Exact Security Analysis of ASCON

Bishwajit Chakraborty, Chandranan Dhar, and Mridul Nandi

Indian Statistical Institute, Kolkata, India
{bishu.math.ynwa,chandranandhar,mridul.nandi}@gmail.com

Abstract. The Ascon cipher suite, offering both authenticated encryption with associated data (AEAD) and hashing functionality, has recently emerged as the winner of the NIST Lightweight Cryptography (LwC) standardization process. The AEAD schemes within Ascon, namely Ascon-128 and Ascon-128a, have also been previously selected as the preferred lightweight authenticated encryption solutions in the CAESAR competition. In this paper, we present a tight and comprehensive security analysis of the Ascon AEAD schemes within the random permutation model. Existing integrity analyses of Ascon (and any Duplex AEAD scheme in general) commonly include the term $DT/2^c$, where D and T represent data and time complexities respectively, and c denotes the capacity of the underlying sponge. In this paper, we demonstrate that Ascon achieves AE security when T is bounded by $\min\{2^\kappa, 2^c\}$ (where κ is the key size), and DT is limited to 2^b (with b being the size of the underlying permutation, which is 320 for Ascon). Our findings indicate that in accordance with NIST requirements, Ascon allows for a tag size as low as 64 bits while enabling a higher rate of 192 bits, surpassing the recommended rate.

Keywords: Ascon, AEAD, tight security, lightweight cryptography

1 Introduction

The Sponge function, initially proposed by Bertoni et al. at the ECRYPT Hash Workshop [3], serves as a mode of operation for variable output-length hash functions and has gained significant popularity. This is evident from the numerous Sponge-based constructions submitted in the NIST SHA-3 competition, with Keccak [7] being the notable winner. At a high level, a Sponge construction utilizes a fixed permutation π of size b and a b-bit state, which is divided into a c-bit capacity and an $r := (b - c)$-bit rate for the Sponge. The Sponge construction begins by initializing the state to zero and padding the input message using a padding function, followed by dividing it into r-bit blocks. Then, the absorption phase of the Sponge construction commences, where the message is XOR-ed with the rate part of the sponge while interleaved with applications of π. Once the absorption phase is complete, the squeezing phase begins. In this phase, the first r bits of the state are outputted as output blocks, again interleaved with applications of π.

The Duplex construction [4] is a variant of the Sponge construction and serves as a widely used approach for constructing authenticated encryption schemes. The Duplex construction maintains a state between calls and processes input strings while producing output strings that depend on all previously received inputs. At a high level, the Duplex mode is a stateful construction that comprises an initialization interface and a duplexing interface. Initialization creates an initial state using the underlying permutation π, and each duplexing call to π absorbs and squeezes r bits of data. The usage of keyed Duplex approach in constructing authenticated encryption modes is evident from the numerous submissions in competitions like CAESAR (including the winner Ascon [13,14]) and the recently concluded NIST LwC competition (with 26 total Duplex-type submissions, notably including the winner Ascon). The security analysis of keyed Duplex-type AEAD modes involves considering two parameters: the data complexity D (representing the total number of initialization and duplexing calls to π) and the time complexity T (representing the total number of direct calls to π).

1.1 Ascon

Ascon was initially introduced as a candidate in Round 1 of the CAESAR competition [11]. Subsequent versions (v1.1 and v1.2) incorporated minor modifications to the original design (version 1 [13]). The latest version (v1.2 [14]), declared as the winner of the NIST Lightweight Cryptography (LwC) project [19], includes the Ascon-128 and Ascon-128a authenticated ciphers, as well as the Ascon-Hash hash function and the Ascon-Xof extendable output function. All the schemes in the suite ensure 128-bit security and utilize a common 320-bit permutation internally, enabling the implementation of both duplex-based AEAD and sponge-based extendable-output hashing with a single lightweight primitive.

The authenticated encryption mode of Ascon is based on the duplex construction [4], specifically the MonkeyDuplex construction [6]. However, unlike MonkeyDuplex, Ascon's mode employs double-keyed initialization and double-keyed finalization to enhance its robustness. For a detailed description of the Ascon AEAD mode, please refer to Section 4.

1.2 Existing Security Analysis

It has come to our attention that previous analyses of Ascon predominantly regard it as a variant of Duplex construction (as indicated in [14]), with no specific security analysis dedicated to Ascon available in the literature. Hence, we briefly discuss the security bounds of generic Duplex constructions here. At a high level, the Sponge construction is known to achieve $2^{c/2}$ bits security, where c is the capacity of the Sponge. This security level has been extended to its keyed variations, such as MonkeyDuplex. The first result which indicates that the duplex-based modes can provide security beyond the birthday bound on the capacity c, was by Bertoni et al. [5]. However, they could only achieve

this only when the time complexity (roughly, this is the number of permutation computations an adversary does) remains well below $2^{c/2}$. In fact, the dominating term in their security analysis was

$$\frac{D^2 + DT}{2^c},$$

where D is the data complexity and T is the time complexity. In 2014 [16], and later in 2019 [17], Jovanovic et al. achieve an improved security of the form

$$\frac{(D+T)q_d}{2^c}$$

where q_d is the number of decryption queries. Andreeva et al. [2] show that the time complexity can be made close to $2^c/\mu$ where μ is the total multiplicity (i.e., the number of queries with a repeated nonce). As the nonce is allowed to repeat in decryption queries, the μ can be as large as q_d (the number of decryption queries). Hence, their security bound is essentially of the form

$$\frac{q_d T}{2^c}.$$

Considering full-state keyed Duplex, Daemen et al. [12] establish stronger bounds for the Duplex mode of operation. These bounds are based on comparing the Duplex mode to an *Ideal Extendable Input Function (IXIF)*. They also do this in a multi-user setting and take into account both respectful and misusing adversaries. The results indicate that the data limit or key could potentially be increased further. One of the dominating terms in their security bound is $\frac{LT}{2^c}$ where L represents the number of construction queries that have some common prefix to some prior query. So, an adversary can easily achieve $L = q_d$ (the number of decryption queries) as nonce is allowed to repeat in decryption queries. So, their bound essentially reduces to $\frac{q_d T}{2^c}$.

Recently, Chakraborty et al. [9] introduced a generic AEAD construction called the Transform-then-Permute (TtP) construction. They demonstrated that well-known constructions such as the keyed Sponge Duplex construction, Beetle [8], and SpoC [1] can be viewed as specific examples of this generic construction. In their work, they provided rigorous proof for a tight security bound of the TtP construction in the form of $\frac{\mu_T D}{2^c}+$ other smaller order terms, where μ_T is a parameter defined in their paper [9]. For a special class of TtP constructions where the decryption feedback function (defined in their paper) is invertible, they showed that $\mu_T = \mathcal{O}\left(\max T/2^r, T/2^\tau, T^2/2^b\right)$. This result indicates that these constructions achieve security levels much higher than $q_d T/2^c$ when D (data complexity) is significantly smaller than T (time complexity). Importantly, this holds true for the upper limits of D and T as specified by the NIST guidelines for Lightweight Cryptography (LwC). However, for other TtP constructions, such as the keyed Sponge Duplex and Ascon constructions, where the decryption feedback function is not invertible, bounding μ_T was left as an open problem for future research.

As observed, a common constraint in the existing analyses of Ascon, as well as other Duplex constructions, is the condition $DT \ll 2^c$, or similar variants where D may be replaced by q_d. It is important to note that no forgery attack matching this bound has been discovered. Notably, the best-known attack on Duplex constructions by Gilbert et al. [15] establishes a lower bound of the form $DT \gg 2^{3c/2}$.

1.3 Our Contribution

In this paper, inspired by the recently concluded NIST LwC competition, we try to provide an improved security bound for the Ascon AEAD. As already stated above, previous analyses of Ascon have treated it as a variant of the Duplex construction, overlooking its unique key robustness features, namely the double-keyed initialization and double-keyed finalization.

Our analysis establishes a tight security bound, considering the tag size τ in bits, key size κ in bits, capacity c in bits, and state size b in bits. The derived bound is given by

$$\frac{T}{2^{\min\{\kappa,c\}}} + \frac{D}{2^{\min\{\tau,c\}}} + \frac{TD}{2^b}.$$

Comparing our result with the recent analysis by Gilbert et al. [15], it becomes evident that Ascon surpasses other generic Duplex constructions in terms of security, solidifying its status as a true champion. Notably, our proof leverages the double-keyed finalization process of Ascon during tag generation, which plays a vital role in achieving such a tight and improved security bound. It should be emphasized that our proof methodology is not applicable to classical sponge constructions, as they do not incorporate a key at the final stage. Furthermore, the recent attack by Gilbert et al. [15] conclusively demonstrates that Ascon consistently offers higher security than other sponge-based modes of operation.

Lastly, in the context of NIST LwC requirements ($D \leq 2^{53}, T \leq 2^{112}$, $\kappa \geq 128$, $\tau \geq 64$), our conclusion is that a capacity size of $c = 128$ (given $b = 320$) and $\tau = 64$ is sufficient to ensure adequate security for Ascon. This choice enables a higher rate of 192 bits, thereby significantly enhancing efficiency without compromising security within the random permutation model. We believe this represents a substantial improvement compared to existing analyses.

1.4 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. Additionally, in Section 3, we elaborate on function graph structures that play a crucial role in our subsequent analyses. Moving forward, in section 4, we present a detailed examination of the Ascon AEAD scheme. We present our primary result, 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. In section 5, we present a rigorous proof of our main theorem, using the H-coefficient technique. Finally, in Section 6, we discuss the tightness of our bound, and conclude the paper.

2 Preliminaries

2.1 Notations

For all $a \leq b \in \mathbb{N}$, let $[b]$ and $[a, b]$ denote the sets $\{1, 2, \ldots, b\}$ and $\{a, a+1, \ldots, b\}$ respectively. For $n, k \in \mathbb{N}$, such that $n \geq k$, we define the falling factorial $(n)_k := n(n-1)\cdots(n-k+1)$. Note that $(n)_k \leq n^k$.

Let $\{0,1\}^n$ denote the set of bit strings of length n, and $\{0,1\}^+$ denote the set of bit strings of arbitrary length. Let λ denote the empty string and we write $\{0,1\}^* = \{\lambda\} \cup \{0,1\}^+$. For any bit string $x = x_1 x_2 \cdots x_k \in \{0,1\}^k$ of length k, and for $n \leq k$, we write $\lceil x \rceil_n := x_1 \cdots x_n$ (resp. $\lfloor x \rfloor_n := x_{k-n+1} \cdots x_k$) to denote the most (resp. least) significant n bits of x. We use $\|$ to denote the bit concatenation operation. We also abuse the notation $(x_1, \ldots, x_r)$ to denote the bit concatenation operation $x_1 \| \cdots \| x_r$ where $x_i \in \{0,1\}^*$. So, if $V := x \| z := (x, z) \in \{0,1\}^r \times \{0,1\}^c$ then $\lceil V \rceil_r = x$ and $\lfloor V \rfloor_c = z$. We use $\oplus$ to denote bitwise xor operation.

Padding and Parsing Bit String. Let $r > 0$ be an integer and $X \in \{0,1\}^*$. Let $d = |X| \% r$ (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 $(X_1, \ldots, X_x) \xleftarrow{r}_* X$ where $X_1 \| \cdots \| X_x = X$, $|X_1| = \cdots = |X_{x-1}| = r$ and

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

2.2 Authenticated Encryption with Associated Data: Definition and Security Model

An authenticated encryption scheme with associated data functionality (called AEAD in short), is a tuple of algorithms $\mathsf{AE} = (\mathsf{E}, \mathsf{D})$, called the encryption and decryption algorithms, respectively, and defined 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* $\mathcal{T}$, where

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

Here, rej indicates the tag ciphertext pair is invalid and hence rejected. Further, we require the correctness condition: $\mathsf{D}(K, N, A, \mathsf{E}(K, N, A, M)) = M$ for any $(K, N, A, M) \in \mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{M}$. For all key $K \in \mathcal{K}$, we write $\mathsf{E}_K(\cdot)$ and $\mathsf{D}_K(\cdot)$ to denote $\mathsf{E}(K, \cdot)$ and $\mathsf{D}(K, \cdot)$, respectively. In this paper, we have $\mathcal{K} = \{0,1\}^\kappa, \mathcal{N} = \{0,1\}^\nu, \mathcal{T} = \{0,1\}^\tau$ and $\mathcal{A}, \mathcal{M} = \mathcal{C} \subseteq \{0,1\}^*$.

AEAD Security in the Random Permutation Model.

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 without replacement sampling of X from $\mathcal{X}$. 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 which maps (N, A, M) to (C, T) such that $|C| = |M|$. Let

- $\Pi \xleftarrow{\$} \mathsf{Perm}(b)$,
- $\Gamma \xleftarrow{\$} \mathsf{Func}(\mathcal{N} \times \mathcal{A} \times \mathcal{M}, \mathcal{M} \times \mathcal{T})$, and
- rej denotes the degenerate function from $(\mathcal{N}, \mathcal{A}, \mathcal{M}, \mathcal{T})$ to $\{\mathsf{rej}\}$.

We use the superscript $\pm$ to denote bidirectional access to Π.

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 AEAD advantage of an adversary $\mathscr{A}$ against AE_Π is defined as*

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

Here $\mathscr{A}^{\mathsf{E}_\mathsf{K}, \mathsf{D}_\mathsf{K}, \Pi^\pm}$ denotes $\mathscr{A}$'s response after its interaction with E_K, D_K, and $\Pi^\pm$ respectively. Similarly, $\mathscr{A}^{\Gamma, \mathsf{rej}, \Pi^\pm}$ denotes $\mathscr{A}$'s response after its interaction with Γ, rej, and $\Pi^\pm$ respectively.

In this paper, we assume that the adversary is adaptive, that is it neither makes any duplicate queries nor makes any query for which the response is already known due to some previous query. Let q_e, q_d and q_p denote the number of queries to $\mathsf{E}_\mathsf{K}, \mathsf{D}_\mathsf{K}$ and $\Pi^\pm$ respectively. Let σ_e and σ_d denote the sum of input (associated data and message) lengths across all encryption and decryption queries respectively. Also, let $\sigma := \sigma_e + \sigma_d$ denote the combined construction query resources.

Remark 1. Here σ corresponds to the online or data complexity, and q_p corresponds to the offline or time complexity of the adversary. Any adversary that adheres to the resource constraints mentioned above is called an $(q_p, \sigma_e, \sigma_d)$-adversary.

2.3 H-coefficient Technique

Consider a deterministic and computationally unbounded adversary $\mathscr{A}$ trying to distinguish the real oracle (say $\mathcal{O}_{\mathrm{re}}$) from the ideal oracle (say $\mathcal{O}_{\mathrm{id}}$). Let

the transcript ω denote the query-response tuple of $\mathscr{A}$'s interaction with its oracle. Sometimes, at the end of the query-response phase of the game, if the oracle chooses to reveal any additional information to the distinguisher, then the extended definition of the transcript may also include that information. Let Θ_{re} (resp. Θ_{id}) denote the random transcript variable when $\mathscr{A}$ interacts with $\mathcal{O}_{\text{re}}$ (resp. $\mathcal{O}_{\text{id}}$). The probability of realizing a given transcript ω in the security game with an oracle $\mathcal{O}$ is known as the *interpolation probability* of ω with respect to $\mathcal{O}$. Since $\mathscr{A}$ is deterministic, this probability depends only on the oracle $\mathcal{O}$ and the transcript ω. A transcript ω is said to be *attainable* if $\Pr[\Theta_{\text{id}} = \omega] > 0$. In this paper, $\mathcal{O}_{\text{re}} = (\mathsf{E}_{\mathsf{K}}, \mathsf{D}_{\mathsf{K}}, \Pi^{\pm})$, $\mathcal{O}_{\text{id}} = (\Gamma, \mathsf{rej}, \Pi^{\pm})$, and the adversary is trying to distinguish $\mathcal{O}_{\text{re}}$ from $\mathcal{O}_{\text{id}}$ in AEAD sense.

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

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

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

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

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

A proof of theorem 1 can be found in multiple papers including [21,10,18].

2.4 Expected Multicollision in a Uniform Random Sample

Let $S := (x_i)_{i \in I}$ be tuple of elements from a set T. For any $x \in T$, we define $\mathsf{mcoll}_x(S) = |\{i \in I : x_i = x\}|$ (the number of times x appears in the tuple). Finally, we define multicollision of S as the $\mathsf{mcoll}(S) := \max_{x \in T} \mathsf{mcoll}_x(S)$. In this section, we revisit some multicollision results discussed in [9].

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

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

Lemma 1. *[9] 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 $\mathsf{Ex}[\mathsf{mcoll}(\mathsf{X}_1, \ldots, \mathsf{X}_q)] \leq \mathsf{mcoll}(q, N)$.*

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 $\mathsf{X}_1, \ldots, \mathsf{X}_q$. Thus, we have the following general result:

Lemma 2 (general multicollision bound). *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 $\mathsf{Z}_i := trunc_\tau(\mathsf{X}_i) \oplus trunc_\tau(\mathsf{Y}_i)$ for $i \in [q_1]$ and $\mathsf{Z}_i := (\mathsf{X}_i \oplus \mathsf{Y}_i)$ for $i \in [q_1 + 1, q]$ where $trunc_\tau$ represents some τ-bit truncation. For $\tau \geq 2$,*

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

3 Function Graph Structures

3.1 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. We call it an *injective partial function* if q_i's are also distinct. We 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 $p \notin \mathsf{domain}(\mathcal{L})$, $\mathcal{L}(p) = \perp$ (a special symbol to mean that the value is undefined).[1] For $f : \{0,1\}^b \dashrightarrow \{0,1\}^b$, $c \in [b-1]$, we define $\lfloor f \rfloor_c : \{0,1\}^b \dashrightarrow \{0,1\}^c$ such that $\lfloor f \rfloor_c(x) = \lfloor f(x) \rfloor_c$ whenever $f(x) \neq \perp$.

Definition 2. *Let $\mathcal{L} : \{0,1\}^b \dashrightarrow \{0,1\}^c$ for $r := b - c > 0$. We associate a labeled directed graph $G := G^{\mathcal{L}}$, called* (labeled) partial function graph*, over the set of vertices*

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

with the label set $\{0,1\}^r$ and the following labeled edge set

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

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 $u_0 \xrightarrow{x^l} u_l$. It is easy to see that if $u \xrightarrow{x} v_1$ and $u \xrightarrow{x} v_2$ then $v_1 = v_2$ (this follows from the fact that $\mathcal{L}$ is a partial function).

[1] A function is a partial function for which every output is defined.

3.2 Sampling Process of a Labeled Walk

Let $f : \{0,1\}^b \dashrightarrow \{0,1\}^b$, $x = x^k$ be a k-tuple label, $k \geq 0$, and $v_0 \in \{0,1\}^c$. We now describe a process that extends the partial function f to f' so that there is a walk

$$v_0 \xrightarrow{x_1} v_1 \xrightarrow{x_2} \cdots \xrightarrow{x_k} v_k$$

in the graph $G^{\lfloor f \rfloor_c}$. The process we define below is denoted as

$$\mathsf{Rand\_Extn}^f(z_0, x^k),$$

which randomly extends the elements of the partial function f whenever required to complete the walk.

$\underline{\mathsf{Rand\_Extn}^f(z_0, x^k)}$:

Initialize $f' = f$.
For $j = 1$ to k:

1. $v_j = f'(x_j, z_{j-1})$.
2. If $v_j = \perp$ then
   - $v_j \xleftarrow{\$} \{0,1\}^b$ and
   - $f' \leftarrow f' \cup \{(x_j \| z_{j-1}, v_j)\}$
3. $z_j = \lfloor v_j \rfloor_c$.

The described process provides a clear and effective method for successfully completing a labeled walk. It operates based on a simple rule: when the current value falls within the defined domain, we utilize the corresponding output to progress further in the walk. In cases where the current value is outside the domain, we employ a random sampling approach to determine the next output. This ensures the completion of the walk.

3.3 Partial XOR-Function Graph

Now, consider a partial function $\mathcal{P} : \{0,1\}^b \dashrightarrow \{0,1\}^b$ and $r \in [b-1]$. We define a new partial function $\mathcal{P}^{\oplus} : \{0,1\}^b \times \{0,1\}^r \dashrightarrow \{0,1\}^b$ as follows. Let $u = u' \| u''$ where $u' \in \{0,1\}^r$. Now,

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

Note that the above may not be defined, in which case we define the output $\perp$ as before. We similarly define partial function graph $G^{\oplus} := G^{\mathcal{P}^{\oplus}}$ with label edges denoted as $u \xrightarrow{x}_{\oplus} v$ (whenever $\mathcal{P}^{\oplus}(u, x) = v$). A walk

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

is denoted as $u_0 \xrightarrow{x^l}_{\oplus} u_l$. Similar to $\mathsf{Rand\_Extn}^f$ Algorithm, we now define a randomized extension algorithm for $\mathcal{P}^{\oplus}$, denoted as $\mathsf{xorRand\_Extn}^{\mathcal{P}}(v_0, x^k)$, $v_0 \in \{0,1\}^b, x_i \in \{0,1\}^r$.

$\underline{\mathsf{xorRand\_Extn}^{\mathcal{P}}(v_0, x^k)}$:

Initilaize $\mathcal{P}' = \mathcal{P}$.
For $j = 1$ to k:

1. $v_j = \mathcal{P}'(v_{j-1} \oplus (x_j \| 0^c))$.
2. If $v_j = \perp$ then
   - $v_j \xleftarrow{\$} \{0,1\}^b$ and
   - $\mathcal{P}' \leftarrow \mathcal{P}' \cup \{v_{j-1} \oplus (x_j \| 0^c), v_j)\}$

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

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

4 Ascon AEAD

In this section, we define the Ascon AEAD [14] construction. Note that the Ascon AEAD is a simple variation of the Duplex construction. Let b denote the state size of the underlying permutation Π and $0 < r < b$ be the number of bits of associated data/message processed per permutation call. We call r the rate of the Ascon construction, and $c = b - r$ is called the capacity. Let κ, ν, τ denote the key size, nonce size, and tag size respectively such that

- $\tau \leq \kappa \leq c$,
- $\kappa + \nu \leq b$,
- $\kappa + r \leq b$.

We fix a constant $IV \in \{0,1\}^{b-\kappa-\nu}$. The AEAD uses a permutation π (Ascon permutation), modeled to be the random permutation while we analyze its security.

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 $K \in \{0,1\}^\kappa$. Broadly we divide the encryption algorithm into three phases: (i) initialization, (ii) associate data and message processing, and (iii) tag generation, run sequentially.

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

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

Before we process associate data and messages, we first parse them:

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

Note that a can be zero in which case it is parsed as an empty string. But $m \geq 1$.

Associate Data and Message Processing. 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

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

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).

Remark 2. The Ascon construction uses two different permutations, p^a and p^b, where a and b indicate the specific rounds used for the underlying permutation p (called the Ascon permutation). In the Ascon implementation, p^a is employed during the initialization phase and for tag generation and verification. On the other hand, p^b is utilized for processing associated data, messages, and ciphertext. For instance, in the Ascon-128 construction, a is set to 12, while b is set to 8.

When modeling Ascon in the random permutation model, there are two options: either using the same permutation $\pi = p^a = p^b$, or utilizing independent permutations π_1 and π_2. Our analysis focuses on the assumption that the permutations are the same, which is generally more challenging to prove compared to assuming independent random permutations. A similar analysis (with bounds of the same order) can be made for the independent random permutation model.

Verification Algorithm. The decryption algorithm performs a verification process to ensure the correctness of the ciphertext and tag pair. If the verification is successful, the algorithm proceeds to generate the corresponding message. While the details of message computation are omitted in this analysis, readers can refer to [14] for a comprehensive explanation. It is important to note that 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. $(A_1, \ldots, A_a) \xleftarrow{r} \mathsf{pad}_1(A)$ and $(C_1, \ldots, C_l) \xleftarrow{r} \mathsf{pad}_2(C)$.

2. Compute $V_0 := \textsc{Init}^{\pi}(K, N)$.
3. We 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 $z_a = \lfloor V_a \rfloor_c$.
   - Case $l = 1$: We define $U = C'_l \parallel (\lfloor V_a \rfloor_{b-d} \oplus 10^*1)$.
   - Case $l \geq 2$: We 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. Rejects if $T \neq \textsc{Tag}^{\pi}(K, F)$, otherwise, it accepts.

4.1 Security bound of Ascon

Theorem 2 (Main Theorem). *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 blocks. Define $\sigma := \sigma_e + \sigma_d$. Then, we can upper bound the AEAD advantage of $\mathscr{A}$ against* Ascon *as follows:*

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

4.2 Interpretation of Theorem 2

We interpret our bound in light of the requirements proposed by NIST for the LwC competition, and the choices of the parameters, namely rate (and hence capacity) and tag size. Ascon operates with a state size (size of the permutation) $b = 320$ bits. We assume $q_p \leq 2^{112}$ and $r\sigma \leq 2^{53}$ as prescribed by NIST.

We note that the only terms which depend on the choice of r and τ are $\mathsf{mcoll}(\sigma_e, 2^r)$, $\mathsf{mcoll}(q_e, 2^\tau)$, and $\mathsf{mcoll}(\sigma + q_p, 2^\tau)$.

First, from the definition of $\mathsf{mcoll}(q, N)$, we have

$$\mathsf{mcoll}(\sigma_e, 2^r) \leq 3 \quad \forall r \geq 128.$$

Now, we fix two choices for tag size τ : 64 bits (the minimum tag size required by NIST) and 128 bits (the tag size recommended by the designers of Ascon). Again, from the definition of $\mathsf{mcoll}(q, N)$, we have

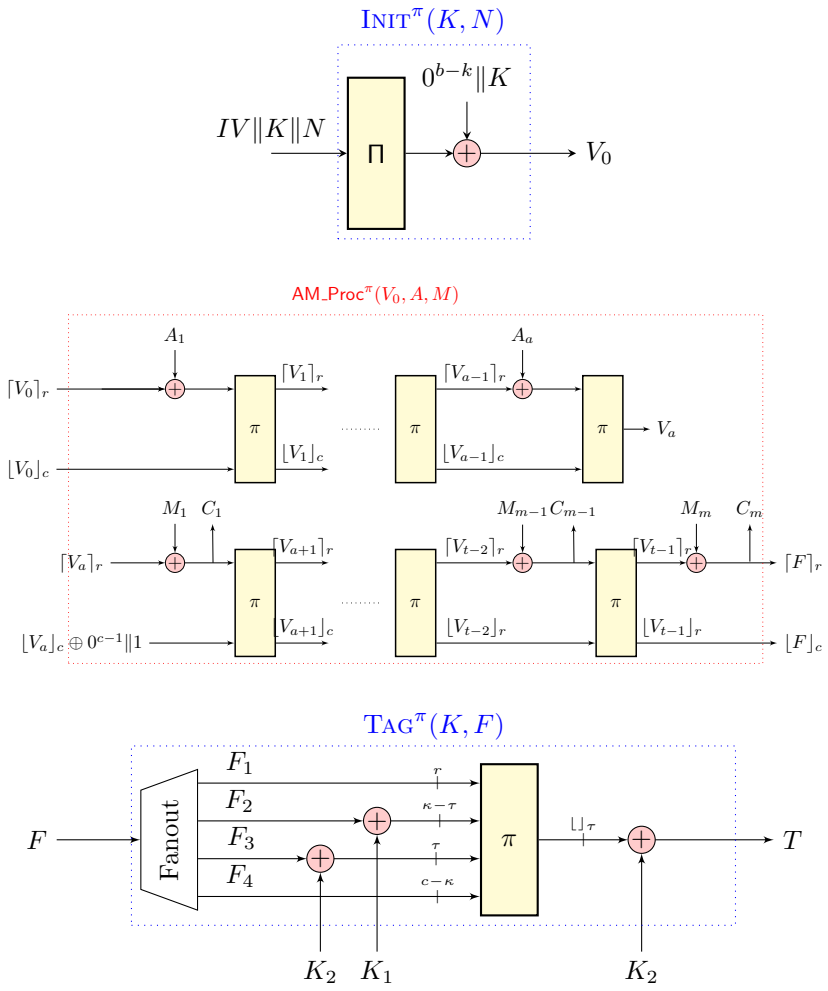

Fig. 1. Encryption in ASCON AEAD. The final ciphertext is $C = \lceil C_1 \| \cdots |C_m \rceil_{|M|}$. Here $t := a+m$, $K_1 = \lceil K \rceil_{\kappa-\tau}$, $K_2 = \lfloor K \rfloor_\tau$. The Fanout operation parses $F = F_1\|F_2\|F_3\|F_4$ such that $|F_1| = r$, $|F_2| = \kappa - \tau$, $|F_3| = \tau$ and $|F_4| = c - \kappa$. It is easy to follow that in the decryption protocol, the permutation input generated after processing C_1 is simply $C_i \| \lfloor V_{a+i-1} \oplus 0^{b-1}\|1 \rfloor_{b-r}$. Similarly after the i-th ciphertext processing where $1 < i \leq m-1$, the permutation input is simply $C_i \| \lfloor V_{a+i-1} \rfloor_{b-r}$. For processing the last block, the $|C| - r(m-1)$ most significant bits of M_m are calculated using V_{t-1} and C_m and then pad_2 is applied to determine the remaining bits of M_m. Finally, this M_m is used in the same way as the encryption protocol to generate F.

– For $\tau = 64$:

$$\mathsf{mcoll}(q_e, 2^\tau) \leq \frac{4\log_2 \sigma}{\log_2 \log_2 \sigma} < 40, \quad \text{and} \quad \mathsf{mcoll}(\sigma + q_p, 2^\tau) \leq \frac{5(q_p + \sigma)}{2^\tau}.$$

Here we assume $q_p \gg 2^\tau$.

– For $\tau = 128$:

$$\mathsf{mcoll}(q_e, 2^\tau) \leq 3, \quad \text{and} \quad \mathsf{mcoll}(\sigma + q_p, 2^\tau) \leq \frac{4\log_2(q_p+\sigma)}{\log_2\log_2(q_p+\sigma)} < 75.$$

So, if $r \geq 128, \tau = 64$, we have

$$\begin{aligned}\mathbf{Adv}^{\mathrm{AEAD}}_{\textsc{Ascon}}(\mathscr{A}) &\leq \frac{2q_d}{2^\tau} + \frac{\sigma_e^2}{2^b} + \frac{\sigma_d(q_p+\sigma_d)}{2^b} + \frac{3(\sigma_d+q_p)}{2^c} + \frac{40q_d}{2^c} \\ &\quad + \frac{q_p+\sigma}{2^\kappa} + \frac{q_eq_d + (q_e+q_d)(\sigma+q_p)}{2^b} + \frac{5(q_p+\sigma_e)q_d}{2^{\kappa+\tau}} \\ &= O(\frac{q_d}{2^\tau}) + O(\frac{\sigma q_p}{2^b}) + O(\frac{q_p}{2^\kappa}) + O(\frac{q_p}{2^c})\end{aligned}$$

(assuming $\sigma \leq q_p$).

If $r \geq 128, \tau = 128$, we have

$$\begin{aligned}\mathbf{Adv}^{\mathrm{AEAD}}_{\textsc{Ascon}}(\mathscr{A}) &\leq \frac{2q_d}{2^\tau} + \frac{\sigma_e^2}{2^b} + \frac{\sigma_d(q_p+\sigma_d)}{2^b} + \frac{3(\sigma_d+q_p)}{2^c} + \frac{3q_d}{2^c} \\ &\quad + \frac{q_p+\sigma}{2^\kappa} + \frac{q_eq_d + (q_e+q_d)(\sigma+q_p)}{2^b} + \frac{75q_d}{2^\kappa} \\ &= O(\frac{q_d}{2^\tau}) + O(\frac{\sigma q_p}{2^b}) + O(\frac{q_p}{2^\kappa}) + O(\frac{q_p}{2^c}).\end{aligned}$$

Given that the key size κ is at least 128 bits (required by NIST), we can see that Ascon is secure even when $c = 128$ (implying $r = 192$), and $\tau = 64$.

Remark 3. Our assumption $\kappa \leq c$ is only for the sake of simplicity as this implies the key is always xor-ed in the capacity part. However, it can be easily verified that the analysis remains the same even when $\kappa > c$.

5 Proof of Theorem 2

5.1 Description of the Real World

The real-world samples $K \xleftarrow{\$} \{0,1\}^\kappa$ 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 would be of the form

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

where 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). After all queries have been made, all inputs-outputs used in Π for all encryption and decryption queries have been included

in the offline transcript. Let $\mathsf{P}_{\mathsf{fin}}$ denote the extended partial function and clearly, all encryption and decryption queries are determined by $\mathsf{P}_{\mathsf{fin}}$. Note that the key K is also determined from the domain of $\mathsf{P}_{\mathsf{fin}}$. It is implicitly understood that the domain and range elements of $\mathsf{P}_{\mathsf{fin}}$ are given in order of the execution of the underlying permutation to compute all encryption and decryption queries. Let

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

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

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

5.2 Description of the Ideal World

Now we describe how the ideal oracle behaves with the adversary $\mathscr{A}$. This description consists of two primary phases: (i) the online phase, which encompasses the actual interaction between the adversary and the ideal oracle, and (ii) the offline phase, which occurs after the online phase and involves the ideal oracle sampling intermediate variables to ensure compatibility with the Ascon construction.

The offline phase is further segmented into several stages, each dependent on events defined over the preceding stages. In the event of a bad event occurring at any stage, the ideal oracle has the option to either abort or exhibit arbitrary behavior. To effectively analyze the situation, 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 simplify notation, we utilize the same notations 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 $(\mathsf{N}_i, \mathsf{A}_i, \mathsf{M}_i)$, $\forall i \in [q_e]$, respond randomly:

$$\mathsf{C}_i \xleftarrow{\$} \{0,1\}^{|\mathsf{M}_i|},\ \mathsf{T}_i \xleftarrow{\$} \{0,1\}^{\tau},\ \ \mathsf{return}(\mathsf{C}_i, \mathsf{T}_i).$$

- On i-th Decryption Query $(\mathsf{N}'_i, \mathsf{A}'_i, \mathsf{C}'_i, \mathsf{T}'_i)$, $i \in [q_d]$, 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

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

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

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

for which the encryption query is made later. It is important to note that the adversary is not allowed to make a decryption query that matches a previous encryption query. However, there is a possibility that a decryption query accidentally matches an encryption query made subsequently. This situation is referred to as a "bad event" and is of concern. Since the adversary has the capability to make nonce-respecting encryption queries only, we can establish an upper bound for the probability of bad_1 as given in the Lemma below. Although the proof for this is omitted here, it can be straightforwardly derived from the description of the ideal world for encryption queries (by looking at the randomness of the tag values).

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

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.

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

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

Bad Event: 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. The proof of the above statement is straightforward as it is easy to see that $\mathsf{V}_{i,j}$'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 P_E. This would ensure

the compatibility of $\mathsf{P}_1 \sqcup \mathsf{P}_\mathsf{E}$ (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 $\mathsf{N}_j = \mathsf{N}'_i$, we define $p_i = -1$.
2. Otherwise, there exists a unique j for which $\mathsf{N}_j = \mathsf{N}'_i$ (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 $p_i = -1$:
   – $\mathsf{V}'_{i,0} \xleftarrow{\$} \{0,1\}^b$.
   – 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 $0 \leq p_i \leq a'_i$:
   – $\mathsf{V}'_{i,p_i} := \mathsf{V}_{j,p_i}$.
   – 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 $a'_i < p_i < t_i$:
   - $\mathsf{V}'_{i,p_i} := \mathsf{V}_{j,p_i}$.
   - 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
$$\mathsf{V}'_{i,p_i} \xrightarrow{\mathsf{C}'_{i,p_i-a'_i+1}} \mathsf{V}'_{i,p_i+1} \xrightarrow{\mathsf{C}'_{i,p_i-a'_i+2}} \cdots \xrightarrow{\mathsf{C}'_{i,c_i-1}} \mathsf{V}'_{i,a'_i+c_i-1}.$$
4. For $i = 1$ to q_d with $p_i = t_i$:
   - $\mathsf{V}'_{i,a'_i+c_i-1} := \mathsf{V}_{j,a'_i+c_i-1}$.

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

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

Given that this bad event does not hold, $\mathsf{P}_\mathsf{E} \cup \mathsf{P}_1$ 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 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 $p_i = c'_i - 1$ 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 $p_i \neq c'_i - 1$. 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 2, we have the above bound. □

We also have to consider some other ways to become a valid forgery. Now, we reach the time to sample the key

$$K = (K_1, K_2) \xleftarrow{\$} \{0,1\}^\kappa, \quad K_2 \in \{0,1\}^\tau.$$

Let

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

Now, we can define all remaining input-outputs for the underlying permutation used in the initialization and tag generation phase as follows:

1. For all $i \in [q_e]$,
   - $\mathsf{I}_i := IV \| K \| \mathsf{N}_i$, $\mathsf{O}_i := \mathsf{V}_{i,0} \oplus 0^{b-\kappa} \| K$,
   - $\mathsf{X}_i := \mathsf{F}_i \oplus 0^r \| K \| 0^{c-\kappa}$, $\mathsf{Y}_i := \alpha_i \| (\mathsf{T}_i \oplus K_2)$, where $\alpha_i \xleftarrow{\$} \{0,1\}^{b-\tau}$.
2. For all $j \in \mathcal{J}$,
   - $\mathsf{I}'_j := IV \| K \| \mathsf{N}'_j$ 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 $\mathsf{N}'_j = \mathsf{N}_i$, and we define $\mathsf{I}'_j := \mathsf{I}_i$, $\mathsf{O}'_j := \mathsf{O}_i$.

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

Bad Event (Permutation Compatibility of $\mathsf{P}_{\text{IN-TAG}}$ and P_2). We define $\mathsf{bad}_7 = 1$ if one of the following holds:

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

Once again, if this bad event does not hold, $\mathsf{P}_3 := \mathsf{P}_2 \sqcup \mathsf{P}_{\mathsf{in\text{-}tag}}$ is an injective partial function. By using the randomness of K, $\mathsf{V}_{i,0}$ and $\mathsf{V}'_{i,0}$ we can easily bound the probability of bad_7 as stated below.

Lemma 8. $\Pr(\mathsf{bad}_7 = 1) \leq \dfrac{q_p + \sigma}{2^\kappa} + \dfrac{q_e q_d + (q_e + q_d)(\sigma + q_p)}{2^b}$.

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 $\mathsf{X}'_i := \mathsf{F}'_i \oplus 0^r \| K \| 0^{c-\kappa}$. If $\mathsf{X}'_i \in \mathsf{domain}(\mathsf{P}_3)$ then we define $\mathsf{Y}'_i = \mathsf{P}_3(\mathsf{X}'_i)$. Else, $\mathsf{Y}'_i \xleftarrow{\$} \{0,1\}^b$.

Bad Event (Decryption Queries are Not Rejected). We divide this into two cases depending on whether $\mathsf{X}'_i \in \mathsf{domain}(\mathsf{P}_3)$ or not:

- Let $\mathsf{F}'_i = (\beta'_i \| x'_i \| \gamma'_i)$, where $|\beta'_i| = r + \kappa - \tau$, $|x'_i| = \tau$ and $|\gamma'_i| = c - \kappa$. We set $\mathsf{bad}_8 = 1$ if

$$\exists i \in [q_d], \qquad \mathsf{X}'_i \in \mathsf{domain}(\mathsf{P}_3) \ \wedge \ \lfloor \mathsf{P}_3(\mathsf{X}'_i) \rfloor_\tau \oplus K_2 = \mathsf{T}'_i.$$

  If $\mathsf{bad}_8 = 1$, then
  (i) for some $(\beta_j \| x_j \| \gamma_j) \in \mathsf{domain}(\mathsf{P}_3)$, $\mathsf{X}'_i = (\beta_j \| x_j \| \gamma_j)$, $|\beta_j| = r + \kappa - \tau$, $|x_j| = \tau$ and $|\gamma_j| = c - \kappa$, and
  (ii) $x_j \oplus y_j = \mathsf{T}'_i \oplus x'_i$ where $y_j = \lfloor \mathsf{P}_3(\beta_j \| x_j \| \gamma_j) \rfloor_\tau$.
  Let ρ_4 denote the multicollision on the values of $(x_a \oplus y_a)_a$ varying over all elements of P_3. Hence, the number of choices of j is at most ρ_4. Then, by the randomness of K,

$$\Pr(\mathsf{bad}_8 = 1 \mid \rho_4 = \rho) \leq \frac{\rho q_d}{2^\kappa}.$$

  So, using the expectation of ρ_4, and applying Lemma 2,we have

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

- $\mathsf{X}'_i \notin \mathsf{domain}(\mathsf{P}_3)$. Let $y_i = \lfloor \mathsf{Y}'_i \rfloor_\tau$. We set $\mathsf{bad}_9 = 1$ if there exists $i \in [q_d]$ such that $y_i \oplus K_2 = \mathsf{T}'_i$. Similarly, by the randomness of y_i, we have

  Lemma 10. $\Pr(\mathsf{bad}_9 = 1) \leq \dfrac{q_d}{2^\tau}$.

Let bad denote the union of all bad events, namely $\cup_{i=1}^{9} \mathsf{bad}_i$. By Lemmas 3 through 10, we have shown that

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

If all these bad events do not occur, then all the decryption queries are correctly rejected for the injective partial function P_3.

Let $\mathsf{P}_{\mathsf{fin}} := \mathsf{P}_3 \cup \big((\mathsf{X}'_i, \mathsf{Y}'_i)_{i\in[q_d]}\big)$. In the offline transcript, we provide all the input-outputs of $\mathsf{P}_{\mathsf{fin}}$. Then,

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

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 our main theorem.

6 Final Discussion

In this paper, we have proved a bound for Ascon AEAD, 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 in the following form:

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

One can easily see that these bounds are tight:

- $\frac{q_p}{2^\kappa}, \frac{q_d}{2^\kappa}$ correspond to generic attacks which guess the key in primitive calls or decryption queries.
- $\frac{q_d}{2^\tau}$ is also a generic attack that guesses the tag in decryption queries.
- Attacks for the terms $\frac{\sigma_e^2}{2^b}, \frac{q_p}{2^c}, \frac{q_d}{2^c}, \frac{\sigma_d^2}{2^b}$ and $\frac{q_p\sigma_d}{2^b}$ can be constructed by observing state collisions in the encryption, primitive and decryption queries.

Further, when $\tau \leq \min\{\kappa, c\}$, the obtained security bound can be reduced to

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

where T is the time complexity and D is the data complexity of the adversary.

We would like to again emphasize that our analysis cannot be directly applied to general Sponge constructions without the double-keyed tag generation/verification protocol. Exploring the security of sponge constructions and achieving improved security, considering the gap between the current known security bounds and recent attacks [15], poses an interesting research problem.

Finally, in the multi-user setting, it is worth noting that our analysis indicates that the first term in the bound for bad_7 (Lemma 8) becomes $\frac{\mu(q_p+\sigma)}{2^\kappa}$, where μ denotes the number of users. Therefore, our current result does not directly extend to the multi-user setting, and a separate analysis would be required to address it.

References

1. Riham AlTawy, Guang Gong, Morgan He, Ashwin Jha, Kalikinkar Mandal, Mridul Nandi, and Raghvendra Rohit. Spoc. *Submission to NIST LwC Standardization Process (Round 2)*, 2019.
2. Elena Andreeva, Joan Daemen, Bart Mennink, and Gilles Van Assche. Security of keyed sponge constructions using a modular proof approach. In Gregor Leander, editor, *Fast Software Encryption - 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers*, volume 9054 of *Lecture Notes in Computer Science*, pages 364–384. Springer, 2015.
3. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Sponge functions. In *ECRYPT Hash Workshop 2007. Proceedings*, 2007.
4. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Duplexing the sponge: Single-pass authenticated encryption and other applications. In Ali Miri and Serge Vaudenay, editors, *Selected Areas in Cryptography - 18th International Workshop, SAC 2011, Toronto, ON, Canada, August 11-12, 2011, Revised Selected Papers*, volume 7118 of *Lecture Notes in Computer Science*, pages 320–337. Springer, 2011.
5. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. On the security of the keyed sponge construction. In *Symmetric Key Encryption Workshop 2011. Proceedings*, 2011.
6. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Permutation-based encryption, authentication and authenticated encryption. In *DIAC 2012*, 2012.
7. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Keccak. In *Advances in Cryptology - EUROCRYPT 2013. Proceedings*, pages 313–314, 2013.
8. Avik Chakraborti, Nilanjan Datta, Mridul Nandi, and Kan Yasuda. Beetle family of lightweight and secure authenticated encryption ciphers. *Cryptology ePrint Archive*, 2018.
9. 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.
10. Shan Chen and John P. Steinberger. Tight security bounds for key-alternating ciphers. In *Advances in Cryptology - EUROCRYPT 2014. Proceedings*, pages 327–350, 2014.
11. The CAESAR Committee. Caesar: competition for authenticated encryption: security, applicability, and robustness, 2014.
12. Joan Daemen, Bart Mennink, and Gilles Van Assche. Full-state keyed duplex with built-in multi-user support. In *Advances in Cryptology - ASIACRYPT 2017. Proceedings, Part II*, pages 606–637, 2017.
13. Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. ASCON v1. Submission to the CAESAR Competition, 2014.
14. Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. ASCON. Submission to NIST LwC Standardization Process (FINALIST), 2019.
15. Henri Gilbert, Rachelle Heim Boissier, Louiza Khati, and Yann Rotella. Generic attack on duplex-based aead modes using random function statistics. In *Advances in Cryptology–EUROCRYPT 2023: 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part IV*, pages 348–378. Springer, 2023.

16. Philipp Jovanovic, Atul Luykx, and Bart Mennink. Beyond 2 c/2 security in sponge-based authenticated encryption modes. In *Advances in Cryptology - ASIACRYPT 2014. Proceedings, Part I*, pages 85–104, 2014.
17. Philipp Jovanovic, Atul Luykx, Bart Mennink, Yu Sasaki, and Kan Yasuda. Beyond conventional security in sponge-based authenticated encryption modes. *J. Cryptology*, 32(3):895–940, 2019.
18. 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.
19. 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`.
20. Jacques Patarin. *Etude des Générateurs de Permutations Pseudo-aléatoires Basés sur le Schéma du DES*. PhD thesis, Université de Paris, 1991.
21. Jacques Patarin. The "coefficients H" technique. In *Selected Areas in Cryptography - SAC 2008. Revised Selected Papers*, pages 328–345, 2008.