. Let $z_a = \lfloor V_a \oplus 0^*1 \rfloor_c$.

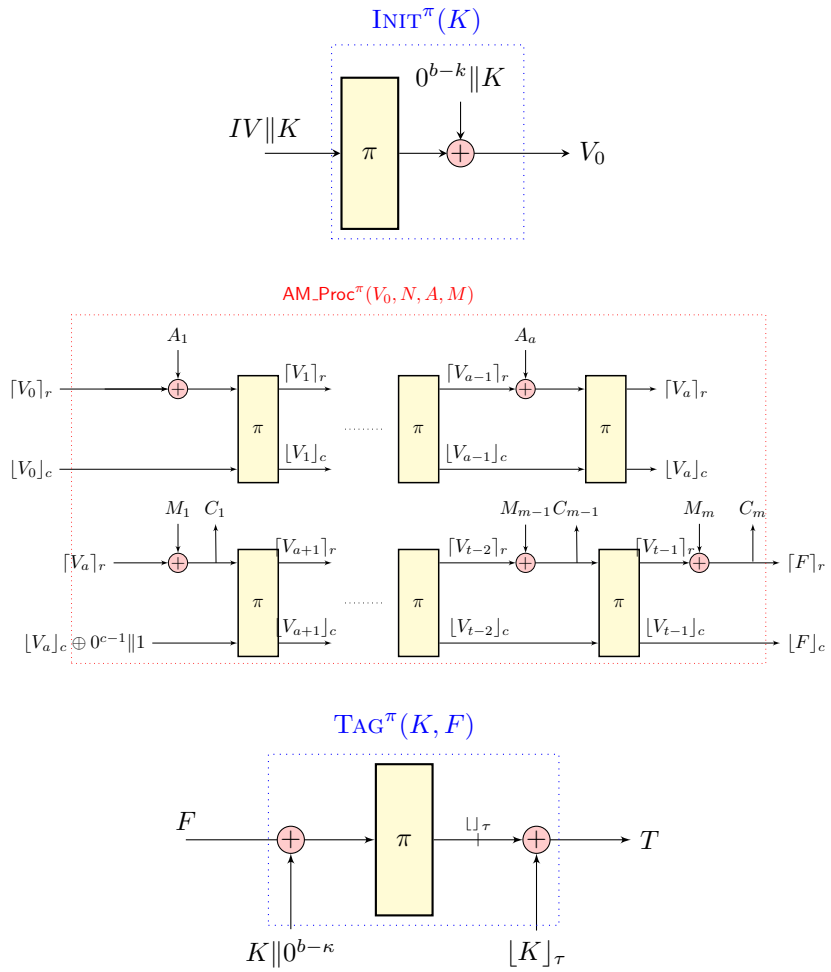

Fig. 4. Encryption in LK-ASCON AEAD. The difference with conventional ASCON is that there is no nonce at the input of INIT. The nonce is parsed with the associated data, and fed at the AM_Proc step. Also, the key addition at the input of TAG is $K\|0^{b-\kappa}$, meaning even if $\kappa \leq c$, it is xored at the rate part.

- Case $l = 1$: Define $F = C'_l \parallel (\lfloor V_a \rfloor_{b-d} \oplus 10^*1)$.
- Case $l \geq 2$: Compute

$$z_a \xrightarrow{C_1} z_{a+1} \xrightarrow{C_2} \cdots \xrightarrow{C_{l-2}} z_{a+l-2}$$

  We define $F = C'_l \parallel \lfloor \pi(C_{l-1}\|z_{a+l-2}) \oplus 10^* \rfloor_{b-d}$.

5. Reject if $T \neq \text{TAG}^\pi(K, F)$, otherwise, accept.

Remark 4. The functions $\textsc{Init}^{\pi}$, $\mathsf{AM\_Proc}^{\pi}$ and $\textsc{Tag}^{\pi}$ are different for Ascon and LK-Ascon. In fact, for the first two, even the domains are different. We have intentionally reused the notations to emphasize the similarity in the processes of the two AEAD modes.

5.1 Security bounds on LK-Ascon

We give two security bounds on Ascon: AEAD advantage for nonce-respecting multi-user LK-Ascon, and authenticity advantage for nonce-misuse multi-user LK-Ascon, along with their interpretations.

Theorem 3. *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, and q_d decryption queries with a total number of σ_d data block. Define $\sigma := \sigma_e + \sigma_d$. Then, we can upper bound the mu-AEAD advantage of $\mathscr{A}$ against LK-Ascon as follows:*

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

where

$$\omega_{r,\kappa} = \begin{cases} 2q_e, & \text{if } r \leq \kappa \\ q_e + \mathsf{mcoll}(q_e, 2^{r-\kappa}) \cdot 2^{r-\kappa} & \text{otherwise} \end{cases}.$$

Interpretation of the Theorem. The first thing we would like our modified construction to have is achieve the same security as Ascon when $\kappa \leq c$ (though this necessitates a larger IV). Upon interpreting our bound within the context of the NIST LWC requirements, it becomes evident that we maintain the same level of security as previously, encompassing both a 128-bit rate and 192-bit rate, as well as both a 64-bit tag and 128-bit tag.

Next, note that for any arbitrary κ, the above bound is of the order

$$\mathcal{O}(\frac{\mu q_p}{2^\kappa} + \frac{q_d}{2^\tau} + \frac{q_p}{2^c} + \frac{\sigma q_p}{2^b}),$$

when interpreted with NIST parameters, which is the exact same as that of multi-user AEAD security of conventional Ascon. Thus, LK-Ascon achieves the same security as Ascon, while enabling an increase in key size of upto 320 bits. Particularly, we would like to note that Ascon-256 achieves the same security as Ascon by just doubling the key size and keeping everything else same (note that for other key sizes, we need to change the IV as well).

Theorem 4. *Consider a possibly nonce-misusing authentication adversary $\mathscr{A}$ making q_p permutation queries, q_e encryption queries with a total number of σ_e data blocks, and q_d decryption queries with a total number of σ_d data blocks. Define $\sigma := \sigma_e + \sigma_d$. Then, we can upper bound the mu-auth advantage of $\mathscr{A}$ against LK-*Ascon *as follows:*

$$\begin{aligned}\mathbf{Adv}^{\mathrm{mu-auth}}_{LK-\textsc{Ascon}}(\mathscr{A}) \leq \frac{\mu^2}{2^\kappa} + \frac{2q_d}{2^\tau} + \frac{\sigma_e^2}{2^c} + \frac{\sigma_d(q_p+\sigma_d)}{2^b} + \frac{\mathsf{mcoll}(\sigma_e, 2^r)(\sigma_d+q_p)}{2^c} \\ + \frac{\mu(q_p+\sigma)}{2^\kappa} + \frac{\mathsf{mcoll}(q_e, 2^\tau)q_d}{2^c} + \frac{\mathsf{mcoll}(\sigma+q_p, 2^\tau)q_d}{2^\kappa} \\ + \frac{q_e^2+q_d^2+q_eq_d+(q_e+q_d)(\sigma+q_p)}{2^b} + \frac{\omega_{r,\kappa}(\sigma+q_p)}{2^b}.\end{aligned}$$

This shows that like Ascon, LK-Ascon too maintains authenticity under nonce-misuse. In the next section, we give a proof overview of Theorem 3. The proof of Theorem 4 is a straightforward extension, and we outline the sketch in Appendix E.

5.2 Proof Overview of Theorem 3

The proof follows the structure outlined in Section 3.5 for Ascon. However, there are notable differences, summarized as follows:

- The case $\kappa = c$ does not result in key nullification because the positions where keys are XOR-ed are distinct. This aspect simplifies the proof slightly.
- New challenges arise due to alterations in constraints. The nonce is now processed with associated data, leading to the identical output of the initialization phase, V_0, for each query to the same user. Additionally, if the nonce spans more than one block, up to a certain point i, all V_i values can be the same for two queries. This challenge is addressed by defining the longest common prefix, especially for encryption queries.
- The analysis diverges depending on whether $\kappa \geq r$. If $\kappa \geq r$, the final block of the ciphertext is fully masked by the key XOR in the input of the finalization phase. Otherwise, if $\kappa < r$, the final ciphertext block is only partially masked.

A detailed proof is provided in Appendix D.

6 Conclusion

In this paper, we derived a multi-user security bound for the Ascon AEAD mode, the winner of the recently concluded NIST LWC competition. This mode follows a Sponge type of construction. Notably, the inclusion of a key XOR operation during the tag generation phase allows us to derive a bound of the following order:

$$\frac{\mu T}{2^\kappa} + \frac{T}{2^c} + \frac{D}{2^\tau} + \frac{DT}{2^b}$$

where T is the time complexity and D is the data complexity of the adversary. We also show that Ascon maintains this authenticity security even in the nonce misuse setting, although confidentiality is not guaranteed.

Finally, we introduce a variant of Ascon, called LK-Ascon, which allows an increase in key size, upto 320 bits. Notably, increasing the key size enhances the security of Ascon in the multi-user seting in addition to providing better security against quantum key recovery attacks utilizing Grover's algorithm. Like Ascon, LK-Ascon also maintains its authenticity security in the nonce misuse setting.

Acknowledgements

Bishwajit Chakraborty is supported by the NRF-ANR project SELECT ("NRF-2020-NRF-ANR072"). Mridul Nandi is partially supported by DRDO project.

References

BDPA11. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Duplexing the sponge: Single-pass authenticated encryption and other applications. In *SAC*, volume 7118 of *Lecture Notes in Computer Science*, pages 320–337. Springer, 2011.

BDPA12. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Permutation-based encryption, authentication and authenticated encryption. In *DIAC 2012*, 2012.

Ber05. Daniel J. Bernstein. The Poly1305-AES Message-Authentication Code. In Henri Gilbert and Helena Handschuh, editors, *FSE*, volume 3557 of *Lecture Notes in Computer Science*, pages 32–49. Springer, 2005.

Ber08a. Daniel J. Bernstein. Chacha, a variant of salsa20. *Workshop Record of SASC*, 8:3–5, 2008.

Ber08b. Daniel J. Bernstein. The salsa20 family of stream ciphers. In Matthew J. B. Robshaw and Olivier Billet, editors, *New Stream Cipher Designs - The eSTREAM Finalists*, volume 4986 of *Lecture Notes in Computer Science*, pages 84–97. Springer, 2008.

BN08. Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *J. Cryptol.*, 21(4):469–491, 2008.

CDN23a. Bishwajit Chakraborty, Chandranan Dhar, and Mridul Nandi. Exact security analysis of ASCON. In *ASIACRYPT 2023, Proceedings, Part III*, volume 14440 of *Lecture Notes in Computer Science*, pages 346–369. Springer, 2023.

CDN23b. Bishwajit Chakraborty, Chandranan Dhar, and Mridul Nandi. Exact security analysis of ASCON. *IACR Cryptol. ePrint Arch.*, page 775, 2023.

CJN20. Bishwajit Chakraborty, Ashwin Jha, and Mridul Nandi. On the security of sponge-type authenticated encryption modes. *IACR Transactions on Symmetric Cryptology*, pages 93–119, 2020.

Com14. The CAESAR Committee. Caesar: competition for authenticated encryption: security, applicability, and robustness, 2014.

CS14. Shan Chen and John P. Steinberger. Tight security bounds for key-alternating ciphers. In *Advances in Cryptology - EUROCRYPT 2014. Proceedings*, pages 327–350, 2014.

DEMS14. Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. ASCON v1. Submission to the CAESAR Competition, 2014.

DEMS19. Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. ASCON. Submission to NIST LwC Standardization Process (FINALIST), 2019.

DNT19. Avijit Dutta, Mridul Nandi, and Suprita Talnikar. Beyond birthday bound secure MAC in faulty nonce model. In *EUROCRYPT Proceedings, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 437–466. Springer, 2019.

ML23. Bart Mennink and Charlotte Lefevre. Generic security of the ascon mode: On the power of key blinding. *IACR Cryptol. ePrint Arch.*, page 796, 2023.

MN17. 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.

MV04. David A. McGrew and John Viega. The security and performance of the galois/counter mode (GCM) of operation. In *INDOCRYPT*, volume 3348 of *Lecture Notes in Computer Science*, pages 343–355. Springer, 2004.

NIS18. 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`.

Pat91. Jacques Patarin. *Etude des Générateurs de Permutations Pseudo-aléatoires Basés sur le Schéma du DES*. PhD thesis, Université de Paris, 1991.

Pat08. Jacques Patarin. The "coefficients H" technique. In *Selected Areas in Cryptography - SAC 2008. Revised Selected Papers*, pages 328–345, 2008.

SCM08. Joseph Salowey, Abhijit Choudhury, and David A. McGrew. AES galois counter mode (GCM) cipher suites for TLS. *RFC*, 5288:1–8, 2008.

A Graph Structures for Functions

The details of this section are thoroughly discussed in [CDN23a]. 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 3. *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, we can define a randomized extension algorithm for $\mathcal{P}^{\oplus}$, denoted as $\mathsf{xorRand\_Extn}^{\mathcal{P}}(v_0, x^k)$, $v_0 \in \{0,1\}^b, x_i \in \{0,1\}^r$.

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

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

B Proof of Theorem 1

Since we employ the H-coefficient technique for the proof, we first need to describe the real and ideal worlds.

Description of the Real World

The real-world samples μ keys $K_1, \ldots, K_\mu \xleftarrow{\$} \{0,1\}^\kappa$ and a random permutation Π. All queries are then responded to honestly following Ascon AEAD as defined above (including direct primitive queries to Π). A transcript in the real world is of the form

$$\Theta_{\text{re,on}} = \big((\mathsf{u}_i, \mathsf{N}_i, \mathsf{A}_i, \mathsf{M}_i, \mathsf{C}_i, \mathsf{T}_i)_{i\in[q_e]},\ (\mathsf{u}'_i, \mathsf{N}'_i, \mathsf{A}'_i, \mathsf{C}'_i, \mathsf{T}'_i, \mathsf{M}'_i)_{i\in[q_d]},\ \ \mathsf{P}\big),$$

where $\mathsf{u}_i, \mathsf{u}'_i$ represent user numbers for encryption and decryption queries, and P represents the query responses for primitive queries (represented in terms of the partial function for Π). When the i-th decryption query is rejected we write $M'_i = \mathsf{rej}$ (we keep this as one of the necessary conditions for a good transcript in the ideal world).

Once all queries have been executed, every input-output pair utilized in Π for both encryption and decryption queries is incorporated into the offline transcript. Let $\mathsf{P}_{\mathsf{fin}}$ represent the extended partial function, and it is evident that all encryption and decryption queries are determined by $\mathsf{P}_{\mathsf{fin}}$. It is essential to note that the keys $K_1, \ldots, K_\mu$ are also determined from the domain of $\mathsf{P}_{\mathsf{fin}}$. Implicitly, the domain and range elements of $\mathsf{P}_{\mathsf{fin}}$ are presented in the order of the execution of the underlying permutation to compute all encryption and decryption queries. Let

$$\Theta_{\text{re}} = \big((\mathsf{u}_i, \mathsf{N}_i, \mathsf{A}_i, \mathsf{M}_i, \mathsf{C}_i, \mathsf{T}_i)_{i\in[q_e]},\ (\mathsf{u}'_i, \mathsf{N}'_i, \mathsf{A}'_i, \mathsf{C}'_i, \mathsf{T}'_i, \mathsf{M}'_i)_{i\in[q_d]},\ \ \mathsf{P}_{\mathsf{fin}}\big)$$

denote the extended real world transcript. For any real world realizable transcript $\theta = \big((u_i, N_i, A_i, M_i, C_i, T_i)_{i\in[q_e]},\ (u'_i, N'_i, A'_i, C'_i, T'_i, M'_i)_{i\in[q_d]},\ \ P_{\mathsf{fin}}\big)$,

$$\Pr(\Theta_{\text{re}} = \theta) = \Pr(P_{\mathsf{fin}} \subseteq \Pi) = 1/(2^b)_{|P_{\mathsf{fin}}|}.$$

Description of the Ideal World

We now elaborate on how the ideal oracle interacts with the adversary $\mathscr{A}$. This depiction consists of two main phases: (i) the online phase, covering the actual interaction between the adversary and the ideal oracle, and (ii) the offline phase, occurring subsequent to the online phase, where the ideal oracle samples intermediate variables to ensure compatibility with the Ascon construction.

The offline phase is further divided into multiple stages, each contingent on events defined over the preceding stages. In the case of a bad event occurring at any stage, the ideal oracle has the flexibility to either abort or exhibit arbitrary behavior. To facilitate a comprehensive analysis, we aim to establish an upper bound on the probability of all such bad events. Consequently, at any given stage, we assume that all prior bad events have not occurred. To streamline notation, we employ the same symbols for the transcripts in both the real and ideal worlds.

Online Phase. The adversary can make three types of queries in an interleaved manner without any repetition: (i) encryption queries (ii) decryption queries, and (iii) primitive queries.

- On i-th Encryption Query $(\mathsf{u}_i, \mathsf{N}_i, \mathsf{A}_i, \mathsf{M}_i)$, $\forall i \in [q_e]$, respond randomly:
$$\mathsf{C}_i \xleftarrow{\$} \{0,1\}^{|\mathsf{M}_i|},\ \mathsf{T}_i \xleftarrow{\$} \{0,1\}^{\tau},\quad \mathsf{return}(\mathsf{C}_i, \mathsf{T}_i).$$
- On i-th Decryption Query $(\mathsf{u}'_i, \mathsf{N}'_i, \mathsf{A}'_i, \mathsf{C}'_i, \mathsf{T}'_i)$, $i \in [q_d]$, reject straightaway: Ideal oracle returns rej for all decryption queries (here we assume that the adversary does not make any decryption query that is obtained from a previous encryption query).
- On i-th Primitive Query $(\mathsf{Q}_i, \mathsf{dir}_i) \in \{0,1\}^b \times \{+1,-1\}$, $i \in [q_p]$, respond honestly: We maintain a list P of responses of primitive queries, representing the partial (injective) function of a random permutation Π. Initially, $\mathsf{P} = \emptyset$.
  1. If $\mathsf{dir}_i = +1$, we set $\mathsf{U}_i = \mathsf{Q}_i$. Let $\mathsf{V}_i \xleftarrow{\$} \{0,1\}^b \setminus \mathsf{range}(\mathsf{P})$, $\mathsf{P} \leftarrow \mathsf{P} \cup \{(\mathsf{U}_i, \mathsf{V}_i)\}$, return V_i.
  2. If $\mathsf{dir}_i = -1$, we set $\mathsf{V}_i = \mathsf{Q}_i$. Let $\mathsf{U}_i \xleftarrow{\$} \{0,1\}^b \setminus \mathsf{domain}(\mathsf{P})$, $\mathsf{P} \leftarrow \mathsf{P} \cup \{(\mathsf{U}_i, \mathsf{V}_i)\}$, return U_i.

After all queries have been made we denote the online transcript (visible to the adversary) as

$$\Theta_{\mathrm{id,on}} = \big((\mathsf{u}_i, \mathsf{N}_i, \mathsf{A}_i, \mathsf{M}_i, \mathsf{C}_i, \mathsf{T}_i)_{i\in[q_e]},\ (\mathsf{u}'_i, \mathsf{N}'_i, \mathsf{A}'_i, \mathsf{C}'_i, \mathsf{T}'_i, \mathsf{rej})_{i\in[q_d]},\ \mathsf{P}\big).$$

Bad Event. We set $\mathsf{bad}_1 = 1$, if

$$(\mathsf{u}_i, \mathsf{N}_i, \mathsf{A}_i, \mathsf{C}_i, \mathsf{T}_i) = (\mathsf{u}'_j, \mathsf{N}'_j, \mathsf{A}'_j, \mathsf{C}'_j, \mathsf{T}'_j), \quad i \in [q_e],\ j \in [q_d]$$

for which the encryption query is made later. It is crucial to highlight that the adversary is prohibited from making a decryption query that matches a

previous encryption query. However, there exists a possibility that a decryption query accidentally matches a subsequently made encryption query, termed a "bad event", which requires attention. Given that the adversary can make only nonce-respecting encryption queries, we can establish an upper bound for the probability of bad_1 as provided in the following lemma.

Lemma 3. $\Pr(\mathsf{bad}_1 = 1) \leq \frac{q_d}{2^\tau}$.

The proof follows trivially since we need to match the tag for some decryption query.

Offline Phase. The offline phase is divided into three main stages, performed sequentially: (i) setting internal states of encryption queries, (ii) setting internal states of decryption queries, and (iii) sampling a key, and verifying compatibility with the online phase.

First, we set the input-output pairs for all permutations used in processing associated data and message part of each encryption query. For $i \in [q_e]$ (i.e., for i-th encryption query) we perform the following:

1. We first parse all data we have in the online transcript.

$$(\mathsf{A}_{i,1}, \ldots, \mathsf{A}_{i,a_i}) \xleftarrow{r} \mathsf{pad}_1(\mathsf{A}_i)$$
$$(\mathsf{M}_{i,1}, \ldots, \mathsf{M}_{i,m_i}) \xleftarrow{r}_* \mathsf{M}_i$$
$$(\mathsf{C}_{i,1}, \ldots, \mathsf{C}_{i,m_i}) \xleftarrow{r}_* \mathsf{C}_i$$

2. Let $t_i = a_i + m_i$, $d_i = |\mathsf{M}_{i,m_i}| = |\mathsf{C}_{i,m_i}|$. We now sample

$$\mathsf{V}_{i,0}, \ldots, \mathsf{V}_{i,a_i-1} \xleftarrow{\$} \{0,1\}^b$$
$$\mathsf{Z}_{i,a_i}, \ldots, \mathsf{Z}_{i,t_i-1} \xleftarrow{\$} \{0,1\}^c, \delta_i^* \xleftarrow{\$} \{0,1\}^{r-d_i}$$

   The values of $\mathsf{V}_{i,j}$ would determine all inputs and outputs for associate data processing. Similarly, $\mathsf{C}_i, \mathsf{Z}_{i,j}, \delta_i^*$ would determine the input and outputs for message processing.

3. We now set all inputs and outputs of the permutation used in associate data and message processing. Note that while $a_i = 0$ is possible, $m_i \geq 1$.

   If $a_i > 0$, we define the following:
   - $\mathsf{U}_{i,j} = \mathsf{V}_{i,j-1} \oplus (\mathsf{A}_{i,j} \| 0^c)$, $\forall j \in [a_i]$.
   - $\mathsf{V}_{i,a_i} = (\mathsf{C}_{i,1} \oplus \mathsf{M}_{i,1}) \| \mathsf{Z}_{i,a_i}$.

   If $m_i \geq 2$:
   - $\mathsf{U}_{i,a_i+1} = \mathsf{C}_{i,1} \| (\mathsf{Z}_{i,a_i} \oplus 0^{c-1}1)$.
   - $\mathsf{U}_{i,a_i+j} = \mathsf{C}_{i,j} \| \mathsf{Z}_{i,a_i+j-1}$, $2 \leq j \leq m_i - 1$.
   - $\mathsf{V}_{i,a_i+j} = (\mathsf{C}_{i,j+1} \oplus \mathsf{M}_{i,j+1}) \| \mathsf{Z}_{i,a_i+j-1}$, $\forall j \in [m_i - 2]$.
   - $\mathsf{V}_{i,t_i-1} = (\mathsf{C}_{i,m_i} \oplus \mathsf{M}_{i,m_i}) \| \delta_i^* \| \mathsf{Z}_{i,t_i-1}$.

– $\mathsf{F}_i = \mathsf{C}_{i,m_i} \| \delta_i^* \| \mathsf{Z}_{i,t_i-1}$.

Otherwise:
– $\mathsf{F}_i = \mathsf{C}_{i,m_1} \| \delta_1^* \| (\mathsf{Z}_{i,a_i} \oplus 0^{c-1}1)$.

We define $\mathsf{P_E}$ to be the partial function mapping $\mathsf{U}_{i,j}$ to $\mathsf{V}_{i,j}$ for all $i \in [q_e]$, $j \in [t_i - 1]$, provided all $\mathsf{U}_{i,j}$'s are distinct. In this case, it is easy to see that

$$\mathsf{V}_{i,0} \xrightarrow{\mathsf{A}_{i,1}}_{\oplus} \mathsf{V}_{i,1} \xrightarrow{\mathsf{A}_{i,2}}_{\oplus} \cdots \xrightarrow{\mathsf{A}_{i,a_i}}_{\oplus} \mathsf{V}_{i,a_i};$$

$$\mathsf{V}_{i,a_i} \oplus 0^{b-1}1 \xrightarrow{\mathsf{M}_{i,1}}_{\oplus} \mathsf{V}_{i,a_i+1} \cdots \xrightarrow{\mathsf{M}_{i,m_i-1}}_{\oplus} \mathsf{V}_{i,t_i-1}.$$

Moreover, $\mathsf{P_E}$ would be an injective partial function if $\mathsf{V}_{i,j}$'s are all distinct.

Bad Event ($\mathsf{P_E}$ is Not an Injective Partial Function). We set

1. $\mathsf{bad}_2 = 1$ if for some $(i,j) \neq (i',j')$, either $\mathsf{U}_{i,j} = \mathsf{U}_{i',j'}$ or $\mathsf{V}_{i,j} = \mathsf{V}_{i',j'}$,
2. $\mathsf{bad}_3 = 1$ if for some $i \neq i' \in [q_e]$, $\mathsf{F}_i = \mathsf{F}_{i'}$ (if this happens then it would force $\mathsf{T}_i = \mathsf{T}_{i'}$ to hold).

Lemma 4. $\Pr(\mathsf{bad}_2 = 1 \vee \mathsf{bad}_3 = 1) \leq \dfrac{\sigma_e^2}{2^b}$.

Proof. $\mathsf{V}_{i,j}$'s are randomly sampled and $\mathsf{U}_{i,j}$'s are defined through a bijective mapping of $\mathsf{V}_{i,j-1}$ values. The same applies to F_i values. Given that we have at most $\binom{\sigma_e}{2}$ choices for inputs and outputs, we get the above bound by simply using the union bound. □

Contingent on the condition that none of the aforementioned bad events occur, we would like to set the input-output pairs for all permutations used in associated data and ciphertext processing for all decryption queries. Here, we only use P to run the randomized extension. Later, we set a bad event if it is not disjoint (both from the domain and the range) with $\mathsf{P_E}$. This would ensure the compatibility of $\mathsf{P}_1 \sqcup \mathsf{P_E}$ (where P_1 is the randomized extension of P) and would also help later in upper bounding the forging probability of a decryption query. For $i \in [q_d]$ (i.e., for the i-th decryption query) with $t_i \geq 2$, we perform the following:

We first parse all data as we have done for encryption queries:

$$(\mathsf{A}'_{i,1}, \ldots, \mathsf{A}'_{i,a'_i}) \xleftarrow{r} \mathsf{pad}_1(\mathsf{A}'_i)$$
$$(\mathsf{C}'_{i,1}, \ldots, \mathsf{C}'_{i,c_i}) \xleftarrow{r}_* \mathsf{C}'_i$$

Let $t'_i = a'_i + c_i$, $d'_i = |\mathsf{C}_{i,c_i}|$. Now, we define p_i indicating the length of the longest common prefix with an encryption query.

Definition of p_i, $i \in [q_d]$.

1. If there does not exist any $j \in [q_e]$ such that $(\mathsf{u}_j, \mathsf{N}_j) = (\mathsf{u}'_i, \mathsf{N}'_i)$, we define $p_i = -1$.

2. Otherwise, there exists a unique j for which $(\mathsf{u}_j, \mathsf{N}_j) = (\mathsf{u}'_i, \mathsf{N}'_i)$ (since the adversary is nonce-respecting and hence every nonce in encryption queries to the same user is distinct). Define p_i denote the length of the largest common prefix of
   – $(\mathsf{A}'_{i,1}, \ldots, (\mathsf{A}'_{i,a'_i}, *), \mathsf{C}'_{i,1}, \ldots, \mathsf{C}'_{i,c_i})$ and
   – $(\mathsf{A}_{j,1}, \ldots, (\mathsf{A}_{j,a_j}, *), \mathsf{C}_{j,1}, \ldots, \mathsf{C}_{j,m_i})$.
   Here $*$ is used to distinguish associate data blocks and ciphertext blocks.

Now, for each $i \in [q_d]$, depending on the value of p_i, we perform the following:

Associated Data and Ciphertext Processing.

1. For $i = 1$ to q_d with $p_i = -1$:
   – If $(\mathsf{u}'_i, \mathsf{N}'_i) = (\mathsf{u}'_j, \mathsf{N}'_j)$ for some $j \in [i-1]$, $\mathsf{V}'_{i,0} := \mathsf{V}'_{j,0}$. Otherwise, $\mathsf{V}'_{i,0} \xleftarrow{\$} \{0,1\}^b$.
   – If $a'_i > 0$, run $\mathsf{xorRand\_Extn}^P(\mathsf{V}'_{i,0}, (\mathsf{A}'_{i,1}, \ldots, \mathsf{A}'_{i,a'_i}))$ to obtain a walk
   $$\mathsf{V}'_{i,0} \xrightarrow{\mathsf{A}'_{i,1}}_{\oplus} \mathsf{V}'_{i,1} \xrightarrow{\mathsf{A}'_{i,2}}_{\oplus} \cdots \xrightarrow{\mathsf{A}'_{i,a'_i}}_{\oplus} \mathsf{V}'_{i,a'_i}.$$
   – If $c_i > 1$, run $\mathsf{Rand\_Extn}^P(\mathsf{V}'_{i,a_i} \oplus 0^*1, \mathsf{C}'_{i,1} \| \ldots \| \mathsf{C}'_{i,c_i-1})$ to obtain a walk
   $$\mathsf{V}'_{i,a'_i} \oplus 0^*1 \xrightarrow{\mathsf{C}'_{i,1}} \mathsf{V}'_{i,a'_i+1} \xrightarrow{\mathsf{C}'_{i,2}} \cdots \xrightarrow{\mathsf{C}'_{i,c_i-1}} \mathsf{V}'_{i,a'_i+c_i-1}.$$
2. For $i = 1$ to q_d with $0 \le p_i \le a'_i$:
   – $\mathsf{V}'_{i,p_i} := \mathsf{V}_{j,p_i}$, where $j \in [q_e]$ such that $(\mathsf{u}'_i, \mathsf{N}'_i) = (\mathsf{u}'_j, \mathsf{N}'_j)$.
   – If $a'_i > p_i$, run $\mathsf{xorRand\_Extn}^P(\mathsf{V}'_{i,p_i}, (\mathsf{A}'_{i,p_i+1}, \ldots, \mathsf{A}'_{i,a'_i}))$ to obtain a walk
   $$\mathsf{V}'_{i,p_i} \xrightarrow{\mathsf{A}'_{i,p_i+1}}_{\oplus} \mathsf{V}'_{i,p_i+1} \xrightarrow{\mathsf{A}'_{i,p_i+2}}_{\oplus} \cdots \xrightarrow{\mathsf{A}'_{i,a'_i}}_{\oplus} \mathsf{V}'_{i,a'_i}.$$
   – If $c_i > 1$, run $\mathsf{Rand\_Extn}^P(\mathsf{V}'_{i,a_i} \oplus 0^*1, \mathsf{C}'_{i,1} \| \ldots \| \mathsf{C}'_{i,c_i-1})$ to obtain a walk
   $$\mathsf{V}'_{i,a'_i} \oplus 0^*1 \xrightarrow{\mathsf{C}'_{i,1}} \mathsf{V}'_{i,a'_i+1} \xrightarrow{\mathsf{C}'_{i,2}} \cdots \xrightarrow{\mathsf{C}'_{i,c_i-1}} \mathsf{V}'_{i,a'_i+c_i-1}.$$
3. For $i = 1$ to q_d with $a'_i < p_i < t_i - 1$:
   – $\mathsf{V}'_{i,p_i} := \mathsf{V}_{j,p_i}$, where $j \in [q_e]$ such that $(\mathsf{u}'_i, \mathsf{N}'_i) = (\mathsf{u}'_j, \mathsf{N}'_j)$.
   – If $p_i < t_i - 1$, run $\mathsf{Rand\_Extn}^P(\mathsf{V}'_{i,p_i}, \mathsf{C}'_{i,p_i-a'_i+1} \| \ldots \| \mathsf{C}'_{i,c_i-1})$ to obtain a walk
   $$\mathsf{V}'_{i,p_i} \xrightarrow{\mathsf{C}'_{i,p_i-a'_i+1}} \mathsf{V}'_{i,p_i+1} \xrightarrow{\mathsf{C}'_{i,p_i-a'_i+2}} \cdots \xrightarrow{\mathsf{C}'_{i,c_i-1}} \mathsf{V}'_{i,a'_i+c_i-1}.$$
4. For $i = 1$ to q_d with $p_i = t_i - 1$:
   – $\mathsf{V}'_{i,a'_i+c_i-1} := \mathsf{V}_{j,a'_i+c_i-1}$, where $j \in [q_e]$ such that $(\mathsf{u}'_i, \mathsf{N}'_i) = (\mathsf{u}'_j, \mathsf{N}'_j)$.

For all the cases above, we define

$$\mathsf{F}'_i = \begin{cases} \mathsf{C}'_{i,c_i} \| 10^* \| \lfloor \mathsf{V}'_{i,a'_i+c_i-1} \rfloor_c & \text{if } c_i \geq 2 \\ \mathsf{C}'_{i,c_i} \| 10^* \| (\lfloor \mathsf{V}'_{i,a'_i+c_i-1} \rfloor_c \oplus 0^{c-1}1) & \text{if } c_i = 1 \end{cases}.$$

Note that for each $i \in [q_d]$, P is updated by both the randomized extension algorithms, and although we start with a permutation, the resulting extended function P_1 need not be injective.

Bad Event (P_1 is Not an Injective Partial Function). We define $\mathsf{bad}_4 = 1$ if there exist (X, Y) and (X', Y') in the set P_1 such that $Y = Y'$. It is important to note that P is an injective partial function, and thus this bad event can only occur when at least one of the values Y or Y' is obtained during the offline phase. Considering that both inputs and outputs are uniformly sampled, the probability of bad_4 can be straightforwardly bounded using the union bound.

Lemma 5. $\Pr(\mathsf{bad}_4 = 1) \leq \dfrac{\sigma_d(q_p + \sigma_d)}{2^b}$.

Bad Event (Permutation Compatibility of P_E and P_1). We now set $\mathsf{bad}_5 = 1$ if

$$\mathsf{domain}(\mathsf{P}_1) \cap \mathsf{domain}(\mathsf{P}_\mathsf{E}) \neq \emptyset \text{ or } \mathsf{range}(\mathsf{P}_1) \cap \mathsf{range}(\mathsf{P}_\mathsf{E}) \neq \emptyset.$$

Given that this bad event does not hold, $\mathsf{P}_\mathsf{E} \sqcup \mathsf{P}_1$ is an injective partial function that is desired for a random permutation.

Lemma 6. $\Pr(\mathsf{bad}_5 = 1) \leq \dfrac{\mathsf{mcoll}(\sigma_e, 2^r) \times (\sigma_d + q_p)}{2^c}$.

Proof. Let ρ_1 (and ρ_2) denote the multicollision on the values of $\lceil x \rceil_r$, for all $x \in \mathsf{domain}(\mathsf{P}_\mathsf{E})$ (and for all $x \in \mathsf{range}(\mathsf{P}_\mathsf{E})$ respectively). Then, by the randomness of the randomized extension process and randomized xor-extension process, $\Pr(\mathsf{bad}_5 = 1 \mid \max\{\rho_1, \rho_2\} = \rho) \leq \rho(\sigma_d + q_p)/2^c$. Hence, using the expectation of $\max\{\rho_1, \rho_2\}$, and applying Lemma 1 and Remark 2, we get the above bound. □

Bad Event (Correctly Forging). We now set bad events whenever we have a correct forging in the ideal world based on the injective partial function $\mathsf{P}_2 := \mathsf{P}_1 \sqcup \mathsf{P}_\mathsf{E}$ constructed so far. We set $\mathsf{bad}_6 = 1$ if

$$(\mathsf{F}'_i, \mathsf{T}'_i) = (\mathsf{F}_j, \mathsf{T}_j), \quad i \in [q_d], \quad j \in [q_e].$$

This is similar to bad_3 as this would force a decryption query to be valid.

Lemma 7. $\Pr(\mathsf{bad}_6 = 1) \leq \dfrac{\mathsf{mcoll}(q_e, 2^\tau) q_d}{2^c}$.

Proof. We divide this into two cases. First, consider $p_i = t'_i - 1$ and $\mathsf{T}'_i = \mathsf{T}_j$. Then $\mathsf{F}'_i \neq \mathsf{F}_j$, and hence bad_6 does not occur.

Next, we assume $p_i \neq t'_i - 1$. Let ρ_3 denote the number of multicollision of T_j values. By using the randomness of Z_{j,t_i-1} and using the multicollision we have, $\Pr(\mathsf{bad}_6 = 1 \mid \rho_3 = \rho) \leq \frac{\rho q_d}{2^c}$. Hence, using the expectation of ρ_3, and applying Lemma 1, we have the above bound. □

Now, we reach the time to sample the keys $K_1, K_2, \ldots K_\mu \xleftarrow{\$} \{0,1\}^\kappa$. For each K_i, let $K_i = (K_{i,1}, K_{i,2})$ where $K_{i,2} \in \{0,1\}^\tau$.

Bad Event (Key-Collision). We set $\mathsf{bad}_7 = 1$ when the keys of two users collide. It is easy to prove the following lemma:

Lemma 8. $\Pr(\mathsf{bad}_7 = 1) \leq \dfrac{\mu^2}{2^\kappa}$.

Let

$$\mathcal{J} = \{j \in [q_d] : (\mathsf{u}'_j, \mathsf{N}'_j) \neq (\mathsf{u}_i, \mathsf{N}_i)\ \forall i \in [q_e]\}.$$

Now, we can define the input-outputs for the underlying permutation used in the initialization phase as follows:

1. For all $i \in [q_e]$,

$$\mathsf{I}_i := IV \| K_{\mathsf{u}_i} \| \mathsf{N}_i, \quad \mathsf{O}_i := \begin{cases} \mathsf{V}_{i,0}, & \text{if } \kappa = c \text{ and } t_i = 1 \\ \mathsf{V}_{i,0} \oplus 0^{b-\kappa} \| K_{\mathsf{u}_i}, & \text{otherwise} \end{cases}$$

2. For all $j \in \mathcal{J}$,

$$\mathsf{I}'_j := IV \| K'_{\mathsf{u}_j} \| \mathsf{N}'_j, \quad \mathsf{O}'_j := \begin{cases} \mathsf{V}'_{j,0}, & \text{if } \kappa = c \text{ and } t_j = 1 \\ \mathsf{V}'_{j,0} \oplus 0^{b-\kappa} \| K'_{\mathsf{u}_j}, & \text{otherwise} \end{cases}$$

3. For all other $j \in [q_d]$, there exists $i \in [q_e]$ such that $(\mathsf{u}'_j, \mathsf{N}'_j) = (\mathsf{u}_i, \mathsf{N}_i)$, and we define $\mathsf{I}'_j := \mathsf{I}_i$, $\mathsf{O}'_j := \mathsf{O}_i$.

Here, K_{u_i} and K'_{u_j} corresponds to the key of users u_i and u'_j respectively. Define $\mathsf{P}_{\mathsf{init}} = \big((\mathsf{I}_i, \mathsf{O}_i)_{i \in [q_e]},\ (\mathsf{I}'_j, \mathsf{O}'_j)_{j \in \mathcal{J}}\big)$.

Bad Event (Permutation Compatibility of $\mathsf{P}_{\mathrm{init}}$ and P_2). We define $\mathsf{bad}_8 = 1$ if one of the following holds:

1. $\mathsf{I}_i, \mathsf{I}'_j \in \mathsf{domain}(\mathsf{P}_2)$ for some $i \in [q_e], j \in [q_d]$.
2. $\mathsf{O}_i = \mathsf{O}_j$ for $i, j \in [q_e]$ or $\mathsf{O'}_i = \mathsf{O'}_j$ for $i, j \in [q_d]$ such that $(\mathsf{u}'_j, \mathsf{N}'_j) \neq (\mathsf{u}'_i, \mathsf{N}'_i)$.
3. $\mathsf{O}_i = \mathsf{O}'_j$ for $i \in [q_e]$ and $j \in [q_d]$ such that $(\mathsf{u}_i, \mathsf{N}_i) \neq (\mathsf{u}'_j, \mathsf{N}'_j)$.
4. $\mathsf{O}_i, \mathsf{O}'_j \in \mathsf{range}(\mathsf{P}_2)$ for some $i \in [q_e], j \in [q_d]$.

Lemma 9. $\Pr(\mathsf{bad}_8 = 1) \leq \dfrac{\mu(q_p + \sigma)}{2^\kappa} + \dfrac{q_e^2 + q_d^2 + q_e q_d + (q_e + q_d)(\sigma + q_p)}{2^b}$.

Proof. In the first case, since the IV and the nonce are in adversarial control, for I_i or I'_j to be in the domain of P_2, a key must collide with the $(b-\kappa-\nu+1)$-th to the $(b-\nu)$-th bit of an element of the set $\mathsf{domain}(\mathsf{P}_2)$. Since there are μ keys, and the size of P_2 is $(q_p+\sigma)$, the probability of this event is bounded by $\frac{\mu(q_p+\sigma)}{2^\kappa}$.

In the second case, if either $\kappa < c$, or both $t_i \neq 1, t_j \neq 1$, then $\mathsf{O}_i = \mathsf{O}_j$ implies $\mathsf{V}_{i,0} \oplus 0^{b-\kappa} \| K_{u_i} = \mathsf{V}_{j,0} \oplus 0^{b-\kappa} \| K_{u_j}$. For fixed i, j, this happens with probability at most $1/2^b$. The same can be easily verified if one or both of t_i and t_j is 1 and $\kappa = c$. Hence, the probability that $\mathsf{O}_i = \mathsf{O}_j$ occurs for some $i, j \in [q_e]$ is $\frac{q_e^2}{2^b}$. A similar analysis holds when we consider the decryption queries with different nonces, and we get the probability $\frac{q_d^2}{2^b}$.

The analysis of the third case is similar to that of the second case, and since we consider a match between encryption and decryption queries, this happens with probability $\frac{q_e q_d}{2^c}$.

The last case considers a full state match between the outputs of $(q_e + q_d)$ encryption or decryption queries, and the range of $(\sigma+q_p)$ elements of P_2. Hence, we get the required bound. □

Define $\mathsf{P}_3 := \mathsf{P}_2 \sqcup \mathsf{P}_{\mathsf{init}}$. Now, we settle tag computation for all encryption queries. Note that the user is already specified by setting the initialization phase, and hence we can work with a single key K. For all $i \in [q_e]$, we define $\mathsf{X}_i := \mathsf{F}_i \oplus (0^r \| K \| 0^{c-\kappa})$, $\mathsf{Y}_i := \alpha_i \| (\mathsf{T}_i \oplus K_2)$, where $\alpha_i \xleftarrow{\$} \{0,1\}^{b-\tau}$. Define $\mathsf{P}_{\mathsf{tag}} = \big((\mathsf{X}_i, \mathsf{Y}_i)_{i \in [q_e]}\big)$.

Bad Event (Permutation Compatibility of $\mathsf{P}_{\mathsf{tag}}$ and P_3). We define $\mathsf{bad}_9 = 1$ if either of the following holds:

1. $\mathsf{domain}(\mathsf{P}_{\mathsf{tag}}) \cap \mathsf{domain}(\mathsf{P}_3) \neq \emptyset$, or
2. $\mathsf{range}(\mathsf{P}_{\mathsf{tag}}) \cap \mathsf{range}(\mathsf{P}_3) \neq \emptyset$.

Given this bad event does not hold, $\mathsf{P}_4 := \mathsf{P}_3 \sqcup \mathsf{P}_{\mathsf{tag}}$ is once again an injective partial function.

Lemma 10. $\Pr(\mathsf{bad}_9 = 1) \leq \dfrac{\mathsf{mcoll}(q_e, 2^{r+c-\kappa})(\sigma + q_p)}{2^\kappa} + \dfrac{q_e(\sigma + q_p)}{2^b}$.

Proof. We divide this into two cases, depending on whether $\kappa = c$ or not. If $\kappa < c$, let $\lambda_i = \lceil F_i \rceil_r \| \lfloor F_i \rfloor_{c-\kappa} = \lceil X_i \rceil_r \| \lfloor X_i \rfloor_{c-\kappa}$. Let ρ_4 denote the multicollision of λ_i values. Then, by the randomness of K and using the multicollision, we have $\Pr(\mathsf{domain}(\mathsf{P}_{\mathsf{tag}}) \cap \mathsf{domain}(\mathsf{P}_3) \neq \emptyset \mid \rho_4 = \rho) \leq \frac{\rho(\sigma+q_p)}{2^\kappa}$. So, using the expectation of ρ_4, and using Remark 2, we have

$$\Pr(\mathsf{domain}(\mathsf{P}_{\mathsf{tag}}) \cap \mathsf{domain}(\mathsf{P}_3) \neq \emptyset) \leq \frac{\mathsf{mcoll}(q_e, 2^{r+c-\kappa})(\sigma + q_p)}{2^\kappa}.$$

in this case.

If $\kappa = c$, let $\Omega_i = \{\mathsf{X}_i \in \mathsf{domain}(\mathsf{P}_{\mathsf{tag}}) \mid t_i = 1\}$. It is enough to bound $\Pr(\Omega_i \cap \mathsf{domain}(\mathsf{P}_3) \neq \emptyset)$. Let $\lambda_i = \lceil F_i \rceil_r = \lceil X_i \rceil_r$. Let ρ_5 denote the multicollision of λ_i values. Then, by the randomness of $\lfloor \mathsf{F}_i \rfloor_c = \lfloor \mathsf{V}_{i,0} \rfloor_c \oplus 0^*1$ and using the

multicollision, we have $\Pr(\Omega_i \cap \mathsf{domain}(\mathsf{P}_3) \neq \emptyset \mid \rho_5 = \rho) \leq \frac{\rho(\sigma+q_p)}{2^c}$. So, using the expectation of ρ_5, and using Remark 2, we have

$$\Pr(\Omega_i \cap \mathsf{domain}(\mathsf{P}_3) \neq \emptyset) \leq \frac{\mathsf{mcoll}(q_e, 2^r)(\sigma + q_p)}{2^c}.$$

Using the randomness of α_i and K, it can be easily seen that

$$\Pr(\mathsf{range}(\mathsf{P}_{\mathsf{tag}}) \cap \mathsf{range}(\mathsf{P}_3) \neq \emptyset) \leq \frac{q_e(\sigma + q_p)}{2^b}.$$

Hence, we get the above bound. □

Finally, we settle the tag computation of all decryption queries and we set bad whenever a valid forgery occurs. For all $i \in [q_d]$, we define

$$\mathsf{X}'_i := \begin{cases} \mathsf{F}'_i, & \text{if } \kappa = c \text{ and } t_i = 1 \\ \mathsf{F}'_i \oplus (0^r \| K \| 0^{c-\kappa}), & \text{otherwise} \end{cases}$$

If $\mathsf{X}'_i \in \mathsf{domain}(\mathsf{P}_4)$ then we define $\mathsf{Y}'_i = \mathsf{P}_4(\mathsf{X}'_i)$. Else, $\mathsf{Y}'_i \xleftarrow{\$} \{0,1\}^b$.

Bad Event (Decryption Queries are Not Rejected). We divide this into two cases depending on whether $\mathsf{X}'_i \in \mathsf{domain}(\mathsf{P}_4)$ or not:

– We set $\mathsf{bad}_{10} = 1$ if

$$\exists i \in [q_d], \qquad \mathsf{X}'_i \in \mathsf{domain}(\mathsf{P}_4) \ \wedge \ \lfloor \mathsf{P}_4(\mathsf{X}'_i) \rfloor_\tau \oplus K_2 = \mathsf{T}'_i.$$

Again, we first consider the case $\kappa < c$. Let $\mathsf{F}'_i = (\beta'_i \| \gamma'_i \| \delta'_i)$, where $|\beta'_i| = r + \kappa - \tau$, $|\gamma'_i| = \tau$ and $|\delta'_i| = c - \kappa$. If $\mathsf{bad}_{10} = 1$, then
(i) for some $(x_j \| y_j \| z_j) \in \mathsf{domain}(\mathsf{P}_4)$, $\mathsf{X}'_i = (x_j \| y_j \| z_j)$, $|x_j| = r + \kappa - \tau$, $|y_j| = \tau$ and $|z_j| = c - \kappa$, and
(ii) $y_j \oplus w_j = \mathsf{T}'_i \oplus \gamma'_i$ where $w_j = \lfloor \mathsf{P}_4(x_j \| y_j \| z_j) \rfloor_\tau$.
Let ρ_6 denote the multicollision on the values of $(y_a \oplus w_a)_a$ varying over all elements of P_4. Hence, the number of choices of j is at most ρ_6. Then, by the randomness of K,

$$\Pr(\mathsf{bad}_{10} = 1 \mid \rho_6 = \rho) \leq \frac{\rho q_d}{2^\kappa}.$$

Now, coming to the $\kappa = c$ case, we only need to consider the case when $t'_i = 1$, otherwise the above proof applies. For a fixed $i \in [q_d]$ such that $t'_i = 1$, $\Pr(\mathsf{X}'_i \in \mathsf{domain}(\mathsf{P}_4)) \leq (\sigma + q_p)/2^c$. Now, given $\mathsf{X}'_i \in \mathsf{domain}(\mathsf{P}_4)$, $\Pr(\lfloor \mathsf{P}_4(\mathsf{X}'_i) \rfloor_\tau \oplus K_2 = \mathsf{T}'_i) = 1/2^\tau$. Taking union bound, and using the expectation of ρ_6 for the $\kappa < c$ case, we have

Lemma 11. $\Pr(\mathsf{bad}_{10} = 1) \leq \frac{\mathsf{mcoll}(\sigma + q_p, 2^\tau) q_d}{2^\kappa} + \frac{q_d(\sigma + q_p)}{2^{c+\tau}}$.

– $\mathsf{X}'_i \notin \mathsf{domain}(\mathsf{P}_4)$. Let $y_i = \lfloor \mathsf{Y}'_i \rfloor_\tau$. We set $\mathsf{bad}_{11} = 1$ if there exists $i \in [q_d]$ such that $y_i \oplus K_2 = \mathsf{T}'_i$. By the randomness of y_i, we have

Lemma 12. $\Pr(\mathsf{bad}_{11} = 1) \leq \dfrac{q_d}{2^\tau}$.

Let bad denote the union of all bad events, namely $\cup_{i=1}^{11} \mathsf{bad}_i$. By Lemmas 3 through 12, we have shown that

$$\begin{aligned}\Pr(\mathsf{bad} = 1) \leq\ & \frac{\mu^2}{2^\kappa} + \frac{2q_d}{2^\tau} + \frac{\sigma_e^2}{2^b} + \frac{\sigma_d(q_p+\sigma_d)}{2^b} + \frac{\mathsf{mcoll}(\sigma_e, 2^r)(\sigma_d+q_p)}{2^c} \\ & + \frac{\mu(q_p+\sigma)}{2^\kappa} + \frac{\mathsf{mcoll}(q_e, 2^\tau)q_d}{2^c} + \frac{\mathsf{mcoll}(\sigma+q_p, 2^\tau)q_d}{2^\kappa} \\ & + \frac{q_e^2+q_d^2+q_eq_d+(q_e+q_d)(\sigma+q_p)}{2^b} + \frac{q_e(\sigma+q_p)}{2^b} \\ & + \frac{\mathsf{mcoll}(q_e, 2^{r+c-\kappa})(\sigma+q_p)}{2^\kappa} + \frac{q_d(\sigma+q_p)}{2^{c+\tau}}.\end{aligned}$$

If all these bad events do not occur, then all the decryption queries are correctly rejected for the injective partial function P_4.

Let $\mathsf{P}_{\mathsf{fin}} := \mathsf{P}_4 \cup \big((\mathsf{X}'_i, \mathsf{Y}'_i)_{i\in[q_d]}\big)$. In the offline transcript, we provide all the input-outputs of $\mathsf{P}_{\mathsf{fin}}$. Then,

$$\Theta_{\mathrm{id}} = \big((\mathsf{u}_i, \mathsf{N}_i, \mathsf{A}_i, \mathsf{M}_i, \mathsf{C}_i, \mathsf{T}_i)_{i\in[q_e]},\ (\mathsf{u}'_i, \mathsf{N}'_i, \mathsf{A}'_i, \mathsf{C}'_i, \mathsf{T}'_i, \mathsf{rej})_{i\in[q_d]},\ \mathsf{P}_{\mathsf{fin}}\big).$$

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,

$$\Pr(\Theta_{\mathrm{id}} = \theta) = \Pr(P \subseteq \Pi) \times 2^{-b(|\mathsf{P}_{\mathsf{fin}}|-|\mathsf{P}|)} \leq 1/(2^b)_{|\mathsf{P}_{\mathsf{fin}}|} = \Pr(\Theta_{\mathrm{re}} = \theta)$$

By using the H-coefficient technique, we complete the proof of Theorem 1.

C Proof of Theorem 2

We highlight the parts where it differs from the proof of Theorem 1. We reuse bad event numbers for easier understanding.

The description of the real world is exactly the same as in the nonce respecting setting.

Description of the Ideal World ONLINE PHASE. On i-th encryption query $(\mathsf{u}_i, \mathsf{N}_i, \mathsf{A}_i, \mathsf{M}_i)$, parse A_i and M_i and determine the longest common prefix p_i as follows:

1. If there does not exist any $j \in [i-1]$ such that $(\mathsf{u}_j, \mathsf{N}_j) = (\mathsf{u}_i, \mathsf{N}_i)$, we define $p_i = -1$.
2. Otherwise, there exists at least one j for which $(\mathsf{u}_j, \mathsf{N}_j) = (\mathsf{u}_i, \mathsf{N}_i)$ (since the adversary can misuse nonces). For each such j, let $p_{i,j}$ denote the length of the largest common prefix of
   – $(\mathsf{A}_{i,1}, \ldots, (\mathsf{A}_{i,a_i}, *), \mathsf{M}_{i,1}, \ldots, \mathsf{M}_{i,m_i})$ and
   – $(\mathsf{A}_{j,1}, \ldots, (\mathsf{A}_{j,a_j}, *), \mathsf{M}_{j,1}, \ldots, \mathsf{M}_{j,m_j})$.
   Here $*$ is used to distinguish associate data blocks and message blocks.
   Finally define $p_i = \max_{j<i} p_{i,j}$.

If $p_i \leq a_i$, respond randomly:

$$\mathsf{C}_{i,j} \xleftarrow{\$} \{0,1\}^r \, \forall j \in [m_i], \; \mathsf{T}_i \xleftarrow{\$} \{0,1\}^\tau, \quad \mathsf{return}(\mathsf{C}_i := \lceil \mathsf{C}_{i,1} \| \cdots \| \mathsf{C}_{i,m_i} \rceil_{\mathsf{M}_i}, \mathsf{T}_i).$$

If $p_i > a_i$, set $\mathsf{C}_{i,j} = \mathsf{C}_{i',j}$ for $j \leq p_i - a_i$ (where $i' \in [i-1]$ is the query with the longest common prefix) and $\mathsf{C}_{i,p_i+1} = \mathsf{M}_{i,p_i+1} \oplus \mathsf{M}_{i',p_i+1} \oplus \mathsf{C}_{i',p_i+1}$. The rest of the ciphertext blocks and the tag are defined randomly. Finally,

$$\mathsf{return}(\mathsf{C}_i := \lceil \mathsf{C}_{i,1} \| \cdots \| \mathsf{C}_{i,m_i} \rceil_{\mathsf{M}_i}, \mathsf{T}_i).$$

Decryption and primitive queries are handled as before, i.e. decryption queries are rejected straightaway and primitive queries are responded to faithfully. The event bad_1 and its bound are also the same as before.

Offline phase. (i) Setting internal states of encryption queries: Unlike the single user nonce-respecting setting, here we have three cases. Note that all the data are already parsed, and hence we begin directly at Step 2 (of the nonce-respecting setting).

- If $p_i = -1$, proceed exactly as in the nonce-respecting setting.
- If $p_i < a_i$, $\mathsf{V}_{i,j} = \mathsf{V}_{i',j}$ for all $j \in [0, p_i]$, where $i' \in [i-1]$ is the query index with maximum common prefix. The rest of the $\mathsf{V}_{i,j}$, $\mathsf{Z}_{i,j}$ and δ_i^* are defined randomly as before.
- If $a_i \leq p_i < t_i$, $\mathsf{V}_{i,j} = \mathsf{V}_{i',j}$ for all $j \in [0, a_i]$, $\mathsf{Z}_{i,j} = \mathsf{Z}_{i',j}$ for all $j \in [a_i + 1, p_i + 1]$, where $i' \in [i-1]$ is the query index with maximum common prefix. The rest of the $\mathsf{Z}_{i,j}$ and δ_i^* are defined randomly as before.

Step 3 is exactly the same as the nonce-respecting setting. Again, we define P_E to be the partial function mapping $\mathsf{U}_{i,j}$ to $\mathsf{V}_{i,j}$ for all $i \in [q_e]$, $j \in [t_i - 1]$, provided all $\mathsf{U}_{i,j}$'s are distinct.

The events bad_2 and bad_3 differ a bit a bit from before. Since a nonce-misuse adversary can force any desired value to the outer part of a permutation call, the probability of the union of events bad_2 and bad_3 is bounded by $\sigma_e^2/2^c$, instead of $\sigma_e^2/2^b$ as in the nonce-respecting scenario.

(ii) Setting internal states of decryption queries: This process is also similar to that of the the nonce-respecting setting. The only exception is that the length of the longest common prefix p_i needs to be defined as in the nonce-misuse encryption case above, since more than one encryption query can have the same nonce. Even after this small change in definition, it can be easily verified that the bad events bad_4 through bad_6 and their proofs are also the same as the nonce-respecting case.

(iii) Sampling keys and verifying compatibility with the online phase: This process is also similar to the multi-user nonce-respecting scenario. We have the same bad event bad_7 with the same bound.

For all $i \in [q_e]$ and $j \in [q_d]$, I_i and I'_j are defined exactly as in the nonce-respecting case. Note that for $i, j \in [q_e]$, we can have $(\mathsf{u}_i, \mathsf{N}_i) = (\mathsf{u}_j, \mathsf{N}_j)$, resulting in $\mathsf{I}_i = \mathsf{I}_j$, but then $\mathsf{V}_{i,0} = \mathsf{V}_{j,0}$ which implies $\mathsf{O}_i = \mathsf{O}_j$, and thus we do not have

any inconsistency in the definitions. Hence, the event bad_8 can be defined as the same as above and we have the same upper bound.

Now, coming to the finalization, for $i, j \in [q_e]$, $\mathsf{F}_i \neq \mathsf{F}_j$ even if $(\mathsf{u}_i, \mathsf{N}_i) = (\mathsf{u}_j, \mathsf{N}_j)$ and hence, the bad_9 can be defined as above. The events bad_{10} and bad_{11} are also the same as above including their bounds. We can then define bad as the union of events bad_1 through bad_{11}.

Thus, we again have an ideal world transcript

$$\Theta_{\mathrm{id}} = \big((\mathsf{u}_i, \mathsf{N}_i, \mathsf{A}_i, \mathsf{M}_i, \mathsf{C}_i, \mathsf{T}_i)_{i\in[q_e]},\ (\mathsf{u}'_i, \mathsf{N}'_i, \mathsf{A}'_i, \mathsf{C}'_i, \mathsf{T}'_i, \mathsf{rej})_{i\in[q_d]},\ \mathsf{P}_{\mathsf{fin}}\big).$$

In setting too, note that any good transcript $\theta \in \Theta_{\mathrm{id}}$, we sample either inputs or outputs of $\mathsf{P}_{\mathsf{fin}} \setminus \mathsf{P}$ uniformly. Thus,

$$\Pr(\Theta_{\mathrm{id}} = \theta) = \Pr(P \subseteq \Pi) \times 2^{-b(|\mathsf{P}_{\mathsf{fin}}|-|\mathsf{P}|)} \leq 1/(2^b)_{|\mathsf{P}_{\mathsf{fin}}|} = \Pr(\Theta_{\mathrm{re}} = \theta)$$

By using the H-coefficient technique, we complete the proof.

D Proof of Theorem 3

The proof structure follows the proof of Theorem 1. However, since the encryption and verification algorithms differ slightly, we present the full proof. As we have done throughout, we reuse bad event notations, and omit proofs of statements already proved above if the proofs are also same.

Description of the Real World: The real-world samples $K_1, \ldots, K_\mu \xleftarrow{\$} \{0,1\}^\kappa$ and a random permutation Π. All queries are then responded to honestly following LK-Ascon AEAD as defined above (including direct primitive queries to Π). 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

$$\Theta_{\mathrm{re}} = \big((\mathsf{u}_i, \mathsf{N}_i, \mathsf{A}_i, \mathsf{M}_i, \mathsf{C}_i, \mathsf{T}_i)_{i\in[q_e]},\ (\mathsf{u}'_i, \mathsf{N}'_i, \mathsf{A}'_i, \mathsf{C}'_i, \mathsf{T}'_i, \mathsf{M}'_i)_{i\in[q_d]},\ \ \mathsf{P}_{\mathsf{fin}}\big)$$

denote the extended real world transcript.

For any real world realizable transcript θ,

$$\Pr(\Theta_{\mathrm{re}} = \theta) = \Pr(P_{\mathsf{fin}} \subseteq \Pi) = 1/(2^b)_{|P_{\mathsf{fin}}|}.$$

Description of the Ideal World: The ideal world is again divided into two phases: the online phase, and the offline phase.

Online Phase. The adversary can make three types of queries in an interleaved manner without any repetition: (i) encryption queries (ii) decryption queries, and (iii) primitive queries.

– On i-th encryption query $(\mathsf{u}_i, \mathsf{N}_i, \mathsf{A}_i, \mathsf{M}_i)$, $\forall i \in [q_e]$, respond randomly.
– On i-th decryption query $(\mathsf{u}'_i, \mathsf{N}'_i, \mathsf{A}'_i, \mathsf{C}'_i, \mathsf{T}'_i)$, $\forall i \in [q_d]$, reject straightaway.
– On i-th primitive query $(\mathsf{Q}_i, \mathsf{dir}_i) \in \{0,1\}^b \times \{+1, -1\}$, $\forall i \in [q_p]$, respond honestly.

After all queries have been made we denote the online transcript (visible to the adversary) as

$$\Theta_{\text{id,on}} = \big((\mathsf{u}_i, \mathsf{N}_i, \mathsf{A}_i, \mathsf{M}_i, \mathsf{C}_i, \mathsf{T}_i)_{i\in[q_e]},\ (\mathsf{u}'_i, \mathsf{N}'_i, \mathsf{A}'_i, \mathsf{C}'_i, \mathsf{T}'_i, \mathsf{rej})_{i\in[q_d]},\ \mathsf{P}\big).$$

Bad Event. We set $\mathsf{bad}_1 = 1$, if

$$(\mathsf{u}_i, \mathsf{N}_i, \mathsf{A}_i, \mathsf{C}_i, \mathsf{T}_i) = (\mathsf{u}'_j, \mathsf{N}'_j, \mathsf{A}'_j, \mathsf{C}'_j, \mathsf{T}'_j), \quad i \in [q_e],\ j \in [q_d]$$

for which the encryption query is made later.

Lemma 13. $\Pr(\mathsf{bad}_1 = 1) \le \dfrac{q_d}{2^\tau}$.

Offline Phase. The offline phase is divided into three stages, performed sequentially: (i) setting internal states of encryption queries, (ii) setting internal states of decryption queries, and (iii) sampling a key, and verifying compatibility with the online phase.

We first set the input-output pairs for all permutations used in processing nonce, associated data and message part of each encryption query. For each $i \in [q_e]$, we perform the following:

1. We first parse all data we have in the online transcript.

$$(\mathsf{A}_{i,1}, \ldots, \mathsf{A}_{i,a_i}) \xleftarrow{r} \mathsf{pad}_1(\mathsf{N}_i, \mathsf{A}_i)$$
$$(\mathsf{M}_{i,1}, \ldots, \mathsf{M}_{i,m_i}) \xleftarrow{r}_* \mathsf{M}_i$$
$$(\mathsf{C}_{i,1}, \ldots, \mathsf{C}_{i,m_i}) \xleftarrow{r}_* \mathsf{C}_i$$

2. Let $t_i = a_i + m_i$, $d_i = |\mathsf{M}_{i,m_i}| = |\mathsf{C}_{i,m_i}|$. Note that $a_i \ge 1$. Since the adversary is nonce-respecting, if there exists $j < i$ such that $\mathsf{u}_i = \mathsf{u}_j$, then $\mathsf{N}_i \ne \mathsf{N}_j$. Let $k = \lceil \frac{|\mathsf{N}|}{r} \rceil$. Then, this implies $(\mathsf{A}_{i,1}, \ldots, \mathsf{A}_{i,k}) \ne (\mathsf{A}_{j,1}, \ldots, \mathsf{A}_{j,k})$. Let $k' \le k$ be the first index such that $\mathsf{V}_{i,k'} \ne \mathsf{V}_{j,k'}$. Then, we set $\mathsf{V}_{i,0} = \mathsf{V}_{j,0}, \ldots, \mathsf{V}_{i,k'-1} = \mathsf{V}_{j,k'-1}$. Otherwise, we sample $\mathsf{V}_{i,0}, \ldots, \mathsf{V}_{i,k'-1} \xleftarrow{\$} \{0,1\}^b$. We also sample

$$\mathsf{V}_{i,k'}, \ldots, \mathsf{V}_{i,a_i-1} \xleftarrow{\$} \{0,1\}^b$$
$$\mathsf{Z}_{i,a_i}, \ldots, \mathsf{Z}_{i,t_i-1} \xleftarrow{\$} \{0,1\}^c, \delta_i^* \xleftarrow{\$} \{0,1\}^{r-d_i}.$$

The values of $\mathsf{V}_{i,j}$ would determine all inputs and outputs for associate data processing. Similarly, $\mathsf{C}_i, \mathsf{Z}_{i,j}, \delta_i^*$ would determine the input and outputs for message processing.

3. We now set all inputs and outputs of the permutation used in associate data and message processing.
   - $\mathsf{U}_{i,j} = \mathsf{V}_{i,j-1} \oplus (\mathsf{A}_{i,j} \| 0^c)$, $\forall j \in [a_i]$.
   - $\mathsf{V}_{i,a_i} = (\mathsf{C}_{i,1} \oplus \mathsf{M}_{i,1}) \| \mathsf{Z}_{i,a_i}$.

   If $m_i \geq 2$:
   - $\mathsf{U}_{i,a_i+1} = \mathsf{C}_{i,1} \| (\mathsf{Z}_{i,a_i} \oplus 0^{c-1}1)$.
   - $\mathsf{U}_{i,a_i+j} = \mathsf{C}_{i,j} \| \mathsf{Z}_{i,a_i+j-1}$, $2 \leq j \leq m_i - 1$.
   - $\mathsf{V}_{i,a_i+j} = (\mathsf{C}_{i,j+1} \oplus \mathsf{M}_{i,j+1}) \| \mathsf{Z}_{i,a_i+j-1}$, $\forall j \in [m_i - 2]$.
   - $\mathsf{V}_{i,t_i-1} = (\mathsf{C}_{i,m_i} \oplus \mathsf{M}_{i,m_i}) \| \delta_i^* \| \mathsf{Z}_{i,t_i-1}$.
   - $\mathsf{F}_i = \mathsf{C}_{i,m_i} \| \delta_i^* \| \mathsf{Z}_{i,t_i-1}$.

   Otherwise:
   - $\mathsf{F}_i = \mathsf{C}_{i,m_1} \| \delta_i^* \| (\mathsf{Z}_{i,a_i} \oplus 0^{c-1}1)$.

We define $\mathsf{P_E}$ to be the partial function mapping $\mathsf{U}_{i,j}$ to $\mathsf{V}_{i,j}$ for all $i \in [q_e]$, $j \in [t_i - 1]$, provided all $\mathsf{U}_{i,j}$'s are distinct. In this case, it is easy to see that

$$\mathsf{V}_{i,0} \xrightarrow{\mathsf{A}_{i,1}}_{\oplus} \mathsf{V}_{i,1} \xrightarrow{\mathsf{A}_{i,2}}_{\oplus} \cdots \xrightarrow{\mathsf{A}_{i,a_i}}_{\oplus} \mathsf{V}_{i,a_i}; \mathsf{V}_{i,a_i} \oplus 0^{b-1}1 \xrightarrow{\mathsf{M}_{i,1}}_{\oplus} \mathsf{V}_{i,a_i+1} \cdots \xrightarrow{\mathsf{M}_{i,m_i-1}}_{\oplus} \mathsf{V}_{i,t_i-1}.$$

Moreover, $\mathsf{P_E}$ would be an injective partial function if $\mathsf{V}_{i,j}$'s are all distinct.

Bad Event ($\mathsf{P_E}$ is Not an Injective Partial Function). We set

1. $\mathsf{bad}_2 = 1$ if for some $(i,j) \neq (i',j')$, either $\mathsf{U}_{i,j} = \mathsf{U}_{i',j'}$ or $\mathsf{V}_{i,j} = \mathsf{V}_{i',j'}$. Additionally, if $\mathsf{u}_i = \mathsf{u}_{i'}$, we do not consider the equalities $\mathsf{U}_{i,k} = \mathsf{U}_{i',k}$ or $\mathsf{V}_{i,k} = \mathsf{V}_{i',k}$ for $k < \lceil \frac{|\mathsf{N}|}{r} \rceil$ since they are set as equal.
2. $\mathsf{bad}_3 = 1$ if for some $i \neq i' \in [q_e]$, $\mathsf{F}_i = \mathsf{F}_{i'}$. Ideally, we should not allow this if $\mathsf{u}_i = \mathsf{u}_{i'}$, but we are giving an upper bound anyway.

It can be easily checked that

Lemma 14. $\Pr(\mathsf{bad}_2 = 1 \vee \mathsf{bad}_3 = 1) \leq \dfrac{\sigma_e^2}{2^b}$.

We would now like to set the input-output pairs for all permutations used in the noce, associated data and ciphertext processing for all decryption queries. For $i \in [q_d]$ (i.e., for the i-th decryption query) with $t_i \geq 2$, we perform the following:

We first parse all data as we have done for encryption queries:

$$(\mathsf{A}'_{i,1}, \ldots, \mathsf{A}'_{i,a'_i}) \xleftarrow{r} \mathsf{pad}_1(\mathsf{N}'_i, \mathsf{A}'_i)$$
$$(\mathsf{C}'_{i,1}, \ldots, \mathsf{C}'_{i,c_i}) \xleftarrow{r}_* \mathsf{C}'_i$$

Let $t'_i = a'_i + c_i$, $d'_i = |\mathsf{C}_{i,c_i}|$. Now, we define p_i indicating the length of the longest common prefix with an encryption query.

Definition of p_i, $i \in [q_d]$.

1. If there does not exist any $j \in [q_e]$ such that $\mathsf{u}_j = \mathsf{u}'_i$, we define $p_i = -1$.
2. Else if there does not exist any $j \in [q_e]$ such that $(\mathsf{u}_j, \mathsf{N}_j) = (\mathsf{u}'_i, \mathsf{N}'_i)$, we define $p_0 = 0$.
3. Otherwise, there exists a unique j for which $(\mathsf{u}_j, \mathsf{N}_j) = (\mathsf{u}'_i, \mathsf{N}'_i)$ (since the adversary is nonce-respecting and hence every nonce in encryption queries is distinct). Define p_i denote the length of the largest common prefix of
   – $(\mathsf{A}'_{i,1}, \ldots, (\mathsf{A}'_{i,a'_i}, *), \mathsf{C}'_{i,1}, \ldots, \mathsf{C}'_{i,c_i})$ and
   – $(\mathsf{A}_{j,1}, \ldots, (\mathsf{A}_{j,a_j}, *), \mathsf{C}_{j,1}, \ldots, \mathsf{C}_{j,m_i})$.
   Here $*$ is used to distinguish associate data blocks and ciphertext blocks.

Now, for each $i \in [q_d]$, depending on the value of p_i, we perform the following:

Associated Data and Ciphertext Processing.

1. For $i = 1$ to q_d with $p_i = -1$:
   – If $\mathsf{u}'_i = \mathsf{u}'_j$ for some $j \in [i-1]$, $V'_{i,0} := V'_{j,0}$. Otherwise, $\mathsf{V}'_{i,0} \xleftarrow{\$} \{0,1\}^b$.
   – Run $\mathsf{xorRand\_Extn}^P(\mathsf{V}'_{i,0}, (\mathsf{A}'_{i,1}, \ldots, \mathsf{A}'_{i,a'_i}))$ to obtain a walk
   $$\mathsf{V}'_{i,0} \xrightarrow{\mathsf{A}'_{i,1}}_{\oplus} \mathsf{V}'_{i,1} \xrightarrow{\mathsf{A}'_{i,2}}_{\oplus} \cdots \xrightarrow{\mathsf{A}'_{i,a'_i}}_{\oplus} \mathsf{V}'_{i,a'_i}.$$
   – If $c_i > 1$, run $\mathsf{Rand\_Extn}^P(\mathsf{V}'_{i,a_i} \oplus 0^*1, \mathsf{C}'_{i,1} \| \ldots \| \mathsf{C}'_{i,c_i-1})$ to obtain a walk
   $$\mathsf{V}'_{i,a'_i} \oplus 0^*1 \xrightarrow{\mathsf{C}'_{i,1}} \mathsf{V}'_{i,a'_i+1} \xrightarrow{\mathsf{C}'_{i,2}} \cdots \xrightarrow{\mathsf{C}'_{i,c_i-1}} \mathsf{V}'_{i,a'_i+c_i-1}.$$
2. For $i = 1$ to q_d with $p_i = 0$:
   – $\mathsf{V}'_{i,0} := \mathsf{V}_{j,0}$, where $j \in [q_e]$ such that $\mathsf{u}'_i = \mathsf{u}_j$.
   – Run $\mathsf{xorRand\_Extn}^P(\mathsf{V}'_{i,0}, (\mathsf{A}'_{i,1}, \ldots, \mathsf{A}'_{i,a'_i}))$ to obtain a walk
   $$\mathsf{V}'_{i,0} \xrightarrow{\mathsf{A}'_{i,1}}_{\oplus} \mathsf{V}'_{i,1} \xrightarrow{\mathsf{A}'_{i,2}}_{\oplus} \cdots \xrightarrow{\mathsf{A}'_{i,a'_i}}_{\oplus} \mathsf{V}'_{i,a'_i}.$$
   – If $c_i > 1$, run $\mathsf{Rand\_Extn}^P(\mathsf{V}'_{i,a_i} \oplus 0^*1, \mathsf{C}'_{i,1} \| \ldots \| \mathsf{C}'_{i,c_i-1})$ to obtain a walk
   $$\mathsf{V}'_{i,a'_i} \oplus 0^*1 \xrightarrow{\mathsf{C}'_{i,1}} \mathsf{V}'_{i,a'_i+1} \xrightarrow{\mathsf{C}'_{i,2}} \cdots \xrightarrow{\mathsf{C}'_{i,c_i-1}} \mathsf{V}'_{i,a'_i+c_i-1}.$$
3. For $i = 1$ to q_d with $0 \le p_i \le a'_i$:
   – $\mathsf{V}'_{i,p_i} := \mathsf{V}_{j,p_i}$, where $j \in [q_e]$ such that $(\mathsf{u}'_i, \mathsf{N}'_i) = (\mathsf{u}_j, \mathsf{N}_j)$.
   – If $a'_i > p_i$, run $\mathsf{xorRand\_Extn}^P(\mathsf{V}'_{i,p_i}, (\mathsf{A}'_{i,p_i+1}, \ldots, \mathsf{A}'_{i,a'_i}))$ to obtain a walk
   $$\mathsf{V}'_{i,p_i} \xrightarrow{\mathsf{A}'_{i,p_i+1}}_{\oplus} \mathsf{V}'_{i,p_i+1} \xrightarrow{\mathsf{A}'_{i,p_i+2}}_{\oplus} \cdots \xrightarrow{\mathsf{A}'_{i,a'_i}}_{\oplus} \mathsf{V}'_{i,a'_i}.$$
   – If $c_i > 1$, run $\mathsf{Rand\_Extn}^P(\mathsf{V}'_{i,a_i} \oplus 0^*1, \mathsf{C}'_{i,1} \| \ldots \| \mathsf{C}'_{i,c_i-1})$ to obtain a walk
   $$\mathsf{V}'_{i,a'_i} \oplus 0^*1 \xrightarrow{\mathsf{C}'_{i,1}} \mathsf{V}'_{i,a'_i+1} \xrightarrow{\mathsf{C}'_{i,2}} \cdots \xrightarrow{\mathsf{C}'_{i,c_i-1}} \mathsf{V}'_{i,a'_i+c_i-1}.$$

4. For $i = 1$ to q_d with $a'_i < p_i < t_i - 1$:
   - $\mathsf{V}'_{i,p_i} := \mathsf{V}_{j,p_i}$, where $j \in [q_e]$ such that $(\mathsf{u}'_i, \mathsf{N}'_i) = (\mathsf{u}_j, \mathsf{N}_j)$.
   - If $p_i < t_i - 1$, run $\mathsf{Rand\_Extn}^P(\mathsf{V}'_{i,p_i}, \mathsf{C}'_{i,p_i-a'_i+1} \| \dots \| \mathsf{C}'_{i,c_i-1})$ to obtain a walk
$$\mathsf{V}'_{i,p_i} \xrightarrow{\mathsf{C}'_{i,p_i-a'_i+1}} \mathsf{V}'_{i,p_i+1} \xrightarrow{\mathsf{C}'_{i,p_i-a'_i+2}} \cdots \xrightarrow{\mathsf{C}'_{i,c_i-1}} \mathsf{V}'_{i,a'_i+c_i-1}.$$
5. For $i = 1$ to q_d with $p_i = t_i - 1$:
   - $\mathsf{V}'_{i,a'_i+c_i-1} := \mathsf{V}_{j,a'_i+c_i-1}$, where $j \in [q_e]$ such that $(\mathsf{u}'_i, \mathsf{N}'_i) = (\mathsf{u}_j, \mathsf{N}_j)$.

For all the cases above, we define

$$\mathsf{F}'_i = \begin{cases} \mathsf{C}'_{i,c_i} \| 10^* \| \lfloor \mathsf{V}'_{i,a'_i+c_i-1} \rfloor_c & \text{if } c_i \geq 2 \\ \mathsf{C}'_{i,c_i} \| 10^* \| (\lfloor \mathsf{V}'_{i,a'_i+c_i-1} \rfloor_c \oplus 0^{c-1}1) & \text{if } c_i = 1 \end{cases}.$$

Bad Event (P_1 is Not an Injective Partial Function). We define $\mathsf{bad}_4 = 1$ if there exist (X, Y) and (X', Y') in the set P_1 such that $Y = Y'$. Considering that both inputs and outputs are uniformly sampled, the probability of bad_4 can be straightforwardly bounded using the union bound.

Lemma 15. $\Pr(\mathsf{bad}_4 = 1) \leq \dfrac{\sigma_d(q_p + \sigma_d)}{2^b}$.

Bad Event (Permutation Compatibility of P_E and P_1). We now set $\mathsf{bad}_5 = 1$ if

$$\mathsf{domain}(\mathsf{P}_1) \cap \mathsf{domain}(\mathsf{P}_\mathsf{E}) \neq \emptyset \text{ or } \mathsf{range}(\mathsf{P}_1) \cap \mathsf{range}(\mathsf{P}_\mathsf{E}) \neq \emptyset.$$

Given that this bad event does not hold, $\mathsf{P}_\mathsf{E} \sqcup \mathsf{P}_1$ is an injective partial function. As before, we have the following lemma.

Lemma 16. $\Pr(\mathsf{bad}_5 = 1) \leq \dfrac{\mathsf{mcoll}(\sigma_e, 2^r) \times (\sigma_d + q_p)}{2^c}$.

Bad Event (Correctly Forging). We now set bad events whenever we have a correct forging in the ideal world based on the injective partial function $\mathsf{P}_2 := \mathsf{P}_1 \sqcup \mathsf{P}_\mathsf{E}$ constructed so far. We set $\mathsf{bad}_6 = 1$ if

$$(\mathsf{F}'_i, \mathsf{T}'_i) = (\mathsf{F}_j, \mathsf{T}_j), \quad i \in [q_d], \quad j \in [q_e].$$

This is similar to bad_3 as this would force a decryption query to be valid if $\mathsf{u}_i = \mathsf{u}_j$.

Lemma 17. $\Pr(\mathsf{bad}_6 = 1) \leq \dfrac{\mathsf{mcoll}(q_e, 2^\tau) q_d}{2^c}$.

Now, we sample the keys $K_1, K_2, \dots K_\mu \xleftarrow{\$} \{0,1\}^\kappa$. For each K_i, let $K_i = (K_{i,1}, K_{i,2})$ where $K_{i,2} \in \{0,1\}^\tau$.

Bad Event (Key-Collision). We set $\mathsf{bad}_7 = 1$ when the keys of two users collide. It is easy to prove the following lemma:

Lemma 18. $\Pr(\mathsf{bad}_7 = 1) \leq \frac{\mu^2}{2^\kappa}$.

Let

$$\mathcal{J} = \{j \in [q_d] : \mathsf{u}'_j \neq \mathsf{u}_i \ \forall i\}.$$

Now, we can define the input-outputs for the underlying permutation used in the initialization phase as follows:

1. for all $i \in [q_e]$, $\mathsf{I}_i := IV \| K_{\mathsf{u}_i}$, $\mathsf{O}_i := \mathsf{V}_{i,0} \oplus 0^{b-\kappa} \| K$,
2. for all $j \in \mathcal{J}$, $\mathsf{I}'_j := IV \| K_{\mathsf{u}_j}$ and $\mathsf{O}'_j := \mathsf{V}'_{j,0} \oplus 0^{b-\kappa} \| K$,
3. For all other $j \in [q_d]$, there exists $i \in [q_e]$ such that $\mathsf{u}'_j = \mathsf{u}_i$, and we define $\mathsf{I}'_j := \mathsf{I}_i$, $\mathsf{O}'_j := \mathsf{O}_i$.

Define $\mathsf{P}_{\mathsf{init}} = \big((\mathsf{I}_i, \mathsf{O}_i)_{i \in [q_e]}, \ \ (\mathsf{I}'_j, \mathsf{O}'_j)_{j \in \mathcal{J}}\big)$.

Bad Event (Permutation Compatibility of P_{INIT} and P_2). We define $\mathsf{bad}_8 = 1$ if one of the following holds:

1. $\mathsf{I}_i, \mathsf{I}'_j \in \mathsf{domain}(\mathsf{P}_2)$ for some $i \in [q_e], j \in [q_d]$.
2. $\mathsf{O}_i = \mathsf{O}_j$ for $i, j \in [q_e]$ such that $\mathsf{u}_i \neq \mathsf{u}_j$ or $\mathsf{O'}_i = \mathsf{O'}_j$ for $i, j \in [q_d]$ such that $\mathsf{u}'_i \neq \mathsf{u}'_j$.
3. $\mathsf{O}_i = \mathsf{O}'_j$ for $i \in [q_e]$ and $j \in [q_d]$ such that $\mathsf{u}_i \neq \mathsf{u}'_j$.
4. $\mathsf{O}_i, \mathsf{O}'_j \in \mathsf{range}(\mathsf{P}_2)$ for some $i \in [q_e], j \in [q_d]$.

As in the case of Ascon, we have the following:

Lemma 19. $\Pr(\mathsf{bad}_8 = 1) \leq \dfrac{\mu(q_p + \sigma)}{2^\kappa} + \dfrac{q_e^2 + q_d^2 + q_e q_d + (q_e + q_d)(\sigma + q_p)}{2^b}$.

Once again, if this bad event does not hold, $\mathsf{P}_3 := \mathsf{P}_2 \sqcup \mathsf{P}_{\mathsf{init}}$ is an injective partial function. Now, we settle tag computation for all encryption queries. For all $i \in [q_e]$, we define $\mathsf{X}_i := \mathsf{F}_i \oplus (K_{u_i} \| 0^{b-\kappa})$, $\mathsf{Y}_i := \alpha_i \| (\mathsf{T}_i \oplus K_{u_i,2})$, where $\alpha_i \xleftarrow{\$} \{0,1\}^{b-\tau}$. Define $\mathsf{P}_{\mathsf{tag}} = \big((\mathsf{X}_i, \mathsf{Y}_i)_{i \in [q_e]}\big)$.

Bad Event (Permutation Compatibility of $\mathsf{P}_{\mathsf{tag}}$ and P_3). We define $\mathsf{bad}_9 = 1$ if either of the following holds:

1. $\mathsf{domain}(\mathsf{P}_{\mathsf{tag}}) \cap \mathsf{domain}(\mathsf{P}_3) \neq \emptyset$, or
2. $\mathsf{range}(\mathsf{P}_{\mathsf{tag}}) \cap \mathsf{range}(\mathsf{P}_3) \neq \emptyset$.

Given this bad event does not hold, $\mathsf{P}_4 := \mathsf{P}_3 \sqcup \mathsf{P}_{\mathsf{tag}}$ is once again an injective partial function.

Lemma 20. $\Pr(\mathsf{bad}_9 = 1) \leq \dfrac{\omega_{r,\kappa}(\sigma + q_p)}{2^b}$, *where*

$$\omega_{r,\kappa} = \begin{cases} 2q_e, & \text{if } r \leq \kappa \\ q_e + \mathsf{mcoll}(q_e, 2^{r-\kappa}) \cdot 2^{r-\kappa} & \text{otherwise} \end{cases}.$$

Proof. We divide this into two cases depending on whether $r \leq \kappa$ or not. If $r \leq \kappa$, then using the randomness of K_{u_i} and $\lfloor \mathsf{F}_i \rfloor_c$, it can be easily shown that

$$\Pr(\mathsf{domain}(\mathsf{P}_{\mathsf{tag}}) \cap \mathsf{domain}(\mathsf{P}_3) \neq \emptyset) \leq \frac{q_e(\sigma + q_p)}{2^b}.$$

However, if $r > \kappa$, then for any $i \in [q_e]$, let $\lambda_i = \lfloor \lceil \mathsf{F}_i \rceil_r \rfloor_{r-\kappa} = \lfloor \lceil \mathsf{X}_i \rceil_r \rfloor_{r-\kappa}$. Let ρ_7 denote the multicollision of λ_i values. Then, by the randomness of K_{u_i} and $\lfloor \mathsf{F}_i \rfloor_c$, and using the multicollision, we have $\Pr(\mathsf{domain}(\mathsf{P}_{\mathsf{tag}}) \cap \mathsf{domain}(\mathsf{P}_3) \neq \emptyset \mid \rho_7 = \rho) \leq \frac{\rho(\sigma+q_p)}{2^\kappa}$. Using the expectation of ρ_7, and using Remark 2, we have

$$\Pr(\mathsf{domain}(\mathsf{P}_{\mathsf{tag}}) \cap \mathsf{domain}(\mathsf{P}_3) \neq \emptyset) \leq \frac{\mathsf{mcoll}(q_e, 2^{r-\kappa})(\sigma + q_p)}{2^{c+\kappa}}.$$

In both the cases, using the randomness of α_i and K_{u_i}, it can be easily seen that

$$\Pr(\mathsf{range}(\mathsf{P}_{\mathsf{tag}}) \cap \mathsf{range}(\mathsf{P}_3) \neq \emptyset) \leq \frac{q_e(\sigma + q_p)}{2^b}.$$

Hence, we get the above bound.

Finally, we settle the tag computation of all decryption queries and we set bad whenever a valid forgery occurs. For all $i \in [q_d]$, we define $\mathsf{X}'_i := \mathsf{F}'_i \oplus (K_{\mathsf{u}_i} \| 0^{b-\kappa})$. If $\mathsf{X}'_i \in \mathsf{domain}(\mathsf{P}_4)$ then we define $\mathsf{Y}'_i = \mathsf{P}_4(\mathsf{X}'_i)$. Else, $\mathsf{Y}'_i \xleftarrow{\$} \{0,1\}^b$.

Bad Event (Decryption Queries are Not Rejected). We divide this into two cases depending on whether $\mathsf{X}'_i \in \mathsf{domain}(\mathsf{P}_4)$ or not:

– We set $\mathsf{bad}_{10} = 1$ if

$$\exists i \in [q_d], \qquad \mathsf{X}'_i \in \mathsf{domain}(\mathsf{P}_4) \ \wedge \ \lfloor \mathsf{P}_4(\mathsf{X}'_i) \rfloor_\tau \oplus K_{u_i,2} = \mathsf{T}'_i.$$

It is easy to check that

Lemma 21. $\Pr(\mathsf{bad}_{10} = 1) \leq \frac{\mathsf{mcoll}(\sigma + q_p, 2^\tau) q_d}{2^\kappa}$.

Proof. Let $\mathsf{F}'_i = (\beta'_i \| \gamma'_i \| \delta'_i)$, where $|\beta'_i| = \kappa - \tau$, $|\gamma'_i| = \tau$ and $|\delta'_i| = b - \kappa$. If $\mathsf{bad}_{10} = 1$, then
(i) for some $(x_j \| y_j \| z_j) \in \mathsf{domain}(\mathsf{P}_4)$, $\mathsf{X}'_i = (x_j \| y_j \| z_j)$, $|x_j| = \kappa - \tau$, $|y_j| = \tau$ and $|z_j| = b - \kappa$, and
(ii) $y_j \oplus w_j = \mathsf{T}'_i \oplus \gamma'_i$ where $w_j = \lfloor \mathsf{P}_4(x_j \| y_j \| z_j) \rfloor_\tau$.
Let ρ_8 denote the multicollision on the values of $(y_a \oplus w_a)_a$ varying over all elements of P_4. Hence, the number of choices of j is at most ρ_8. Then, by the randomness of K_{u_i},

$$\Pr(\mathsf{bad}_{10} = 1 \mid \rho_8 = \rho) \leq \frac{\rho q_d}{2^\kappa}.$$

So, using the expectation of ρ_8, and applying Lemma 2, we have the result.

– $\mathsf{X}'_i \notin \mathsf{domain}(\mathsf{P}_4)$. Let $w_i = \lfloor \mathsf{Y}'_i \rfloor_\tau$. We set $\mathsf{bad}_{11} = 1$ if there exists $i \in [q_d]$ such that $y_i \oplus K_{\mathsf{u}_i,2} = \mathsf{T}'_i$. Similarly, by the randomness of y_i, we have

Lemma 22. $\Pr(\mathsf{bad}_{11} = 1) \leq \dfrac{q_d}{2^\tau}$.

If all these bad events do not occur, then all the decryption queries are correctly rejected for the injective partial function P_4.

Let $\mathsf{P}_{\mathsf{fin}} := \mathsf{P}_4 \cup \big((\mathsf{X}'_i, \mathsf{Y}'_i)_{i\in[q_d]}\big)$. In the offline transcript, we provide all the input-outputs of $\mathsf{P}_{\mathsf{fin}}$. Then,

$$\Theta_{\mathrm{id}} = \big((\mathsf{u}_i, \mathsf{N}_i, \mathsf{A}_i, \mathsf{M}_i, \mathsf{C}_i, \mathsf{T}_i)_{i\in[q_e]},\ (\mathsf{u}'_i, \mathsf{N}'_i, \mathsf{A}'_i, \mathsf{C}'_i, \mathsf{T}'_i, \mathsf{rej})_{i\in[q_d]},\ \mathsf{P}_{\mathsf{fin}}\big).$$

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,

$$\Pr(\Theta_{\mathrm{id}} = \theta) = \Pr(P \subseteq \Pi) \times 2^{-b(|\mathsf{P}_{\mathsf{fin}}|-|\mathsf{P}|)} \leq 1/(2^b)_{|\mathsf{P}_{\mathsf{fin}}|} = \Pr(\Theta_{\mathrm{re}} = \theta)$$

By using the H-coefficient technique, we complete the proof of the theorem.

E Proof of Theorem 4

As in the case of Ascon, this proof is very similar to that of the nonce-respecting setting of LK-Ascon. In fact, the only parts where the proof of Theorem 4 differs from the proof of Theorem 3 are the parts where the proof of Theorem 2 differs from Theorem 1. Since the adversary can now reuse the nonce for encryption queries to the same user, we only need to redefine p_i for encryption queries. The rest of the proof remains the same as that of Theorem 3. This streamlined approach renders the proof notably straightforward, thus we have opted to exclude it.

Definition of p_i

1. If there does not exist any $j \in [i-1]$ such that $\mathsf{u}_j = \mathsf{u}_i$, we define $p_i = -1$.
2. Else if there does not exist any $j \in [i-1]$ such that $\mathsf{u}_j = \mathsf{u}_i$, we define $p_0 = 0$.
3. Otherwise, there exists at least one j for which $(\mathsf{u}_j, \mathsf{N}_j) = (\mathsf{u}_i, \mathsf{N}_i)$ (since the adversary can misuse nonces). For each such j, let $p_{i,j}$ denote the length of the largest common prefix of
   – $(\mathsf{A}_{i,1}, \ldots, (\mathsf{A}_{i,a_i}, *), \mathsf{M}_{i,1}, \ldots, \mathsf{M}_{i,m_i})$ and
   – $(\mathsf{A}_{j,1}, \ldots, (\mathsf{A}_{j,a_j}, *), \mathsf{M}_{j,1}, \ldots, \mathsf{M}_{j,m_j})$.
   Here $*$ is used to distinguish associate data blocks and message blocks.
   Finally define $p_i = \max_{j<i} p_{i,j}$.

F Forgery Against Ascon Authenticity In the Nonce Misuse Setting

Here, we show an online $c/2$-bit forgery against the authenticity of Ascon. Note that the same attack is possible for LK-Ascon too.

1. Fix a nonce N and a one-block message M, and choose $2^{c/2}$ different associated data $A[1], ..., A[2^{c/2}]$.
2. Obtain $2^{c/2}$ pairs of ciphertext and tag $(C[i], T[i]) = \mathsf{E}(K, N, A[i], M)$.
3. For each i, define a new one-block message $M[i]$ such that the outer part after absorbing $M[i]$ is 0^r, i.e.,$M[i] = C[i] \oplus M$.
4. Obtain $2^{c/2}$ pairs of ciphertext and tag $(C'[i], T'[i]) = \mathsf{E}(K, N, A[i], M[i])$.
5. If there exists a pair (i_1, i_2) such that $T[i_1] = T'[i_2]$, then do the following steps. Since the attacker chooses $2^{c/2}$ different associated data, the inner part values are the same with high probability.
6. For $A[i_1]$, define a new one-block message $M^*[i_1]$ such that the outer part after absorbing $M^*[i_1]$ is 1^r.
7. Obtain $(C^*[i_1], T^*[i_1]) = \mathsf{E}(K, N, A[i_1], M^*[i_1])$.
8. Make a decryption query $\mathsf{D}(K, N, A[i_2], 1^r, T^*[i_1])$. One can expect that the forgery succeeds with high probability.