INDIAN STATISTICAL INSTITUTE, KOLKATA

Provable Security in Idealised Models

A thesis submitted in fulfilment of the requirements
for the degree of Doctor of Philosophy
in the
Cryptology Research Group
Applied Statistics Unit

by

Chandranan Dhar

Supervised By

Mridul Nandi

July, 2024

List of Publications

This thesis is based on the following publications. They are mentioned here in their order of appearance in the thesis.

[1] Chandranan Dhar, Yevgeniy Dodis, and Mridul Nandi. "Revisiting Collision and Local Opening Analysis of ABR Hash". In Dana Dachman-Soled, editor, *3rd Conference on Information-Theoretic Cryptography - ITC 2022*, volume 230 of LIPIcs, pages 11:1-11:22. https://doi.org/10.4230/LIPIcs.ITC.2022.11

[2] Bishwajit Chakraborty, Chandranan Dhar, and Mridul Nandi. "Exact security analysis of ASCON". In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology - ASIACRYPT 2023*, Proceedings, Part III, pages 346–369. https://doi.org/10.1007/978-981-99-8727-6_12

[3] Bishwajit Chakraborty, Chandranan Dhar, and Mridul Nandi. "Tight Multi-user Security of ASCON and Its Large Key Extension". In Tianqing Zhu and Yannan Li, editors, *The 29th Australasian Conference on Information Security and Privacy - ACISP 2024*, Proceedings, Part I, pages 57-76. https://doi.org/10.1007/978-981-97-5025-2_4

[4] Arghya Bhattacharjee, Ritam Bhaumik, and Chandranan Dhar. "Universal Context Commitment without Ciphertext Expansion". Submitted to *Advances in Cryptology - ASIACRYPT 2024*. Submission #449.

Statement

I affirm that the thesis entitled "Provable Security in Idealised Models" and the content contained within are the result of my own research efforts. This work was undertaken during my pursuit of a doctoral degree at the Indian Statistical Institute. I attest that none of the material presented in this thesis has been previously submitted for any degree or qualification at this or any other institution. Whenever I have drawn upon or built upon the published work of others, I have provided clear attribution. Any direct quotations from the work of others are properly cited. This thesis represents my own independent work, except for acknowledged quotations and collaborations. I have acknowledged all significant sources of assistance. In cases where the thesis includes collaborative work, I have clearly indicated the contributions of others and obtained their permission to include our joint work in this thesis.

Chandranan Dhar

July 18, 2024

"We live on an island surrounded by a sea of ignorance. As our island of knowledge grows, so does the shore of our ignorance."

John Archibald Wheeler

Abstract

This thesis is a compilation of provable security analyses of various cryptographic constructions in idealised models. The first construction examined is the ABR hash. We revisit the existing proof of the ABR hash in the random oracle model and identify significant errors in the proof. Although we are unable to correct the original proof, we establish the security of the ABR tree of height 3 from scratch, addressing the first non-trivial case.

As our second contribution, we conduct a tight and comprehensive security analysis of the Ascon AEAD mode in the random permutation model. We show that the efficiency of Ascon can be increased by 50%, and the tag size can be halved without losing any security.

In the third contribution, we extend our security analysis of Ascon to the multi-user setting, providing tight security bounds for both nonce-respecting and nonce-misuse adversaries. Additionally, we propose LK-Ascon, a variant of Ascon with a key size of up to 256 bits, offering improved multi-user security compared to Ascon.

As the final contribution, we introduce PACT, a transform that converts any authenticated encryption mode into a context-committing one without any output length expansion. PACT achieves this with a single call to a collision-resistant unkeyed hash function and one call to a block cipher, with the analysis performed in the ideal cipher model. We also propose comPACT, a faster version of PACT which gives a nonce-respecting committing authenticated encryption scheme.

Acknowledgment

First and foremost, I would like to express my deepest gratitude to my Ph.D. supervisor, Professor Mridul Nandi, for his invaluable guidance, sustained patience, and insightful feedback throughout my Ph.D. journey. Besides my supervisor, I am sincerely grateful to the other professors in our unit for their support and constructive feedback during the annual evaluations of my research. I also appreciate all my coursework teachers who made my transition from mathematics to cryptography smoother.

I would like to extend special thanks to my co-authors Yevgeniy Dodis, Bishwajit Chakraborty, Arghya Bhattacharjee, and Ritam Bhaumik. I am equally thankful to my colleagues Bishwajit da, Suprita di, Avishek da, Samir da, Arghya da, Anik da, Jyotirmoy da, Abishanka da, Sayantan da, Animesh da, Debasmita, Soumit, Suman, Gourab, and Subha for their unwavering support in academic discussions and moral support. Without them, life at ISI would have been far less enjoyable. I am also grateful to the staff of the Applied Statistics Unit, ISI Kolkata, for their constant help with official, administrative, and other relevant tasks.

I am deeply indebted to my family—my parents, my brother, and my grandmother—for their continuous support and for always being there when I needed them. Finally, I would like to thank my fiancée, Samayita, who has been by my side through thick and thin during almost my entire Ph.D. tenure and has been my go-to person for everything outside academia.

Chandranan Dhar

ISI, Kolkata

July 18, 2024

Contents

List of Figures

To my future self – for hopefully never having to read this entire thing again, but knowing that it was all worth it.

Chapter 1

Introduction

Cryptography, the science of secure communication, has a rich history dating back to ancient times when ciphers such as substitution ciphers and tools like scytales were used to conceal messages. However, modern cryptography as we know it began to take shape in the mid-twentieth century with the advent of computers and the increasing need for secure communication in the digital age.
One of the major developments in modern cryptography was the distinction between two fundamental approaches: *public-key cryptography* (or *assymetric cryptography*) and *symmetric cryptography*. Public-key cryptography, introduced by Whitfield Diffie and Martin Hellman in 1976 [54], revolutionised the field by allowing secure communication over insecure channels without the need for a shared secret key. This approach relies on the use of two keys, a public key for encryption and a private key for decryption, making it suitable for a wide range of applications such as secure online communication and digital signatures.
On the other hand, symmetric cryptography, which predates public-key cryptography, involves the use of a single shared secret key for both encryption and decryption. While less complex than public-key cryptography, symmetric cryptography is often more efficient and is still widely used in applications where speed and resource constraints are paramount. Since our work focuses on the symmetric setup, we would like to discuss some fundamental elements of it before delving into the original contribution of the thesis.

1.1 Symmetric Cryptography

In symmetric cryptography, the same cryptographic keys are used for both the encryption of plaintext and the decryption of ciphertext. Traditionally, symmetric

encryption employs either block ciphers or stream ciphers as foundational primitives and constructs various cryptographic modes on top of these primitives to achieve diverse security objectives. Both types of primitives aim to provide confidentiality and remain extremely popular, even though a significant part of the design space has recently been occupied by tweakable block ciphers and cryptographic permutations. First, let us briefly discuss these primitives:

- **Block ciphers** are designed to encrypt fixed-size blocks of data, typically 64 or 128 bits, as single units. To accommodate data that does not perfectly align with the block size, padding schemes are used to extend the plaintext to the required length. A block cipher consists of an encryption algorithm and a decryption algorithm, where the decryption process is the inverse of the encryption process. Each algorithm requires two inputs: an input block and a secret key, which ensures that the transformation is both secure and unique. For any given key, the block cipher performs a permutation across the set of all possible input blocks, effectively scrambling the data in a reversible manner. Notably, the AES (Advanced Encryption Standard) [48] is a widely adopted block cipher, standardised by NIST in 2001, which uses 128-bit blocks and supports key sizes of 128, 192, or 256 bits.

- **Tweakable block ciphers**, introduced by Liskov et al. [91] are an advanced variation of traditional block ciphers, designed to offer enhanced flexibility and security. In addition to the usual inputs of a plaintext block and a secret key, tweakable block ciphers incorporate an additional parameter known as the "tweak", which can be viewed as an additional input that modifies the encryption process without the need to change the key. The purpose of the tweak is to provide context-specific transformations, allowing the same key to encrypt identical plaintext blocks differently depending on the tweak value.

- **Stream ciphers** are a type of symmetric cryptographic algorithm designed to encrypt data in a continuous stream, one bit or byte at a time, rather than in fixed-size blocks. This method allows for greater efficiency and speed, making stream ciphers particularly suitable for environments where data arrives in varying sizes or requires real-time encryption, such as in network communication and secure data streaming. A stream cipher generates a keystream, a sequence of pseudo-random bits, which is then combined with the plaintext bits using an operation like XOR to produce the ciphertext.

The keystream is typically generated based on an initial secret key and, in many cases, an additional value called a nonce or initialisation vector (IV) to ensure that identical plaintexts encrypt differently each time. Prominent examples of stream ciphers include RC4 [106], which has been widely used in protocols like SSL/TLS, and the more modern and secure Salsa20 [20] and ChaCha20 [19], known for their robustness and efficiency.

- **Cryptographic permutations** are increasingly being used directly as cryptographic primitives. A cryptographic permutation is a bijective function that maps an input bit string of a fixed length to an output bit string of the same length. Permutations, when employed directly, play a crucial role in various modern cryptographic constructions, most notably sponge functions. Sponge functions operate by iteratively applying a fixed permutation to an internal state to process input data and produce output data. The Keccak algorithm [26], which forms the basis of the SHA-3 hash standard [62], is a prime example of a cryptographic sponge that relies heavily on permutations for its security properties.

Most cryptographic primitives operate on fixed-size blocks, except for stream ciphers, which encrypt data a single bit or byte at a time. Variable-length schemes are often built from these fixed-length primitives by means of a **mode of operation**. These modes define how data is processed across multiple blocks, ensuring confidentiality, integrity, and sometimes authentication.
Common modes include *Electronic Codebook (ECB)* [60], where each block is encrypted independently, making it susceptible to pattern analysis; *Cipher Block Chaining (CBC)* [60], which introduces diffusion by XOR-ing each plaintext block with the previous ciphertext block before encryption; and *Counter (CTR) mode* [60], which generates a keystream by encrypting a counter value with the key and XOR-ing it with the plaintext, allowing parallel encryption. Some modes, such as *Galois/Counter Mode (GCM)* [61], integrate encryption with authentication, providing both confidentiality and integrity. The choice of mode depends on the security needs and performance constraints of the specific application.
Symmetric algorithms are crucial for communication security because they rely on simple operations (e.g. XOR, logical AND, etc.), enabling them to achieve high speeds in both software and hardware implementations. They are significantly faster and more lightweight than public-key algorithms, while also using

much shorter encryption keys. The security and efficiency of modern communications heavily depend on symmetric algorithms, making symmetric cryptography an essential and continually evolving branch of modern cryptography.

1.2 Provable Security

The methods to analyse the security of cryptographic constructions can be divided into two main classes: *the cryptanalytic approach* and *the provable security approach*.

The cryptanalytic approach involves directly attacking the cryptographic construction to identify weaknesses and potential vulnerabilities. This approach employs a variety of techniques to break the security assumptions of the cryptographic system. The cryptanalytic approach can further be divided into several methods based on the techniques used to attack cryptographic algorithms. Three of the most prominent methods are *differential cryptanalysis* [28], *linear cryptanalysis* [92], and *integral cryptanalysis* [88]. Cryptanalytic techniques are invaluable for understanding the practical security of cryptographic constructions, providing real-world insights by revealing how and where a system might fail under attack.

In contrast, the provable security approach involves mathematically proving the security of a cryptographic construction under specific assumptions. This method provides a rigorous theoretical framework to demonstrate that breaking the cryptographic system is infeasible based on certain hard problems. Provable security offers theoretical guarantees that the cryptographic construction is secure against all efficient adversaries within the defined model, providing confidence that, assuming the underlying assumptions hold, the system cannot be broken by any feasible attack.

Both the cryptanalytic and provable security approaches are essential for a comprehensive analysis of cryptographic constructions. In practice, these approaches are complementary. A cryptographic construction is typically subjected to extensive cryptanalytic testing to uncover practical weaknesses while also being supported by provable security arguments to ensure its theoretical robustness. This dual approach helps ensure that cryptographic systems are both practically secure against known attacks and theoretically sound under formal security models. Since our contributions align with the provable security approach, we explore this in more detail.

The provable security approach is mostly applicable to *modes of operation*, which are built upon *primitive functions*. While analysing the security of a mode of operation, there are two major ways to model the primitive functions it is built upon – *the standard model* and *the idealised models*.
In the standard model, the idea is to relate the security of a scheme to the security of the underlying primitive (also known as a *security reduction*). The standard model reflects realistic assumptions and practical constraints, making it highly relevant for real-world applications. However, it can sometimes be challenging to achieve rigorous security proofs in the standard model due to the complexity and potential weaknesses of real-world primitives.
The ideal models, on the other hand, assume that the primitive functions behave like perfect, idealised versions of themselves. For example, a block cipher might be modelled as an ideal cipher (i.e. it behaves as a random permutation when a key is fixed), or a hash function might be modelled as a random oracle. In these models, the primitives are assumed to have no structural weaknesses and behave in a perfectly random manner. The idealised models simplify security analysis by abstracting away the details of the underlying primitives and treating them as black boxes with ideal properties. This can make it easier to prove strong security guarantees for cryptographic constructions.
However, idealised models are often criticised for being overly optimistic, as real-world primitives may not exhibit ideal behaviour. Despite their limitations, the idealised models are useful for gaining insights into the fundamental properties of cryptographic constructions and for establishing baseline security guarantees. They help identify potential weaknesses and inform the design of more robust systems. It is important to note that the idealised models are not uniform across all scenarios; its characteristics vary depending on which primitive is being idealised. We now have a brief look at the different idealised security models used in symmetric cryptography.

1.3 Idealised Security Models

The Random Oracle Model

The *Random Oracle Model (ROM)*, formalised by Bellare and Rogaway [15], assumes that a primitive is replaced by a publicly accessible random function (the random oracle). This means that the adversary cannot compute the result of the

function by itself: it must query the random oracle. The ROM was used for some time before its formalisation [66, 70, 71], and continues to see widespread use today [16, 17]. Proofs conducted in the ROM often yield schemes that are provably secure and more efficient than those proven secure in the standard model, contributing to the ROM's extensive adoption.
Random oracles are theoretical constructs that do not exist in reality. When cryptographic schemes that have been proven secure in the ROM are to be deployed in practice, a real-world counterpart must be chosen to simulate the random oracle. This process is known as "instantiation". Typically, cryptographic hash functions (or compression functions for fixed domains) are used for this purpose. The security proofs in the ROM ensure that if the adversary treats the instantiated random oracle as a black box, without exploiting its internal structure any peculiarities of the instantiation, then the security guarantees hold in the standard model.

The Ideal Cipher Model

The *Ideal Cipher Model (ICM)* is another idealised computational framework, similar to the ROM. Instead of a publicly accessible random function, the ICM involves a publicly accessible random block cipher, or ideal cipher. This block cipher operates with a k-bit key and has n-bit inputs and outputs, selected uniformly at random from all possible block ciphers of this format. Essentially, this is equivalent to having a family of 2^k independent random permutations. All parties, including adversaries, can query the ideal block cipher for both encryption and decryption using any given key. The concept of the ICM dates back to Shannon [113] and has been employed in various cryptographic contexts (see, for example, [30, 51, 63, 87]). Under certain settings, the ROM and the ICM have been shown to be equivalent [79].

The Random Permutation Model

The *Random Permutation Model (RPM)*, akin to the Random Oracle Model (ROM) and Ideal Cipher Model (ICM), provides an idealised framework in cryptography. In the RPM, instead of a random function or block cipher, parties interact with a publicly accessible random permutation. This permutation is chosen uniformly at random from the set of all possible permutations of a given domain. This model assumes that the adversary, like all parties, can query the permutation for any input-output pair under the constraint of the chosen permutation. In

recent times, several works in sponge-based hash and authenticated modes (see, for example [31, 33, 36, 94]) have been done in the RPM.

1.4 Contributions and Outline

This thesis is a compilation of several results related to the provable security of various modes of operation, specifically hash functions and authenticated encryption schemes. Here, we discuss the key contributions briefly in the order they appear in the thesis.

In Chapter 2, we introduce basic cryptographic preliminaries. After briefly outlining the notations used in the thesis, we discuss cryptographic hash functions and authenticated encryption, and define the security notions associated with them that we explore later in the thesis.

In Chapter 3, we revisit the collision security of the ABR hash mode [4], a tree hash mode that is more efficient compared to the Merkle tree. The analysis is conducted in the ROM, where we model the compression function used for the ABR hash as a random oracle. We identify significant gaps in the existing security proof and highlight some missing cases. Although we are unable to patch the existing proof, we demonstrate the collision security of ABR_3 (the ABR of height 3), the first non-trivial case, but are unable to extend the result to the general case. Furthermore, we show that the ABR hash cannot have any non-trivial local opening that provides birthday security. The contents of this chapter have been published in [52].

In Chapter 4, we bound the AEAD security advantage of the Ascon mode, which was previously shown to be at least as secure as the Duplex mode. By modelling the Ascon permutation as an ideal permutation, we show that Ascon is significantly more secure than Duplex in the single-user nonce-respecting setting. While the dominating term in the security bound for the Duplex is of the order $DT/2^c$, where D and T represent the data and time complexities respectively, and c is the capacity of the underlying sponge, we show that for Ascon, the term $DT/2^c$ can be improved to $(DT/2^b + T/2^c)$. Interpreting this in light of the NIST LWC requirements, our bounds allow the rate of Ascon to be increased by 50%, thus increasing efficiency without degrading security. We also show that the tag size of Ascon can be halved. The contents of this chapter have been published in [34].

In Chapter 5, we extend our results for Ascon to the multi-user setting. We show that unless the number of users becomes excessively large, the Ascon AEAD

mode remains secure in this setting. However, since the dominating terms in this case are $(T/2^c + \mu T/2^\kappa)$, where μ denotes the number of users and κ is the key size of Ascon, security degrades as the number of users μ increases. To mitigate this, we propose the LK-Ascon (Large Key Ascon) mode, an Ascon-like mode with a larger key size (up to 256 bits, instead of the standard 128 bits of Ascon). We show that LK-Ascon is secure even when the number of users reaches the data limit. Furthermore, we demonstrate that although privacy is not guaranteed, both Ascon and LK-Ascon achieve authenticity security in the nonce-misuse setting as well. The contents of this chapter have been published in [35].

As our final contribution, in Chapter 6, we propose PACT, a transform that converts any authenticated encryption into a committing one (we call this a CMT transform). PACT is the first CMT transform that is both universal (i.e., applicable to any AE scheme) and output-length-preserving (i.e., does not increase the ciphertext size). PACT uses one call to an unkeyed hash function and one call to a block cipher (which we model as an ideal cipher for our security results). To prove that PACT is committing, we propose a new notion for AE schemes, which we call "ciphertext collision advantage". We also propose comPACT, a lighter version of PACT which is also committing and preserves standard AEAD security as long as nonces are not misused. In the same chapter, we show that most authenticated schemes for which prior CMT transforms are not applicable are not committing, thus making universal transforms necessary. The contents of this chapter are available at the E-print server [27].

Finally, in Chapter 7, we conclude the thesis and discuss some potential future works.

Chapter 2

Preliminaries

2.1 Notation

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 $\{0,1\}^* = \{\lambda\} \cup \{0,1\}^+$. We call elements of $\{0,1\}^n$ blocks. A k-to-r (block) function or random oracle has domain $\{0,1\}^{kn}$ and range $\{0,1\}^{rn}$. 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 $(n)_k \leq n^k$. For any bit string $x = x_1x_2\cdots x_k \in \{0,1\}^k$, we write $|x|$ to denote its bit-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. For a set $\mathcal{X}$, $x \xleftarrow{\$} \mathcal{X}$ denotes that x is sampled from $\mathcal{X}$ uniformly at random, and $y \xleftarrow{\mathsf{wor}} \mathcal{X}$ denotes without replacement sampling of y from $\mathcal{X}$.

A partial function τ from D to R is a subset $\tau \subseteq D \times R$ such that for every $x \in D$, there are at most one y with $(x,y) \in \tau$. We define domain $\mathrm{dom}(\tau) := \{x : \exists y, (x,y) \in \tau\}$ and range $\mathrm{ran}(\tau) = \{y : \exists x, (x,y) \in \tau\}$ of a partial function τ. We use the shorthand notation $A \cup x$ and $A \backslash x$ to denote $A \cup \{x\}$ and $A \backslash \{x\}$ respectively. For any q-tuple x^q, we define $\mathrm{mc}(x^q) = \mathsf{max}_a |\{i : x_i = a\}|$. For two lists $\mathcal{L}_1$ and $\mathcal{L}_2$, we define $\mathrm{mc}(\mathcal{L}_1 \oplus \mathcal{L}_2) = \mathsf{max}_a |\{(i,i') : L_i \oplus L_{i'} = a, L_i \in \mathcal{L}_1, L_{i'} \in \mathcal{L}_2\}|$. It can be similarly extended for XOR of more than two lists.

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

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

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 denote $(X_1, \ldots, X_x) \xleftarrow{r}_* X$ when $X_1 \| \cdots \| X_x = X$, $|X_1| = \cdots = |X_{x-1}| = r$ 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 Distinguishing Advantage

For two oracles $\mathcal{O}_0$ and $\mathcal{O}_1$, an algorithm $\mathcal{A}$ which tries to distinguish between $\mathcal{O}_0$ and $\mathcal{O}_1$ is called a *distinguishing adversary*. $\mathcal{A}$ plays an interactive game with $\mathcal{O}_b$ where b is unknown to $\mathcal{A}$, and then outputs a guess for b; $\mathcal{A}$ wins when the guessed bit matches b. The *distinguishing advantage* of $\mathcal{A}$ is defined as

$$\mathbf{Adv}_{\mathcal{O}_1,\mathcal{O}_1}(\mathcal{A}) := \Big| \Pr_{\mathcal{O}_0}[\mathcal{A} \Rightarrow 1] - \Pr_{\mathcal{O}_0}[\mathcal{A} \Rightarrow 1] \Big|,$$

where the subscript of Pr denotes the oracle with which $\mathcal{A}$ is playing. $\mathcal{O}_0$ conventionally represents an ideal primitive, while $\mathcal{O}_1$ represents either an actual construction or a mode of operation built using some other ideal primitives. We use the standard terms real oracle and ideal oracle for $\mathcal{O}_1$ and $\mathcal{O}_0$, respectively. Typically, the goal of the function F represented by $\mathcal{O}_1$ is to emulate the ideal primitive F^* represented by $\mathcal{O}_0$. A security game is a distinguishing game with an optional set of additional restrictions chosen to reflect the desired security goal. When we talk of distinguishing advantage between F and F^* with a specific security game $\mathcal{G}$ in mind, we include $\mathcal{G}$ in the superscript, e.g., $\mathbf{Adv}^{\mathcal{G}}_{F,F^*}(\mathcal{A})$. We note that this notation is general enough to capture games where each oracle implements multiple functions, e.g., F can handle both encryption and decryption queries by accepting an extra bit to indicate the direction of queries. Also, we sometimes drop the resource or the ideal primitive or both, and simply write, e.g.,

$\mathbf{Adv}_F^{\mathcal{G}}(\mathcal{A})$, when the omitted variables are clear from the context. We interchangeably use the distinguishing advantage notation with games instead of oracles when the adversary is trying to distinguish between two games.

2.3 Cryptographic Hash Function

A *cryptographic hash function* $\mathsf{H} : \{0,1\}^* \to \{0,1\}^n$ is a function that maps bit strings of arbitrary length to fixed-length digests of size n. Sometimes, a hash function can be keyed, meaning it can take a key as an additional input. Cryptographic hash functions have many applications, notably in digital signatures, message authentication codes (MACs), and other forms of authentication. Although this thesis primarily focuses on unkeyed hash functions, we will occasionally encounter keyed hash functions when analysing certain schemes.

A *compression function* $f : \{0,1\}^m \to \{0,1\}^n$ for $m > n$ is a hash function limited to inputs of a fixed length m. If $m = n$, we call the function *non-compressing*. Compression functions serve as the building blocks for constructing hash functions. However, the primitives used to build hash functions are not always compressing. For instance, while the primitives of the Merkle-Damgård Hash [49, 98] and the Merkle tree [97] are compressing, the primitive used in the latest hash standard, SHA-3 [62], is non-compressing. Throughout this thesis, a t-to-1 block compression function will refer to a tn-to-n bit compression function. When we use a smaller compression function f to build a larger compression function H, we will refer to the larger compression function as a hash function and denote it by H^f to indicate that H makes calls to f.

2.3.1 Generic Hash Mode

Let H^f be a t-to-1 block hash function which uses an n-bit compression function (i.e. λ-to-1 compression function f for some $\lambda > 1$) as an oracle. Note that a mode can use more than one compression functions $f_1, \ldots, f_r$. However, as we analyse in the random oracle model, independent random oracles can be obtained from a single random oracle with a larger domain by using the standard domain separation method. The hash function calls f_i on i-th call and so the domains of every call are separated by domain separation. We also denote the family $f := (f_i : i \in [r])$ by f and we call λ-to-1 r r.o. (random oracle). We denote $\tau_{\mathsf{H}}(M \mid f) := \{((1, x_1), y_1), \ldots, ((r, x_r), y_r)\}$ where x_i denotes the input of i-th call of its oracle tuple while computing $\mathsf{H}^f(M)$ and $y_i = f_i(x_i) := f(i, x_i)$. A λ-to-1

transcript τ is a partial function from $[r] \times \{0,1\}^{\lambda n}$ to $\{0,1\}^n$. For a λ-to-1 r r.o. f, we have

$$\forall (i,x) \notin \mathrm{dom}(\tau), y \in \{0,1\}^n, \;\; \Pr(f(i,x) = y \mid \tau \subseteq f) = 2^{-n}.$$

Definition 2.1 (transcript-based hash computation)**.** Given a partial function $\tau \subseteq f$, let $\mathsf{H}^\tau = \{(M, \mathsf{H}^f(M)) : \; \tau_{\mathsf{H}}(M \mid f) \subseteq \tau\}$ be a partial hash function. In other words, H^τ consists of all pairs (M,z) such that $\mathsf{H}^f(M)$ can be computed by simply using the transcript τ and z is the final value. The elements of the set $\mathrm{dom}(\mathsf{H}^\tau)$ are called τ-computable inputs. As $\tau \subseteq f$, we must have $\mathsf{H}^\tau \subseteq \mathsf{H}^f$.

2.3.2 Collision Game

Let $\mathcal{A}$ be an adversary having oracle access of f which makes q queries to each f_i adaptively. As we assume an unbounded time adversary, there is no loss in assuming that $\mathcal{A}$ is deterministic. Thus, the i-th query (x_i, v_i) of $\mathcal{A}$ depends on τ^{i-1} (the transcript of query-responses after $(i-1)$ queries). After the query-response phase, $\mathcal{A}$ returns a pair of distinct inputs (M, M') such that both M, M' are transcript-computable. We say $\mathsf{coll}_{\mathsf{H}}$ holds if $\mathsf{H}^\tau(M) = \mathsf{H}^\tau(M')$, called a *computable collision pair*. We define $\mathbf{Adv}^{\mathsf{coll}}_{\mathsf{H}^f}(\mathcal{A}) := \Pr(\mathsf{coll}_{\mathsf{H}})$.

Definition 2.2 (cross collision)**.** Let H and H' be two hash functions. A *cross-collision τ-computable pair* is a pair (M, M') (not necessarily distinct) such that $\mathsf{H}^\tau(M) = \mathsf{H}'^\tau(M')$. We denote $\mathsf{coll}^\tau_{\mathsf{H},\mathsf{H}'} := \{M \in \mathrm{dom}(\mathsf{H}^\tau) : \; \exists M', \; \mathsf{H}^\tau(M) = \mathsf{H}'^\tau(M')\}$.

2.3.3 Local Opening

We now define the local opening security of a hash function output (viewed as a commitment of an input). Given a hash function mode H^f, a local opening Open^f for H maps a pair (M,i) to $\pi = (m_i, i, \pi')$ (called *proof*) where $M = (m_1, m_2, \ldots, m_c)$ is an input (a tuple of blocks) and $i \in [c]$ is an index.

Correctness of Local Opening. There is an efficient function Ver^f such that for all input M, all index i, $\mathsf{Ver}^f(\mathsf{Open}^f(M,i), \mathsf{H}^f(M)) = 1$.

Security of Local Opening. In the local opening security game, the adversary wins if it produces an output h corresponding to two contradicting local openings for some position i.

Definition 2.3 (local opening advantage)**.** Let H be a hash function and Open be a correct local opening for H with verification function Ver. For any adversary $\mathcal{A}$, we define the local opening advantage as

$$\mathbf{Adv}_{H}^{local}(\mathcal{A}) = \Pr\Big[\mathsf{Ver}(i,m,\pi,h) = \mathsf{Ver}(i,m',\pi',h) = 1, m \neq m' \mid (i,m,m',\pi,\pi',h) \leftarrow \mathcal{A}^{f}\Big]$$

By-Pass Hash Computation. We say that H has a *by-pass computation* $(\mathsf{H}_i : i \in [c])$ corresponding to a local opening Open if for all M, $i \in [c]$,

$$\mathsf{H}_i^f(\mathsf{Open}^f(M,i)) = \mathsf{H}^f(M).$$

In other words, given a proof (output of the Open) and the input block for the index (for which the proof is produced), we can compute the hash output of the input (without knowing the other blocks of the input). The verification algorithm simply checks whether the hash value computed through the by-pass hash is the same as what was committed before. As f is treated as an oracle, it is natural to assume that for all M and for all i,

$$\tau_{\mathsf{Open}}(M,i \mid f) \cup \tau_{\mathsf{H}_i}(\mathsf{Open}^f(M,i) \mid f) = \tau_{\mathsf{H}}(M \mid f).$$

We now define the *inter-collision* advantage for by-pass computation H_i as

$$\mathbf{Adv}_{\mathsf{H}_i}^{\mathsf{coll}*}(\mathcal{A}) = \Pr\Big[\mathsf{H}_i(m,\pi) = \mathsf{H}_i(m',\pi') \text{ and } m \neq m' \mid (m,\pi,m',\pi) \leftarrow \mathcal{A}^{f}\Big].$$

Thus, it is the same as the collision game, except that the adversary needs to find a collision pair for which $m \neq m'$. Suppose $\mathcal{A}$ finds a collision pair $((m,\pi),(m',\pi'))$ for H_i, and let $h = \mathsf{H}_i(m,\pi)$. Then $\mathcal{A}$ can commit h and later on, it can successfully open for either of two inputs m and m' as required. Now we make the following simple observation

$$\mathbf{Adv}_{\mathsf{H}}^{local}(q') = \max_{\mathcal{A}} \max_{i} \mathbf{Adv}_{\mathsf{H}_i}^{\mathsf{coll}*}(\mathcal{A}).$$

The above observation (see [58] for details) helps us to reduce the local opening security to inter-collision security problem for the by-pass hash family.

2.4 Authenticated Encryption

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

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

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

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

For a key $K \in \mathcal{K}$, we use $\mathsf{Enc}_K(\cdot)$ and $\mathsf{Dec}_K(\cdot)$ to denote $\mathsf{Enc}(K, \cdot)$ and $\mathsf{Dec}(K, \cdot)$, respectively. An NAEAD scheme is called *tidy* if

$$\mathsf{Enc}(K, N, A, \mathsf{Dec}(K, N, A, C, T)) = (C, T) \text{ for any choice of } (K, N, A, C, T).$$

All the schemes we consider in this thesis will be tidy.
Deterministic authenticated encryption with associated data, abbreviated as DAE (or DAEAD), is similar to NAEAD except that it does not take any nonce as input. In this thesis, we use the generic term AEAD (or just AE) to denote an authenticated encryption which can be either nonce-based or deterministic.

Remark 2.4. For some AEs, the tag space $\mathcal{T}$ is not explicitly defined. However, in such cases, $|C| > |M|$, i.e. the size of the ciphertext is larger than the size of the message. If $|C| - |M| = t$, then throughout this thesis, without loss of generality, we assume the last t bits of the ciphertext C to be the tag unless otherwise specified.

Remark 2.5. We write $(K, N, A, M, C, T) \in \Pi$ (or $(K, N, A, M, C, T) \notin \Pi$) to denote that a party interacting with an AE Π has already verified that the tuple (K, N, A, M, C, T) is compatible (or incompatible, respectively) with Π. The notation $(C, T) = \Pi.\mathsf{Enc}(K, N, A, M)$ or $M = \Pi.\mathsf{Dec}(K, N, A, C, T)$ is different, either of which only states the correctness condition for the tuple (K, N, A, M, C, T) with respect to Π, but does not guarantee any verification of the correctness by the interacting party.

2.4.1 AEAD Security Definitions

We consider two types of adversaries of an nAE scheme. A *confidentiality adversary* (also called a *privacy adversary*) $\mathcal{A}$ of an nAE scheme $\Pi := (\Pi.\mathsf{Enc}, \Pi.\mathsf{Dec})$ is a distinguishing adversary which tries to distinguish between $\Pi.\mathsf{Enc}$ and an oracle Π^* which works as follows. For an encryption query (N, A, M), it samples an element from $\{0,1\}^{|M|+t}$ uniformly at random, and outputs it. The advantage of $\mathcal{A}$ is called *confidentiality advantage*, denoted by $\mathcal{A}^{\mathsf{conf}}_{\Pi}(\mathcal{A})$.
Sometimes $\mathcal{A}$ is assumed to be *nonce-respecting*, i.e., it is not supposed to repeat a nonce. Depending on whether we make this assumption or not, we use the terms UNAE (Unique Nonce AE) or MRAE (Misuse Resistant AE) in place of AE. In the case of an MRAE scheme, Π^* can be fine-tuned to $\Pi^\dagger$ for the confidentiality game in the following. For every encryption query $(N, A, \cdot)$, it samples an element from $\{0,1\}^{|M|+t}$ uniformly at random without replacement, and outputs it. But if we use Π^* instead of $\Pi^\dagger$, $\mathcal{A}$ can distinguish it from $\Pi.\mathsf{Enc}$ only after it can observe a collision at the output of Π^* for the same (N, A) tuple, which will happen after about $2^{|M|+t}$ encryption queries. Since we do not deal with beyond-birthday-bound security in this thesis, we use Π^* for MRAE as well.
An *authenticity adversary* $\mathcal{B}$ of Π is an adversary which has access to both $\Pi.\mathsf{Enc}$ and $\Pi.\mathsf{Dec}$, and tries to "forge", i.e. make a successful query to $\Pi.\mathsf{Dec}$ and this query is not the result of a previous encryption query, and its advantage is called *authenticity advantage*, denoted by $\mathcal{A}^{\mathsf{auth}}_{\Pi}(\mathcal{B})$. We always impose a restriction on each adversary that it cannot make *pointless* queries, i.e., cannot repeat the same query multiple times or cannot make the query (N, A, C, T) to $\Pi.\mathsf{Dec}$ if it has already made a query (N, A, M) to $\Pi.\mathsf{Enc}$ and received (C, T) in response.
The AEAD security game is a combination of the confidentiality and the authenticity games. Since we use the AEAD security game only in the RPM, we discuss this combination in the next section.

2.4.2 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, parameterised by the number of users μ. For the single-user setting, $\mu = 1$. Let:

- $\pi \xleftarrow{\$} \mathsf{Perm}(b)$ (we use the superscript $\pm$ to denote bidirectional access to π),
- $f_1, \ldots, f_\mu \xleftarrow{\$} \mathsf{Func}(\mathcal{N} \times \mathcal{A} \times \mathcal{M}, \mathcal{M} \times \mathcal{T})$,

- 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 2.6. 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 $\mathcal{A}$ against AE_π is defined as

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

Here $\mathcal{A}^{\mathsf{Enc}_{K_i}, \mathsf{Dec}_{K_i}, \pi^{\pm}}$ denotes $\mathcal{A}$'s response after its interaction with Enc_{K_i}, Dec_{K_i}, and $\pi^{\pm}$ (i.e., both forward and backward queries to π) respectively. Similarly, $\mathcal{A}^{f_i, \mathsf{rej}, \pi^{\pm}}$ denotes $\mathcal{A}$'s response after its interaction with f_i, rej, and $\pi^{\pm}$ respectively. In this thesis, 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 Enc_{K_i}, all Dec_{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 2.7. 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 2.8. 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 $\mathcal{A}$ against AE_π is defined as

$$\mathbf{Adv}^{\mathsf{mu-conf}}_{\mathsf{AE}_\pi}(\mathcal{A}) := \left| \Pr_{\substack{(K_i)_{i=1}^{\mu} \xleftarrow{\$} \mathcal{K} \\ \pi^{\pm}}} [\mathcal{A}^{(\mathsf{Enc}_{K_i})_{i=1}^{\mu}, \pi^{\pm}} = 1] - \Pr_{\substack{(f_i)_{i=1}^{\mu} \\ \pi^{\pm}}} [\mathcal{A}^{(f_i)_{i=1}^{\mu}, \pi^{\pm}} = 1] \right|,$$

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

$$\mathbf{Adv}^{\mathsf{mu-auth}}_{\mathsf{AE}_\pi}(\mathcal{A}) := \Pr_{\substack{(K_i)_{i=1}^{\mu} \xleftarrow{\$} \mathcal{K} \\ \pi^{\pm}}} [\mathcal{A}^{(\mathsf{Enc}_{K_i},\mathsf{Dec}_{K_i})_{i=1}^{\mu},\pi^{\pm}} \text{ forges }].$$

In the context of authenticity, we use the term "$\mathcal{A}$ forges" to describe a situation where $\mathcal{A}$ 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 [59], it can be shown that

$$\mathbf{Adv}^{\mathsf{mu-auth}}_{\mathsf{AE}_\pi}(\mathcal{A}) \leq \left| \Pr_{\substack{(K_i)_{i=1}^{\mu} \xleftarrow{\$} \mathcal{K} \\ \pi^{\pm}}} [\mathcal{A}^{(\mathsf{Enc}_{K_i},\mathsf{Dec}_{K_i})_{i=1}^{\mu},\pi^{\pm}} = 1] - \Pr_{\substack{(f_i)_{i=1}^{\mu} \\ \pi^{\pm}}} [\mathcal{A}^{(f_i)_{i=1}^{\mu},\mathsf{rej},\pi^{\pm}} = 1] \right|.$$

Proposition 2.9. *[13] There exist adversaries $\mathcal{A}$, $\mathcal{A}'$ and $\mathcal{A}''$ having same query complexities such that*

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

2.4.3 Committing Security

The two prevalent notions of committing security in the literature are: committing solely to the key K, which we call CMT_k security, and committing to the complete context, denoted as (K, N, A), which we call CMT security. Bellare and Hoang [9] demonstrated that incorporating the message M into the context is unnecessary, as committing to (K, N, A) alone is equivalent to committing to (K, N, A, M).

CMT_k **game for AE.** In the CMT_k game against an AE scheme Π, an adversary $\mathcal{A}$ outputs (K, N, A, M) and (K', N', A', M'); $\mathcal{A}$ wins if:

- $K \neq K'$;
- $\Pi.\mathsf{Enc}(K, N, A, M) = \Pi.\mathsf{Enc}(K', N', A', M')$.

We write $\mathcal{A}^{\mathrm{CMT}_k}_{\Pi}(\mathcal{A})$ to denote the probability that $\mathcal{A}$ wins.

CMT **game for AE.** In the CMT game against an AE scheme Π, an adversary $\mathcal{A}$ outputs (K, N, A, M) and (K', N', A', M'); $\mathcal{A}$ wins if:

- $(K, N, A) \neq (K', N', A')$;
- $\Pi.\mathsf{Enc}(K, N, A, M) = \Pi.\mathsf{Enc}(K', N', A', M')$.

We write $\mathcal{A}_{\Pi}^{\mathrm{CMT}}(\mathcal{A})$ to denote the probability that $\mathcal{A}$ wins.

2.5 Block Cipher

A block cipher is a function $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ (where n is a positive integer) such that for any $K \in \mathcal{K}$, $E_K(\cdot) := E(K, \cdot)$ is a permutation on $\{0,1\}^n$. K is called the key of the block cipher. In other words, a block cipher is a set of permutations indexed by a key. The typical expectation from a block cipher is that for a randomly sampled secret key K, E_K should behave like a random permutation.

Ideal Cipher. Let $\mathcal{BC}$ be the set of all block ciphers from $\mathcal{K} \times \{0,1\}^n$ to $\{0,1\}^n$. An ideal cipher from $\mathcal{K} \times \{0,1\}^n$ to $\{0,1\}^n$ is an element from $\mathcal{BC}$ chosen uniformly at random. Ideal ciphers are not practical primitives, but are often used as a theoretical representation of a perfect block cipher while studying block cipher-based constructions where the block cipher key is dynamically generated.

2.6 H-coefficient Technique

Consider a deterministic and computationally unbounded adversary $\mathcal{A}$ trying to distinguish between a real oracle (say $\mathcal{O}_{\mathrm{re}}$) and an ideal oracle (say $\mathcal{O}_{\mathrm{id}}$). Let the transcript τ denote the query-response tuple of $\mathcal{A}$'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 $\mathcal{A}$ interacts with $\mathcal{O}_{\mathrm{re}}$ (resp. $\mathcal{O}_{\mathrm{id}}$). The probability of realising a given transcript τ in the security game with an oracle $\mathcal{O}$ is known as the *interpolation probability* of τ with respect to $\mathcal{O}$. Since $\mathcal{A}$ is deterministic, this probability depends only on the oracle $\mathcal{O}$ and the transcript τ. A transcript ω is said to be *realisable* if $\Pr(\Theta_{\mathrm{id}} = \tau) > 0$.

Theorem 2.10 (H-coefficient technique [104, 105])**.** *Let Ω be the set of all realisable 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 $\mathcal{A}$*, we have the following bound on its distinguishing advantage:*
$$\mathbf{Adv}^{\mathcal{O}_{\mathrm{re}},\mathcal{O}_{\mathrm{id}}}(\mathcal{A}) \leq \epsilon_{\mathsf{bad}} + \epsilon_{\mathsf{ratio}}.$$

A proof of theorem 2.10 can be found in multiple papers including [42, 95, 105].

2.7 Expected Multicollision in a Uniform Random Sample

Let $S := (x_i)_{i \in I}$ be a tuple with 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 [37].
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 2.11. *[37] Let* $\mathcal{D}$ *be a set of size* $N \geq 4$, $n = \log_2 N$. *Given random variables* $X_1, \ldots, X_q \xleftarrow{\$} \mathcal{D}$, *we have* $\mathbb{E}[\mathsf{mcoll}(X_1, \ldots, X_q)] \leq \mathsf{mcoll}(q, N)$.

Remark 2.12. Similar bounds as in the above Lemma 2.11 can be achieved in the case of non-uniform samplings. Let $Y_1, \ldots, Y_q \xleftarrow{\mathsf{wor}} \{0,1\}^b$ 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 $X_1, \ldots, X_q$.

We also have the following general result:

Lemma 2.13 (general multicollision bound)**.** *Let* $\mathcal{A}$ *be an adversary which makes queries to a* b*-bit random permutation* $\pi^{\pm}$ *and* τ*-bit to* τ*-bit random function* f. *Let* $(X_1, Y_1), \ldots, (X_{q_1}, Y_{q_1})$ *and* $(X_{q_1+1}, Y_{q_1+1}), \ldots, (X_{q_1+q_2}, Y_{q_1+q_2})$ *be the tuples of input-output corresponding to* π *and* f *respectively obtained by the* $\mathcal{A}$. *Let* $q := q_1 + q_2 \leq 2^b$ *and* $Z_i := trunc_\tau(X_i) \oplus trunc_\tau(Y_i)$ *for* $i \in [q_1]$ *and* $Z_i := (X_i \oplus Y_i)$ *for*

$i \in [q_1 + 1, q]$ *where* $trunc_\tau$ *represents some* τ*-bit truncation. For* $\tau \geq 2$,

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

Chapter 3

Revisiting Collision and Local Opening Analysis of ABR Hash

The question of constructing the most efficient tn-to-n-bit collision-resistant hash function H from a smaller (say, $2n$-to-n-bit) compression function f is one of the fundamental questions in symmetric key cryptography. This question has a rich history and was open for general t, until a recent breakthrough result by Andreeva, Bhattacharyya and Roy at Eurocrypt'21, who designed an elegant mode (which we call ABR) achieving roughly $2t/3$ calls to f, which matches the famous Stam's bound from CRYPTO'08. Unfortunately, we have found serious issues in the claims made by the authors. These issues appear quite significant, and range from verifiably false statements to noticeable gaps in the proofs (e.g., omissions of important cases and unjustified bounds). We were unable to patch up the current proof provided by the authors. Instead, we prove from scratch the security of the ABR construction for the first non-trivial case $t = 11$ (ABR mode of height 3), which was incorrectly handled by the authors. In particular, our result matches Stam's bound for $t = 11$. While the general case is still open, we hope our techniques will prove useful in finally settling the question of the optimal efficiency of hash functions.

3.1 Introduction

The *Merkle-Damgård construction* [49, 98] is a sequential construction used in MD5, SHA-1, SHA-2 and many other hash functions. On the other hand, the *Merkle tree* [97] is a parallel construction that is used in hash-based signatures (valued due to their post-quantum security), version control systems such as Git,

and cryptocurrencies like Ethereum. It is well known that the Merkle-Damgård construction and the Merkle tree are collision-resistant provided so are the compression functions. The number of compression function calls is (essentially) the same for both constructions. When we use $2n$-to-n-bit compression functions, we can process t blocks of messages by making t or $(t-1)$ calls to the compression function.

Although both of these widely used constructions are quite efficient and rely solely on the collision resistance of the compression function, practical compression functions are believed to possess more properties than mere collision resistance. Consequently, it is of interest to investigate the most efficient way to build a t-to-1 collision-resistant hash function, even when modelling the compression function as ideal (i.e., a random oracle). In particular, it is worth examining whether the classical Merkle-Damgård and Merkle tree constructions can be improved under such idealised modelling. This question has garnered significant attention from the cryptography community, which we survey below.

Lower bound on the number of calls. We start with lower bounds (i.e., attacks). In [29], Black et al. formally analyse the security-efficiency trade-off of compression functions, demonstrating that a $2n$-to-n-bit compression function making a single call to a fixed-key n-bit block cipher cannot achieve collision resistance. Later Rogaway and Steinberger [112] generalise the result for permutation-based hash. For a general hash function based on a compression function, Stam [115] conjectures a lower bound on the number of compression function calls. Specifically, a collision with at most $2^{n(\lambda-(t-0.5)/r)}$ queries on a t-to-1 block hash function can be found after making r calls to λ-to-1 block compression functions. Equivalently, for optimal birthday security, the number of hash calls must be at least $r \geq (2t-1)/(2\lambda-1)$. This bound is popularly known as the Stam's bound. Stam has demonstrated the bound for some cases under a uniformity assumption. Later by Steinberger [116] and by Steinberger, Sun and Yang [117], a formal proof of the Stam's bound is shown.

Hence, for the most widely studied case of $\lambda = 2$, we have a lower bound $r \geq (2t-1)/3$, leaving an efficiency gap of a factor of 1.5 when compared to the Merkle-Damgård and Merkle tree constructions.

Upper bound on the number of calls. For the upper bounds, much of earlier work concentrates on the setting of the "non-compressing" case of $\lambda = 1$, and often focuses on the case of small t (e.g., $t = 2$), implicitly suggesting that — once the

2-to-1 function is built, — one should do further extensions with either Merkle-Damgård or Merkle trees. For instance, Shrimpton and Stam [114] propose a 2-to-1 compression function based on three calls of a non-compressing function, which matches Stam's bound for $\lambda = 1$ and $t = 2$. Rogaway and Steinberger [111] design comparable results when the non-compressing primitive is an invertible permutation, which they also show is optimal for this setting [112].

For general (large) t, Mennink and Preneel [96] consider the non-compressing case $\lambda = 1$ and propose an elegant tree-based mode of operation making $(2t - 1)$ calls to the non-compressing round function, which matches Stam's bound. Unfortunately, they are only able to prove below-birthday security of $2^{n/3}$ queries for this construction. They also conjecture that the construction achieves optimal birthday security $2^{n/2}$, but their proof is limited to a very restricted special-case attacker scenario. These attacks make all their random oracle calls "layer-by-layer" (as opposed to in any order). The authors acknowledge that this simplifying assumption significantly aids in proving the security of this specific case but comes at the cost of generality. They provided evidence suggesting that their current analysis is unlikely to suffice for proving optimal security against unrestricted attackers.

Recently, two papers have appeared to tackle the compressing case $\lambda = 2$. In [58], Dodis et al. optimally settle the case $t = 5$, by introducing the T5 construction that processes five n-bit message blocks using three $2n$-to-n-bit compression function calls, which matches Stam's bound for $t = 5$ and $\lambda = 2$. Further, they suggest extending the T5 construction to a larger value of t using either Merkle-Damgård or Merkle trees. In both scenarios, they achieve significant efficiency gains compared to earlier methods, which required t compression calls: both variants now make approximately $3t/4$ calls to the compression functions. Still, once $t > 5$, this does not match the current lower bound of $2t/3$ calls. [58] also mention a natural, but more aggressive, variant of this extended construction for the case of Merkle trees. However, they remark that this construction — even if proven collision-resistant (which is open), — would lose the efficient "local opening" properties of their simpler tree construction with $3t/4$ compression calls. Namely, one can no longer open one message block by only opening $O(\log t)$ internal values in the tree (as any such opening cannot have birthday security, despite satisfying correctness).

Finally, a significant breakthrough at Eurocrypt'21 by Andreeva, Bhattacharyya, and Roy [4] purportedly resolves the general case. They propose a hash function ABR_l based on a perfect binary tree of height l. The hash ABR_l can process

$t = (2^l + 2^{l-1} - 1)$ blocks with $r = (2^l - 1)$ calls of compression functions. This matches Stam's bound $r \geq (2t-1)/3$. Interestingly, the ABR construction closely resembles the tree construction of Mennink and Preneel [96] using non-compressing primitives, with the distinction that all compression calls at the leaf level now incorporate an extra input (due to $\lambda = 2$ instead of $\lambda = 1$), while internal calls to the compression function can also handle an additional input using a slightly more complex rule involving two simple XOR operations. Thus, intuitively, the authors appear to have addressed the challenges encountered in [96] regarding general adversaries, with a construction very similar to that of [96].
Additionally, the work of [4] claims that the ABR_l mode also has attractive local opening properties, albeit with a slightly longer proof length ($2l$ instead of l of Merkle trees), while still requiring only l compression calls to verify such local opening.

Are we done? Unfortunately, we have found serious issues in several claims made by the authors of [4], whom we call ABR hereafter. These issues appear quite significant, and range from verifiably false statements to noticeable gaps in the proof (e.g., omissions of important cases and unjustified bounds). Regrettably, we currently lack a straightforward solution to address these issues.

3.1.1 Contributions

Our results can be roughly divided into three categories:

(1) explicit refutation of several claims made in [4];

(2) serious technical issues in the proof presented in [4];

(3) a correct (but very different from [4]) proof for the for the ABR_3 construction (i.e. $t = 11$ and $r = 7$), which ABR mishandled.

We detail these below.

Local opening insecurity of ABR. As already mentioned, ABR propose a very efficient local opening for ABR_l. It opens about $2l$ blocks and makes l calls to verify. However, we show that a collision pair of the verification function can be found in $O(2^{n/2l})$ queries, which is significantly below birthday security already for $l = 2$. Hence, the suggested local opening can be broken in the above complexity. Moreover, we have shown that *any* non-trivial local opening of ABR_l satisfying a

"by-pass verification" property (which is a natural class of openings that seems to include any natural opening one can think of) is broken below the birthday bound. For example, even opening $(t-1)$ out of t inputs cannot be birthday-secure, where $t = 2^l + 2^{l-1} - 1 = 2^{\Omega(l)}$. In contrast, previous tree-like constructions (e.g., [58]) achieve birthday security with logarithmic opening length $O(l)$. This is discussed in Section 3.3.

There are two surprising aspects to this mistake. First, our attack is completely standard (using standard generalised birthday attack [119]). Second, the local opening subsection in the ABR paper does not even mention anything about security, focusing solely on the correctness of the opening. This is quite unexpected.

Mistakes with the main proof. While the local opening mistake above is indisputable, the technical mistakes in the main collision resistance proof of ABR are harder to explain in detail (at least in the introduction, before the technical notation is developed). They are also harder to state with conviction since they often do a combination of the following pitfalls:

(a) involve imprecise statements,

(b) state a bound which might be true, but appears completely non-obvious to us (to the extent of being the most difficult part of the proof);

(c) point to an "analogous" earlier case, but we fail to see why the previous argument generalises;

(d) state some bound which appears to be correct only if one makes some restricting assumption on the attacker (but no such assumptions are made by the authors, who claim a fully general result!);

(e) silently omitting an important special case of the proof (i.e., the proof is non-exhaustive).

The totality of these issues makes the proof presented by [4] at best unverifiable, and at worst incorrect. In particular, we still believe that the end result is correct, but fixing it would require a substantially harder proof.

At a very high level, the correct collision analysis for a tree-based function like ABR_l is complex mostly due to the *adaptive nature* of queries, and the queries made to different layers in the tree might not come in monotone order (i.e., may not be in order of the level of the nodes). Indeed, this is precisely the reason why the

earlier birthday security result of Mennink and Preneel [96] only held for "in order" adversaries. Fortunately, the outputs of the leaf nodes can be given beforehand, as the input of those has no role in finding a collision. More formally, we can make a simple argument to force the attacker to "evaluate" the first layer compression calls before any of the subsequent calls as follows. We give the attacker q random outputs (where q is the total number of queries made by the attacker) at the very beginning, but allow the adversary to *arbitrarily label* the corresponding input values at any point in the game. This is fine, since those input values do not participate in any other computation, but now all the outputs in the first layer are known before a single compression call is made to the lower layers. This allows for relatively simple analysis for the special case $l = 2$, and the authors of [4] indeed start with the correct analysis of this special case.[1]

Unfortunately, this argument completely fails after the first layer. (Indeed, handling this case will be one of the most difficult parts of our analysis, when we provide a correct proof for $l = 3$ in this paper.) In particular, we see the following high-level issues with the proof presented in [4] for $l \geq 3$. (More lower-level issues are discussed in Section 3.4.3 in the paper.)

1. ABR claimed a relation between collision and the number of computable hash outputs (termed as load). We will show in Section 3.4.4 that the relation is not true in general by giving a counterexample. This seems to hold for ABR if queries to the root node are performed at the end (which is the case for ABR_2). However, it seems non-obvious to us why a similar relation holds when the adversary makes out-of-order queries.

2. We have also found issues while bounding load. ABR consider "input multi-collision" for every node up to $O(n)$. However, due to the multiplicative nature of the number of multi-collisions as one goes down in the tree, we find that $O(n^i)$ multi-collision must be considered for the nodes at the i-th level. This would degrade the bound for load claimed by ABR, and invalidate the claimed birthday security at the end (unless the number of levels i is constant, in which case one can hide the extra n^i bound in the "O-tilde"-notation). This will be discussed in Step 1 of Section 3.4.3.

[1] Another correct proof for $t = 5$ (corresponding to tree depth $l = 2$) was made for the T5 compression function by [58]. Interestingly, the authors did not notice the simplifying non-adaptivity argument above and had to work relatively hard to handle out-of-order queries (e.g., it involved a careful expectation analysis and applying Markov's inequality; see proof of Proposition 5 in [58], which is over a page). This shows that handling out-of-order attackers is indeed highly non-trivial.

3. Even if the load analysis is somehow fixed, ABR seem to consider the last query happens in the final node (or at the node where the load is considered). This is effectively equivalent to in-order adversaries, but does not seem to be the case for general attackers. See Step 2 of Section 3.4.3.

4. Moreover, both messages of a collision pair can be generated due to a single query response (termed as *twin collision pair*). ABR completely ignore this case. This is discussed in detail in the last paragraph of Section 3.4.3.

We leave a more detailed explanation of these (and other issues (a)-(e)) later in the paper.

Collision analysis of ABR_3. On a positive, our main technical result shows that the ABR_3 construction for $t = 11$ indeed achieves birthday security (roughly $n^5q^2/2^n$, where q is the number of compression function queries) with an optimally small number of $r = 7$ compression calls (see Section 3.5). While forming only the first step in recouping the incorrect results of [4], we are optimistic that our approach could be extended to finally settle the general case correctly. For example, compared to best-known correct proofs for $t = 5$ (e.g., ABR_2 from [4], or the T5 compression function from [58]), we can no longer assume that the second layer calls to the compression function are made before all the third-layer calls, which is the main (unresolved) difficulty in the work of [96], and one of the key mistakes in the analysis of [4] (as we explained above). Thus, our proof is the first that handles non-trivial "out-of-order" adversaries correctly.
We also hope our proof of ABR_3 provides a sharp contrast to the flawed proof of [4], even for this special case. For example, we already mentioned handling general "out-of-order" adversaries. In a different vein, we also consider the twin-collision analysis for ABR_3 which is completely missing from [4]. This analysis requires a non-trivial multi-collision analysis on a sum of our compression functions, and we also need to bound some other failure events to analyze the non-twin collision security of ABR_3. None of these arguments appeared in [4].

3.2 Re-introduction of ABR Hash

We first start by defining a generalised tree hash structure, and then re-introduce the ABR Hash as a special tree hash, as opposed to introducing it as it is in [4]. This is because we feel some things have not been properly defined by the authors there, and these issues need to be addressed properly.

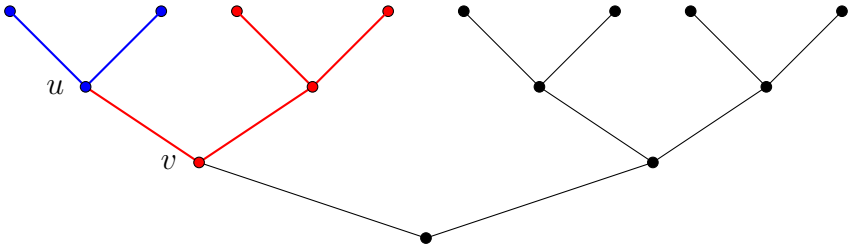

FIGURE 3.1: In this figure, $\mathcal{F}_v$ is the sub-tree rooted at v, i.e. the union of the red and blue sub-trees, $\mathcal{F}_{-v}$ is the black sub-tree, and $\mathcal{F}_{v-u}$ is the red sub-tree.

A *full binary tree* (FBT) is a binary tree in which every node v other than the leaf nodes has two children, denoted as v_{L} (left child) and v_{R} (right child). A *perfect binary tree* (PBT) is a full binary tree in which all the leaf nodes are at the same level (called height of the tree).

Example 3.1 (perfect binary tree of height l)**.** *Let l be a fixed positive integer and $\mathcal{T}$ be a perfect binary tree of height l over all vertices (j, b), $j \in [l], b \in [2^{l-j}]$ with $(l, 1)$ being the root. For every two vertices (j, b) and $(j+1, \lceil b/2 \rceil)$, we associate an edge. We call $(j-1, 2b-1)$ and $(j-1, 2b)$ the left and right child of (j, b) respectively. Note that $\mathcal{T} = \mathcal{T}_{(l,1)}$.*

Some Notations and Definitions on Binary Trees

For a binary tree $\mathcal{F}$, let $\mathscr{L}_{\mathcal{F}}$ and $V(\mathcal{F})$ denote the set of leaf nodes and all nodes of $\mathcal{F}$ respectively. Any non-leaf node is called an intermediate node. For a non-root intermediate node v of $\mathcal{F}$, we consider the following two full binary trees:

1. $\mathcal{F}_v$: the full binary sub-tree rooted at v.
2. $\mathcal{F}_{-v}$: the sub-tree $(\mathcal{F} \setminus \mathcal{F}_v) \cup v$.

For a tree $\mathcal{F}$, and a vertex v of $\mathcal{F}$, we write V_v, $\mathscr{L}_v$ and V_v^* to denote the set of all nodes, leaf nodes and intermediate (non-leaf) nodes respectively for the tree $\mathcal{F}_v$. For any $u \in V_v^* \setminus v$, we write $\mathcal{F}_{v-u} = (\mathcal{F}_v \setminus \mathcal{F}_u) \cup u$. We write V_{v-u} to denote the set of vertices of $\mathcal{F}_{v-u}$. For the sake of notational simplicity, we ignore the suffix v when v is the root. In this section, we only consider trees of the form $\mathcal{F}_v$ and $\mathcal{F}_{v-u}$. Refer to Figure 3.1 for a pictorial representation.
To each node $v \in V$ of a perfect binary tree $\mathcal{T}$, an independent 2-to-1 block compression function (modelled as a random oracle) f_v is assigned. We use the notation f to denote the collection of random oracles $\{f_v : v \in \mathcal{T}\}$.

Definition 3.1 (message for tree hash)**.** A message m for any full binary sub-tree $\mathcal{F}$ of a perfect binary tree $\mathcal{T}$ having the same root is a function $m : V(\mathcal{F}) \to \{0,1\}^n \cup \{0,1\}^{2n}$ such that for all $u \in \mathscr{L}_{\mathcal{F}} \cap \mathscr{L}_{\mathcal{T}}$, $m(u) \in \{0,1\}^{2n}$, otherwise, $m(u) \in \{0,1\}^n$. A *complete message* m is a message at the root of $\mathcal{T}$.

Thus, for every leaf node of $\mathcal{F}$ (which is also a leaf node of the perfect binary tree), we associate $2n$ bit messages. For all other vertices, we associate an n bit message. We write $\mathbb{M}_{\mathcal{F}}$ to denote the set of all messages for $\mathcal{F}$. We simply write $\mathbb{M}_v$ and $\mathbb{M}_{v-u}$ instead of $\mathbb{M}_{\mathcal{T}_v}$ and $\mathbb{M}_{\mathcal{T}_{v-u}}$ respectively.
For a message m for $\mathcal{T}_v$ (also called m at the node v), and $u \in V_v$, we write $m|_u = m|_{\mathcal{T}_u}$, the message restricted to $\mathcal{T}_u$. Similarly, we write $m_{\mathsf{L}} := m|_{v_{\mathsf{L}}}$ and $m_{\mathsf{R}} := m|_{v_{\mathsf{R}}}$. We also write $m|_{v-u\to h}$ to denote a message for $\mathcal{T}_{v-u}$ which is same as the restricted function $m|_{\mathcal{T}_{v-u}}$, except at u, where it assigns h (instead of $m(u)$). In the context of this chapter, this means we replace the message $m(u)$ at node u by the intermediate hash output of $\mathcal{T}_u$, the tree rooted at u, and consider the message for the remaining tree, $\mathcal{T}_{v-u}$.

Definition 3.2 (Generalised Tree Hash)**.** Let $\mathcal{F}$ be a full binary sub-tree of a perfect binary tree $\mathcal{T}$ and let $m \in \mathbb{M}_{\mathcal{F}}$. For every $v \in \mathcal{F}$, we associate an intermediate hash output O_v and an intermediate input I_v recursively as follows:

1. $v \in \mathscr{L}_{\mathcal{F}} \setminus \mathscr{L}_{\mathcal{T}}$, $|m(v)| = n$: $O_v = m(v)$ and there is no input,
2. $v \in \mathscr{L}_{\mathcal{F}} \cap \mathscr{L}_{\mathcal{T}}$, $|m(v)| = 2n$: $O_v = f_v(m(v))$, $I_v = m(v)$,
3. otherwise: $|m(v)| = n$ and we define

$$I_v = (O_{v_{\mathsf{L}}} \oplus m(v),\ O_{v_{\mathsf{R}}} \oplus m(v)), \text{ and } O_v = f_v(O_{v_{\mathsf{L}}} \oplus m(v),\ O_{v_{\mathsf{R}}} \oplus m(v)) \oplus O_{v_{\mathsf{R}}}.$$

O_ω is the final hash output corresponding to $\mathcal{F}$ where ω is the root of $\mathcal{F}$. We also call I_ω final input.

Let us see what this means. If $\mathcal{F} = \mathcal{T}$, the above definition implies that for a leaf node v, the message at v, which itself is the input, is $2n$ bits long, and the output is just $f_v(m(v))$, where f_v is the 2-to-1 block compression function attached to it, and for an intermediate node, the message is n bits long, and the input and output are as defined above. If $\mathcal{F}$ is a proper sub-tree of $\mathcal{T}$, then there might exist vertices, which are leaves of $\mathcal{F}$, but not of $\mathcal{T}$. For such a vertex v, the message is n bits long, and the message itself is considered the output of the vertex. This vertex does not have any input.

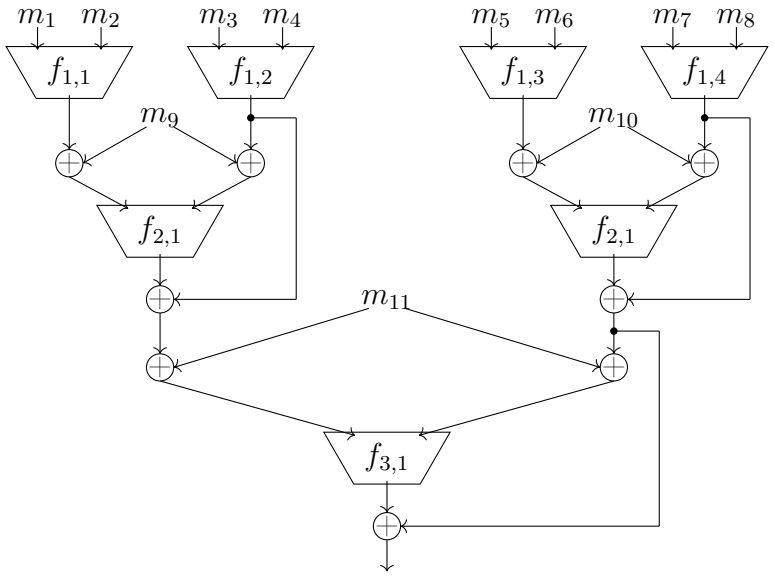

FIGURE 3.2: ABR of height 3

The ABR hash function. The ABR hash is the hash output based on a perfect binary tree $\mathcal{T}$ of height l. In terms of Definition 3.2, the case $\mathcal{F} = \mathcal{T}$ corresponds to an ABR tree, and the final hash output is the ABR hash output. Thus, ABR_l is a $(2^l + 2^{l-1} - 1)$-to-1 block hash function for $l > 1$. We refer to Figure 3.2 for a pictorial view of ABR_3. For a trivial tree $\mathcal{F} = \{w\}$, with a message $m(\omega) \in \{0,1\}^{2n}$, $\mathcal{F}(m) = f_\omega(m)$.

We write $\mathsf{H}^\tau(m)$ and $\mathsf{in}^\tau(m)$ to denote the transcript based hash and the final input respectively, whenever defined for the message m for a tree $\mathcal{F}$. If $\mathsf{H}^\tau(m)$ is defined we call m τ-computable or simply computable message. We write $\perp$ to mean that it is undefined. Note that a tree is uniquely determined from the message. We write dom_v^τ and dom_{v-u}^τ to denote the set of all computable messages at v and for $\mathcal{T}_{v-u}$ respectively. Similarly, we write ran_v^τ and ran_{v-u}^τ to denote the set of all computable hashes at v and for $\mathcal{T}_{v-u}$ respectively. The size of the set ran_v^τ, called *load at* v, is denoted as $L_{\tau,v}$.

3.3 Local Opening Analysis of ABR Hash Function

In section 3.2, we have defined a hash function based on a tree $\mathcal{F}$ for a message over the tree $\mathcal{F}$. In this section, we consider a variant of the message function and a hash function for the variant message. This is required to properly define the local opening of the ABR tree.

Message for a full binary tree. Let $\mathcal{F}$ be a full binary tree and $L \subseteq \mathscr{L}_{\mathcal{F}}$. Let $\mathcal{M}_{\mathcal{F},L}$ be the set of all functions $m : V(\mathcal{F}) \to \{0,1\}^n \cup \{0,1\}^{2n}$ such that for all $v \in \mathscr{L}_{\mathcal{F}} \setminus L$, $m(v) \in \{0,1\}^{2n}$ and for all other vertices v, $m(v) \in \{0,1\}^n$. We call m a message (or a message function) for $\mathcal{F}$.

Definition 3.3 (Generalised Tree Hash, a variant)**.** Let $m \in \mathcal{M}_{L,\mathcal{F}}$ be a message function for $\mathcal{F}$. For every $v \in \mathcal{F}$, the intermediate hash output O_v is defined recursively as follows:

1. $v \in L$, $|m(v)| = n$: $O_v = m(v)$,
2. $v \in \mathscr{L}_{\mathcal{F}} \setminus L$, $|m(v)| = 2n$: $O_v = f_v(m(v))$, $I_v = m(v)$,
3. $v \notin \mathscr{L}_{\mathcal{F}}$: we define

$$I_v = (h_1 \oplus m(v),\ h_2 \oplus m(v)) \text{ and } O_v = f_v(h_1 \oplus m(v),\ h_2 \oplus m(v)) \oplus h_2,$$

where $h_1 = O_{v_{\mathsf{L}}}$ and $h_2 = O_{v_{\mathsf{R}}}$.

The hash output corresponding to $\mathcal{F}$ is defined as $\mathcal{F}^f(m) := O_\omega$ where ω is the root of $\mathcal{F}$. We also call $I_\omega := \mathcal{F}^f_{\mathsf{in}}(m)$ final input. It is clear from the definition that for any node $v \notin L$, $\mathcal{F}^f_v(m|_v) = O_v$ and $\mathcal{F}^f_{v,\mathsf{in}}(m|_v) = I_v$.

Visualising the tree is not difficult. As an example, when $\mathcal{F} = \mathsf{ABR}_3$, we have Figure 3.2, where L is a subset of the leaf nodes, say $(1,1)$ and $(1,2)$. We now define local opening of the Generalised Tree Hash.

Definition 3.4. Let m be a message for a perfect binary tree $\mathcal{T}$. For any full binary sub-tree $\mathcal{F}$ and a set $\mathscr{L}_{\mathcal{F}} \setminus \mathscr{L}_{\mathcal{T}} \subseteq L \subseteq \mathscr{L}_{\mathcal{F}}$, we define a message $m' := \mathsf{Open}^f_{\mathcal{F},L}(m) \in \mathcal{M}_{\mathcal{F},L}$ for $\mathcal{F}$ as follows.

1. $v \in L$: $m'(v) = \mathcal{T}^f_v(m_v)$.
2. Otherwise: $m'(v) = m(v)$.

Now, we first analyse the local opening security of ABR_l proposed by [4] and then show that no non-trivial opening of ABR can achieve birthday bound security.

3.3.1 Local Opening Analysis of ABR Hash

We describe the by-pass hash corresponding to the message block m_1 for ABR_l. It is based on the full sub-tree $\mathcal{F}$ consisting of nodes $\{(i,1) : i \in [l]\} \cup \{(i,2) : i \in [l-1]\}$

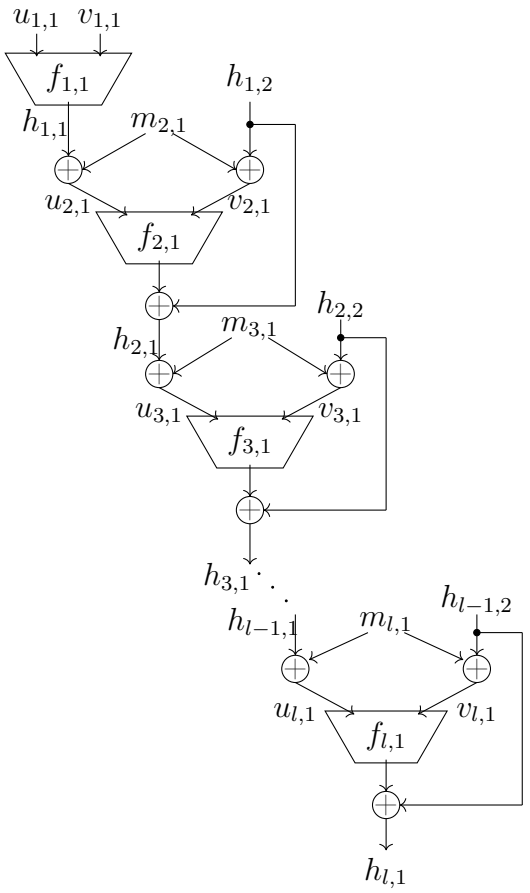

FIGURE 3.3: A specific local opening of ABR_l.

and $L = \{(1,2),(2,2),\ldots,(l-1,2)\}$. Refer to Figure 3.3. Note that the number of blocks in $\mathsf{Open}_{\mathcal{F},L}(m)$ is $2l$, and in the sub-tree $\mathcal{F}$ corresponding to $\mathsf{Open}_{\mathcal{F},L}(m)$, the number of calls to underlying compression function f is l. According to Stam's bound, there exists a collision attack with at most $2^{n/2l}$ queries. We give an attack that matches this bound.

Let $I(i,1) = (u_{i,1}, v_{i,1})$ be the input of $f_{i,1}$ and let $y_{i,1}$ be the output. Let $h_{i,2}$ be the message for node $(i,2)$. Let $h_{i,1} = y_{i,1} \oplus h_{i-1,2}$ for $i > 1$ and $h_{1,1} = y_{1,1}$. Then, $h_{i,1}$ is the output at node $(i,1)$. Also, let $m_{i,1}$ be the message associated with a non-leaf node $(i,1)$. We wish to find a collision at the output of node $(l,1)$, i.e. we need to find two messages m' and m'' for $\mathcal{F}$ such that $\mathcal{F}_{(l,1)}(m') = \mathcal{F}_{(l,1)}(m'')$. Given any message for $\mathcal{F}$, the output at node $(l,1)$ is given by $h_{l,1}$.

Note that $h_{1,1} = f_{1,1}(u_{1,1}, v_{1,1})$. After computing $h_{i-1,1}$, we proceed to compute $h_{i,1}$. We note that $h_{i-1,2}$ is a message block for $\mathcal{F}$. The input at node $(i,1)$, $I(i,1) = (h_{i-1,1} \oplus m_{i,1}, h_{i-1,2} \oplus m_{i,1}) = (u_{i,1}, v_{i,1})$ and the output at node $(i,1)$ is:

$$h_{i,1} = f_{i,1}(I(i,1)) \oplus h_{i-1,2} = f_{i,1}(u_{i,1}, v_{i,1}) \oplus h_{i-1,2}$$

$$= f_{i,1}(u_{i,1}, v_{i,1}) \oplus u_{i,1} \oplus v_{i,1} \oplus h_{i-1,1}$$
$$= g_{i,1}(u_{i,1}, v_{i,1}) \oplus h_{i-1,1}$$

where $g_{i,1}(u_{i,1}, v_{i,1}) = f_{i,1}(u_{i,1}, v_{i,1}) \oplus u_{i,1} \oplus v_{i,1}$. By induction, the final hash computation is

$$h_{l,1} = g_{l,1}(u_{l,1}, v_{l,1}) \oplus g_{l-1,1}(u_{l-1,1}, v_{l-1,1}) \oplus \ldots \oplus g_{1,1}(u_{1,1}, v_{1,1}).$$

Since the functions $f_{i,1}$ are random and independent, so are $g_{i,1}$'s. Thus $h_{l,1}$ is the XOR of l random functions. Thus, a collision is expected at node $(l, 1)$ with $2^{n/2l}$ queries. One can also apply a generalised birthday attack with complexity $2^{n/(1+\lceil \log 2l \rceil)}$.

Now, let us also look at the *target collision resistance* of the above local opening of ABR_l. Target collision resistance describes the ability of an adversary to find a second pre-image for a fixed message. Target collision resistance has many practical applications. For example, if a client sends a file F to the server and then wants the server to send part of the file F_i along with a proof of correctness, then as long as the server does not control the choice of the file F, the server would need to find a targeted collision to break security and reveal an incorrect value F_i'. Here, for a fixed message m, the final hash computation $h_{l,1}$ is fixed. Hence, for target collision resistance we wish the XOR of l random functions to collide with this value of $h_{l,1}$. This collision is expected with $2^{n/l}$ queries.

3.3.2 Decomposition of ABR Hash

Now we decompose ABR hash computation on $\mathcal{T}$ through a full binary proper sub-tree $\mathcal{F}$ sharing the same root and a set L.

Lemma 3.5 (decomposition lemma for any full binary tree)**.** *For all full binary sub-tree $\mathcal{F}$ of a perfect binary tree $\mathcal{T}$ and a set of nodes $\mathscr{L}_{\mathcal{F}} \setminus \mathscr{L}_{\mathcal{T}} \subseteq L \subseteq \mathscr{L}_{\mathcal{F}}$, we have*

$$\mathcal{T}^f = \mathcal{F}^f \circ \mathsf{Open}^f_{\mathcal{F},L}.$$

Proof. Let m be a message for $\mathcal{T}$. $\mathcal{T}^f(m)$ represents the hash output based on the perfect binary tree $\mathcal{T}$. For any node v of $\mathcal{T}$, the restricted message over $\mathcal{T}_v$ is m_v. Hence, $\mathcal{T}^f(m)$ computes $\mathcal{T}_v^f(m_v)$ for all nodes $v \in \mathcal{T}$.
For any full binary sub-tree $\mathcal{F}$ of $\mathcal{T}$, $m' = \mathsf{Open}^f_{\mathcal{F},L}$ is defined as above. For any $v \in L$: $m'(v) = \mathcal{T}_v^f(m_v)$. We calculate the hash outputs for the restricted

messages on these nodes first. Since for all other $v \in \mathcal{F}$, $m'(v) = m(v)$, and $\mathcal{F}$ is a sub-tree of $\mathcal{T}$, $\mathcal{F}^f(m')$ actually computes $\mathcal{T}_v^f(m_v)$ for all $v \in \mathcal{F} \setminus \mathscr{L}_{\mathcal{F}}$. Thus, $\mathcal{F}^f \circ \mathsf{Open}_{\mathcal{F}}^f(m)$ also computes $\mathcal{T}_v^f(m_v)$ for all nodes $v \in \mathcal{T}$ and produces the same output $\mathcal{T}^f(m)$. □

If $\mathcal{F} = \mathcal{T}$ and $L = \emptyset$ then $\mathsf{Open}_{\mathcal{F},L}^f(m) = m$. For any other proper local opening we cannot ensure birthday bound security. We prove the following theorem:

Theorem 3.6. *No non-trivial opening of* ABR *can achieve birthday bound security.*

Proof. According to Stam's bound, there exists a collision attack with at most $2^{n(\lambda-(t-0.5)/r)}$ queries on a t-to-1 block hash function making r calls to λ-to-1 block compression functions. We have $\lambda = 2$. If we want to achieve $2^{n/2}$ collision security, $t \leq 1.5r + 0.5$. In other words, if $t > 1.5r + 0.5$, then we have a collision attack with query complexity $2^{\frac{n}{2}(1-\delta/r)}$, $\delta := t - 1.5r - 0.5$.

For ABR of height l, we have $t = 2^l + 2^{l-1} - 1$ and $r = 2^l - 1$. This satisfies $t = 1.5r + 0.5$, and it is optimal. We show that for any non-trivial opening $\mathsf{Open}_{\mathcal{F},L}$ of ABR, $\mathcal{F}$ satisfies $t > 1.5r + 0.5$. Let us consider the simplest non-trivial opening, corresponding to $L = \{(1,1)\}$. Then, for $m = (m_1, m_2, m')$, where m_1, m_2 are the first two message blocks and m' is the remaining part, $\mathsf{Open}_{\mathcal{F},L}(m) = (f_{1,1}(m_1, m_2), m')$. Then, $t = 2^l + 2^{l-1} - 2$, and $r = 2^l - 2$ ($f_{1,1}$ is not called). This satisfies $t > 1.5r + 0.5$. If $\mathsf{Open}_{\mathcal{F},L}$ consists of only one sub-tree computation of height h, then for $\mathcal{F}$, we have $t = (2^l + 2^{l-1} - 1) - (2^h + 2^{h-1} - 1) + 1$ and $r = 2^l - 2^h$, which satisfies $t > 1.5r + 0.5$.

A general opening $\mathsf{Open}_{\mathcal{F},L}$ of ABR may consist of more than one complete sub-tree computation. Let the number of complete sub-tree computations in $\mathsf{Open}_{\mathcal{F},L}$ be k, and for each $1 \leq i \leq k$, let h_i be the height of the i-th sub-tree. Then, for $\mathcal{F}$, we have

$$t = (2^l + 2^{l-1} - 1) - \sum_{i=1}^{k} (2^{h_i} + 2^{h_i - 1} - 1) + k, \qquad r = (2^l - 1) - \sum_{i=1}^{k} (2^{h_i} - 1).$$

It can be easily seen that $t > 1.5r + 0.5$. Thus, no non-trivial opening of ABR can achieve birthday bound security. □

3.4 Collision Analysis of ABR hash

In this section, we first define certain items which will be required to analyse the collision.

Definition 3.7 (input multi-collision)**.** For any $x \in \{0,1\}^n$, let $\mathrm{MC}_v^\tau(x)$, called input multi-collision set at v (with x as input multi-collision value), denote the set of all messages m at v with $\mathsf{in}^\tau(m) = x$. also, let

$$\mathrm{mc}_v^\tau(x) = |\mathrm{MC}_v^\tau(x)|, \quad \mathrm{mc}_v^\tau = \max_{x\in\{0,1\}^n} \mathrm{mc}_v^\tau(x).$$

When v is the root node, we skip the notation v.

We define the newly generated messages and the hashes at a node v due to the addition of the query-response (x, y) to the transcript τ as

$$\mathsf{New}_v^\tau(x,y) := \mathrm{dom}_v^{\tau\cup(x,y)} \setminus \mathrm{dom}_v^\tau, \quad \mathsf{NewH}_v^\tau(x,y) := \mathrm{ran}_v^{\tau\cup(x,y)} \setminus \mathrm{ran}_v^\tau.$$

Clearly, $\mathsf{NewH}_v^\tau(x,y) = \mathsf{H}^{\tau\cup(x,y)}(\mathsf{New}_v^\tau(x,y))$ (image set of $\mathsf{H}^{\tau\cup(x,y)}$ for the domain $\mathsf{New}_v^\tau(x,y)$). Note that x need not be queried at v. However, to have a new computable message, x should be queried at some node, say u, in $\mathcal{T}_v$. Analysing the behaviour of the set $\mathsf{New}_v^\tau(x,y)$ (or its size) is easy when $u = v$ or when u is one of the children of v. However, it becomes more complex when u is far away from v.

- Case $u = v$: $\mathsf{New}_v^\tau(x,y) = \mathrm{MC}_v^\tau(x)$ (and does not depend on y) and we call these messages freshly generated *immediate* messages.
- Case $u \in \mathcal{T}_v \setminus v$: The newly generated messages at v is

$$\mathsf{New}_v^\tau(x,y) = \{m|_v\colon \mathsf{in}^\tau(m|_u) = x, m|_{v-u\to h} \in \mathrm{dom}_{v-u}^\tau, h = y \oplus \mathsf{H}^\tau(m|_{u_\mathsf{R}})\}.$$

So, we have $\mathbb{E}_y(|\mathsf{New}_v^\tau(x,y)|) = \dfrac{\mathrm{mc}_u^\tau(x) \times |\mathrm{dom}_{v-u}^\tau|}{2^n}$.
Now we discuss how the size of the computable message space $|\mathrm{dom}_{v-u}^\tau|$ can be written when u is one of the children or grandchildren of v.

Example 3.2. *Suppose $u = v_\mathsf{R}$. In this case,*

$$\begin{aligned}\mathsf{New}_v^\tau(x,y) =& \{m|_v\colon \mathsf{in}^\tau(m_\mathsf{R}) = x, y = \mathsf{H}^\tau(m_{\mathsf{RR}}) \oplus \mathsf{H}^\tau(m_\mathsf{L}) \oplus x_1 \oplus x_2,\\ & \mid (v,(x_1,x_2)) \in \mathrm{dom}(\tau), m(v) = x_1 \oplus \mathsf{H}^\tau(m_\mathsf{L})\}.\end{aligned}$$

So, $\mathbb{E}_y(|\mathsf{New}_v^\tau(x,y)|) \le \dfrac{\mathrm{mc}_{v_\mathsf{R}}^\tau(x) \times |\mathrm{ran}_{v_\mathsf{L}}^\tau| \times |\tau_v|}{2^n}$, *where τ_v denotes the set of elements in the transcript of the form $((v,x),y)$.*

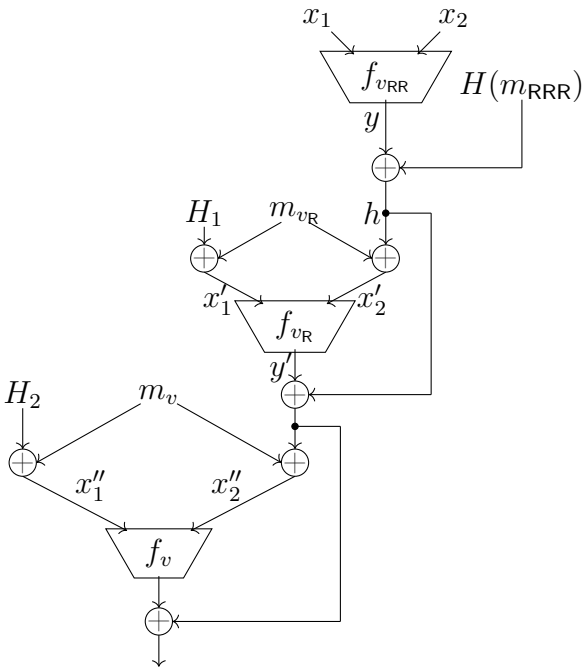

FIGURE 3.4: The graph of $\mathcal{T}_{v-u}$ when u is the rightmost grandchild of v.

Example 3.3. *In the previous case, we could write the expectation of the number of newly generated messages in terms of input multi-collision and range size of tree hash. Now, we consider $u = v_{\mathsf{RR}}$, i.e. u is a grandchild of v. Refer to Figure 3.4. Let $h = y \oplus \mathsf{H}^\tau(m_{\mathsf{RRR}})$. First, let us look at $|\mathrm{dom}^\tau_{v-v_{\mathsf{RR}}}|$.*

$$\begin{aligned}|\mathrm{dom}^\tau_{v-v_{\mathsf{RR}}}|=\{m|_v\colon\ & H_1 \oplus h = x'_1 \oplus x'_2, H_2 \oplus y' \oplus h = x''_1 \oplus x''_2,\\ & \mid H_1 = \mathsf{H}^\tau(m_{\mathsf{RL}}), H_2 = \mathsf{H}^\tau(m_{\mathsf{L}}), (v_{\mathsf{R}}, (x'_1, x'_2), y'), (v, (x''_1, x''_2), *) \in \tau\}.\end{aligned}$$

Note that this implies $H_1 \oplus H_2 \oplus \bar{y'} = x''_1 \oplus x''_2$, where $\bar{y'} = x'_1 \oplus x'_2 \oplus y'$. Thus,

$$|\mathrm{dom}^\tau_{v-v_{\mathsf{RR}}}| = \mathrm{mc}(\mathrm{ran}^\tau_{v_{\mathsf{L}}} \oplus \mathrm{ran}^\tau_{v_{\mathsf{RL}}} \oplus \bar{f}_{v_{\mathsf{R}}}) \times |\tau_v|,$$

where $\bar{f}_{v_{\mathsf{R}}}(u_1, u_2) = u_1 \oplus u_2 \oplus f_{v_{\mathsf{R}}}(u_1, u_2)$. Hence,

$$\mathbb{E}_y(|\mathsf{New}^\tau_v(x, y)|) \le \frac{\mathrm{mc}^\tau_{v_{\mathsf{RR}}}(x) \times \mathrm{mc}(\mathrm{ran}^\tau_{v_{\mathsf{L}}} \oplus \mathrm{ran}^\tau_{v_{\mathsf{RL}}} \oplus \bar{f}_{v_{\mathsf{R}}}) \times |\tau_v|}{2^n}.$$

Adversary and its queries. Let $\mathcal{L}_v$ denote the lists of all responses of f_v, for all leaf node v. We can assume that these lists are given to the adversary at the beginning of the game. This is without loss of generality as the inputs to f_v's have no role in the collision event. However, this is not true for all intermediate nodes (the non-leaf nodes), and so the adaptivity of the intermediate nodes must

be considered. We assume that an adversary makes exactly q queries to each node. Let $q' := qr$ denote the total number of queries where $r = |V^*|$ and V^* is the set of non-leaf or intermediate nodes. Let Q_v denote the set of query numbers for the node v, $v \in V^*$. So for any non-leaf node v, $|Q_v| = q$. Let (x_i, y_i) denote the i-th query-response pair made to the node v_i. So given transcript τ^{i-1} (transcript after $(i-1)$ queries), the distribution of y is uniform over $\{0,1\}^n$. For notational simplicity, we use simply i in a superscript instead of τ^i (the transcript after i-th query) in all the above notations defined so far. For example, $\mathsf{H}^i(m)$ denotes the transcript-based hash of m where the transcript is τ^i. We write New_v^i instead of $\mathsf{New}_v^{\tau^{i-1}}(x_i, y_i)$, which represents the set of all newly generated computable messages at node v immediately after obtaining i-th query-response. We also ignore the superscript τ^i completely when we all the queries have been made, i.e. $i = q'$. For example, we write $\mathrm{mc}_v(x)$ instead of $\mathrm{mc}_v^\tau(x)$ when τ is the final transcript, obtained at the end of all the queries.

- For any computable message m at v, we write $\mathrm{Fin}(m) := i$ to encode the final query index after which m is computable.
- For all m for which $m_\mathsf{L}, m_\mathsf{R}$ are τ-computable, we define $\mathrm{Fin}^*(m) = i$ such that
  $\max\{\mathrm{Fin}(m_\mathsf{L}), \mathrm{Fin}(m_\mathsf{R})\} = i$, (i.e. immediately after i-th query the final-input for the message m is computable).

3.4.1 Steps of Collision Analysis

Proper internal collision. We say that a *proper internal collision* happens at $v = (j, b)$ for a transcript τ if for some distinct messages m, m' at v, (i) $\mathsf{H}^\tau(m) = \mathsf{H}^\tau(m')$, (ii) $\mathsf{in}^\tau(m) \neq \mathsf{in}^\tau(m')$, and (iii) no collision happens for H_u^τ for all $u \in V(\mathcal{T}_v)$, $u \neq v$. By using standard reduction, a collision of ABR must have a proper internal collision at some node. So it is sufficient to bound the probability of a proper internal collision at the root node of ABR as H_v is identical to ABR_s where s denotes the level of the node v. We write $\mathsf{coll} := \mathsf{coll}_l$ to denote the proper internal collision at the root node of $\mathcal{T}$ of height l. The probability of collision of ABR_l can be then bounded as $\sum_{i \le l} 2^{l-i} \Pr(\mathsf{coll}_i)$.

Now, two types of collision can happen for any proper collision at the root. Let us consider the i-th query. This query itself can generate two new computable messages for which the collision occurs. This is the first type of collision. Also,

the hash output of one among the new computable messages generated by the i-th query can match with one of the hash outputs generated by the previous queries.

Definition 3.8 (types of collision)**.** The two types of collision can be formalized as follows:

- We call a collision pair (M, M') *twin* at the i-th query, $i \in [q']$ if $M, M' \in \mathsf{New}^i$. In this case $\mathsf{in}^i_{v_i}(M) = \mathsf{in}^i_{v_i}(M') = x_i$, where v_i is the node where the i-th query is made.
- The collision pair is called *non-twin* at the i-th query if exactly one of M and M' is a member of New^i, and the other message is τ^{i-1}-computable.

We write coll^i to denote that the proper internal collision happens at the i-th query. Moreover, if it is a twin-collision (or non-twin collision) we denote the event as $\mathsf{coll}^{i,\mathrm{tw}}$ (or $\mathsf{coll}^{i,\mathrm{ntw}}$ respectively). Thus,

$$\mathsf{coll} = \bigcup_{i \in [q']} (\mathsf{coll}^{i,\mathrm{ntw}} \cup \mathsf{coll}^{i,\mathrm{tw}}).$$

It is easy to see that a twin collision at the root node is not possible as a collision at the right child of the root node is necessary. In notation, $\mathsf{coll}^{i,\mathrm{tw}} = \emptyset$, whenever $v_i = \omega$.

Non-Twin Collision Analysis

For any non-root, non-leaf node v, we consider cross-collision between H_{-v} and H_ω. Let CC^i_v denote the set of all pairs (m, m') such that (i) m is a complete message, m' is a message for $\mathcal{T}_{-v}$ and (ii) $\mathsf{H}^i(m) = \mathsf{H}^i_{-v}(m')$. Now, a *non-twin collision* can happen at the i-th query (to the node v_i) if the freshly generated hash of a message at v_i matches with the v_i-th message block of m' for a cross-collision pair (m, m') of CC^{i-1}_v. Thus,

$$\Pr(\mathsf{coll}^{i,\mathrm{ntw}}) \leq \frac{\mathrm{mc}^{i-1}_v(x_i) \times |\mathsf{CC}^{i-1}_v|}{2^n}. \tag{3.1}$$

Now, if $v = \omega$ then the freshly generated hash at the root node is a hash. So, we have,

$$\Pr(\mathsf{coll}^{i,\mathrm{ntw}}) \leq \frac{\mathrm{mc}^{i-1}_\omega(x_i) \times L}{2^n}. \tag{3.2}$$

Twin Collision Analysis

For any non-root, non-leaf node v and $\delta \in \{0,1\}^n \setminus \{0^n\}$, let $\mathsf{C}_{\delta,v}$, called δ-collision, denote the set of all pairs (m, m') such that $\mathsf{H}^\tau_{-v}(m) = \mathsf{H}^\tau_{-v}(m')$ and $m(v) \oplus m'(v) =$

δ. We have seen that no twin collision possible at the root node. We define a set

$$\Delta^i = \{\mathsf{H}^{i-1}(m_{\mathsf{R}}) \oplus \mathsf{H}^{i-1}(m'_{\mathsf{R}}) : m, m' \in \mathrm{MC}_{v_i}^{i-1}(x_i)\}.$$

Now,

$$\Pr(\mathsf{coll}^{i,\mathrm{tw}}) \leq \frac{\sum_{\delta\in\Delta} \mathrm{mc}_v^{i-1}(x_i) \times |\mathsf{C}_{\delta,v}^{i-1}|}{2^n}. \tag{3.3}$$

Note that the size of Δ can be at most $(\mathrm{mc}_v^{i-1}(x_i))^2$.
Thus, we have seen a collision analysis requires to bound the following random variables.

1. $\mathrm{mc}_v^{i-1}(x_i)$ for all i (and so for all nodes v),
2. L: load of the hash,
3. $|\mathrm{dom}_{-v}^{i-1}|$: load for $\mathcal{T}_{-v}$ which is required to bound the load L,
4. $|\mathsf{C}_{\delta,v}^{i-1}|$: size of δ-collision, and
5. $|\mathsf{CC}_v^{i-1}|$: size of cross-collision.

In the following subsection, we present the collision analysis of ABR_2 in which we only need the input multi-collision and load (which is also bounded in terms of input multi-collision). Later, we also present a collision analysis of ABR_3 for which the above terms are present.

3.4.2 Collision Analysis of ABR_2 by ABR

As discussed above, we can assume that all queries to the compression functions at the leaf node have been made beforehand and let q denote the number of queries to each oracle. Let $\mathcal{L}_1, \mathcal{L}_2$ be two lists of outputs of the leaf node functions, and let $\omega := (2, 1)$ denote the root (the only non-leaf node for $\mathcal{T}$ of height 2). Note that the proper collision at height 1 is the same as the collision of the lists $\mathcal{L}_1, \mathcal{L}_2$. The proper collision at a leaf node can happen with probability at most $q^2/2^n$.
So, we now consider collision at the root $(2, 1)$. For this, we now define a bad event mc_ω that $\mathrm{mc}_\omega^q > n$. Equivalently, the event can be expressed as $\mathsf{mc}(\mathcal{L}_1 \oplus \mathcal{L}_2) > n$. Note that we do not have any non-leaf node other than the root node. So, the load for hash values L can be upper bounded as nq, given that mc_ω does not hold. Moreover, cross-collision and δ-collision are also not possible as we do not have

any non-leaf, non-root node. Now, it is well known that

$$\Pr(\mathsf{mc}(\mathcal{L}_1 \oplus \mathcal{L}_2) > n) \leq \frac{q^2}{2^n}$$

(see [4] for details). Thus, the collision probability is bounded by $\frac{(n^2+2)q^2}{2^n}$.

3.4.3 Collision Analysis of $\mathsf{ABR}_h, h \geq 3$ by ABR

The proof of [4] is divided into two main parts: (i) bounding the load and (ii) bounding proper collision probability in terms of the load. ABR fix a parameter ρ (which is chosen to be $n+1$, however, the exact value is not relevant to our discussion). Let $L_{i,v} = \sum_{j \leq i, j \in Q_v} |\mathsf{NewH}_v^i|$ represent the total number of generated hash values at v after all i queries. If there is no collision (which is true while we consider proper internal collision), $L_{i,v}$ is the same as the size of the set $|\mathrm{dom}_v^i|$. To bound load, ABR considered the following bad events (in our notations):

1. $\mathsf{bad}_{1,v}$: $\mathrm{mc}_v^{q'} > \rho$ at v. Let $\mathsf{bad}_1 := \cup_v \mathsf{bad}_{1,v}$.
2. $\mathsf{bad}_{2,v}$: $L_{q',v} \geq \rho q$. Let $\mathsf{bad}_2 := \cup_v \mathsf{bad}_{2,v}$.

Given $\mathsf{bad}_1, \mathsf{bad}_2$ do not hold, clearly $L \leq 2\rho q$.

Step-1: Bounding $\Pr(\mathsf{bad}_1)$

Let $\mathsf{bad}_{1,\leq i} = \cup_{(j,b):j \leq i} \mathsf{bad}_{1,v}$. So it is sufficient to bound $\Pr(\mathsf{bad}_{1,(j,b)} \wedge \neg \mathsf{bad}_{1,<j})$. Let us fix a query x at $v = (j,b)$. Now, ABR implicitly claimed the following:

Claim 1 [4]: If $\mathrm{MC}_{(j,b)}^{q'}(x) \supseteq \{m_1, \ldots, m_\rho\}$ then $\mathsf{in}_{(j-1,2b)}(m_{i,\mathsf{R}})$'s are distinct.
We note that this claim is not correct. As there can be ρ multi-collision at node $(j-1, 2b)$, each query can potentially give at most ρ multi-collision at node (j,b). Hence we can have ρ^2 multi-collision at node (j,b). Thus, a corrected version of the above claim requires to revise the parameter ρ depending on the level. So, we may redefine $\mathsf{bad}_{1,(j,b)}$: $\mathrm{mc}_v > \rho^j$ which could solve the issue. This is a fixable minor issue (but will have an impact on the claimed bound).
Now to continue with the bound, let us assume that $\mathrm{MC}_v^{q'}(x) \supseteq \{m_1, \ldots, m_\rho\}$ such that $\mathsf{in}(m_{i,\mathsf{R}})$'s are distinct and $x = (a,b)$. So we can choose ρ query indices out of q queries to $v_2 := v_{\mathsf{R}}$ in $\binom{q}{\rho}$ ways. For any such choices of ρ tuple $(i_1, i_2, \ldots, i_\rho)$ (all queried to v_2), we have

$$\Pr(f(x_{i_1}) \oplus \mathsf{H}(m_{1,\mathsf{RR}}) = b, \ldots, f(x_{i_\rho}) \oplus \mathsf{H}(m_{\rho,\mathsf{RR}}) = b) = \frac{2\rho q}{2^{n\rho}}$$

as there ρq many choices of $\mathsf{H}(m_{i,\mathsf{RR}})$ values (as we assume the load at v_{RR} is less than $2\rho q$). However, the above is true when we consider the cases where $\mathrm{Fin}^*(m_i) = j_i$ where $v_{j_i} = v_2$ for all i. The most important case in which the input multi-collision is contributed due to the final queries that are not on the right child is not considered in the proof by [4].

Step-2: Bounding $\Pr(\mathsf{bad}_2)$

Let $\mathsf{bad}_{2,\leq i} = \cup_{(j,b):j\leq i}\mathsf{bad}_{2,v}$. So it is sufficient to bound

$$\Pr(\mathsf{bad}_{2,(j,b)} \wedge \neg\mathsf{bad}_{1,<j} \wedge \neg\mathsf{bad}_1 \wedge \neg\mathsf{coll}).$$

The main idea to bound the above probability is to bound the expected number of newly generated hash at $v = (j, b)$ over all queries. Then the bad event probability can be bounded by applying Markov's inequality. We have already seen that

$$\mathbb{E}_y(|\mathsf{New}_v^i| \mid \tau^{i-1}) = \frac{\mathrm{mc}_{v_i}^{\tau}(x_i) \times |\mathrm{dom}_{v-v_i}^i|}{2^n}.$$

Moreover, we have shown that bounding $|\mathrm{dom}_{v-v_i}^i|$ becomes more complex when v_i is neither v nor a child of v (see Example 3.3). ABR tried to argue differently. They showed a bound expectation of load due to all queries of its children (see Example 3.2). Then, they continued this argument two levels up (i.e. for the queries on grandchildren as we consider in Example 3.3). However, they did not analyse this case properly. In particular, they did not consider to bound the $\mathrm{mc}(\mathrm{ran}_{v_{\mathsf{L}}}^{\tau} \oplus \mathrm{ran}_{v_{\mathsf{RL}}}^{\tau} \oplus \bar{f}_{v_{\mathsf{R}}})$. Finally, they claimed the general case by using induction which is clearly unverifiable.

Step-3: Proving collision in terms of the load

ABR stated that as analysed for ABR_2, given (i) no collision for all primitive, (ii) $\neg\mathsf{bad}_{1,\leq l}$ and (iii) $\neg\mathsf{bad}_{2,\leq l}$, the proper internal collision probability at the root node is $\mathbb{E}(L^2)/2^n$ where L is the total number computable hash values.
There is a fundamental gap in the high level of the proof. As ABR did not explain anything supporting his claim, we show that this statement is not true in general. In particular, we show (in the next subsection) a *hash mode based on* 2*-to-*1 *compression function whose load is at most* q^2 *(for any* q*-query adversary), however, a collision can be found in* $O(n)$ *queries.* So the above claim cannot be made in general.

Missing step: twin collision analysis

We find that the twin-collision analysis of the ABR hash is missed completely. The bound for δ-collision is not obvious and it requires bounding the probability of some more bad events. In the following section, we have analysed ABR_3 in which the twin-collision analysis requires a bad event dealing with the multi-collision of XOR of random oracle compression function outputs for two distinct inputs. We do not know any method to bound the number of cross-collision pairs for a general height tree.

3.4.4 Relationship between Load and Collision Probability

A hash function with a high load is unlikely to be collision-resistant. For example, $\mathsf{xor}(x_1, \ldots, x_r) = f_1(x_1) \oplus \cdots \oplus f_r(x_r)$ has load 2^r after 2 queries to each oracle f_i. It is easy to see that the hash function xor is not collision-resistant. Let $r = n$. Then, after making two queries to each function, we have sufficiently many computable messages. It is then very easy to find computable collision pairs by solving a linear system of equations. In general, if the load becomes the order of $2^{n/2}$ then one may expect a collision. However, the converse need not be true. In other words, we have a hash function where the load can not be high, but still, a collision pair can be generated efficiently.

Example of collision insecure hash functions with low load

Let MD^f be the MD hash which takes n blocks and the initial value is also replaced by one message block (so exactly $n-1$ calls of f are required). We define $MD^f_n(M) = MD^{f_1}(M) \| \cdots \| MD^{f_n}(M)$ which is n^2-to-n^2 hash function. Now we define a hash function $\mathsf{H}(M_1, M_2)$ for $M_1, M_2 \in \{0,1\}^{n^2}$:

1. Let $(C_1, C_2) = (MD^f_n(M_1) \oplus M_2, MD^g_n(C_1) \oplus M_1)$ (two round LR construction which is invertible).

2. Let $h_1, \ldots, h_n$ be $2n$-to-n functions. The final hash output is defined as $h_1(x_1) \oplus \cdots \oplus h_n(x_n)$ where $C_1 \| C_2 = x_1 \| \cdots \| x_n$, $x_i \in \{0,1\}^{2n}$.

Note that we cannot compute (C_1, C_2) for more than q^2 messages assuming there are no collisions in f and g functions. So, $L(q) \leq q^2$ for any q-query adversary.

A collision attack. Now, we construct a collision-finding algorithm for the above hash. It first finds a collision pair for xor function $h_1 \oplus \cdots \oplus h_n$ (can be

achieved easily by making $2n$ queries altogether). Let (C, C') be a collision pair. We can easily invert C and C' to obtain M and M' respectively. Clearly, (M, M') is a computable collision pair.

3.5 Analysis of ABR_3

In this section, we show that the ABR_3 construction achieves birthday security. In particular, we prove the following theorem:

Theorem 3.9 (collision theorem for ABR_3)**.** *For any adversary $\mathcal{A}$ making at most q queries to each compression function modelled to be a random oracle, we have*

$$\mathbf{Adv}^{\mathsf{coll}}_{\mathsf{H}^f}(\mathcal{A}) \leq \frac{6n^5q^2 + 3n^4q^2 + 2n^4q + 2n^2q^2 + 13q^2}{2^n}. \tag{3.4}$$

Let $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3, \mathcal{L}_4$ be the four lists of size q each corresponding to the outputs of $f_{1,1}, f_{1,2}, f_{1,3}, f_{1,4}$ respectively. We can assume that these lists are given to the adversary at the beginning of the game. This is without loss of generality as the inputs to $f_{1,i}$'s are independent from the rest of the transcripts. Also, for ease of notation, from now on we denote $f_{2,1}$ by f_1, $f_{2,2}$ by f_2 and $f_{3,1}$ by f_3. If the input to any of the functions is $u = (u_1, u_2)$, we define $u^{\oplus} = u_1 \oplus u_2$. Also, if $f_3(u) = v$, then we define $\bar{f}_3(u) = u^{\oplus} \oplus v$. As f_3 is a random oracle, the output distributions of $\bar{f}_3$ are uniform and independent. Let Q_j be the set of queries to f_j. We assume $|Q_j| = q$ for $j = 1, 2, 3$. Also, let Q_j^i denote the set of queries to Q_j up to the i-th query (including the i-th one). Let $\mathcal{G}_1 = O_{(2,1)}$ denote the set of intermediate hash outputs at node $(2, 1)$ and $\mathcal{G}_2 = O_{(2,2)}$. Let $\mathcal{H}$ denote the set of final hash outputs of ABR_3. Refer to Figure 3.5 for a pictorial representation. We follow the general approach as described before. We have already shown the collision bound for ABR_2 and so it is sufficient to bound proper collision at the root for ABR_3.
As we have seen above, the collision analysis requires us to bound some random variables. We first define some bad events to bound these random variables.

Definition 3.10 (list collision)**.** The first bad event we consider is:

- bad_0: There exists a collision in at least one of the lists $\mathcal{L}_1$, $\mathcal{L}_2$, $\mathcal{L}_3$, $\mathcal{L}_4$, $\{f_1(u) : u \in Q_1\}$, $\{f_2(u) : u \in Q_2\}$, $\{f_3(u) : u \in Q_3\}$.

Since f is modelled as a random function, the collision probability in any of the lists is at most $q^2/2^n$. Hence, $\Pr(\mathsf{bad}_0) \leq 7q^2/2^n$.

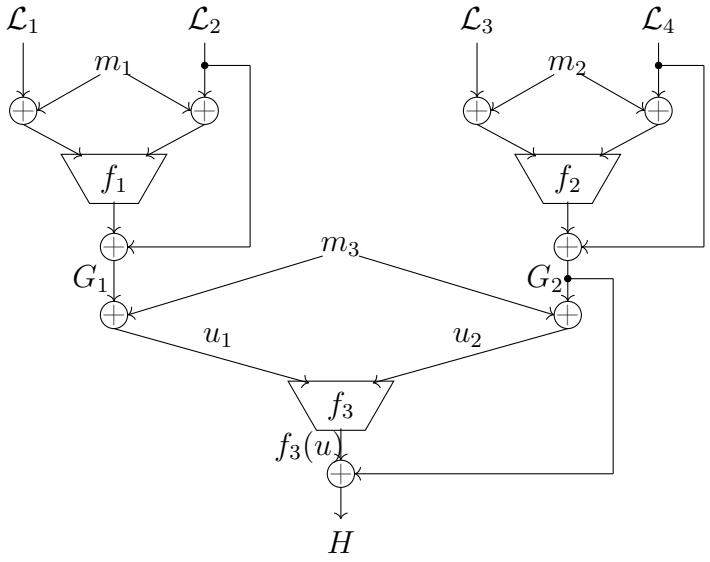

FIGURE 3.5: ABR_3 according to our new notation when the query $u = (u_1||u_2)$ is made to f_3.

Definition 3.11 (bad event on input multi-collision)**.** We define the following bad events:

- bad_1: $\mathrm{mc}(\mathcal{L}_1 \oplus \mathcal{L}_2) > n$, or $\mathrm{mc}(\mathcal{L}_3 \oplus \mathcal{L}_4) > n$,
- bad_2 : $\mathrm{mc}(\mathcal{G}_1 \oplus \mathcal{G}_2) > n^2$.

We now state some simple observations related to input multi-collision:

1. Given that bad_1 does not hold, $\mathrm{mc}_{(2,1)}, \mathrm{mc}_{(2,2)} \leq n$ and so $|\mathcal{G}_1|, |\mathcal{G}_2| \leq nq$.
2. Given that bad_2 does not hold, $\mathrm{mc}_{(3,1)} \leq n^2$ and so $L_{(3,1)} \leq n^2 q$.
3. Note, $|\mathrm{dom}(\mathcal{T}_{-(2,1)})|, |\mathrm{dom}(\mathcal{T}_{-(2,2)})| \leq nq^2$. So, $\mathbb{E}(L_{(2,1)}), \mathbb{E}(L_{(2,2)}) \leq n^2 q^3/2^n$. By Markov's inequality, $\Pr(L > 3n^2 q) \leq 2q^2/2^n$ ($3n^2 q$ because we include $L_{(3,1)}$ as well).
4. By using a similar argument as we applied for multi-collision, we have $\Pr(\mathsf{bad}_1) \leq 2q^2/2^n$.
5. Now, given that bad_1 does not hold and bad_2 holds, there must exist at least n distinct inputs to f_2 leading to n^2 input multi-collision. So, we can similarly prove $\Pr(\mathsf{bad}_2) \leq q^2/2^n$.

We say that bad_{mc} holds if either bad_1 or bad_2 happens, or $L > 3n^2 q$. Then, from above, $\Pr(\mathsf{bad}_{mc}) \leq 3q^2/2^n$. We now define bad events which would be used to bound cross-collision.

Definition 3.12 (bad event on cross-collision)**.** We define the following bad events:

- bad_3: $|\{(G_2, f_3(u), H) : G_2 \oplus f_3(u) \oplus H = 0; G_2 \in \mathcal{G}_2, u \in Q_3, H \in \mathcal{H}\}| > 3n^4q$.
- bad_4: $|\{(G_1, \bar{f}_3(u), H) : G_1 \oplus \bar{f}_3(u) \oplus H = 0; G_1 \in \mathcal{G}_1, u \in Q_3, H \in \mathcal{H}\}| > 3n^4q$.

We say that bad_{cc} holds if any one of the above happens.

If the i-th query is made at f_2, an intermediate hash output G_2 generated at this level due to this query can match with a query u already done to f_3 to generate a final hash output H which was already previously generated by the first $(i-1)$ queries. The event bad_3 implies that the number of such triplets $(G_2, f_3(u), H)$ is more than $3n^4q$. bad_4 has a similar implication when we consider $\mathcal{G}_1$ instead of $\mathcal{G}_2$.

Lemma 3.13. $\Pr(\mathsf{bad}_{cc} \wedge \neg\mathsf{bad}_{mc}) \leq 2q^2/2^n$.

Proof. $\Pr(\mathrm{mc}(\mathcal{G}_2 \oplus \mathrm{ran}(f_3)) > n^2) \leq q^2/2^n$. The proof is similar to that of event bad_2. Hence, for a fixed $H \in \mathcal{H}$, we have

$$\Pr[|\{(G_2, f_3(u), H) : G_2 \oplus f_3(u) \oplus H = 0; G_2 \in \mathcal{G}_2, u \in Q_3\}| > n^2] \leq q^2/2^n.$$

Now, there are $3n^2q$ choices for H. Therefore, $\Pr(\mathsf{bad}_3 \wedge \neg\mathsf{bad}_{mc}) \leq q^2/2^n$. A similar argument works for bad_4. Hence,

$$\Pr(\mathsf{bad}_{cc} \wedge \neg\mathsf{bad}_{mc}) \leq 2q^2/2^n.$$

Given that bad_{cc} does not hold, $|\mathsf{CC}_{(2,1)}| \leq 3n^4q$ (or $|\mathsf{CC}_{(2,2)}| \leq 3n^4q$ respectively). We finally define bad events which would be used to bound δ-collision pairs.

Definition 3.14 (bad event on δ-collision)**.** We define the following bad event:

- bad_5: $\mathsf{mc}(\bar{f}_3(u) \oplus \bar{f}_3(u')) > n$.

We say that bad_δ holds if the above happens.

Lemma 3.15. $\Pr(\mathsf{bad}_\delta) \leq \frac{q^2}{2^n}$.

Proof. Since $f_3(u)$ is random, $\bar{f}_3(u) = f_3(u) \oplus u^{\oplus}$ is also random. Therefore, bounding bad_5 is similar to bounding bad_1. □

Given that bad_δ does not hold, $|\mathsf{C}_\delta| \leq n$. Let $\mathsf{bad} = \mathsf{bad}_0 \cup \mathsf{bad}_{mc} \cup \mathsf{bad}_{cc} \cup \mathsf{bad}_\delta$. Then, $\Pr(\mathsf{bad}) \leq \frac{13q^2}{2^n}$.

Collision Analysis

We assume that bad does not hold. Since $\mathsf{coll} = \bigcup_{i\in[q],v\in V\setminus\mathscr{L}} (\mathsf{coll}_v^{i,\mathrm{ntw}} \cup \mathsf{coll}_v^{i,\mathrm{tw}})$, we need to bound $\mathsf{coll}_v^{i,\mathrm{ntw}}$ and $\mathsf{coll}_v^{i,\mathrm{tw}}$ for $v = (2,1),(2,2),(3,1)$. In the following lemmas, we bound them, assuming bad does not occur. We already know that $\mathsf{coll}_{(3,1)}^{i,\mathrm{tw}}$ does not occur.

Lemma 3.16. $\Pr(\mathsf{coll}_{(3,1)}^{i,\mathrm{ntw}} \mid \neg\mathsf{bad}) \leq \frac{3n^4q}{2^n}$.

Proof. As seen above in equation 3.2,

$$\Pr(\mathsf{coll}_{(3,1)}^{i,\mathrm{ntw}}) \leq \frac{\mathrm{mc}_{(3,1)}^{i-1}(x_i) \times L}{2^n}.$$

Given $\neg\mathsf{bad}$, $\mathrm{mc}_{(3,1)} \leq n^2$ and $L \leq 3n^2q$. Hence, $\Pr(\mathsf{coll}_{(3,1)}^{i,\mathrm{ntw}} \mid \neg\mathsf{bad}) \leq \frac{3n^4q}{2^n}$. □

Lemma 3.17. $\Pr(\mathsf{coll}_{(2,1)}^{i,\mathrm{ntw}} \mid \neg\mathsf{bad}) \leq \frac{3n^5q}{2^n}$.

Proof. As seen above in equation 3.2,

$$\Pr(\mathsf{coll}_{(2,1)}^{i,\mathrm{ntw}}) \leq \frac{\mathrm{mc}_{(2,1)}^{i-1}(x_i) \times |\mathsf{CC}_{(2,1)}^{i-1}|}{2^n}.$$

Given $\neg\mathsf{bad}$, $\mathrm{mc}_{(2,1)} \leq n$ and $|\mathsf{CC}_{(2,1)}| \leq 3n^4q$. Hence, $\Pr(\mathsf{coll}_{(2,1)}^{i,\mathrm{ntw}} \mid \neg\mathsf{bad}) \leq \frac{3n^5q}{2^n}$. □

Lemma 3.18. $\Pr(\mathsf{coll}_{(2,2)}^{i,\mathrm{ntw}} \mid \neg\mathsf{bad}) \leq \frac{3n^5q}{2^n}$.

Proof. This proof is similar to that of the previous lemma.

$$\Pr(\mathsf{coll}_{(2,2)}^{i,\mathrm{ntw}}) \leq \frac{\mathrm{mc}_{(2,2)}^{i-1}(x_i) \times |\mathsf{CC}_{(2,2)}^{i-1}|}{2^n}.$$

Given $\neg\mathsf{bad}$, $\mathrm{mc}_{(2,2)} \leq n$ and $|\mathsf{CC}_{(2,2)}| \leq 3n^4q$. Hence, $\Pr(\mathsf{coll}_{(2,2)}^{i,\mathrm{ntw}} \mid \neg\mathsf{bad}) \leq \frac{3n^5q}{2^n}$. □

Lemma 3.19. $\Pr(\mathsf{coll}_{(2,1)}^{i,\mathrm{tw}} \mid \neg\mathsf{bad}) \leq \frac{n^4}{2^n}$.

Proof. As seen above in equation 3.3,

$$\Pr(\mathsf{coll}_{(2,1)}^{i,\mathrm{tw}}) \leq \frac{\sum_{\delta\in\Delta} \mathrm{mc}_{(2,1)}^{i-1}(x_i) \times |\mathsf{C}_{\delta,(2,1)}^{i-1}|}{2^n}.$$

Given $\neg\mathsf{bad}$, $\mathrm{mc}_{(2,1)} \leq n$, $|\Delta| \leq (\mathrm{mc}^{i-1}_{(2,1)}(x_i))^2 \leq n^2$ and $|\mathsf{C}_{\delta,(2,1)}| \leq n$. Hence, $\Pr(\mathsf{coll}^{i,\mathrm{tw}}_{(2,1)} \mid \neg\mathsf{bad}) \leq \dfrac{n^4}{2^n}$. □

Lemma 3.20. $\Pr(\mathsf{coll}^{i,\mathrm{tw}}_{(2,2)} \mid \neg\mathsf{bad}) \leq \frac{n^4}{2^n}$.

Proof. This proof is similar to that of the previous lemma.

$$\Pr(\mathsf{coll}^{i,\mathrm{tw}}_{(2,2)}) \leq \frac{\sum_{\delta\in\Delta} \mathrm{mc}^{i-1}_{(2,2)}(x_i) \times |\mathsf{C}^{i-1}_{\delta,(2,2)}|}{2^n}.$$

Given $\neg\mathsf{bad}$, $\mathrm{mc}_{(2,2)} \leq n$, $|\Delta| \leq (\mathrm{mc}^{i-1}_{(2,2)}(x_i))^2 \leq n^2$ and $|\mathsf{C}_{\delta,(2,2)}| \leq n$. Hence, $\Pr(\mathsf{coll}^{i,\mathrm{tw}}_{(2,2)} \mid \neg\mathsf{bad}) \leq \dfrac{n^4}{2^n}$. □

From the above lemmas, we have

$$\begin{aligned}\Pr(\mathsf{coll} \mid \neg\mathsf{bad}) &\leq \sum_{i\in[q],v\in V\setminus\mathscr{L}} \Pr(\mathsf{coll}^{i,\mathrm{ntw}}_{v} \mid \neg\mathsf{bad}) + \Pr(\mathsf{coll}^{i,\mathrm{tw}}_{v} \mid \neg\mathsf{bad}) \\ &\leq \frac{6n^5q^2 + 3n^4q^2 + 2n^4q}{2^n}.\end{aligned}$$

Therefore,

$$\begin{aligned}\Pr(\mathsf{coll}) &\leq \Pr(\mathsf{coll} \mid \neg\mathsf{bad}) + \Pr(\mathsf{bad}) \\ &\leq \frac{6n^5q^2 + 3n^4q^2 + 2n^4q + 13q^2}{2^n}.\end{aligned}$$

Note that we have bound the proper collision probability at the root for ABR_3. Since bad_0 does not occur, collision does not occur at the leaf node. As seen in Section 3.4.2, the probability that proper collision occurs at node $(2,1)$ (resp. $(2,2)$) is bounded above by $\frac{n^2q^2}{2^n}$. Hence, the theorem is proved.

3.6 Summary

In this chapter, we revisit the collision security of the ABR hash and identify significant gaps in its analysis. Important missing cases have also been highlighted. Additionally, we have demonstrated collision security for ABR of height 3, identifying several new bad events for ABR_3 that were not considered for the general hash. The collision security analysis for the general hash remains unresolved, leaving the optimality of Stam's bound for an arbitrary domain hash an open question.
Furthermore, we have discovered that the ABR hash cannot achieve any non-trivial local opening that provides birthday-bound security. This reveals a limitation in

its application for local openings. Specifically, the efficient local opening proposed by [4] can be compromised with a query complexity of $O(2^{n/2l})$.

Chapter 4

Exact Security Analysis of Ascon

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) standardisation 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 chapter, we present a tight and comprehensive security analysis of the Ascon AEAD mode in 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 chapter, we demonstrate that Ascon achieves AEAD 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 rates recommended by the designers of Ascon.

4.1 Introduction

The Sponge function, initially proposed by Bertoni et al. at the ECRYPT Hash Workshop [22], 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 [26] being the notable winner. At a high level, a Sponge construction utilises

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 initialising 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 [23] 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 initialisation interface and a duplexing interface. Initialisation 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 [55, 56]) 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 initialisation and duplexing calls to π) and the time complexity T (representing the total number of direct calls to π).

4.1.1 Ascon

Ascon was initially introduced as a candidate in Round 1 of the CAESAR competition [45]. Subsequent versions (v1.1 and v1.2) incorporated minor modifications to the original design (version 1 [56]). The latest version (v1.2 [55]), declared as the winner of the NIST Lightweight Cryptography (LWC) project [103], 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 utilise 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 [23], specifically the MonkeyDuplex construction [25]. However, unlike

MonkeyDuplex, Ascon's mode employs double-keyed initialisation and double-keyed finalisation to enhance its robustness. For a detailed description of the Ascon AEAD mode, please refer to Section 4.3.

4.1.2 Existing Security Analysis

It has come to our attention that previous analyses of the Ascon mode predominantly regard it as a variant of the Duplex construction (as indicated in [55]), 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, is by Bertoni et al. [24]. However, they could 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 is

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

where D is the data complexity and T is the time complexity. In 2014 [85], and later in 2019 [86], 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. [6] 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. [47] 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)*. Essentially, they prove a bound on the advantage of distinguishing a full-state keyed duplex from an 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. [37] introduce a generic AEAD construction called the Transform-then-Permute (TtP) construction. They demonstrate that well-known constructions such as the keyed Sponge Duplex construction, Beetle [32], and SpoC [3] can be viewed as specific examples of this generic construction. In their work, they provide 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 [37]. 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 LWC guidelines. 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.

In a concurrent work [94], Lefevre and Mennink also present a dedicated security analysis of Ascon. While they focus on a different setting (authenticity under nonce misuse and state recovery, multi-user security), they could show the impact of strengthened initialisation and finalisation of Ascon in the case of authenticity under state recovery. However, in the case of conventional single-user nonce-based authenticity, their bounds reduce to $\frac{q_d T}{2^c}$.

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. [69] establishes a lower bound of the form $DT \gg 2^{3c/2}$.

4.1.3 Contributions

In this chapter, motivated by the recently concluded NIST LWC competition, we try to provide an improved security bound for the Ascon AEAD mode. 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 initialisation and double-keyed finalisation.
Our analysis establishes a tight security bound, considering the tag size τ bits, key size κ bits, capacity c bits, and state size b bits. The derived bound is given by

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

Comparing our result with the recent analysis by Gilbert et al. [69], 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 finalisation process of Ascon during tag generation, which plays a vital role in achieving such a tight and improved security bound. It should be emphasised 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. [69] 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.

4.1.4 Chapter Outline

In Section 4.2, we first elaborate on function graph structures that play a crucial role in our subsequent analyses. Moving forward, in section 4.3, 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 4.4, we present a rigorous proof of our main theorem, using the H-coefficient technique. Finally, in Section 4.5, we discuss the tightness of our bound, and summarise the paper.

4.2 Function Graph Structures

4.2.1 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 4.1. Let $\mathcal{L} : \{0,1\}^b \dashrightarrow \{0,1\}^c$ for $r := b - c > 0$. We associate a labelled directed graph $G := G^{\mathcal{L}}$, called *(labelled) 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 labelled edge set

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

We call it (labelled) 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).

4.2.2 Sampling Process of a Labelled Walk

Let $f : \{0,1\}^b \dashrightarrow \{0,1\}^b$, $x^k = (x_1, \ldots, x_k)$ be a k-tuple label, $k \geq 0$, and $z_0 \in \{0,1\}^c$. We now describe a process that extends the partial function f to f' so that there is a walk

$$z_0 \xrightarrow{x_1} z_1 \xrightarrow{x_2} \cdots \xrightarrow{x_k} z_k$$

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

Algorithm 1 $\mathsf{Rand\_Extn}^f(z_0, x^k)$

Initialise $f' = f$
for $j = 1$ to k **do**
    $v_j = f'(x_j, z_{j-1})$
    if $v_j = \perp$ **then**
        $v_j \leftarrow \{0,1\}^b$
        $f' \leftarrow f' \cup \{(x_j \| z_{j-1}, v_j)\}$
    $z_j = \lfloor v_j \rfloor_c$

in the graph $G^{\lfloor f \rfloor_c}$. The process, defined in Algorithm 1 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.

The described process provides a clear and effective method for successfully completing a labelled walk. It operates on the basis of 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.

4.2.3 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 labelled 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 randomised extension algorithm for $\mathcal{P}^{\oplus}$ in Algorithm 2, denoted as $\mathsf{xorRand\_Extn}^{\mathcal{P}}(v_0, x^k)$,

Algorithm 2 $\mathsf{xorRand\_Extn}^{\mathcal{P}}(v_0, x^k)$

Initialise $\mathcal{P}' = \mathcal{P}$
for $j = 1$ to k **do**
  $v_j = \mathcal{P}'(v_{j-1} \oplus (x_j \| 0^c))$
  if $v_j = \perp$ **then**
    $v_j \leftarrow \{0,1\}^b$
    $\mathcal{P}' \leftarrow \mathcal{P}' \cup \{(v_{j-1} \oplus (x_j \| 0^c), v_j)\}$

$v_0 \in \{0,1\}^b, x_i \in \{0,1\}^r$.

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:

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

4.3 The Ascon AEAD Mode

In this section, we define the Ascon AEAD [55] mode. 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 an $IV \in \{0,1\}^{b-\kappa-\nu}$. The AEAD uses a permutation π (Ascon permutation), modelled to be the random permutation while we analyse its security. 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. 4.1.

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) initialisation, (ii) associated data and message processing, and (iii) tag generation, run sequentially.

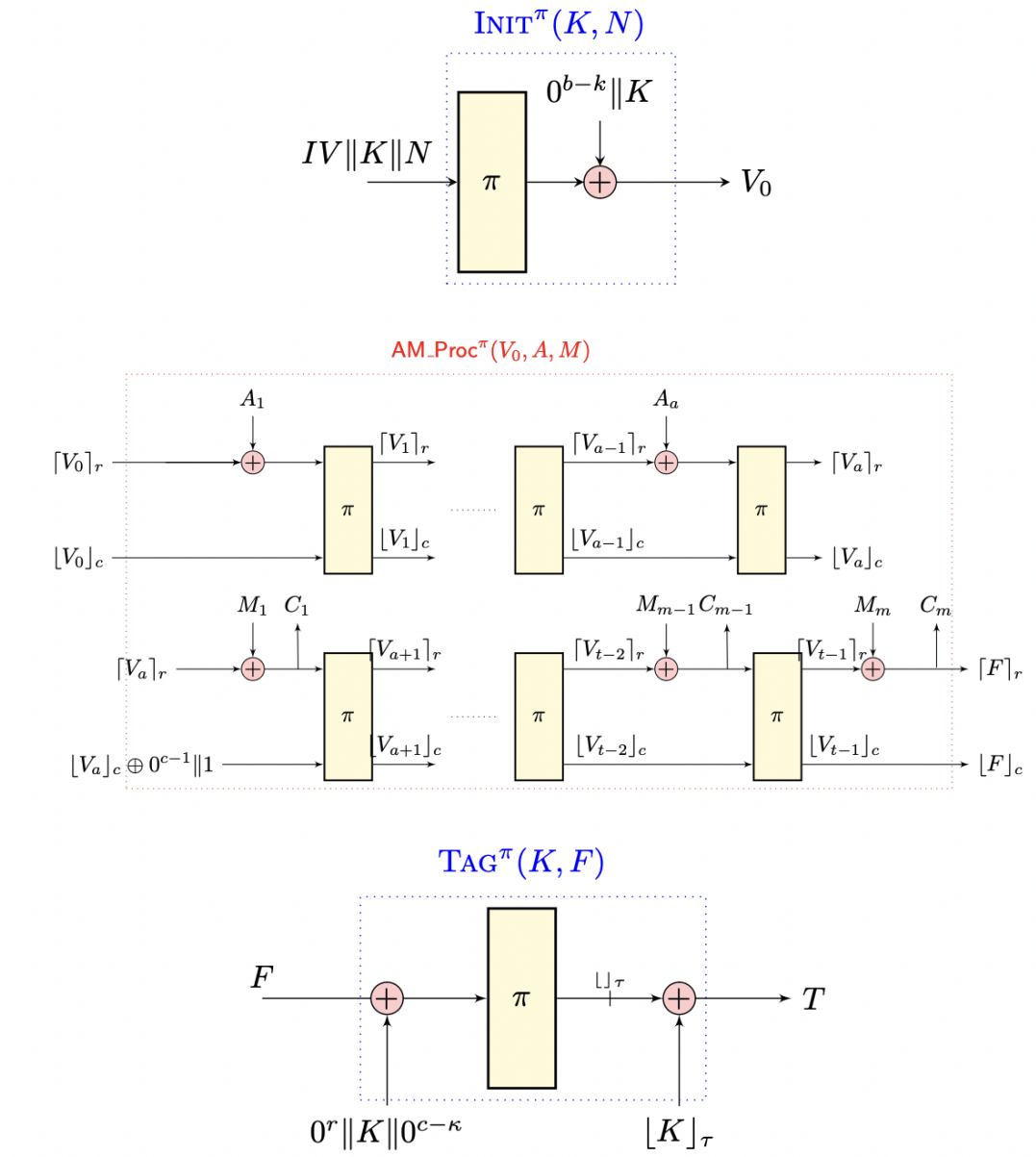

FIGURE 4.1: Encryption in ASCON AEAD. The final ciphertext is $C = \lceil C_1\|\cdots|C_m\rceil_{|M|}$. Here, $t = a + m$.

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

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

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

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

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

Associated 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

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

Tag Generation. Finally, we compute

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

The Ascon AEAD returns (C, T).

Remark 4.2. 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 initialisation phase and for tag generation and verification. On the other hand, p^b is utilised for processing associated data, messages, and ciphertext. For instance, in the Ascon-128 construction, a is set to 12, while b is set to 6. When modelling Ascon in the random permutation model, there are two options: either using the same permutation $\pi = p^a = p^b$, or using 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.

Decryption 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 [55] 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 $F = C_l' \parallel (\lfloor V_a \rfloor_{b-d} \oplus 10^*1)$.
   - Case $l \geq 2$: We compute

$$z_a \oplus 0^*1 \xrightarrow{C_1} z_{a+1} \xrightarrow{C_2} \cdots \xrightarrow{C_{l-2}} z_{a+l-2}.$$

   We define $F = C_l' \parallel (\lfloor \pi(C_{l-1} \| z_{a+l-2}) \rfloor_{b-d} \oplus 10^*)$.
5. Rejects if $T \neq \textsc{Tag}^\pi(K, F)$, otherwise, it accepts.

4.3.1 Security bound of Ascon

Theorem 4.3 (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:*

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

4.3.2 Interpretation of Theorem 4.3

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 give upper bounds to $\mathsf{mcoll}(\sigma_e, 2^r)$, $\mathsf{mcoll}(q_e, 2^\tau)$, and $\mathsf{mcoll}(\sigma + q_p, 2^\tau)$, depending on the choice of r and τ. Note that $\mathsf{mcoll}(q_e, 2^{r+c-\kappa}) \leq \mathsf{mcoll}(\sigma_e, 2^r)$, so we do not need to bound it separately.
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

- 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

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{AEAD}}_{\mathrm{ASCON}}(\mathscr{A}) &\leq \frac{2q_d}{2^\tau} + \frac{\sigma_e^2}{2^b} + \frac{\sigma_d(q_p + \sigma_d)}{2^b} + \frac{3(\sigma_d + q_p)}{2^c} + \frac{40q_d}{2^c} \\
&\quad + \frac{4(q_p + \sigma)}{2^\kappa} + \frac{q_e^2 + q_d^2 + q_e q_d + (q_e + q_d)(\sigma + q_p)}{2^b} \\
&\quad + \frac{5(q_p + \sigma_e)q_d}{2^{\kappa+\tau}} + \frac{q_d(\sigma + q_p)}{2^{c+\tau}} \\
&= O(\frac{q_d}{2^\tau}) + O(\frac{\sigma q_p}{2^b}) + O(\frac{q_p}{2^\kappa}) + O(\frac{q_p}{2^c})
\end{aligned}$$

(assuming $\sigma \leq q_p$).

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

$$\begin{aligned}\mathbf{Adv}^{\mathrm{AEAD}}_{\textsc{Ascon}}(\mathscr{A}) \leq\ & \frac{2q_d}{2^\tau} + \frac{\sigma_e^2}{2^b} + \frac{\sigma_d(q_p+\sigma_d)}{2^b} + \frac{3(\sigma_d+q_p)}{2^c} + \frac{3q_d}{2^c} \\ & + \frac{4(q_p+\sigma)}{2^\kappa} + \frac{q_e^2+q_d^2+q_eq_d+(q_e+q_d)(\sigma+q_p)}{2^b} \\ & + \frac{75q_d}{2^\kappa} + \frac{q_d(\sigma+q_p)}{2^{c+\tau}} \\ =\ & O(\frac{q_d}{2^\tau}) + O(\frac{\sigma q_p}{2^b}) + O(\frac{q_p}{2^\kappa}) + O(\frac{q_p}{2^c}).\end{aligned}$$

Thus, in terms of order, a tag size of 64 bits yields the same security as a tag size of 128 bits. 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 $\tau = 64$.

4.3.3 Tightness of the Bound

In the worst case scenario, the obtained bound is of the following order:

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

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.

4.4 Proof of Theorem 4.3

We first prove the theorem under the additional assumption that $\kappa < c$. Then, in Section 4.4.3, we extend the proof to the $\kappa = c$ case.

4.4.1 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

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

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

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

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

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

4.4.2 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 mode.
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 behaviour. To effectively analyse 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 use 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 $\mathsf{V}_i \xleftarrow{\$} \{0,1\}^b \setminus \mathsf{range}(\mathsf{P})$, $\mathsf{P} \leftarrow \mathsf{P} \cup \{(\mathsf{U}_i, \mathsf{V}_i)\}$, $\mathsf{return}\ \mathsf{V}_i$.
  2. If $\mathsf{dir}_i = -1$, we set $\mathsf{V}_i = \mathsf{Q}_i$. Let $\mathsf{U}_i \xleftarrow{\$} \{0,1\}^b \setminus \mathsf{domain}(\mathsf{P})$, $\mathsf{P} \leftarrow \mathsf{P} \cup \{(\mathsf{U}_i, \mathsf{V}_i)\}$, $\mathsf{return}\ \mathsf{U}_i$.

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

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

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 is capable of making nonce-respecting encryption queries only, we can establish an upper bound for the probability of bad_1 as given in Lemma 5.7 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 4.4. $\Pr(\mathsf{bad}_1 = 1) \leq \frac{q_d}{2^\tau}$.

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:

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

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

Bad Event ($\mathsf{P}_{\mathsf{Enc}}$ 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 4.5. $\Pr(\mathsf{bad}_2 = 1 \vee \mathsf{bad}_3 = 1) \leq \dfrac{\sigma_e^2}{2^b}$.

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 randomised extension. Later, we set a bad event if it is not disjoint (both from the domain and the range) with $\mathsf{P}_{\mathsf{Enc}}$. This would ensure the compatibility of $\mathsf{P}_1 \sqcup \mathsf{P}_{\mathsf{Enc}}$ (where P_1 is the randomised 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 of the i-th decryption query and 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 $0 \le p_i \le a'_i$:
   - $\mathsf{V}'_{i,p_i} := \mathsf{V}_{j,p_i}$.
   - 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 randomised 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 4.6. $\Pr(\mathsf{bad}_4 = 1) \leq \dfrac{\sigma_d(q_p + \sigma_d)}{2^b}$.

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

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

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

Lemma 4.7. $\Pr(\mathsf{bad}_5 = 1) \leq \dfrac{\mathsf{mcoll}(\sigma_e, 2^r) \times (\sigma_d + q_p)}{2^c}$.

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

$\Pr(\mathsf{bad}_5 = 1 \mid \max\{\rho_1, \rho_2\} = \rho) \le \rho(\sigma_d + q_p)/2^c$. Hence, using the expectation of $\max\{\rho_1, \rho_2\}$, and applying Lemma 2.11 and Remark 2.12, 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 $\mathsf{P}_2 := \mathsf{P}_1 \sqcup \mathsf{P}_{\mathsf{Enc}}$ 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 4.8. $\Pr(\mathsf{bad}_6 = 1) \le \dfrac{\mathsf{mcoll}(q_e, 2^\tau) q_d}{2^c}$.

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, $\Pr(\mathsf{bad}_6 = 1 \mid \rho_3 = \rho) \le \frac{\rho q_d}{2^c}$. Hence, using the expectation of ρ_3, and applying Lemma 2.11, 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

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

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

1. For all $i \in [q_e]$, $\mathsf{I}_i := IV \| K \| \mathsf{N}_i$, $\mathsf{O}_i := \mathsf{V}_{i,0} \oplus 0^{b-\kappa} \| K$,
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 $\mathsf{P}_{\mathsf{init}} = ((\mathsf{I}_i, \mathsf{O}_i)_{i \in [q_e]}, \quad (\mathsf{I}'_j, \mathsf{O}'_j)_{j \in \mathcal{J}})$.

Bad Event (Permutation Compatibility of $\mathsf{P}_{\mathsf{init}}$ 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, $\mathsf{P}_3 := \mathsf{P}_2 \sqcup \mathsf{P}_{\mathsf{init}}$ 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 4.9. $\Pr(\mathsf{bad}_7 = 1) \leq \dfrac{q_p + \sigma}{2^\kappa} + \dfrac{q_e q_d + (q_e + q_d)(\sigma + q_p)}{2^b}$.

Now, we settle tag computation for all encryption queries. 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 $\mathsf{P}_{\mathsf{tag}} = ((\mathsf{X}_i, \mathsf{Y}_i)_{i \in [q_e]})$.

Bad Event (Permutation Compatibility of $\mathsf{P}_{\mathsf{tag}}$ and P_3). We define $\mathsf{bad}_8 = 1$ 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 4.10. $\Pr(\mathsf{bad}_8 = 1) \leq \dfrac{\mathsf{mcoll}(q_e, 2^{r+c-\kappa})(\sigma + q_p)}{2^\kappa} + \dfrac{q_e(\sigma + q_p)}{2^b}$.

Proof. 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 2.12, we have

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

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 $\mathsf{X}'_i := \mathsf{F}'_i \oplus (0^r \| K \| 0^{c-\kappa})$. 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:

- Let $\mathsf{F}'_i = (\beta'_i \| \gamma'_i \| \delta'_i)$, where $|\beta'_i| = r + \kappa - \tau$, $|\gamma'_i| = \tau$ and $|\delta'_i| = c - \kappa$. We set $\mathsf{bad}_9 = 1$ if

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

  If $\mathsf{bad}_9 = 1$, 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_5 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_5. Then, by the randomness of K,

$$\Pr(\mathsf{bad}_9 = 1 \mid \rho_5 = \rho) \leq \frac{\rho q_d}{2^\kappa}.$$

  So, using the expectation of ρ_5, and applying Lemma 2.13,we have

  Lemma 4.11. $\Pr(\mathsf{bad}_9 = 1) \leq \dfrac{\mathsf{mcoll}(\sigma + q_p, 2^\tau) q_d}{2^\kappa}$.

- $\mathsf{X}'_i \notin \mathsf{domain}(\mathsf{P}_4)$. Let $y_i = \lfloor \mathsf{Y}'_i \rfloor_\tau$. We set $\mathsf{bad}_{10} = 1$ 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 4.12. $\Pr(\mathsf{bad}_{10} = 1) \leq \dfrac{q_d}{2^\tau}$.

Let bad denote the union of all bad events, namely $\cup_{i=1}^{10} \mathsf{bad}_i$. By Lemmas 4.4 through 4.12, we have shown that

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

If all these bad events do not occur, then all the decryption queries are correctly rejected for the injective partial function P_4.
Let $\mathsf{P}_{\mathsf{fin}} := \mathsf{P}_4 \cup ((\mathsf{X}'_i, \mathsf{Y}'_i)_{i\in[q_d]})$. In the offline transcript, we provide all the input-outputs of $\mathsf{P}_{\mathsf{fin}}$. Then,

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

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 when $\kappa < c$.

4.4.3 Extension of the Theorem when $\kappa = c$

The reason why the above proof cannot be carried over to this case is that when $\kappa = c$, for any encryption or decryption query of block length 1, the inner key of the initialisation phase cancels out the inner key of the finalisation phase, as can be seen in Fig. 4.2.

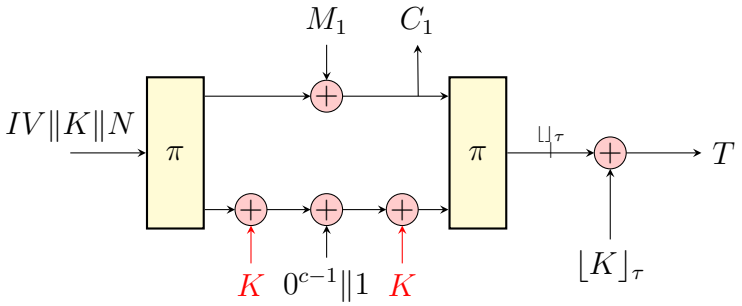

FIGURE 4.2: Encryption in ASCON AEAD when $\kappa = c$, and we have only a single block message without any associated data. The two keys additions in red cancel each other out. The final ciphertext is $C = \lceil C_1 \rceil_{|M|}$, and tag is T.

Note that whenever the construction query is at least two blocks long, the keys do not cancel out, and the above process can be carried out without any change. Even when the construction query is of length one block, the description of the real world and the online phase of the ideal world in this case is exactly the same as the previous case (the $\kappa < c$ case). Moreover, the associated data and message/ciphertext for encryption and decryption queries is processed as above.

The only differences arise after sampling the key. We present a detailed proof of this case here.

We misuse the notation a bit, and use the same bad event notations as above for easier understanding. Events bad_1 to bad_6 are exactly the same as the $\kappa < c$ case, as there are no differences with the previous case before sampling the key. We directly move on to the sampling of the key K and define $\mathcal{J}$ as above.

We define the input-outputs for the underlying permutation used in the initialisation as follows:

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

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

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

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

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

Bad Event (Permutation Compatibility of $\mathsf{P}_{\mathrm{init}}$ and P_2). We define $\mathsf{bad}_7 = 1$ 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 $i \in [q_e]$ 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 $i \in [q_e], j \in [q_d]$.
4. $\mathsf{O}_i = \mathsf{O}_j$ for $i, j \in [q_e]$, or $\mathsf{O}'_i = \mathsf{O}'_j$ for $i, j \in [q_d]$ such that $\mathsf{N}'_i \neq \mathsf{N}'_j$.

It can be easily seen that the first three cases remain exactly the same even after these new definitions of O_i and O'_j. This is because both the proofs use the randomness of $\mathsf{V}_{i,0}$ and $\mathsf{V}'_{j,0}$ only for checking the output collision and compatibility. The last case arises because, for $i, j \in [q_e]$ such that $t_i = 1, t_j \neq 1$, we can have $\mathsf{V}_{i,0} = \mathsf{V}_{j,0} \oplus 0^{b-\kappa} \| K$. However, the probability of this is bounded above by $1/2^b$. The same applies for decryption queries too. Thus, we have

Lemma 4.13. $\Pr(\mathsf{bad}_7 = 1) \leq \dfrac{q_p + \sigma}{2^\kappa} + \dfrac{q_e^2 + q_d^2 + q_e q_d + (q_e + q_d)(\sigma + q_p)}{2^b}$.

Define $\mathsf{P}_3 := \mathsf{P}_2 \sqcup \mathsf{P}_{\mathsf{init}}$. Now, coming to tag computation for encryption queries, we define

$$\mathsf{X}_i := \begin{cases} \mathsf{F}_i, & \text{if } t_i = 1 \\ \mathsf{F}_i \oplus (0^r \| K \| 0^{c-\kappa}), & \text{otherwise} \end{cases}, \quad \mathsf{Y}_i := \alpha_i \| (\mathsf{T}_i \oplus K_2),$$

where $\alpha_i \xleftarrow{\$} \{0,1\}^{b-\tau}$. Define $\mathsf{P}_{\mathsf{tag}} = ((\mathsf{X}_i, \mathsf{Y}_i)_{i \in [q_e]})$.

Bad Event (Permutation Compatibility of $\mathsf{P}_{\mathrm{tag}}$ and P_3). We define $\mathsf{bad}_8 = 1$ 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 4.14. $\Pr(\mathsf{bad}_8 = 1) \leq \dfrac{\mathsf{mcoll}(q_e, 2^r)(\sigma + q_p)}{2^c} + \dfrac{q_e(\sigma + q_p)}{2^b}$.

Since $c = \kappa$ here, note that this bound is the same as in the previous case.

Proof. 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_6 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 $\Pr(\Omega_i \cap \mathsf{domain}(\mathsf{P}_3) \neq \emptyset \mid \rho_6 = \rho) \leq \frac{\rho(\sigma + q_p)}{2^c}$. So, using the expectation of ρ_6, and using Remark 2.12, we have

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

□

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

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

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 $\mathsf{bad}_9 = 1$ if

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

  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 including the $t'_i \neq 1$ case, we have

  Lemma 4.15. $\Pr(\mathsf{bad}_9 = 1) \leq \dfrac{\mathsf{mcoll}(\sigma + q_p, 2^\tau) q_d}{2^\kappa} + \dfrac{q_d(\sigma + q_p)}{2^{c+\tau}}$.

- If $\mathsf{X}'_i \notin \mathsf{domain}(\mathsf{P}_4)$, let $y_i = \lfloor \mathsf{Y}'_i \rfloor_\tau$. We set $\mathsf{bad}_{10} = 1$ if there exists $i \in [q_d]$ such that $y_i \oplus K_2 = \mathsf{T}'_i$. As before, by the randomness of y_i, we have

  Lemma 4.16. $\Pr(\mathsf{bad}_{10} = 1) \leq \dfrac{q_d}{2^\tau}$.

Again, let bad denote the union of all bad events, namely $\cup_{i=1}^{10} \mathsf{bad}_i$. We have shown that

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

If all these bad events do not occur, then all the decryption queries are correctly rejected for the injective partial function P_4.
Let $\mathsf{P}_{\mathsf{fin}} := \mathsf{P}_4 \cup ((\mathsf{X}'_i, \mathsf{Y}'_i)_{i \in [q_d]})$. In the offline transcript, we provide all the input-outputs of $\mathsf{P}_{\mathsf{fin}}$. Then,

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

By defining the good transcript, and using H-coefficient technique as above, we complete the proof for the $\kappa = c$ case.

4.5 Summary

In this chapter, we have proved a bound for the AEAD mode of Ascon, the winner of the recently concluded NIST LWC competition. This mode follows a Duplex type of construction. Notably, the double-keyed finalisation (i.e., inclusion of key XOR operations before and after the permutation call during the tag generation/verification phase) of Ascon allows us to derive a bound in the following order:

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

where T is the time complexity and D is the data complexity of the adversary. Further, when $\tau \leq \kappa \leq c$, the obtained security bound can be reduced to

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

We have also shown that the bound is tight. We would like to again emphasise that our analysis cannot be directly applied to general Sponge constructions without the double-keyed tag generation/verification protocol.

Chapter 5

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

The Ascon cipher suite has recently become the preferred standard in the NIST Lightweight Cryptography standardisation 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. In this chapter, we eliminate these constraints and provide a comprehensive security analysis of the Ascon AEAD mode in the multi-user setting. 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.

5.1 Introduction

In Chapter 4, we conduct the first dedicated security analysis of the Ascon mode. We leverage the double-keyed initialisation and finalisation of Ascon, demonstrating the removal of the term $DT/2^c$ for the Ascon AEAD. We achieve a bound of the order

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

We also demonstrate that the bound is tight. However, this bound is only attainable in the single-user nonce-respecting setting, where nonces cannot be reused across encryption queries.

In a concurrent work [94], Lefevre and Mennink also present a dedicated security analysis of the Ascon mode. 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 initialisation and finalisation 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 $\frac{q_d T}{2^c}$.

5.1.1 Contributions

In this chapter, 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 our single-user result, 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 [94]. We also show that the achieved bound is tight. 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 bounds are 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 [94], where the authors could not overcome the hurdle of $DT/2^c$ under any attack setting.

Setting	**Security**	**Chapter 4**	[94]	**This chapter**
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	-	-	$\frac{\mu T}{2^\kappa}+\frac{DT}{2^b}+\frac{D^2+T}{2^c}$

Figure 5.1: 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.

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 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 resilience against key recovery attacks that utilise 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, labelled 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 authenticity 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 Chapter 4 and [94] can be found in Fig. 5.1.

5.1.2 Chapter Outline

In section 5.2, we present the extension of our results on Ascon AEAD mode in the multi-user setting. 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 5.3, we present the authenticity security of Ascon in the nonce misuse setting. In Section 5.4, we define LK-Ascon, a variant of Ascon and state the security of LK-Ascon, along with a proof outline. In Section 5.5, we present the proofs of the theorems, and finally summarise the chapter in Section 5.6.

5.2 Extending the Result to the Multiuser Setting

In this section, we extend our result (Theorem 4.3) to the multiuser (mu) setting. We assume $\kappa \leq c$ directly here. Let the number of users be μ. The description of the AEAD mode is exactly the same as the su setting, except the μ users have independent keys $K_1, K_2, \ldots K_\mu$ and the encryption and decryption queries of the adversary are user-specific, i.e. (U_i, N_i, A_i, M_i) and $(U_i', N_i', A_i', C_i', T_i')$ respectively, where U_i and U_i' correspond to the user number. In this setting, we have the following theorem:

Theorem 5.1. *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:*

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

5.2.1 Interpretation of Theorem 5.1

First, note that when $\mu = 1$ (corresponding to the su setting), the aforementioned bound aligns with the security bound established in Chapter 4. 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.

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

As attacks for the remaining terms arere already shown in Chapter 4, our bounds are tight.

5.2.3 Proof Overview of Theorem 5.1

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. The 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 an 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 $\mathsf{P}_{\mathsf{Enc}}$.
3. Randomly extend P to P_1 by setting input-outputs for decryption queries. Set $\mathsf{P}_2 := \mathsf{P}_1 \cup \mathsf{P}_{\mathsf{Enc}}$.
4. Finally, sample keys $K_1, \ldots, K_\mu$. Set input-output pairs for initialisation first, and then move on to the finalisation 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, and
- decryption queries are not rejected.

Bounding the bad events concludes the proof. A detailed proof of the theorem is presented in Section 5.5.1.

5.3 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 5.2. *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:*

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

This bound is the similar to that of Theorem 5.1, 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 Section 5.5.2. To show that the bound is tight, we now demonstrate a forgery that establishes the $\sigma_e^2/2^c$ bound.

Forgery against Ascon authenticity in the nonce misuse setting.

Here, we show an online $c/2$-bit forgery against the authenticity of Ascon.

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 $(C[i], T[i]) = \mathsf{Enc}(K, N, A[i], M)$.
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 $(C'[i], T'[i]) = \mathsf{Enc}(K, N, A[i], M[i])$.
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 $(C^*[i_1], T^*[i_1]) = \mathsf{Enc}(K, N, A[i_1], M^*[i_1])$.

8. Make a decryption query $\mathsf{Dec}(K, N, A[i_2], 1^r, T^*[i_1])$. One can expect that the forgery succeeds with high probability.

5.4 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 5.2.1, where the term $\frac{\mu q_p}{2^\kappa}$ emerges as the dominant factor in Theorem 5.1, 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 generalised 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 $\kappa + \nu \leq b$ and $\kappa \leq c$, 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 $\tau \leq \kappa < b$, 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), modelled to be the random permutation while we analyse its security.

Remark 5.3. While we define the mode for any $\tau \leq \kappa < b$, 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 $\tau \leq \kappa$ 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) initialisation, (ii) nonce, associated data and message processing, and (iii) tag generation, run sequentially.

INITIALISATION. 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 initialisation 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

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

TAG GENERATION. Finally, we compute

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

The LK-ASCON AEAD returns (C, T). A pictorial description of the encryption algorithm of LK-ASCON can be found in Fig. 5.2.

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 $V_0 := \textsc{Init}^{\pi}(K)$.

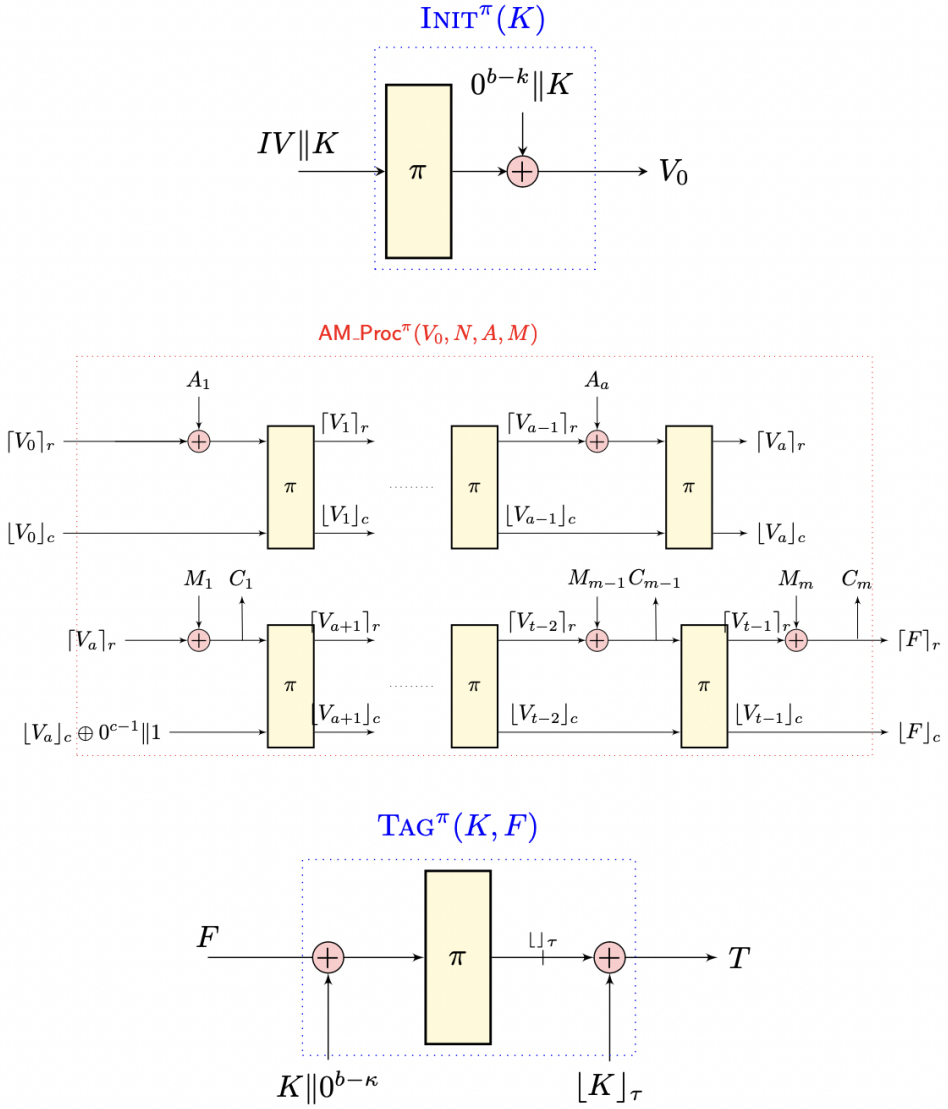

FIGURE 5.2: Encryption in LK-ASCON AEAD. The difference with conventional ASCON is that there is no nonce at the input of INIT. The nonce is parsed with the associated data, and fed at the AM_Proc step. Also, the key addition at the input of TAG is $K\|0^{b-\kappa}$, meaning even if $\kappa \leq c$, it is xored at the rate part.

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

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

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

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

Remark 5.4. 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 emphasise the similarity in the processes of the two AEAD modes.

5.4.1 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 5.5. *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:*

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

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 5.6. *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:*

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

This shows that like Ascon, LK-Ascon too maintains authenticity under nonce-misuse. In the next section, we give a proof overview of Theorem 5.5. The proof of Theorem 5.6 is a straightforward extension, and we outline the sketch in Section 5.5.4.

5.4.2 Proof Overview of Theorem 5.5

The proof follows the structure outlined in Section 5.2.3 for Ascon. However, there are notable differences, summarised 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 finalisation phase. Otherwise, if $\kappa < r$, the final ciphertext block is only partially masked.

A detailed proof is provided in Section 5.5.3.

5.5 Proofs

5.5.1 Proof of Theorem 5.1

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

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

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

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

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

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

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

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

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

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

Bad Event ($\mathsf{P}_{\mathsf{Enc}}$ 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 5.8. $\Pr(\mathsf{bad}_2 = 1 \vee \mathsf{bad}_3 = 1) \leq \dfrac{\sigma_e^2}{2^b}$.

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 randomised extension. Later, we set a bad event if it is not disjoint (both from the domain and the range) with $\mathsf{P}_{\mathsf{Enc}}$. This would ensure the compatibility of $\mathsf{P}_1 \sqcup \mathsf{P}_{\mathsf{Enc}}$ (where P_1 is the randomised 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 $\mathsf{Rand\_Extn}^P(\mathsf{V}'_{i,p_i}, \mathsf{C}'_{i,p_i-a'_i+1} \| \dots \| \mathsf{C}'_{i,c_i-1})$ 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 $p_i = t_i - 1$:
   - $\mathsf{V}'_{i,a'_i+c_i-1} := \mathsf{V}_{j,a'_i+c_i-1}$, 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 randomised 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 5.9. $\Pr(\mathsf{bad}_4 = 1) \leq \dfrac{\sigma_d(q_p + \sigma_d)}{2^b}$.

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

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

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

Lemma 5.10. $\Pr(\mathsf{bad}_5 = 1) \leq \dfrac{\mathsf{mcoll}(\sigma_e, 2^r) \times (\sigma_d + q_p)}{2^c}$.

Proof. Let ρ_1 (and ρ_2) denote the multicollision on the values of $\lceil x \rceil_r$, for all $x \in \mathsf{domain}(\mathsf{P}_{\mathsf{Enc}})$ (and for all $x \in \mathsf{range}(\mathsf{P}_{\mathsf{Enc}})$ respectively). Then, by the randomness of the randomised extension process and randomised 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 2.11 and Remark 2.12, 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 $\mathsf{P}_2 := \mathsf{P}_1 \sqcup \mathsf{P}_{\mathsf{Enc}}$ 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 5.11. $\Pr(\mathsf{bad}_6 = 1) \leq \dfrac{\mathsf{mcoll}(q_e, 2^\tau) q_d}{2^c}$.

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 2.11, 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 5.12. $\Pr(\mathsf{bad}_7 = 1) \leq \dfrac{\mu^2}{2^\kappa}$.

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

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

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 initialisation 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 $\mathsf{P}_{\mathsf{tag}} = ((\mathsf{X}_i, \mathsf{Y}_i)_{i \in [q_e]})$.

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

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 2.12, 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 2.12, 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 5.15. $\Pr(\mathsf{bad}_{10} = 1) \leq \dfrac{\mathsf{mcoll}(\sigma + q_p, 2^\tau) q_d}{2^\kappa} + \dfrac{q_d(\sigma + q_p)}{2^{c+\tau}}$.

- $\mathsf{X}'_i \notin \mathsf{domain}(\mathsf{P}_4)$. 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 5.16. $\Pr(\mathsf{bad}_{11} = 1) \leq \dfrac{q_d}{2^\tau}$.

Let bad denote the union of all bad events, namely $\cup_{i=1}^{11}\mathsf{bad}_i$. By Lemmas 5.7 through 5.16, we have shown that

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

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

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

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

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,

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

By using the H-coefficient technique, we complete the proof of Theorem 5.1.

5.5.2 Proof of Theorem 5.2

We highlight the parts where it differs from the proof of Theorem 5.1. 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 $\mathsf{P}_{\mathsf{Enc}}$ 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 finalisation, 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

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

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

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

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

5.5.3 Proof of Theorem 5.5

The proof structure follows the proof of Theorem 5.1. 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

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

denote the extended real world transcript. For any real world realisable transcript θ,

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

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

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

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

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.
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 $a_i \geq 1$. Since the adversary is nonce-respecting, if there exists $j < i$ such that $\mathsf{u}_i = \mathsf{u}_j$, then $\mathsf{N}_i \neq \mathsf{N}_j$. Let $k = \lceil \frac{|\mathsf{N}|}{r} \rceil$. Then, this implies $(\mathsf{A}_{i,1}, \ldots, \mathsf{A}_{i,k}) \neq (\mathsf{A}_{j,1}, \ldots, \mathsf{A}_{j,k})$. Let $k' \leq k$ be the first index such that $\mathsf{V}_{i,k'} \neq \mathsf{V}_{j,k'}$. 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 $\mathsf{P}_{\mathsf{Enc}}$ 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, $\mathsf{P}_{\mathsf{Enc}}$ would be an injective partial function if $\mathsf{V}_{i,j}$'s are all distinct.

Bad Event ($\mathsf{P}_{\mathsf{Enc}}$ 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 5.18. $\Pr(\mathsf{bad}_2 = 1 \vee \mathsf{bad}_3 = 1) \leq \dfrac{\sigma_e^2}{2^b}$.

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 $\mathsf{Rand\_Extn}^P(\mathsf{V}'_{i,a_i} \oplus 0^*1, \mathsf{C}'_{i,1} \| \dots \| \mathsf{C}'_{i,c_i-1})$ 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 $\mathsf{xorRand\_Extn}^P(\mathsf{V}'_{i,p_i}, (\mathsf{A}'_{i,p_i+1}, \dots, \mathsf{A}'_{i,a'_i}))$ 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 $\mathsf{Rand\_Extn}^P(\mathsf{V}'_{i,a_i} \oplus 0^*1, \mathsf{C}'_{i,1} \| \dots \| \mathsf{C}'_{i,c_i-1})$ 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

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

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

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

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

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

Lemma 5.20. $\Pr(\mathsf{bad}_5 = 1) \leq \dfrac{\mathsf{mcoll}(\sigma_e, 2^r) \times (\sigma_d + q_p)}{2^c}$.

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 $\mathsf{P}_2 := \mathsf{P}_1 \sqcup \mathsf{P}_{\mathsf{Enc}}$ 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 5.21. $\Pr(\mathsf{bad}_6 = 1) \leq \dfrac{\mathsf{mcoll}(q_e, 2^\tau) q_d}{2^c}$.

Now, we sample the keys $K_1, K_2, \ldots K_\mu \xleftarrow{\$} \{0,1\}^\kappa$. 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 5.22. $\Pr(\mathsf{bad}_7 = 1) \leq \frac{\mu^2}{2^\kappa}$.

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

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

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 $\mathsf{P}_{\mathsf{tag}} = ((\mathsf{X}_i, \mathsf{Y}_i)_{i \in [q_e]})$.

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 5.24. $\Pr(\mathsf{bad}_9 = 1) \leq \dfrac{\omega_{r,\kappa}(\sigma + q_p)}{2^b}$, *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 2.12, 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 5.25. $\Pr(\mathsf{bad}_{10} = 1) \leq \frac{\mathsf{mcoll}(\sigma+q_p, 2^\tau) q_d}{2^\kappa}$.

  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 2.13, 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 5.26. $\Pr(\mathsf{bad}_{11} = 1) \leq \dfrac{q_d}{2^\tau}$.

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

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

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

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.

5.5.4 Proof of Theorem 5.6

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 5.6 differs from the proof of Theorem 5.5 are the parts where the proof of Theorem 5.2 differs from Theorem 5.1. 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 5.5. 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 $p_i = \max_{j<i} p_{i,j}$.

5.6 Summary

In this chapter, we derive a multi-user security bound for the Ascon AEAD mode, the winner of the recently concluded NIST LWC competition. This mode follows a Duplex 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 using Grover's algorithm. Like Ascon, LK-Ascon also maintains its authenticity security in the nonce misuse setting.

Chapter 6

Universal Context Commitment without Ciphertext Expansion

An ongoing research challenge in symmetric cryptography is to design an authenticated encryption (AE) with a commitment to the secret key or preferably to the entire context. One way to achieve this is to use a transform on an existing AE scheme, if possible with no output length expansion. At EUROCRYPT'22, Bellare and Hoang proposed the HtE transform, which lifts key-commitment to context-commitment. In the same year at ESORICS'22, Chan and Rogaway proposed the CTX transform, which works on any AE scheme where the tag is not required for decryption. However, for AE schemes which are not key-committing to begin with and which use the tag for decryption, no such transform exists till date. The latter category encompasses all AE schemes based on the design paradigms SIV, MAC-then-Encrypt, and Encode-then-Encipher. In this work, we propose PACT, a transform to convert any authenticated encryption into a context-committing one without any output length expansion. In addition, PACT preserves both nonce-respecting and nonce-misuse security of the legacy AE scheme. PACT achieves this with only one call to a collision-resistant unkeyed hash function and one call to a block cipher. In addition, we propose a lighter transform comPACT which gives a context-committing nonce-respecting AE scheme.

6.1 Introduction

Authenticated Encryption (AE) plays a pivotal role in symmetric cryptography by facilitating both encryption and authentication of plaintext simultaneously. Frequently, AE also offers the ability to authenticate additional data, which is

transmitted without encryption unlike the plaintext. This scenario is commonly known as Authenticated Encryption with Associated Data (AEAD). Contemporary AE schemes predominantly rely on nonces [109], which in its most basic form require users to provide an additional nonce that must be unique for each encryption. Such schemes are known as Unique Nonce AE schemes (UNAE). In contrast, Deterministic Authenticated Encryption (DAE) [110] is used in scenarios where leveraging existing entropy or redundancy in inputs makes more sense, thereby avoiding the overhead of nonces. This is particularly advantageous, for instance, when encrypting cryptographic keys.

Over time, the comprehension of AE security has undergone several revisions. Misuse-Resistant AE (MRAE) [110] was introduced to ensure that nonce repetitions do not compromise the security of the scheme, provided that a combination of nonce, associated data, and message values is not repeated. Subsequently, AE security was broadened to include scenarios where unverified plaintext are released [5]. Further developments led to the notion of robust AE [78] which allows the user to specify the ciphertext expansion, i.e., how much longer the ciphertext is compared to the plaintext. In recent years, a diverse array of leakage-resilient AEAD notions [8, 21, 107] have spawned. More recent works have emphasised that nonce and associated data present in plaintext could potentially divulge crucial information, such as identifying sessions and users, thereby advocating for anonymous AE [39] and AE2 [14].

Despite the existence of AE schemes with desirable features nonetheless, the primary focus has consistently centered on ensuring confidentiality and authenticity across different scenarios. Both the CAESAR competition for authenticated encryption [45] and the recent NIST lightweight cryptography (LWC) standardisation process [102] underscored the importance of AE schemes that guarantee security by delivering both confidentiality and authenticity.

However, a recent wave of attacks has demonstrated the necessity for a reevaluation of our conception of a secure AE scheme. The Facebook message franking attack [57] show how to break Facebook's message franking scheme, which means a malicious user can send an objectionable image to a recipient but that recipient cannot report it as abuse. Vulnerabilities have also been observed in "Subscribe with Google" [2] and Shadowsocks proxy servers [90]. In the latter, adversaries can build a practical partitioning oracle attack that quickly recovers passwords from the servers. Indeed, all these attacks can be attributed to a common issue: the presence of ciphertexts that decrypt correctly under multiple keys. This

vulnerability persists despite efforts to ensure confidentiality and authenticity, underscoring the necessity for an additional security concept.

Committing Authenticated Encryption. In pursuit of this goal, commitment security [9] was introduced, necessitating that every ciphertext serves as a commitment to the key (CMT_k) or even to the entire context (CMT). The latter notion represents the strongest form and is formalised through the following security game: The adversary provides two tuples (K, N, A, M), (K', N', A', M'), each comprising a key, nonce, associated data, and message. The adversary wins if their contexts differ, i.e., $(K, N, A) \neq (K', N', A')$, and if $\Pi.\mathsf{Enc}(K, N, A, M) = \Pi.\mathsf{Enc}(K', N', A', M')$ holds, where Π denotes the scheme under scrutiny.
The aforementioned attacks underscore the serious repercussions of employing non-committing authenticated encryption. Given the likelihood of additional undiscovered attacks, addressing this issue becomes imperative. In this regard, it is essential to subject AE schemes used in practice to scrutiny regarding committing security. This evaluation process has already commenced, with several commonly used AE schemes (such as GCM, ChaCha20-Poly1305, SIV, CCM, EAX, OCB3) undergoing examination [2, 9, 93]. Regrettably, a majority of these schemes have been found lacking in achieving committing security.
To attain committing security, a viable strategy entails constructing authenticated encryption schemes from the ground up, ensuring they inherently offer commitment. This approach alleviates worries about committing attacks when these schemes are deployed in diverse protocols. Nonetheless, a more efficient method involves devising techniques to convert existing AE schemes, which we call legacy AE schemes into commitment-secure ones, preferably meeting the criteria of CMT security.

Transforming Legacy AE Schemes into CMT **Secure Schemes.** Several generic transforms or enhancements of legacy AE into committing AE schemes have been suggested. Initially, these transforms focused solely on achieving CMT_k security, but our interest lies in transforms that achieve CMT security. Bellare and Hoang [9] introduce the first generic transform, known as HtE (Hash-then-Encrypt), which converts a CMT_k scheme into a CMT scheme. Fig. 6.1 gives the complete specification of HtE. HtE imposes no ciphertext overhead and preserves both UNAE and MRAE security. While this serves as a promising starting point, it is crucial for an AE scheme to also achieve CMT_k security. To address this, the same paper introduces two transforms: UtC (UNAE-then-Commit) and RtC

$\mathsf{HtE}[\Pi, \mathcal{H}].\mathsf{Enc}(K, N, A, M)$
1: $K^* \leftarrow \mathcal{H}(K, N, A)$
2: $(C, T) \leftarrow \Pi.\mathsf{Enc}(K^*, N, \epsilon, M)$
3: **return** (C, T)

$\mathsf{HtE}[\Pi, \mathcal{H}].\mathsf{Dec}(K, N, A, C, T^*)$
1: $K^* \leftarrow \mathcal{H}(K, N, A)$
2: $X \leftarrow \Pi.\mathsf{Dec}(K^*, N, \epsilon, C, T)$
3: **return** X

FIGURE 6.1: Specification of HtE. HtE.Enc and HtE.Dec are the encryption and decryption algorithms of HtE respectively. Π is the input AE scheme, and Π.Enc and Π.Dec are its encryption and decryption algorithms respectively. K, N, A, M, C, and T denote the key, the nonce, the associated data, the message, the ciphertext and the tag, respectively. $\mathcal{H}$ is a hash function whose output-size is equal to the key-size of Π.

(MRAE-then-Commit), which transform UNAE and MRAE schemes, respectively, into CMT_k schemes. However, both of these transforms result in ciphertext expansion. Consequently, any nonce-based AE can be transformed into a CMT scheme using either UtC+HtE or RtC+HtE, but both schemes entail ciphertext overheads. As a separate contribution, the authors also suggest modifications for GCM and AES-GCM-SIV to enhance their CMT_k security. Specifically, they introduce CAU and CAU-SIV as generalisations of GCM and AES-GCM-SIV respectively, and propose adjustments to yield CAU-C1 and CAU-SIV-C1 respectively. Unlike UtC and RtC, these modifications do not lead to any ciphertext expansion, allowing the application of HtE to obtain CMT schemes without ciphertext expansion.

Chan and Rogaway [40] introduce the CTX transform, marking the first generic transform that avoids ciphertext expansion. CTX transforms an AE scheme Π into a CMT scheme CTX[Π], provided that the encryption algorithm of Π can be decomposed into two independent algorithms Π.Enc and Π.Auth, where Π.Enc produces the ciphertext C and Π.Auth generates the tag T. Fig. 6.2 gives the complete specification of CTX. While the authors argue that commonly used UNAE schemes like GCM and OCB meet these structural requirements, most frequently employed MRAE schemes such as SIV and its variants like GCM-SIV, NSIV, schemes employing the Mac-then-Encrypt paradigm or the Encode-then-Encipher paradigm, and several DAE schemes fall beyond the scope of CTX.

In the third NIST workshop on block cipher modes of operation 2023, Bellare et al. [11, 12] proposed CTX+ transform[1], offering a swifter alternative to the CTX transform. Unlike CTX, which involves processing the associated data twice - once for encryption and once for hashing, CTX+ transform demonstrates that using the associated data solely for hashing suffices to achieve CMT security. Although this enhancement renders CTX more efficient, the CTX+ transform is only applicable to

[1]The same transform has been formalised as CTY in [10].

$\mathsf{CTX}[\Pi, \mathcal{H}].\mathsf{Enc}(K, N, A, M)$

1: $C \leftarrow \Pi.\mathsf{Enc}(K, N, A, M)$
2: $T \leftarrow \Pi.\mathsf{Auth}(K, N, A, M)$
3: $T^* \leftarrow \mathcal{H}(K, N, A, T)$
4: **return** (C, T^*)

$\mathsf{CTX}[\Pi, \mathcal{H}].\mathsf{Dec}(K, N, A, C, T^*)$

1: $M \leftarrow \Pi.\mathsf{Dec}(K, N, A, C)$
2: $T \leftarrow \Pi.\mathsf{Auth}(K, N, A, M)$
3: **If** $T^* \neq \mathcal{H}(K, N, A, T)$
4: **return** $\perp$
5: **return** M

FIGURE 6.2: Specification of CTX. CTX.Enc and CTX.Dec are the encryption and decryption algorithms of CTX respectively. Π is the input AE scheme. Π.Enc and Π.Auth are the algorithms of Π to generate the ciphertext and the tag respectively; and Π.Dec is the decryption algorithm of Π. K, N, A, M, C, and T denote the key, the nonce, the associated data, the message, the ciphertext and the tag, respectively. $\mathcal{H}$ is a hash function whose output-size is equal to the tag-size.

the same limited range of AE schemes as CTX. During the same workshop, Naito et al. [99, 100] introduced a CMT transform labelled KIVR, which purportedly avoids ciphertext expansion by leveraging redundancy inherent in the plaintext. While this transform could prove advantageous for protocols where redundancy in plaintexts is commonplace, it deviates from assumptions typically encountered in classical settings.

Challenges. To our knowledge, there is currently no CMT transform that meets both universality, meaning it can be applied to any legacy AE scheme, and output-length-preserving, meaning it avoids ciphertext expansion. We stress the significance of maintaining output length for any CMT transform from a practical standpoint, as preserving ciphertext-size is crucial for compatibility with existing hardware, databases, or communication protocols.

6.1.1 Contributions

In this chapter, we propose a new transform, which we call PACT. To give a brief overview, we start with an AE scheme Π that produces a ciphertext C and a tag T using $\Pi.\mathsf{Enc}(K, N, A, M)$. PACT modifies the tag T to a new tag T^* by hashing (K, N, A) using an unkeyed hash function and then making a block cipher call with the hash output as the key and T as the input.

At an overview level, PACT differs from transforms like HtE and CTX in the fact that PACT includes a block cipher call on top of the hash call. This design choice ensures that the modification of the tag from T to T^* is reversible, preserving the original tag T. This allows for decryption using $\Pi.\mathsf{Dec}(K, N, A, C, T^*)$ for all Π, even if their decryption algorithm utilises T.

To the best of our knowledge, this transform is the first of its kind to achieve the following objectives:

1. CMT *transform*, i.e., converts an AE scheme into a CMT secure AE scheme.
2. *Universal transform*, i.e., it is applicable to any AE scheme.
3. *Output-length-preserving transform*, i.e., it does not incur any additional output length expansion compared to the original AE scheme.

To start with, it is worth noting that HtE and CTX are the sole existing transforms that satisfy the first and third objectives. Regarding the second criterion, we observe that numerous AE schemes are incompatible with CTX. For instance, schemes following the SIV paradigm or the Mac-then-Encrypt paradigm utilise the tag for encryption, rendering the decryption function dependent on the tag, thus precluding the application of CTX. Similarly, AE schemes adhering to the Encode-then-Encipher paradigm necessitate the tag for decryption, as the output of enciphering contains both the ciphertext and the tag, making them unsuitable for CTX as well. Additionally, during our survey, we observed that certain AE schemes possess an MRAE counterpart to their UNAE design, with the MRAE designs typically employing the tag for encryption, thereby rendering them unsuitable for CTX. Moreover, among these AE schemes, many are insecure even against a constant-time CMT_k adversary, thus ruling out the applicability of HtE. Consequently, PACT emerges as the sole CMT transform applicable to these schemes. It is noteworthy that unlike PACT, both HtE and CTX rely on the Random Oracle or PRF assumption for the underlying hash function. In contrast, we solely rely on the collision-resistance property of the hash and utilise a single block cipher call, which we model as an ideal cipher for security proof.
For the concrete security analysis of PACT, we introduce a new security notion for authenticated encryption schemes, termed the "ciphertext collision advantage". We demonstrate that to achieve a robust CMT security bound for PACT, it is essential for the legacy AE scheme to possess a small ciphertext collision advantage. Conversely, we contend that an AE scheme deemed "good" should inherently exhibit a small ciphertext collision advantage, roughly of the order $(q^2/2^{|S|} + q^2/2^{|T|})$, where the number of AE computations by the ciphertext collision adversary is q, and $|S|, |T|$ refer to the state-size (e.g. block-size for block cipher-based AEs) and tag-size of the AE respectively. Furthermore, we substantiate this assertion by examining several classes of AE schemes to confirm that they indeed possess a small

ciphertext collision advantage. It is important to note that we do not view this requirement as a special condition solely for a legacy AE scheme to be compatible with PACT.

Once we give detailed CMT and AE security analysis of PACT, we compare it with HtE and CTX to show that PACT is not heavier than its counterparts from the design point of view. To underline that, we also suggest a concrete instantiation of PACT.

While PACT is universally applicable, we also propose a more efficient version, termed comPACT. It applies to the cases where we do not need to preserve the MRAE security of the legacy AE, i.e., it remains only UNAE secure after applying comPACT. The difference lies in how associated data is handled: in PACT, it is processed by both the legacy AE and the hash function, whereas comPACT only involves the associated data for the hash function, leaving the legacy AE with empty associated data. The CMT security of PACT extends to comPACT, with similar security analyses for privacy and authenticity for UNAE schemes. We also compare comPACT with CTX+, and give an attack to show that neither of them is MRAE-secure.

Universal CMT_k **Transform.** So far most of the practical commitment attacks on the AE schemes which are used in important applications have been CMT_k attacks which motivated the cryptography community to explore this particular research direction in the first place. Surprisingly all the generic length-preserving transforms so far are CMT transforms. One might intuitively think that a good exercise could be to try and design a generic length-preserving CMT_k-only transform which prevents the practical CMT_k attacks and potentially has a lighter design as compared to the generic CMT transforms. We tried to explore this direction and found some intuition about why it might be a little difficult.

A general underlying theme of all the CMT transforms proposed so far is to use a collision-resistant function on the key-nonce-associated data tuple, which is very much in line with the definition of CMT security itself. Now, if we try to extrapolate this idea to design a CMT_k-only transform, we do not need the collision-resistant function, and can simply use the key only. Unfortunately, however, this strategy does not work in most of the AE schemes. The issue is that once an input-output tuple is fixed, the associated data can often be used to adjust an entire new input tuple and force the output to be equal to the previous output. As

a result, one needs a somewhat novel idea to achieve a potentially lighter CMT_k-only transform. In the following we illustrate this with the concrete example of Deoxys-II [83, 84]. The encryption algorithm of Deoxys-II is given in Fig. 6.3.
Suppose a CMT_k adversary of Deoxys-II obtains a (K, N, A, M, C, T) tuple by a Deoxys-II computation. Then it queries (K, T) to the block cipher E to obtain T^*. Then it fixes some (K', N') with $K' \neq K$ and makes a (K', T^*) query to E^{-1} to obtain T'. Now, given K', N', C, T', the adversary can compute an unverified message M'. Since the adversary has yet to choose an associated data, it can just choose one block associated data A' such that M' is verified. In this way, the adversary generates two tuples (K, N, A, M, C, T^*) and (K', N', A', M', C, T^*) of the CMT_k-only transform and breaks its CMT_k security.

6.1.2 Related Works

The research in the field of committing encryption goes back to 2003 with Gertner and Herzberg [68], who considered the problem in both the symmetric and asymmetric settings. Abdalla et al. [1] gave definitions for what they termed robustness. This work was in the asymmetric setting and required an adversary to produce a ciphertext that validly decrypts under two different keys. Their notion encompassed keys that are honestly generated. Later, Farshim et al. [64] pointed out that, for some applications, robustness against adversarially chosen keys is critical. They strengthened Abdalla et al.'s notion to address this observation. Farshim et al. [65] contextualised Abdalla et al.'s robustness in the AE setting, initialising the study of what we call committing AE. Shortly after, Grubbs et al. [73] defined compactly committing AE with the goal of constructing schemes that support message franking. Dodis et al. [57] also targeted message franking and further develop the definitions from [73] by introducing encryptment as a core component of compactly committing AE, which was later explored further in [75–77]. Albertini et al. [2] observed the possibility of mitigating the attacks described in [73] and [57] under a weak form of commitment.
Bellare and Hoang [9] introduced new encryption-based and decryption-based commitment security notions, which also incorporate the number of inputs the adversary can produce for a single output. Chan and Rogaway [40] introduced another notion, which incorporates the degree of control the adversary has on the key. Other contributions in [9] and [40] have already been discussed. Menda et al. [93] introduced the notion of context discoverability and investigated discoverability attacks on popular schemes. Len et al. [90] demonstrated password-recovery attacks

on non-committing password-based AE schemes. Chen et al. [43] investigated both key-commitment and context-commitment security of a few AE schemes that are built from a variable-length tweakable cipher or wide block cipher (WBC) via Encode-then-Encipher (EtE) approach [18]. Krämer et al. [89] investigated the commitment security of the finalists of the NIST Lightweight Cryptography Standardisation Process [102]. Daemen et al. [46] designed sponge-based commitment secure AE modes from SHAKE and TurboSHAKE. Degabriele et al. [50] proved commitment security of SpongeWrap. Struck and Weishäupl [118] designed leakage resilient commitment secure AE schemes, but Dhar et al. [53] showed that most leakage resilient AE schemes in practise are already committing.
Bellare et al. [11, 12] proposed a new committing AE scheme with no nonce or associated data which works on very short messages. Bellare and Hoang later formalised it in [10]. They first define a ciphertext-shortening transform SC. The SC transform takes a collision-resistant invertible PRF as an additional input. However, SC is neither committing, nor does it preserve standard AE security. Hence, they compose SC and CTY, and show that the resulting transform is both AE secure and committing. However, since CTY suffers the same weaknesses as CTX, the resulting transform is not universal.

6.1.3 Outline of the Chapter

In Section 6.2, we introduce the notion of ciphertext collision advantage for an AE scheme, and explain why we believe it should be small for all good AE schemes. In Section 6.3, we introduce the PACT transform, and state our security results on the commitment security, privacy, and authenticity of PACT; we also introduce the lighter transform comPACT suitable for nonce-respecting AE schemes. In Section 6.4, we illustrate the applications of PACT and comPACT by exhibiting some well-known AE schemes for which we have no known generic context-committing transform without ciphertext expansion. In Section 6.5, we provide the security proofs for the results from Section 6.3. Finally, in Section 6.7 we summarise our work.

6.2 Ciphertext Collision in AE Schemes

In this section, we introduce a concept related to AE schemes called the "ciphertext collision". In essence, for any two key-nonce-associated data tuples (K_1, N_1, A_1) and (K_2, N_2, A_2), and a tag T_1, a ciphertext collision in a scheme Π refers to

the situation where upon receiving a randomly generated tag T_2 one obtains two messages M_1 and M_2 such that for some ciphertext C it holds that

$$\Pi.\mathsf{Enc}(K_1, N_1, A_1, M_1) = (C, T_1), \text{ and } \Pi.\mathsf{Enc}(K_2, N_2, A_2, M_2) = (C, T_2).$$

This is typically unnecessary in AE design scenarios where adversarial access to the key is not assumed. However, when an adversary can choose the key, it can launch commitment attacks using the ciphertext collision. Therefore, from a committing security standpoint, our goal is to ensure that the probability of ciphertext collision in an AE scheme is minimal. We first formally define the relevant terms.

6.2.1 Ciphertext Collision Advantage

Let Π be an AE scheme with tag space $\mathcal{T}$. In the course of a *ciphertext collision security game*, an adversary $\mathcal{A}$ chooses q_1 key-nonce-associated data-tag tuples $\{(K_i^0, N_i^0, A_i^0, T_i^0) \mid i \in [q_1]\}$. For each $i \in [q_1]$, it further chooses m_i key-nonce-associated data tuples $\{(K_i^j, N_i^j, A_i^j) \mid j \in [m_i]\}$, where $m_1 + \cdots + m_{q_1} = q_2$. For each $i \in [q_1], j \in [m_i]$, and an $\epsilon > 0$, a tag T_i^j is sampled from a distribution which guarantees that for any $c \in \mathcal{T}, \Pr[T_i^j = c] \leq \epsilon$. $\mathcal{A}$ can make its choices adaptively and in any order, based on previously sampled values of T_i^j.
At the end of the game, $\mathcal{A}$ outputs a couple of messages M_1 and M_2, and three indices i, j_1 and j_2 with $0 \leq j_1 < j_2$, and wins if there exists a ciphertext C such that $\Pi.\mathsf{Enc}(K_i^{j_1}, N_i^{j_1}, A_i^{j_1}, M_1) = (C, T_i^{j_1})$ and $\Pi.\mathsf{Enc}(K_i^{j_2}, N_i^{j_2}, A_i^{j_2}, M_2) = (C, T_i^{j_2})$. The *ciphertext collision advantage* of $\mathcal{A}$ is defined as

$$\mathbf{Adv}_{\Pi}^{\mathrm{CC}}(\mathcal{A}) := \Pr[\mathcal{A} \text{ wins}].$$

$\mathcal{A}$ is called a *(q_1, q_2, ϵ)-ciphertext collision adversary* of Π.
Here, we provide a rough intuition about what q_1 and q_2 represent. In Section 6.3, we define the PACT transform, which converts any AE scheme Π into a CMT-secure one. The PACT transform takes as input the AE scheme Π, a hash function $\mathcal{H}$, and a block cipher E. In the security analysis of PACT, we consider a (q_1, q_2, ϵ)-ciphertext collision adversary $\mathcal{A}$, where q_1 is the number of ideal cipher queries and q_2 is the number of Π computations made by $\mathcal{A}$. This is a very high-level intuition, with the detailed explanation provided in Section 6.3.
Our next objective is to demonstrate that for any AE scheme Π, the ciphertext collision advantage is not substantial. In the following subsections, we analyse the

$\mathsf{Deoxys\text{-}II}[\tilde{E}].\mathsf{Enc}(K, N, A = A[1]\|\ldots\|A[a], M = M[1]\|\ldots\|M[m])$

1: $\mathsf{Auth} \leftarrow 0$
2: **for** $i \in [1..a]$ **do**
3: $\quad \mathsf{Auth} \leftarrow \mathsf{Auth} \oplus \tilde{E}_K(0010\|i, A[i])$ ▷ $\tilde{E}$ is a tweakable block cipher (TBC)
4: $T \leftarrow \mathsf{Auth}$
5: **for** $j \in [1..m-1]$ **do**
6: $\quad T \leftarrow T \oplus \tilde{E}_K(0000\|j, M[j])$
7: $T \leftarrow T \oplus \tilde{E}_K(0100\|m, \mathsf{ozpad}(M[m]))$ ▷ ozpad is the 10^* padding if $|M[m]| < n$
8: $T \leftarrow \tilde{E}_K(0001\|0^4\|N, T)$
9: **for** $j \in [1..m]$ **do**
10: $\quad Z[j] \leftarrow \tilde{E}_K(1\|T \oplus j, 0^8\|N)$
11: $Z \leftarrow Z[1]\|\ldots\|Z[m]$
12: **return** $(C := M \oplus \lceil Z \rceil_{|M|}, T)$

FIGURE 6.3: Encryption of Deoxys-II

ciphertext collision advantage for two prominent AEs, Deoxys-II (in Section 6.2.2) and ASCON (in Section 6.2.3) in detail. Then, in Section 6.2.4, we argue why this advantage should not be large for any AE scheme. We would like to mention that bounding the ciphertext collision advantage of a general AE scheme depends on some equations intrinsic to the scheme, and thus providing an exact bound for a general scheme is tough. We have reviewed numerous AEs in practice, and our findings support our argument. The choice of Deoxys-II and ASCON for detail analysis is also justified in Section 6.2.4.

6.2.2 Ciphertext Collision Advantage of Deoxys-II

Let us first look at Deoxys-II [83, 84], an AE selected as the first choice for the "in-depth security" portfolio of the CAESAR competition. The encryption algorithm for Deoxys-II is given in Fig. 6.3. Let $\mathcal{A}$ be a (q_1, q_2, ϵ)-ciphertext collision adversary of Deoxys-II. We calculate $\mathbf{Adv}^{\mathrm{CC}}_{\mathsf{Deoxys\text{-}II}}(\mathcal{A})$. In the following analysis Π refers to Deoxys-II.

Let us fix i, j_1, j_2, the indices for which $\mathcal{A}$ finds a collision. First assume that the Π computation at index j_1 happens after the Π computation at index j_2. Let the Π computation at index j_2 yield

$$\Pi.\mathsf{Enc}(K_i^{j_2}, N_i^{j_2}, A_i^{j_2}, M_2) = (C, T_i^{j_2}).$$

Then, for $\mathcal{A}$ to win, we must obtain a message M_1 such that

$$\Pi.\mathsf{Enc}(K_i^{j_1}, N_i^{j_1}, A_i^{j_1}, M_1) = (C, T_i^{j_1}).$$

We divide this into two cases depending on whether $\mathcal{A}$ starts with C and tries to generate an M_1 or starts with some M_1 and tries to obtain C as the ciphertext. In each case, we calculate

$$\Pr[\Pi.\mathsf{Enc}(K_i^{j_1}, N_i^{j_1}, A_i^{j_1}, M_1) = (C, T_i^{j_1})]$$

for the M_1 selected. When $\mathcal{A}$ starts with C at index j_1, the analysis differs depending on whether $j_1 = 0$ or $j_1 > 0$. Therefore, we consider the following three cases:

- Case 1: $\mathcal{A}$ starts with C at index $j_1 > 0$. In this case, given $K_i^{j_1}, N_i^{j_1}, A_i^{j_1}, T_i^{j_1}$, the unverified message $M_1 = M_{1,1}\| \ldots \| M_{1,l}$ is generated blockwise. The next step is the verification of the tag $T_i^{j_1}$. Using $K_i^{j_1}, N_i^{j_1}, A_i^{j_1}, M_1$, the adversary can compute a new tag $\tilde{T}$, and $\tilde{T}$ is independent of $T_i^{j_1}$. For the tag verification to be successful, we must have $\tilde{T} = T_i^{j_1}$. Since, $j_1 > 0$, $T_i^{j_1}$ is sampled following a distribution which guarantees that for any $c \in \mathcal{T}, \Pr[T_i^j = c] \leq \epsilon$. Hence, the probability that the tag verification is successful is at most ϵ.

- Case 2: $\mathcal{A}$ starts with C at index $j_1 = 0$. In this case, T_i^0 is chosen by $\mathcal{A}$. Given $K_i^0, N_i^0, A_i^0, T_i^0$, the unverified message $M_1 = M_{1,1}\| \ldots \| M_{1,l}$ is generated blockwise. The next step is the verification of the tag T_i^0. Since K_i^0, N_i^0, A_i^0 is fixed, so is Auth_i^0. First, let us assume $|M_{1,l}| = n$. Then, the last message block $M_{1,l}$ must satisfy

$$M_{1,l} = \tilde{E}_{K_i^0}^{-1}\Bigg(0^4\|(l-1), (\mathsf{Auth}_i^0 \oplus \sum_{u=1}^{l-1} \tilde{E}(0^4\|(u-1), M_{1,u}) \oplus \tilde{E}_{K_i^0}^{-1}(1\|0^4\|N_i^0, T_i^0))\Bigg). \tag{6.1}$$

  This occurs with probability $1/2^n$. If $|M_{1,l}| < n$, then we replace $M_{1,l}$ by $M_{1,l}\|10^*$ in the above equation and the probability remains the same.

- Case 3: $\mathcal{A}$ starts with an M_1 at index j_1. Then, $\mathcal{A}$ must obtain

$$\Pi.\mathsf{Enc}(K_i^{j_1}, N_i^{j_1}, A_i^{j_1}, M_1) = (C, T_i^{j_1}).$$

  Regardless of whether j_1 equals 0 or not, $\mathcal{A}$ must choose $M_1 = M_{1,1}\| \ldots \| M_{1,l}$ such that $T_i^{j_1}$ is verified. To verify the tag, note that $\mathcal{A}$ can freely choose

any $(l-1)$ blocks of M_1 (for instance, the last $(l-1)$ blocks), while the first block must satisfy

$$M_{1,1} = \tilde{E}^{-1}_{K_i^{j_1}}\Bigg(0, (\mathsf{Auth}_i^{j_1} \oplus \sum_{u=2}^{l} \tilde{E}(0^4 \| (u-1), M_{1,u}) \\ \oplus \tilde{E}^{-1}_{K_i^{j_1}}(1 \| 0^4 \| N_i^{j_1}, T_i^{j_1}))\Bigg). \tag{6.2}$$

One thing to note here is that $M_{1,1}$ must be a full block. Otherwise, the verification of $T_i^{j_1}$ is probabilistic, and if $|M_1| = n-k$, then $\Pr[T_i^{j_1}$ is verified$] = 1/2^k$ accounting for the k bits of fixed padding.

Now that the tag is verified, $\mathcal{A}$ needs to obtain C as the ciphertext. Since $\mathcal{A}$ can choose the last $(l-1)$ blocks of M_1 freely, $\mathcal{A}$ can match the last $(l-1)$ blocks, but given that $M_{1,1}$ is fixed,

$$\Pr[C_1 = M_{1,1} \oplus \tilde{E}_K(1 \| T \oplus 1, 0^8 \| N)] = \frac{1}{2^{|C_1|}}.$$

If $|C_1| = n$, then the probability of $\mathcal{A}$ winning in this case is bounded by $1/2^n$. If $|C_1| = n-k$, then for $\mathcal{A}$ to win both $T_i^{j_1}$ needs to be verified and C_1 needs to match, the probability of which is bounded above by $1/2^k \times 1/2^{n-k} = 1/2^n$.

Next, if we assume that the Π computation at index j_2 happens after the Π computation at index j_1, note that Case 2 does not arise since $j_2 > 0$, and the analysis of the other cases are exactly the same.
Hence, $\Pr[\mathcal{A}$ wins $\mid i, j_i, j_2] = 2\epsilon + 3/2^n$. Summing over $0 \le j_1 < j_2$, we have

$$\Pr[\mathcal{A} \text{ wins} \mid i] \le m_{q_i}(m_{q_i}+1)(2\epsilon + 3/2^n).$$

Finally, summing over all i, we have

$$\Pr[\mathcal{A} \text{ wins}] \le q_2(q_2+1)(2\epsilon + 3/2^n).$$

6.2.3 Ciphertext Collision Advantage of Ascon

Now, we look at Ascon, the NIST LWC standard, and also the primary choice for the "lightweight authenticated encryption" portfolio of the CAESAR competition. The encryption algorithm for Ascon is given in Fig. 6.4. Let $\mathcal{A}$ be a (q_1, q_2, ϵ)-ciphertext collision adversary of Ascon. We calculate $\mathbf{Adv}^{\text{CC}}_{\text{Ascon}}(\mathcal{A})$. In the following analysis, Π refers to Ascon.

$\textsc{Ascon}[P,Q].\mathsf{Enc}(K,N,A=A[1]\|\ldots\|A[a],M=M[1]\|\ldots\|M[m])$

1: $S \leftarrow IV\|K\|N$ ▷ $|S|=b, |A[i]|=|M[i]|=r, c=b-r, |K|=k$
2: $S \leftarrow P(S)\oplus(0^{b-k}\|K)$ ▷ P is the outer permutation of Ascon
3: **for** $i \in [1..a]$ **do**
4: $\quad S \leftarrow Q(\lceil S\rceil_r \oplus A[i])\|\lfloor S\rfloor_c$ ▷ Q is the inner permutation of Ascon
5: $S \leftarrow S\oplus(0^{b-1}\|1)$
6: **for** $j \in [1..m-1]$ **do**
7: $\quad \lceil S\rceil_r \leftarrow \lceil S\rceil_r \oplus M[i]$
8: $\quad C[i] \leftarrow \lceil S\rceil_r$
9: $\quad S \leftarrow Q(S)$
10: $\lceil S\rceil_r \leftarrow \lceil S\rceil_r \oplus M[m]$
11: $C[m] \leftarrow \lceil S\rceil_{M[m]}$
12: $S \leftarrow P(S\oplus(0^r\|K\|0^{b-r-k})$
13: $T \leftarrow \lfloor S\rfloor_\tau \oplus \lfloor K\rfloor_\tau$
14: **return** $(C := C[1]\|\ldots\|C[m], T)$

FIGURE 6.4: Encryption of Ascon

As in the case of Deoxys-II, we fix i, j_1, j_2 and assume that the Π computation at index j_1 happens after the Π computation at index j_2. It is easy to see that the winning event of $\mathcal{A}$ can be divided into the same three cases as above. We analyse the three cases for Ascon now.

- Case 1: $\mathcal{A}$ starts with C at index $j_1 > 0$. Note that Ascon is CTX-friendly. Hence, given $K_i^{j_1}, N_i^{j_1}, A_i^{j_1}, C$, the adversary $\mathcal{A}$ can generate the verification tag $\tilde{T}$ as

$$\tilde{T} = \Pi.\mathsf{Auth}(K,N,A,\Pi.\mathsf{Dec}(K,N,A,C)),$$

  and hence $\tilde{T}$ and $T_i^{j_1}$ are necessarily independent. Since, $j_1 > 0$, $T_i^{j_1}$ is sampled following a distribution which guarantees that for any $c \in \mathcal{T}, \Pr[T_i^j = c] \leq \epsilon$. Hence, the probability that the tag verification is successful is at most ϵ.

- Case 2: $\mathcal{A}$ starts with C at index $j_1 = 0$. In this case, T_i^0 is chosen by $\mathcal{A}$. But since $\mathcal{A}$ gets C after it chooses T_i^0, the tag verification is not guaranteed. In fact, we must have

$$C_l\|X = P^{-1}(Y\|(T_i^0 \oplus \lceil K_i^0\rceil_\tau)), \tag{6.3}$$

  where X is a function of $(K_i^0, N_i^0, A_i^0, C_1, \ldots, C_{l-1})$ and Y can be chosen freely by $\mathcal{A}$. For an ideal permutation P, the probability of this event is $1/2^b$, where b is the permutation-size.

- Case 3: $\mathcal{A}$ starts with an M_1 at index j_1. For a successful ciphertext collision, we must have $M_1 = \Pi.\mathsf{Dec}(K_i^{j_1}, N_i^{j_1}, A_0^{j_1}, C)$. Even if the adversary chooses this M_1, the tag verification boils down to either Case 1 or Case 2 depending on the value of j_1.

Considering the instance when the Π computation at index j_2 happens after the Π computation at index j_1, and summing over all i, j_1, j_2, we have

$$\Pr[\mathcal{A} \text{ wins}] \leq q_2(q_2+1)(4\epsilon + 3/2^b).$$

6.2.4 Generalisation

In this section, we conjecture that if $\epsilon \leq 2/2^{|T|}$, then for a "well-designed" AE scheme, the ciphertext collision advantage of $\mathcal{A}$ is

$$\mathcal{O}\left(\frac{q_2^2}{2^{|T|}} + \frac{q_2^2}{2^{|S|}}\right),$$

where q_2 is total the number of Π computations available to $\mathcal{A}$, $|S|$ is the size of the primitive (block-size for block cipher-based AE or permutation-size for permutation-based AE), and $|T|$ is the tag-size. By "well-designed", we mean that the AE scheme must be built upon near-ideal primitives. We now provide an intuition for why we believe our conjecture is correct. Note that the exact ciphertext collision advantage of a scheme would depend on the scheme itself. For any Π, the cases remain the same as above, although their analyses will differ based on the scheme.

Starting with the first case, note that the analysis hinges on the fact that the tag $\tilde{T}$ generated during the verification process and the randomly chosen starting tag T_i^j are independent for both Deoxys-II and ASCON. We argue that this holds for any well-designed AE scheme.

First, note that in case of a CTX-friendly AE scheme Π, the valid tag $\tilde{T}$ corresponding to a tuple (K_i^j, N_i^j, A_i^j, C) can be obtained by

$$\tilde{T} = \Pi.\mathsf{Auth}(K_i^j, N_i^j, A_i^j, \Pi.\mathsf{Dec}(K_i^j, N_i^j, A_i^j, C)),$$

and hence $\tilde{T}$ and T_i^j are necessarily independent.

Next, for any AE scheme Π where decryption precedes verification in the decryption algorithm, if we select a tuple $(K_i^j, N_i^j, A_i^j, C, T_i^j)$ with a random T_i^j and decrypt, the unverified message M should essentially be almost independent of

T_i^j for any well-designed AE scheme. Many schemes like Mac-then-Encrypt and SIV (most prominent CTX-unfriendly schemes fall in this category) then employ a verification algorithm to generate T^*, stating that verification is successful if $T_i^j = T^*$. This T_i^j and T^* are independent. Therefore, the success probability of $\mathcal{A}$ in this case is of the order $1/2^{|T|}$ for all well-designed schemes.

Next, for Case 2, note that $\mathcal{A}$ chooses a starting tag T_i^0 before receiving any C from which it needs to construct M. The verification tag $\tilde{T}$ must be a function of either M or C; it cannot be solely a function of (K_i^0, N_i^0, T_i^0). We select Deoxys-II and Ascon for detailed discussions because $\tilde{T}$ depends on M in the case of Deoxys-II and on C in the case of Ascon.

If $\tilde{T}$ is a function of C, then since C is selected later, not all values of C will decrypt successfully. For successful decryption (i.e., $\tilde{T} = T_i^0$), C and T_i^0 must be related in some way (e.g., as in equation 6.3 for Ascon). In most schemes, at least one ciphertext block needs to be part of the relation.

If $\tilde{T}$ is a function of M, then a later-selected C can be decrypted to obtain an unverified message M, but for successful verification (i.e., $\tilde{T} = T_i^0$), M and T_i^0 must be related in some way (e.g., as in equation 6.1 for Deoxys-II). In most schemes, at least one message block needs to be part of the relation. In both these subcases, for all well-designed schemes, the success probability of $\mathcal{A}$ is of the order of $1/2^{|S|}$.

Finally, in Case 3 too, note that the starting tag T_i^j is chosen (either by $\mathcal{A}$ or randomly) before $\mathcal{A}$ receives C and selects an M to generate this C. For generating C, successful verification is necessary (i.e., the verification tag $\tilde{T}$ must match T_i^j). Similar to Case 2, at least one block of either C or M needs to be related to T_i^j, depending on whether $\tilde{T}$ is a function of C or M. Therefore, in this case as well, the success probability of $\mathcal{A}$ is of the order of $1/2^{|S|}$.

Then, taking a sum over i, j_1, j_2, our intuition is that the ciphertext collision advantage is

$$\mathcal{O}\left(\frac{q_2^2}{2^{|T|}} + \frac{q_2^2}{2^{|S|}}\right).$$

6.3 The PACT Transform

The PACT Transform. Let Π be an AE scheme, $\mathcal{H}$ be a hash function and E be a block cipher, such that the key-size of E is equal to the output-size of $\mathcal{H}$ and the block-size of E is equal to the tag-size of Π. We describe a transform PACT such that PACT$[\Pi, \mathcal{H}, E]$ is a CMT secure AE scheme.

$\mathsf{PACT}[\Pi, \mathcal{H}, E].\mathsf{Enc}(K, N, A, M)$

1: $(C, T) \leftarrow \Pi.\mathsf{Enc}(K, N, A, M)$
2: $K^* \leftarrow \mathcal{H}(K, N, A)$
3: $T^* \leftarrow E(K^*, T)$
4: **return** (C, T^*)

$\mathsf{PACT}[\Pi, \mathcal{H}, E].\mathsf{Dec}(K, N, A, C, T^*)$

1: $K^* \leftarrow \mathcal{H}(K, N, A)$
2: $T \leftarrow E^{-1}(K^*, T^*)$
3: $X \leftarrow \Pi.\mathsf{Dec}(K, N, A, C, T)$
4: **return** X

FIGURE 6.5: Specification of PACT

We start with an AE scheme Π having an encryption algorithm $\Pi.\mathsf{Enc}(K, N, A, M)$ that generates a ciphertext C and tag T. (If the AE scheme doesn't specify the tag and instead returns an expanded ciphertext C^* with $\lceil C^* \rceil = |M| + t$, we denote the ciphertext core as $C = \lceil C^* \rceil_{|M|}$ and the remaining t bits as the tag T.) Our PACT construction uses the same ciphertext but replaces the tag T with an alternative tag T^*. We compute T^* by hashing (K, N, A) using an unkeyed hash function and then feeding the hash output as the key into a block cipher along with the original tag T as the input. The complete specification of PACT is given in Figure 6.5. Note that we treat the nonce and the associated data in a similar fashion, and the interface of our transform accepts DAE schemes as well.

It is worth mentioning that while hashing the context (K, N, A) has become somewhat standard for committing security (both HtE and CTX employ this method), PACT introduces a block cipher call at the end. This design ensures that the modification of the tag from T to T^* is reversible. Unlike CTX, the information on the original tag T is preserved, allowing for decryption using $\Pi.\mathsf{Dec}(K, N, A, C, T^*)$ for all Π, even if their decryption algorithm utilises T. For the security proofs, we model E as an ideal cipher.

We claim that $\mathsf{PACT}[\Pi, \mathcal{H}, E]$ is CMT-secure as long as $\mathcal{H}$ is collision-resistant, and the advantage of any ciphertext collision adversary of Π is small.

Theorem 6.1. *For any* CMT *adversary* $\mathcal{A}$ *of* $\mathsf{PACT}[\Pi, \mathcal{H}, E]$ *making* q *computations of* Π *and* q_C *queries to the* t*-bit ideal cipher* E*, we can construct a collision adversary* $\mathcal{B}_1$ *of* $\mathcal{H}$ *and a* $(q_C, q, 2/2^t)$*-ciphertext collision adversary* $\mathcal{B}_2$ *of* Π *such that*

$$\mathbf{Adv}^{\mathrm{CMT}}_{\mathsf{PACT}[\Pi,\mathcal{H},E]}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{coll}}_{\mathcal{H}}(\mathcal{B}_1) + \mathbf{Adv}^{\mathrm{CC}}_{\Pi}(\mathcal{B}_2) + \frac{2q_C^2}{2^t} + \frac{2qq_C}{2^t} + \frac{2q_C}{2^t},$$

where $q_C < 2^{t-1}$*, and the computation time of* $\mathcal{B}_1$ *doesn't exceed that of* $\mathcal{A}$ *by more than a constant.*

The proof of Theorem 6.1 is deferred to Section 6.5.1. We have already argued that the ciphertext collision advantage should be $\mathcal{O}(q^2/2^{|S|} + q^2/2^t)$ (since the tag-size

$|T|$ equals the block-size t in the definition of PACT) for any practical AE. Further, the CMT security of PACT is bounded by the collision security of the hash function that it employs. One can expect to break this with about $2^{|K^*|/2}$ operations using a birthday attack, where $|K^*|$ is the key-size of E. Hence, roughly, PACT guarantees $\min\{\frac{|S|}{2}, \frac{|K^*|}{2}, \frac{t}{2}\}$ bits of CMT security.
It is also crucial to ensure that a secure AE scheme Π retains its traditional AE security properties after undergoing the PACT transform. In Theorems 6.2 and 6.3, we demonstrate that PACT maintains confidentiality security and authenticity security, respectively, thus preserving the original AE security of Π.

Theorem 6.2. *Let $\mathcal{A}$ be a confidentiality adversary of $\mathsf{PACT}[\Pi, \mathcal{H}, E]$ making q encryption queries and q_C ideal cipher queries. Then we can construct a collision adversary $\mathcal{B}$ for $\mathcal{H}$ and a confidentiality adversary $\mathcal{B}'$ for Π making q encryption queries to its own oracle, such that*

$$\mathbf{Adv}^{\mathsf{conf}}_{\mathsf{PACT}[\Pi,\mathcal{H},E]}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{coll}}_{\mathcal{H}}(\mathcal{B}) + \mathbf{Adv}^{\mathsf{conf}}_{\Pi}(\mathcal{B}') + \frac{q_C}{2^k} + \frac{2q^2}{2^t},$$

where k and t are the key-size and tag-size of Π respectively. The computation time of $\mathcal{B}$ and $\mathcal{B}'$ don't exceed that of $\mathcal{A}$ by more than a constant.

The proof of Theorem 6.2 is deferred to Section 6.5.2.

Theorem 6.3. *Let $\mathcal{A}$ be an authenticity adversary of $\mathsf{PACT}[\Pi, \mathcal{H}, E]$ making q_1 forging attempts and q_2 ideal cipher queries. Then we can construct a collision adversary $\mathcal{B}$ for $\mathcal{H}$ and an authenticity adversary $\mathcal{B}'$ of Π making q_2 forging attempts, such that*

$$\mathbf{Adv}^{\mathsf{auth}}_{\mathsf{PACT}[\Pi,\mathcal{H},E]}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{coll}}_{\mathcal{H}}(\mathcal{B}) + \mathbf{Adv}^{\mathsf{auth}}_{\Pi}(\mathcal{B}') + \frac{2q_1}{2^t}.$$

where t is the tag-size of Π. The computation time of both $\mathcal{B}$ and $\mathcal{B}'$ don't exceed that of $\mathcal{A}$ by more than a constant.

The proof of Theorem 6.3 is deferred to Section 6.5.3. While we prove these theorems for a nonce-based AE, it is evident from the proofs that the nonce does not have a unique role. Therefore, these proofs can also apply to DAE schemes.

6.3.1 CMT Security: PACT vs Others

We would like to point out that PACT does not fall short compared to other existing transforms in terms of providing CMT security. All existing transforms

guarantee at most $t/2$ bits of CMT security. As already mentioned above, PACT guarantees $\min\{\frac{|S|}{2}, \frac{|K^*|}{2}, \frac{t}{2}\}$ bits of CMT security.
Firstly, the state-size $|S|$ is the block-size for block cipher-based AE schemes, and on the permutation-size for public permutation-based AE schemes. Since the size of the permutations commonly used to design AE schemes is generally greater than the tag-size, this does not worsen our bound. Moreover, for block cipher-based AE schemes, the tag-size is usually not larger than the block-size. Hence, typically, $|S| \geq t$. Therefore, our bound is not worse in that case either.
Secondly, for all practical block ciphers the key-size $|K^*|$ is at least as large as the block-size t. Hence, $|K^*| \geq t$. Therefore, for all practical AEs, PACT also guarantees $t/2$ bits of CMT security.

6.3.2 Efficiency Comparison with HtE and CTX

PACT uses a call to a hash function $\mathcal{H}$ with K, N and A, and a call to a block cipher with T using $\mathcal{H}(K, N, A)$ as the key. While HtE uses a call to a hash function with K, N and A, CTX uses a call to a hash function with K, N, A and T. If we compare PACT to HtE or CTX, the only additional cost of PACT is one single call to a block cipher, in place of processing one additional block in the hash function. On the other hand, PACT requires only the collision-resistance assumption on $\mathcal{H}$, while both HtE and CTX require stronger assumptions on their respective hashes—a PRF assumption for HtE and a random oracle assumption for CTX, so it is possible to potentially instantiate PACT with a lighter hash function. Overall PACT does not lose out in efficiency when compared to HtE and CTX, while having wider applicability.

6.3.3 comPACT for UNAE Schemes

PACT is designed to transform an AE scheme to a CMT secure one. As we have already mentioned, it is a universal transform which is applicable to any AE scheme. But we can make PACT even more efficient if we do not need to preserve the MRAE security of the legacy AE. We call this variant comPACT. If the legacy AE is UNAE secure, it remains so after the comPACT transform. comPACT differs from PACT in the way it processes the associated data. In PACT, the associated data goes into both the legacy AE as well as into the hash function of PACT. In comPACT, the associated data is processed only by the hash function. The legacy AE is called with empty associated data. The CMT security of PACT also holds

comPACT$[\Pi, \mathcal{H}, E]$.Enc(K, N, A, M)
1: $(C, T) \leftarrow \Pi$.Enc(K, N, ϵ, M)
2: $K^* \leftarrow \mathcal{H}(K, N, A)$
3: $T^* \leftarrow E(K^*, T)$
4: **return** (C, T^*)

comPACT$[\Pi, \mathcal{H}, E]$.Dec(K, N, A, C, T^*)
1: $K^* \leftarrow \mathcal{H}(K, N, A)$
2: $T \leftarrow E^{-1}(K^*, T^*)$
3: $X \leftarrow \Pi$.Dec(K, N, ϵ, C, T)
4: **return** X

FIGURE 6.6: Specification of comPACT

for comPACT, as the security analysis is similar. The UNAE confidentiality and authenticity security analyses also carry over without any major changes. Fig. 6.6 gives the complete specification of comPACT.

Nonce-Misuse Attack on comPACT: One can mount a simple 2-query confidentiality attack after applying comPACT to a legacy AE. In the case of comPACT, as the legacy AE doesn't process the associated data, and the ciphertext output of the legacy AE is never updated, any two input tuples which differ only in the associated data values result in the same ciphertext. More precisely, if two input tuples of the form (K, N, A, M) and (K, N, A', M) with $A \neq A'$ result in the output tuples (C, T) and (C', T'), then $C = C'$.

6.3.4 Comparison between comPACT and CTY

The CTX transform processes the associated data twice: once for encryption and once for hashing. The CTY transform [10] is a faster variant, which encrypts with empty AD and processes the AD only for hashing. Since comPACT is derived from PACT in a similar manner, it is natural to compare CTY and comPACT. We note the nonce-misuse attack on comPACT also holds for CTY[2]. Therefore, in terms of security, both transforms offer similar guarantees.
From an applicability standpoint, comPACT has an advantage over CTY, as comPACT is universal whereas CTY is not. Additionally, since we have already established that PACT is as efficient as CTX, their respective faster variants do not lose out on efficiency either.

6.3.5 Instantiation

We now discuss the choices for $\mathcal{H}$ and E. For $\mathcal{H}$, we need a collision-resistant unkeyed hash function, and for E, a block cipher. Also, we need the output-size

[2]This comparison is done in the AE1 notion which we have used throughout the thesis. However, in the AE3 notion defined in [10], both comPACT and CTY would be MRAE-secure.

$\mathsf{GCM\text{-}SIV}[E, \mathsf{GHASH}].\mathsf{Enc}(K_1, K_2, K_3, N, A, M = M[1]\|\ldots\|M[m])$

$V \leftarrow \mathsf{GHASH}_{K_1}(\mathsf{Encode}(A, M)) \oplus N$ ▷ Encode is the GCM encoding function
$T \leftarrow E_{K_2}(V)$
$IV \leftarrow \lceil T \rceil_{n-k} \| 0^k$
$I[1] \leftarrow IV$
$Z[1] \leftarrow E_{K_3}(I[1])$
for $i \in [2..m]$ **do**
  $I[i] \leftarrow \mathsf{inc}(I[i-1])$ ▷ $\mathsf{inc}(X) := \lceil X \rceil_{n-32} \| ((\lfloor X \rfloor_{32} + 1) \mod 2^{32})$
  $Z[i] \leftarrow E_{K_3}(I[i])$
$Z \leftarrow Z[1]\|\ldots\|Z[m]$
return $(C := M \oplus Z, T)$

FIGURE 6.7: Encryption of GCM-SIV

of the hash function to be equal to the key-size of the block cipher. So, one can instantiate PACT using $(K, N, A) \mapsto \lceil$SHA-256 (K, N, A)$\rceil_{128}$ as the hash function and AES-128 as the block cipher. (We recall that for multiple hash inputs we always assume an injective padding.) We don't recommend using tag lengths shorter than 128 bits, because of the birthday-term in the confidentiality bound. Note that we propose instantiation only for AE schemes with 128-bit tag.

6.4 CMT_k Attacks On CTX-unfriendly AE Schemes

In this section, we seek to support our proposal by illustrating constant time CMT_k attacks on various AE schemes. Each of these schemes exhibits characteristics that make them unsuitable for CTX transform, as their decryption process relies on the tag. Moreover, none of these schemes adhere to the CAU-SIV paradigm. Consequently, neither HtE nor CTX can effectively be applied to any of them.

Our CMT_k attacks fall into two main categories. In some AE schemes, we expose vulnerabilities in the scheme that utilise polynomial hashing for the MAC, with the hash key being independent of the encryption key. These schemes are prone to CMT_k attacks whenever a CMT_k adversary can exploit the hash, such as by finding a collision. We expose such a vulnerability for GCM-SIV in Section 6.4.1. In certain other schemes, a CMT_k adversary can backtrack from the tag, they can modify a few blocks of nonce or associated data to launch a CMT_k attack. We give such an attack for Deoxys-II in Section 6.4.2. In Section 6.6, we give CMT_k attacks to some more AE schemes from both categories.

6.4.1 GCM-SIV

Fig. 6.7 gives the encryption algorithm of GCM-SIV [74], a widely used MRAE scheme. GCM-SIV uses three different keys: K_1 for GHASH (which is a polynomial hash), K_2 for the block cipher call on the GHASH output, and K_3 for the block cipher calls in the counter mode encryption. The authors suggest three instances of GCM-SIV. A three-key instance (GCM-SIV$_3$) where K_1, K_2 and K_3 are independently chosen, a two-key instance (GCM-SIV$_2$) where K_1 and K_2 are independently chosen and $K_3 = K_2$, and finally a single-key instance (GCM-SIV$_1$) where a single key K_0 is chosen first, then K_1 and K_2 are derived from K_0 by encrypting 0^{128} and $0^{127}||1$ respectively with AES instantiated with K_0, and $K_3 = K_2$. We observe that both GCM-SIV$_3$ and GCM-SIV$_2$ are vulnerable to a common CMT$_k$ attack, and the fix by Bellare and Hoang (which converts CAU-SIV to CAU-SIV-C1) [9] does not work for either of them. Note that GCM-SIV$_1$ is not vulnerable to this attack, since K_1 and K_2 are not independent.
Let $\mathcal{A}$ be a CMT$_k$ adversary attacking GCM-SIV$_i$ (where $i \in \{2, 3\}$). It proceeds as follows.

1. $\mathcal{A}$ computes (C, T) from (K_1, K_2, K_3, N, A, M).
2. $\mathcal{A}$ chooses $K_1' \neq K_1$ and A', and calculates N' such that

$$N' = \mathsf{GHASH}_{K_1}(\mathsf{Encode}(A, M)) \oplus \mathsf{GHASH}_{K_1'}(\mathsf{Encode}(A', M)) \oplus N,$$

   where M is the same as the first query, and Encode is the GCM encoding function.
3. $\mathcal{A}$ outputs (K_1, K_2, K_3, N, A, M) and $(K_1', K_2, K_3, N', A', M)$.

6.4.2 Deoxys-II and Joltik$^=$

Now, we move on to some schemes where a CMT$_k$ adversary needs to exploit the structure of the schemes to produce an attack. SCT (Synthetic Counter-in-Tweak) or Joltik$^=$ [82], and SCT-2 or Deoxys-II[83, 84] were submitted to the CAESAR competition [45], where Joltik$^=$ was selected as a second-round candidate, and Deoxys-II was selected as the first choice in the final portfolio for defence-in-depth. Both these constructions follow the NSIV(nonce-based SIV) paradigm [108] and are MRAE schemes. Here, we show a CMT$_k$ attack on Deoxys-II. A similar attack also holds for Joltik$^=$. Fig. 6.3 gives the encryption algorithm of Deoxys-II.

Let $\mathcal{A}$ be a CMT_k adversary attacking Deoxys-II. It proceeds as follows.

1. $\mathcal{A}$ computes (C, T) from (K, N, A, M). Let $C = C[1]\|\ldots\|C[m]$.
2. $\mathcal{A}$ chooses K' and N', and fixes $M' = M'[1]\|\ldots\|M'[m]$ in a way such that the XOR of $M'[i]$ and the encryption of $0\|N'$ yields $C[i]$. Formally, $M'[i] = \tilde{E}_{K'}(1\|T \oplus (i-1), 0^8\|N') \oplus C[i](\forall i \in [m])$.
3. $\mathcal{A}$ chooses all but one of the blocks of A' (say, the first block $A'[1]$), and makes forward queries to $\tilde{E}$ with all the associated data and message blocks as inputs (except $A'[1]$). Let X be the xor of all the outputs from $\tilde{E}$, and Y be the output of the backward query to $\tilde{E}$ with T as the input.
4. $\mathcal{A}$ makes a backward query to $\tilde{E}$ with $X \oplus Y$ as the input and sets the output as $A'[1]$.
5. $\mathcal{A}$ outputs (K, N, A, M) and (K', N', A', M').

6.5 Security Proofs

6.5.1 Proof of Theorem 6.1

In this section, we investigate the CMT security of PACT. We assume the adversary does not make any pointless queries, which include:

- repeated queries to Π or E;
- a query (K, N, A, C, T) to Π.Dec following a query (K, N, A, M) to Π.Enc that returned (C, T);
- a query (K, N, A, M) to Π.Enc following a query (K, N, A, C, T) to Π.Dec that returned M;
- a query (K^*, T^*) to E^{-1} following a query (K^*, T) to E that returned T^*;
- a query (K^*, T) to E following a query (K^*, T^*) to E^{-1} that returned T.

First, we list the adversaries we construct to simulate $\mathcal{A}$.

- $\mathcal{B}_1$: A collision adversary of $\mathcal{H}$. It outputs a colliding $\mathcal{H}$ input pair of $\mathcal{A}$, if any such pair exists. Otherwise, it outputs any two random $\mathcal{H}$ inputs.

- $\mathcal{B}_2$: A $(q_C, q, 2/2^t)$-ciphertext collision adversary of Π. Whenever $\mathcal{A}$ queries $T_j^* = E(K_j^*, T_j)$ where $K_i^* = \mathcal{H}(K_i, N_i, A_i)$ and $T_j^* \neq T_k^*$ for $k \in [j-1]$, it makes a new list $\mathcal{L}[T_j^*]$, and submits (K_j, N_j, A_j, T_j). Whenever $\mathcal{A}$ queries $E^{-1}(K_{j'}^*, T_j^*)$ where $K_{i'}^* = \mathcal{H}(K_{i'}, N_{i'}, A_{i'})$, $\mathcal{B}_2$ includes $(K_{j'}, N_{j'}, A_{j'})$ in $\mathcal{L}[T_j^*]$ and submits the same, receives $T_{j'} = E^{-1}(K_{j'}^*, T_j^*)$, and forwards it to $\mathcal{A}$. Finally, if $\mathcal{A}$ makes any two Π computations to obtain the tuples $(K_y, N_y, A_y, M_y, C, T_y)$ and $(K_z, N_z, A_z, M_z, C, T_z)$ where both the tuples are in the same list $\mathcal{L}[T_x^*]$ and $y < z$, $\mathcal{B}_2$ outputs M_y, M_z, x, y and z. Otherwise, $\mathcal{B}_2$ aborts.

Next, we list the bad events.

- B1: $\mathcal{B}_1$ wins.

$$\Pr[\mathsf{B1}] \leq \mathbf{Adv}_{\mathcal{H}}^{\mathsf{coll}}(\mathcal{B}_1).$$

- B2: $\mathcal{B}_2$ wins.

$$\Pr[\mathsf{B2}] \leq \mathbf{Adv}_{\Pi}^{\mathrm{CC}}(\mathcal{B}_2).$$

- B3: $\mathcal{A}$ makes two E queries to obtain the tuples (K, X, Y) and (K', X', Y) such that $\mathcal{A}$ obtains (K', X', Y) after it obtains (K, X, Y), and through a forward query (i.e., $\mathcal{A}$ queries $Y = E(K', X')$). For a fixed pair of queries, the probability of this event can be upper-bounded by $1/(2^t - q_C)$. Because such a pair of E queries can be chosen in $\binom{q_C}{2}$ ways, and $q_C < 2^{t-1}$, applying union bound we obtain

$$\Pr[\mathsf{B3}] \leq 2\binom{q_C}{2}/2^t < 2q_C^2/2^t.$$

- B4: $\mathcal{A}$ makes one Π computation to obtain an input-output tuple (K, N, A, M, C, T), and after that, makes an inverse query to E with K' and T', such that $K \neq K'$ and $E^{-1}(K', T') = T$. As we model E as an ideal cipher, for a fixed Π computation and a fixed E query, the probability of this event can be upper-bounded by $1/(2^t - q_C)$. As $\mathcal{A}$ can make at most q computations of Π and q_C queries to E, applying union bound we obtain

$$\Pr[\mathsf{B4}] \leq 2qq_C/2^t.$$

Suppose $\mathcal{A}$ submits (K_1, N_1, A_1, M_1) and (K_2, N_2, A_2, M_2) with $(K_1, N_1, A_1) \neq (K_2, N_2, A_2)$ and wins. W.l.o.g., we assume that if $\mathcal{A}$ has made either $E(K_1^*, T_1)$ or $E^{-1}(K_1^*, T^*)$, as well as either $E(K_2^*, T_2)$ or $E^{-1}(K_2^*, T^*)$, it has made them in that order, i.e., the latter is made after the former. Then the following must hold for some C, T^*, T_1, T_2, K_1^*, and K_2^*:

$$\begin{aligned}(C, T_1) &= \Pi.\mathsf{Enc}(K_1, N_1, A_1, M_1),\\ (C, T_2) &= \Pi.\mathsf{Enc}(K_2, N_2, A_2, M_2),\\ K_1^* &= \mathcal{H}(K_1, N_1, A_1),\\ K_2^* &= \mathcal{H}(K_2, N_2, A_2),\\ T^* &= E(K_1^*, T_1) = E(K_2^*, T_2).\end{aligned}$$

Next, we list the events, one of which must happen.

- E1: $\mathcal{A}$ makes both the queries (1) $T^* = E(K_1^*, T_1)$ or $T_1 = E^{-1}(K_1^*, T^*)$, and (2) $T^* = E(K_2^*, T_2)$ or $T_2 = E^{-1}(K_2^*, T^*)$.
  - E11: $\mathcal{A}$ queries $E(K_2^*, T_2)$.
  - E12: $\mathcal{A}$ queries $E^{-1}(K_2^*, T^*)$.
    - E121: $(K_1, N_1, A_1, M_1, C, T_1) \in \Pi$, $(K_2, N_2, A_2, M_2, C, T_2) \in \Pi$.
    - E122: $(K_2, N_2, A_2, M_2, C, T_2) \in \Pi$, $(K_1, N_1, A_1, M_1, C, T_1) \notin \Pi$.
    - E123: $(K_1, N_1, A_1, M_1, C, T_1) \in \Pi$, $(K_2, N_2, A_2, M_2, C, T_2) \notin \Pi$.
    - E124: $(K_1, N_1, A_1, M_1, C, T_1) \notin \Pi$, $(K_2, N_2, A_2, M_2, C, T_2) \notin \Pi$.
- E2: $\mathcal{A}$ doesn't make both the queries (1) $T^* = E(K_1^*, T_1)$ or $T_1 = E^{-1}(K_1^*, T^*)$, and (2) $T^* = E(K_2^*, T_2)$ or $T_2 = E^{-1}(K_2^*, T^*)$.

Next, we analyse the winning probability of $\mathcal{A}$.

- E11: This event is impossible because B3 doesn't happen.
- E121: $K_1^* = K_2^*$ is impossible as B1 doesn't happen. And if $K_1^* \neq K_2^*$, then this event is impossible as B4 doesn't happen.
- E122: This event is impossible, and the argument is the same as E121.
- E123: If $\mathcal{A}$ makes a forward query $T^* = E(K_1^*, T_1)$, then a previous forward query $T^* = E(K, T)$ by $\mathcal{A}$ is not possible because B3 doesn't happen. Otherwise, E123 is impossible as B2 doesn't happen.

$E[\mathscr{T}](K^*, T)$

1: $T^* \leftarrow_\$ \{0,1\}^t \setminus \mathscr{T}[K^*, \cdot]$
2: **if** $\mathscr{T}[K^*, T] \neq \perp$ **then**
3: $\quad T^* \leftarrow \mathscr{T}[K^*, T]$
4: **else**
5: $\quad \mathscr{T}[K^*, T] \leftarrow T^*$
6: **return** T^*

$\mathsf{PACT}[\Pi, \mathcal{H}, \mathscr{T}, K](N, A, M)$

1: $(C, T) \leftarrow \Pi.\mathsf{Enc}(K, N, A, M)$
2: $T^* \leftarrow_\$ \{0,1\}^t \setminus \mathscr{T}[\mathcal{H}(K, N, A), \cdot]$
3: **if** $\mathscr{T}[\mathcal{H}(K, N, A), T] \neq \perp$ **then**
4: $\quad T^* \leftarrow \mathscr{T}[\mathcal{H}(K, N, A), T]$
5: **else**
6: $\quad \mathscr{T}[\mathcal{H}(K, N, A), T] \leftarrow T^*$
7: **return** (C, T^*)

FIGURE 6.8: Lazy-sampled versions of E and PACT, using a shared table.

- E124: This event is impossible, and the argument is the same as E123.
- E2: W.l.o.g., suppose $E(K_1^*, T_1)$ or $E^{-1}(K_1^*, T^*)$ is not made by $\mathcal{A}$. Then the probability of the event $E(K_1^*, T_1) = T^*$ (in case $E(K_2^*, T_2)$ or $E^{-1}(K_2^*, T^*)$ is made by $\mathcal{A}$, so that T^* is fixed) or $E(K_1^*, T_1) = E(K_2^*, T_2)$ (in case $E(K_2^*, T_2)$ or $E^{-1}(K_2^*, T^*)$ is not made by $\mathcal{A}$) is upper-bounded by $1/(2^t - q_C)$. As $\mathcal{A}$ can query E maximum q_C times, and $q_C < 2^{t-1}$, we obtain by applying union bound

$$\Pr[\mathcal{A} \text{ wins}] \leq 2q_C/2^t.$$

The final upper bound of the winning probability of $\mathcal{A}$ is obtained by adding the probabilities of all the bad events and the winning probability of $\mathcal{A}$ in the case of E2. □

6.5.2 Proof of Theorem 6.2

We bound the confidentiality advantage of $\mathcal{A}$ against $\mathsf{PACT}[\Pi, \mathcal{H}, E]$ through a series of games. In game $\mathcal{G}_0$, we begin with lazy-sampled versions of PACT and the ideal cipher E, such that PACT doesn't make oracle calls to E, but instead they share the same table $\mathscr{T}$ (Fig. 6.8). The adversary plays against $(\mathsf{PACT}[\Pi, \mathcal{H}, \mathscr{T}, K], E[\mathscr{T}])$, where $K \leftarrow_\$ \mathcal{K}$. In the subsequent games, we transform $\mathsf{PACT}[\Pi, \mathcal{H}, \mathscr{T}, K]$ through a series of hybrids which culminate in $\mathsf{PACT}^\star$ in game $\mathcal{G}_5$, which is identical in output distribution to Π^*. The different hybrid construction oracles used are described in Fig. 6.9. Figure 6.10 lists the series of games and the corresponding construction oracles.

In game $\mathcal{G}_1$, PACT remains unchanged, but uses an independent table $\mathscr{T}'$. Suppose $\mathcal{A}_1$ is an adversary trying to distinguish between $\mathcal{G}_0$ and $\mathcal{G}_1$. Since the different rows $\mathscr{T}[K^*, \cdot]$ of $\mathscr{T}$ are always sampled independently, the games $\mathcal{G}_0$ and $\mathcal{G}_1$ are

$\mathsf{PACT}^{\circ}[\Pi, \mathscr{T}^{\circ}, K](N, A, M)$

$(C, T) \leftarrow \Pi.\mathsf{Enc}(K, N, A, M)$
$T^* \leftarrow_{\$} \{0,1\}^t \setminus \mathscr{T}^{\circ}[N, A, \cdot]$
if $\mathscr{T}^{\circ}[N, A, T] \neq \bot$ **then**
    $T^* \leftarrow \mathscr{T}^{\circ}[N, A, T]$
else
    $\mathscr{T}^{\circ}[N, A, T] \leftarrow T^*$
return (C, T^*)

$\mathsf{PACT}^{\dagger}[\mathscr{T}^{\circ}](N, A, M)$

$(C, T) \leftarrow_{\$} \{0,1\}^{|M|+t}$
$T^* \leftarrow_{\$} \{0,1\}^t \setminus \mathscr{T}^{\circ}[N, A, \cdot]$
if $\mathscr{T}^{\circ}[N, A, T] \neq \bot$ **then**
    $T^* \leftarrow \mathscr{T}^{\circ}[N, A, T]$
else
    $\mathscr{T}^{\circ}[N, A, T] \leftarrow T^*$
return (C, T^*)

$\mathsf{PACT}^{\ddagger}[\mathscr{T}^{\circ}](N, A, M)$

$(C, T) \leftarrow_{\$} \{0,1\}^{|M|+t}$
$T^* \leftarrow_{\$} \{0,1\}^t \setminus \mathscr{T}^{\circ}[N, A, \cdot]$
$\mathscr{T}^{\circ}[N, A, T] \leftarrow T^*$
return (C, T^*)

$\mathsf{PACT}^{\star}(N, A, M)$

$(C, T) \leftarrow_{\$} \{0,1\}^{|M|+t}$
$T^* \leftarrow_{\$} \{0,1\}^t$
return (C, T^*)

FIGURE 6.9: The hybrid construction oracles used in the proof of Theorem 6.2.

Game	Construction Oracle
$\mathcal{G}_0$	$\mathsf{PACT}[\Pi, \mathcal{H}, \mathscr{T}, K]$
$\mathcal{G}_1$	$\mathsf{PACT}[\Pi, \mathcal{H}, \mathscr{T}', K]$
$\mathcal{G}_2$	$\mathsf{PACT}^{\circ}[\Pi, \mathscr{T}^{\circ}, K]$
$\mathcal{G}_3$	$\mathsf{PACT}^{\dagger}[\mathscr{T}^{\circ}]$
$\mathcal{G}_4$	$\mathsf{PACT}^{\ddagger}[\mathscr{T}^{\circ}]$
$\mathcal{G}_5$	$\mathsf{PACT}^{\star}$

FIGURE 6.10: The adversary's construction oracles corresponding to the different games proof of Theorem 6.2. In each game the adversary also queries $E[\mathscr{T}]$. In $\mathcal{G}_0$ the table $\mathscr{T}$ is shared between PACT and E; in $\mathcal{G}_1$ it is replaced in PACT by an identical but independent table $\mathscr{T}'$. In the first three games $K \leftarrow_{\$} \mathcal{K}$ as part of the initialisation.

identical unless at some point in $\mathcal{G}_0$, PACT° and E access the same row of $\mathscr{T}^{\circ}$; this can only happen if $\mathcal{A}_1$ can correctly guess the construction key K and sets $K^* = \mathcal{H}(K, N, A)$ for some N, A in one of its queries to E, so we have

$$\mathbf{Adv}_{\mathcal{G}_0, \mathcal{G}_1}(\mathcal{A}_1) \leq \frac{q_C}{2^k}.$$

In game $\mathcal{G}_2$, we replace PACT with PACT° which doesn't make calls to $\mathcal{H}$ and uses a modified table $\mathscr{T}^{\circ}$ where the rows are indexed with (N, A) instead of $\mathcal{H}(K, N, A)$. Let $\mathcal{A}_2$ be an adversary trying to distinguish between $\mathcal{G}_1$ and $\mathcal{G}_2$. Let $\mathcal{B}$ be a collision adversary which simulates both the challenger and $\mathcal{A}_2$; thus $\mathcal{B}$ knows K and all the online queries (N, A, M) of $\mathcal{A}_2$, and computes $\mathcal{H}(K, N, A)$ for each

such online query. If any two of these inputs X_1 and X_2 have $\mathcal{H}(X_1) = \mathcal{H}(X_2)$, $\mathcal{B}$ outputs (X_1, X_2), and otherwise $\mathcal{B}$ outputs a pair of random hash inputs (Z_1, Z_2). Then the games $\mathcal{G}_1$ and $\mathcal{G}_2$ are identical unless $\mathcal{B}$ wins. Thus,

$$\mathbf{Adv}_{\mathcal{G}_1,\mathcal{G}_2}(\mathcal{A}_2) \leq \mathbf{Adv}^{\mathsf{coll}}_{\mathcal{H}}(\mathcal{B}).$$

In game $\mathcal{G}_3$, we replace PACT° with $\mathsf{PACT}^\dagger$, which samples (C, T) randomly instead of calling $\Pi.\mathsf{Enc}$. Let $\mathcal{A}_3$ be an adversary trying to distinguish between $\mathcal{G}_2$ and $\mathcal{G}_3$. Then we construct a confidentiality adversary $\mathcal{B}'$ for Π as follows: $\mathcal{B}'$ simulates $\mathcal{A}_3$ and lets $\mathcal{A}_3$ play against $(\mathsf{PACT}^?[\mathscr{T}^\circ], E^\circ)$, where $\mathsf{PACT}^?$ is identical to PACT° or $\mathsf{PACT}^\dagger$ except that the first line is replaced by a call to $\mathcal{B}'$'s own oracle. $\mathcal{B}'$ replicates the output bit of $\mathcal{A}_3$. Then we have

$$\mathbf{Adv}_{\mathcal{G}_2,\mathcal{G}_3}(\mathcal{A}_3) \leq \mathbf{Adv}^{\mathsf{conf}}_{\Pi}(\mathcal{B}').$$

In game $\mathcal{G}_4$, we replace $\mathsf{PACT}^\dagger$ with $\mathsf{PACT}^\ddagger$, which updates the table $\mathscr{T}^\circ$ without checking for previous entries. Let $\mathcal{A}_4$ be an adversary trying to distinguish between $\mathcal{G}_3$ and $\mathcal{G}_4$. The two games are identical unless the if-clause in $\mathsf{PACT}^\dagger$ is executed in $\mathcal{G}_3$ for some query, which can only happen if a randomly sampled T collides with an earlier T' for the same (N, A). Thus we have

$$\mathbf{Adv}_{\mathcal{G}_3,\mathcal{G}_4}(\mathcal{A}_4) \leq \frac{q^2}{2^t}.$$

In game $\mathcal{G}_5$, we replace $\mathsf{PACT}^\ddagger$ with $\mathsf{PACT}^\star$, which does not maintain a table and samples T^* uniformly at random. Let $\mathcal{A}_5$ be an adversary trying to distinguish between $\mathcal{G}_4$ and $\mathcal{G}_5$. The two games are identical unless a randomly sampled T^* collides with an earlier T'^* for the same (N, A). Thus we have

$$\mathbf{Adv}_{\mathcal{G}_4,\mathcal{G}_5}(\mathcal{A}_5) \leq \frac{q^2}{2^t}.$$

If $\mathcal{A}$ wins the confidentiality game against $\mathsf{PACT}[\Pi, \mathcal{H}, E]$, at least one of $\mathcal{A}_1, \ldots, \mathcal{A}_5$ must win. Thus we have

$$\begin{aligned}\mathbf{Adv}^{\mathsf{conf}}_{\mathsf{PACT}[\Pi,\mathcal{H},E]}(\mathcal{A}) &\leq \sum_{i=1}^{5} \mathbf{Adv}_{\mathcal{G}_{i-1},\mathcal{G}_i}(\mathcal{A}_i) \\ &\leq \mathbf{Adv}^{\mathsf{coll}}_{\mathcal{H}}(\mathcal{B}) + \mathbf{Adv}^{\mathsf{conf}}_{\Pi}(\mathcal{B}') + \frac{q_C}{2^k} + \frac{2q^2}{2^t}.\end{aligned}$$

This completes the proof. □

6.5.3 Proof of Theorem 6.3

Both $\mathcal{B}'$ and $\mathcal{B}$ simulate $\mathcal{A}$. $\mathcal{B}$ looks at the hash computations as $\mathcal{A}$, and if $\mathcal{A}$ finds two hash inputs X_1 and X_2 with $\mathcal{H}(X_1) = \mathcal{H}(X_2)$, $\mathcal{B}$ outputs (X_1, X_2), and otherwise $\mathcal{B}$ outputs a pair of random hash inputs (Z_1, Z_2). $\mathcal{B}'$ proceeds as described below.

Let the ℓ-th forging attempt of $\mathcal{A}$ be $(N_\ell, A_\ell, C_\ell, T^*_\ell)$ (where $\ell \in [q_1]$). For each $\ell \in [q_1]$, let $\mathcal{L}_\ell$ be the list of all ideal cipher query-response tuples of the form $(K^*_{\ell,i}, T_{\ell,i}, T^*_\ell)$, such that for each $i \in [|\mathcal{L}_\ell|]$ there is a hash computation with input $(K_{\ell,i}, N_\ell, A_\ell)$ and output $K^*_{\ell,i}$. For each $\ell \in [q_1]$, $i \in [|\mathcal{L}_\ell|]$, if $\Pi.\mathsf{Dec}(K_{\ell,i}, N_\ell, A_\ell, C_\ell, T_{\ell,i}) \neq \perp$, then $\mathcal{B}'$ makes a forging attempt $(N_\ell, A_\ell, C_\ell, T_{\ell,i})$.

Let K be the secret key of Π. Suppose the forging attempt $(N_\ell, A_\ell, C_\ell, T^*_\ell)$ by $\mathcal{A}$ is successful. This can only happen when at least one of the following four events occurs:

- B1: $\mathcal{A}$ finds a collision-pair for $\mathcal{H}$. When this happens, $\mathcal{B}$ wins, so

$$\Pr[\mathsf{B1}] \leq \mathbf{Adv}^{\mathsf{coll}}_{\mathcal{H}}(\mathcal{B}).$$

- B2: $(N_\ell, A_\ell, C_\ell, T)$ is a corresponding forging attempt by $\mathcal{B}'$, and $K_{\ell,i} = K$ for $T_{\ell,i} = T$. In this case, whenever $\mathcal{A}$ wins, $\mathcal{B}'$ also wins. Also note that $\mathcal{B}'$ makes a maximum of q_2 forging attempts with this strategy; this is because each pair $(T^*_\ell, K^*_{\ell,i})$ can lead to at most one forging attempt. So we obtain

$$\Pr[\mathsf{B2} \mid \neg\mathsf{B1}] \leq \mathbf{Adv}^{\mathsf{auth}}_{\Pi}(\mathcal{B}').$$

- B3: $(N_\ell, A_\ell, C_\ell, T)$ is a corresponding forging attempt by $\mathcal{B}'$, but $K_{\ell,i} \neq K$ for $T_i = T$. In this case, the forgery by $\mathcal{A}$ is accidental, i.e., $\mathcal{A}$ wins by random guess of the tag. Hence

$$\Pr[\mathsf{B3} \mid \neg\mathsf{B1}] \leq \frac{q_1}{2^t}.$$

- B4: $\mathcal{B}'$ makes no corresponding forging attempt. This follows the same reasoning as B2, so

$$\Pr[\mathsf{B4} \mid \neg\mathsf{B1}] \leq \frac{q_1}{2^t}.$$

The final upper bound of the winning probability of $\mathcal{A}$ is obtained by applying union-bound over the four events. □

6.6 CMT_k Attacks On CTX-unfriendly AE Schemes (Continued)

Here we continue our discussion from Section 6.4 on illustrating constant time CMT_k attacks on various AE schemes where neither HtE nor CTX can be applied.

6.6.1 GCM-SIVr

In GCM-SIVr [80] which runs r instances of GCM-SIV in parallel, all the keys are independent. So GCM-SIV1 is vulnerable to the same attack, which works on $\mathsf{GCM\text{-}SIV}_3$ and $\mathsf{GCM\text{-}SIV}_2$. For GCM-SIV$r$ with $r > 1$, the CMT_k adversary $\mathcal{A}$ proceeds as follows.

1. $\mathcal{A}$ computes (C, T) from (K_1, K_2, K_3, N, A, M). Let $K_i = (K_{i,1}, \ldots, K_{i,r})$ for $i = 1, 3$ and $K_2 = (K_{2,1}, \ldots, K_{2,r^2})$.

2. $\mathcal{A}$ chooses $K_1' \neq K_1$ and all but the first r blocks of A', and calculates them from the following set of r equations.

$$\begin{gathered}\mathsf{GHASH}_{K_{1,1}}(\mathsf{Encode}(A, M)) = \mathsf{GHASH}_{K_{1,r}'}.(\mathsf{Encode}(A', M)) \\ \vdots \\ \mathsf{GHASH}_{K_{1,r}}(\mathsf{Encode}(A, M)) = \mathsf{GHASH}_{K_{1,r}'}.(\mathsf{Encode}(A', M))\end{gathered}$$

3. $\mathcal{A}$ outputs (K_1, K_2, K_3, N, A, M) and $(K_1', K_2, K_3, N, A', M)$.

6.6.2 SCM

Synthetic Counter with Masking, or SCM [44] is yet another MRAE scheme following the NSIV paradigm. Fig. 6.11 gives the encryption algorithm of SCM. We show a CMT_k attack on SCM, when it is instantiated with an ϵ-AXU polynomial hash. Let $\mathcal{A}$ be a CMT_k adversary attacking SCM. It proceeds as follows.

$\mathsf{SCM}[E, \mathcal{H}].\mathsf{Enc}(K_1, K_2, K_3, K_4, N, A, M = M[1]\|\ldots\|M[m])$

$\Delta \leftarrow E_{K_3}(N\|00) \oplus E_{K_3}(N\|01)$
$\Delta' \leftarrow E_{K_3}(N\|00) \oplus E_{K_3}(N\|10)$
$\Delta'' \leftarrow E_{K_3}(N\|00) \oplus E_{K_3}(N\|11)$
$X \leftarrow \mathcal{H}_{K_1}(A, M) \oplus (N\|00)$ ▷ $\mathcal{H}$ is an ϵ-AXU hash
$T \leftarrow E_{K_2}(X) \oplus \Delta''$
for $i \in [1..m]$ **do**
    $X[i] \leftarrow T \oplus 2^{i-1}\Delta$
    $Z[i] \leftarrow E_{K_4}(X[i]) \oplus \Delta'$
$Z \leftarrow Z[1]\|\ldots\|Z[m]$
return $(C := M \oplus Z, T)$

FIGURE 6.11: Encryption of SCM

$\mathsf{DCT}[\mathcal{H}, E, \tilde{E}].\mathsf{Enc}(K_1, K_2, K_3, A, M)$

$(M_L, M_R \leftarrow \mathsf{Encode}(M))$
$X \leftarrow \mathcal{H}_{K_1}(A, M_R)$
$Y \leftarrow M_L \oplus X$
$C_L \leftarrow E_{K_2}(Y)$
$C_R \leftarrow \tilde{E}_{K_3}(C_L, M_R)$
return $(C := C_L\|C_R)$

FIGURE 6.12: Encryption of DCT

1. $\mathcal{A}$ computes (C, T) from (K, N, A, M), where $K = (K_1, \ldots, K_4)$.
2. $\mathcal{A}$ chooses $K_1' \neq K_1$ and all but one of the blocks of A', (say, the first block $A'[1]$), and calculates $A'[1]$ from the equation

$$\mathcal{H}_{K_1}(A, M) = \mathcal{H}_{K_1'}(A', M).$$

3. $\mathcal{A}$ outputs (K, N, A, M) and (K', N, A', M), where $K' = (K_1', K_2, K_3, K_4)$.

6.6.3 DCT

We now have a look at a few DAE schemes, starting with Deterministic Counter in Tweak, or DCT [67]. Fig. 6.12 gives the encryption algorithm of DCT. A CMT_k attack similar to that on SCM also holds for DCT, when it is instantiated with an ϵ-AXU polynomial hash.

6.6.4 BCTR

Fig. 6.13 gives the encryption algorithm of BCTR [38], which is a DAE scheme suitable for disk encryption. We first show a CMT_k attack on BCTR. In the

$\mathsf{BCTR.Enc}[E, \mathsf{BRW}](K_1, K_2, \tau, M = M[1]\| \ldots \|M[m])$

1: $X \leftarrow K_1 \cdot \mathsf{BRW}_{K_1}(M\|\tau)$
2: $T \leftarrow E_{K_2}(X)$
3: **for** $i \in [1..m]$ **do**
4: $\quad Z[i] \leftarrow E_{K_2}(T \oplus i)$
5: **return** $(C := M \oplus Z, T)$

FIGURE 6.13: Encryption of BCTR

$\mathsf{SUNDAE}[E].\mathsf{Enc}(K, A = A[1]\| \ldots \|A[a], M = M[1]\| \ldots \|M[m])$

1: $T \leftarrow E_K(110^{n-2})$
2: **for** $i \in [1..a]$ **do**
3: $\quad T \leftarrow E_K(T \oplus A[i])$
4: $T \leftarrow E_K(4 \cdot (T \oplus A[a]))$
5: **for** $i \in [1..m-1]$ **do**
6: $\quad T \leftarrow E_K(T \oplus M[i])$
7: $T \leftarrow E_K(4 \cdot (X \oplus M[m]))$
8: $V \leftarrow T$
9: **for** $i \in [1..m]$ **do**
10: $\quad V \leftarrow E_K(V)$
11: $\quad C[i] \leftarrow M[i] \oplus V$
12: **return** $(C := (C[1], \ldots, C[m]), T)$

FIGURE 6.14: Encryption of SUNDAE

case of BCTR, the CMT_k adversary $\mathcal{A}$ considers the tuples (K_1, K_2, M, τ) and (K_1', K_2, M, τ') for the attack. It chooses the values for all the inputs except a single tweak block (say, the j-th block of τ, i.e., τ_j), and calculates its value from the equation

$$K_1 \cdot \mathsf{BRW}_{K_1}(M, \tau) = K_1' \cdot \mathsf{BRW}_{K_1'}(M, \tau').$$

6.6.5 SUNDAE and ANYDAE

We now have a look at SUNDAE [7], a lightweight DAE scheme, and its RUP secure modification ANYDAE [41]. Fig. 6.14 gives the encryption algorithm of SUNDAE. We first show a CMT_k attack on SUNDAE. Let $\mathcal{A}$ be a CMT_k adversary attacking SUNDAE. It proceeds as follows.

1. $\mathcal{A}$ computes (C, T) from (K, A, M).
2. $\mathcal{A}$ then chooses a $K' \neq K$, and fixes $M' = M'[1], \ldots, M'[m]$ by setting $M'[1] = E^i_{K'}(T) \oplus C[i]$, where E^i denotes i compositions of E.

LM-DAE$[\tilde{E}, \pi]$.Enc$(K, A = A[1]\| \ldots \|A[a], M = M[1]\| \ldots \|M[m])$

$h_t \leftarrow 0^n$
$h_b \leftarrow 0^n$
for $i \in [1..a-1]$ **do**
    $h_t \leftarrow \tilde{E}_K(1\|h_b, h_t \oplus A[i])$
    $h_b \leftarrow \pi(h_b) \oplus h_t$
$h_t \leftarrow \tilde{E}_K(3\|h_b, h_t \oplus A[a])$
$h_b \leftarrow \pi(h_b) \oplus h_t$
for $i \in [1..m]$ **do**
    $h_t \leftarrow \tilde{E}_K(5\|h_b, h_t \oplus M[i])$
    $h_b \leftarrow \pi(h_b) \oplus h_t$
$h_t \leftarrow \tilde{E}_K((5+6)\|h_b, h_t)$
$h_b \leftarrow \pi(h_b) \oplus h_t$
$h_t \leftarrow \tilde{E}_K((5+6)\|h_b, h_t)$
$h_b \leftarrow \pi(h_b) \oplus h_t$
$T \leftarrow h_t\|h_b$
$h_t[0] \leftarrow h_t$
$h_b[0] \leftarrow h_b$
for $i \in [1..m]$ **do**
    $h_t[i] \leftarrow \tilde{E}_K(0\|h_b[i-1], h_t[i-1])$
    $h_b[i] \leftarrow \pi(h_b[i-1]) \oplus h_t[i]$
$Z \leftarrow h_t[1]\| \ldots \|h_t[m]$
return $(C := M \oplus Z, T)$

FIGURE 6.15: Encryption of LM-DAE

3. $\mathcal{A}$ then chooses all but one of the blocks of A', (say, the first block $A'[1]$), starts going backwards from T using inverse block cipher calls, and calculates $A'[1]$ such that the starting value 110^{n-2} is reached.

4. $\mathcal{A}$ outputs (K, A, M) and (K', A', M').

A similar attack also holds for ANYDAE, if the function ρ_1 is invertible. As a consequence, both MONDAE and TUESDAE, which are two instances of ANYDAE, are vulnerable to the attack.

6.6.6 LM-DAE

Fig. 6.15 gives the encryption algorithm of LM-DAE [101], another DAE scheme. We first show a CMT_k attack on LM-DAE. Let $\mathcal{A}$ be a CMT_k adversary attacking LM-DAE. It proceeds as follows.

1. $\mathcal{A}$ computes (C, T) from (K, A, M).

2. $\mathcal{A}$ chooses $K' \neq K$, and uses K', C and T to compute M'.

3. $\mathcal{A}$ chooses all but two two blocks of A' (say, $A'[1]$ and $A'[2]$).

$\mathsf{MRO}[\tilde{E}].\mathsf{Enc}(K, N, A = A[1]\|\ldots\|A[a], M = M[1]\|\ldots\|M[m])$

$U \leftarrow K\|N$
$S \leftarrow 0$
for $i \in [1..a]$ **do**
    $S \leftarrow S \oplus \tilde{E}_U((i-1)\|0\|0, A[i])$
for $i \in [1..m]$ **do**
    $S \leftarrow S \oplus \tilde{E}_U((i-1)\|1\|0, M[i])$
$S \leftarrow \tilde{E}_U(0\|2\|0, S \oplus an\|mn)$
$T \leftarrow \lceil S \rceil_t$
for $i \in [1..m]$ **do**
    $Z[i] \leftarrow \tilde{E}_U(0\|0\|1, T\|(i-1))$
return $(C := M \oplus Z, T)$

FIGURE 6.16: Encryption of MRO

4. $\mathcal{A}$ starts to compute towards the IV of the construction (i.e., 0^{2n}). Let the output of the penultimate $\tilde{E}^{-1}$ call be X, and the output of the penultimate π^{-1} call be Y.

5. $\mathcal{A}$ calculates $A'[1]$ and $A'[2]$ as follows.

$$A'[2] = X \oplus Y \oplus \pi(0^n),$$
$$\text{and, } A'[1] = \tilde{E}^{-1}(A'[2] \oplus X).$$

6. $\mathcal{A}$ outputs (K, A, M) and (K', A', M').

6.6.7 MRO, MRS and MRSO

Fig. 6.16 gives the encryption algorithm of MRO (Misuse Resistant Offset) [72]. We first show a CMT_k attack on MRO. Let $\mathcal{A}$ be a CMT_k adversary attacking MRO. It proceeds as follows.

1. $\mathcal{A}$ computes (C, T) from (K, N, A, M), where $C = C[1]\|\ldots\|C[m]$.

2. $\mathcal{A}$ chooses $K' \neq K$ and N', defines $U' = (K', N')$, and computes $M'[i] = \tilde{E}_{U'}(0\|0\|1, T\|(i-1)) \oplus C[i], \forall i \in [m]$.

3. $\mathcal{A}$ computes $W' = \tilde{E}^{-1}_{U'}(0\|2\|0, T\|0) \oplus an\|mn$.

4. $\mathcal{A}$ chooses all but one of the blocks of A' (say, the first block $A'[1]$), and computes $X'[i] = \tilde{E}_{U'}((i-1)\|0\|0, A'[i]), \forall i > 1$, and $Y'[i] = \tilde{E}_{U'}((i-1)\|1\|0, M'[i])$ $\forall i$. Let $V' = W' \oplus \Sigma_{i>1} X'[i] + \Sigma_i Y'[i]$.

$\mathsf{BTM}[E,F].\mathsf{Enc}(K,A,M=M[1]\|\ldots\|M[m])$

1: $L \leftarrow E_K(0)$
2: $U \leftarrow E_K(1)$
3: $S \leftarrow F_{L,U}(A,M)$ ▷ F is a bivariate hash
4: $T \leftarrow E_K(S)$
5: $X \leftarrow T \boxplus U$ ▷ $\boxplus$ denotes half-wise addition modulo $2^{n/2}$
6: **for** $i \in [1..m]$ **do**
7: $\quad Z[i] \leftarrow E_K(X \boxplus (i-1))$
8: $Z \leftarrow Z[1]\|\ldots\|Z[m]$
9: **return** $(C := M \oplus Z, T)$

FIGURE 6.17: Encryption of BTM

5. $\mathcal{A}$ computes $A'[1] = \tilde{E}^{-1}_{U'}(0, V')$.

6. $\mathcal{A}$ outputs (K, N, A, M) and (K', N', A', M').

The same attack strategy works for MRS and MRSO as well.

6.6.8 BTM

Fig. 6.17 gives the encryption algorithm of BTM (Bivariate Tag Mixing) [81]. BTM uses a bi-variate hash F that uses two keys L and U and accepts a vector of inputs $(X_1, \ldots, X_d)$, where X_i is of length $x_i n$ bits for each i. (We skip the definition for incomplete blocks here.) F is defined as

$$F_{L,U}(X_1, \ldots, X_d) := U^{d-1} \cdot f_L(X_1) \oplus \ldots \oplus U \cdot f_L(X_{d-1}) \oplus f_L(X_d),$$

where $f_L(X_i)$ denotes the polynomial defined as

$$f_L(X_i) := 2 \cdot (L^{x_i} \oplus L^{x_i - 1} \cdot X_i[1] \oplus \ldots \oplus L \cdot X_i[x_i - 1] \oplus X_i[x_i]).$$

Thus, for two inputs X_1, X_2 such that X_1 is just one n-bit block, we have

$$F_{L,U}(X_1, X_2) = 2 \cdot U \cdot X_1 \oplus f_L(X_2).$$

Thus, if the associated data is a single block of n bits, line 3 of the algorithm in Fig. 6.17 gives

$$S = 2 \cdot U \cdot A \oplus f_L(M).$$

A CMT_k adversary $\mathcal{A}$ can exploit this as follows.

1. $\mathcal{A}$ first chooses (K, A, M) with a single-block A and computes (C, T) as the output of $\mathsf{BTM}[E, F].\mathsf{Enc}(K, A, M)$, as well as the subkeys $L = E_K(0)$ and $U = E_K(1)$.

2. $\mathcal{A}$ chooses $K' \neq K$ and computes $L' = E_{K'}(0)$, $U' = E_{K'}(1)$. (In the unlikely event that $L' = L, U' = U$, try other values of K' till $L' \neq L$ or $U' \neq U$.)

3. $\mathcal{A}$ executes lines 5-9 of the encryption algorithm in Fig. 6.17 starting from T and U' and obtains Z'.

4. $\mathcal{A}$ computes $M' := C \oplus Z'$ and $S' := E^{-1}_{K'}(T)$.

5. $\mathcal{A}$ computes $A' := 2^{-1} \cdot U^{-1} \cdot (S' \oplus f_{L'}(M'))$.

6. $\mathcal{A}$ outputs (K, A, M) and (K', A', M').

6.7 Summary

In this work, we proposed PACT, a CMT transform for any authenticated encryption. PACT is output-length-preserving, works on any AE scheme and preserves both UNAE and MRAE security of the legacy AE scheme. PACT uses only one call to a collision-resistant unkeyed hash function and one call to a block cipher, so it's efficient from an implementation point of view. We also proposed comPACT, a lighter version of PACT which preserves UNAE security.

Chapter 7

Conclusion and Future Work

This thesis explores various aspects of provable security for cryptographic modes of operation, focusing on hash functions and authenticated encryption schemes in idealised models. In Chapter 3, we revisit the collision security of the ABR hash mode, a tree hash mode more efficient than the Merkle tree. We identify gaps in the existing security proof and provide a proof of the collision security for ABR_3 in the random oracle model.
Chapter 4 focuses on the AEAD security of the Ascon mode. By modelling the Ascon permutation as an ideal permutation, we demonstrate that Ascon is significantly more secure than the Duplex mode in the single-user nonce-respecting setting, allowing for an increased rate and halved tag size while maintaining security.
In Chapter 5, we extend the results for Ascon to the multi-user setting, introducing the LK-Ascon mode with a larger key size to maintain security as the number of users increases. We also establish authenticity security in the nonce-misuse setting for both Ascon and LK-Ascon.
Chapter 6 introduces PACT, a transform that converts any AE scheme into a committing one without increasing the ciphertext size. We also propose comPACT, a lighter version of PACT, which maintains standard AEAD security in the nonce-respecting setting. This chapter underscores the necessity for universal transforms due to the non-committing nature of many authenticated schemes.

7.1 Directions for Future Research

In Chapter 3, we were able to prove the collision security of the ABR_3 hash mode. While we conjecture that ABR_l for $l > 3$ should also exhibit collision resistance

up to the birthday bound, the proof appears to be very challenging with our current approach. An alternative strategy might involve proving the collision security of an optimal linear hash, akin to an extension of the Shrimpton-Stam construction [114], or using a compressing primitive. However, even this method may encounter difficulties in bounding the load for "out-of-order" queries.

In Chapter 4, our results indicate that the Ascon AEAD mode can support an increased rate of 192 bits due to an improved security bound. This suggests a potential for enhanced throughput for Ascon, though this conclusion is drawn within the random permutation model and requires cryptanalytic validation. Notably, Ascon-128 employs 6 rounds of the permutation for a 64-bit rate, while Ascon-128a uses 8 rounds for a 128-bit rate. Therefore, more than 8 rounds might be necessary to maintain security with a 192-bit rate.

Additionally, unlike Ascon, the security bounds for Duplex are not tight. The best-known bounds are of the order $DT/2^c$, whereas the most effective attack, as reported by Gilbert et al. [69], is of the order $DT/2^{3c/2}$. Closing this gap would be a valuable direction for future work.

In Chapter 5, to enhance the multi-user security of Ascon and potentially improve its quantum security, we proposed the LK-Ascon mode. While we demonstrated that LK-Ascon is provably more secure than Ascon in the multi-user setting (with similar security in the single-user setting), further cryptanalytic research is necessary before practical application. Additionally, exploring whether the increased key size offers better protection against quantum adversaries would be an intriguing avenue for investigation.

In Chapter 6, we introduced PACT, the first CMT transform for AEAD that is both universal and output-length-preserving. The primary limitation of our transform is that it generally provides only 64 bits of CMT security, whereas 128 bits would be more beneficial against offline committing attacks. Although there is no known universal CMT transform that guarantees 128 bits of CMT security in practice, Bellare et al. [10] proposed the SC transform, which can reduce ciphertext size without compromising committing security when composed with another suitable CMT transform. Designing a universal CMT transform that can be composed with the SC transform and guarantees 128 bits of CMT security, even with increased ciphertext size, would be a significant achievement. Moreover, developing a significantly lighter universal CMT_k transform compared to PACT or comPACT would be another promising direction for future research.

References

[1] Michel Abdalla, Mihir Bellare, and Gregory Neven. Robust encryption. In Daniele Micciancio, editor, *Theory of Cryptography, 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland, February 9-11, 2010. Proceedings*, volume 5978 of *Lecture Notes in Computer Science*, pages 480–497. Springer, 2010.

[2] Ange Albertini, Thai Duong, Shay Gueron, Stefan Kölbl, Atul Luykx, and Sophie Schmieg. How to abuse and fix authenticated encryption without key commitment. In Kevin R. B. Butler and Kurt Thomas, editors, *31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022*, pages 3291–3308. USENIX Association, 2022.

[3] 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.

[4] Elena Andreeva, Rishiraj Bhattacharyya, and Arnab Roy. Compactness of hashing modes and efficiency beyond merkle tree. In *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part II*, pages 92–123. Springer, 2021.

[5] Elena Andreeva, Andrey Bogdanov, Atul Luykx, Bart Mennink, Nicky Mouha, and Kan Yasuda. How to securely release unverified plaintext in authenticated encryption. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 105–125. Springer, 2014.

[6] 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.

[7] Subhadeep Banik, Andrey Bogdanov, Atul Luykx, and Elmar Tischhauser. SUNDAE: small universal deterministic authenticated encryption for the internet of things. *IACR Trans. Symmetric Cryptol.*, 2018(3):1–35, 2018.

[8] Guy Barwell, Daniel P. Martin, Elisabeth Oswald, and Martijn Stam. Authenticated encryption in the face of protocol and side channel leakage. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 693–723. Springer, 2017.

[9] Mihir Bellare and Viet Tung Hoang. Efficient schemes for committing authenticated encryption. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology - EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30 - June 3, 2022, Proceedings, Part II*, volume 13276 of *Lecture Notes in Computer Science*, pages 845–875. Springer, 2022.

[10] Mihir Bellare and Viet Tung Hoang. Succinctly-committing authenticated encryption. *IACR Cryptol. ePrint Arch.*, page 875, 2024.

[11] Mihir Bellare, Viet Tung Hoang, and Cong Wu. The landscape of committing authenticated encryption. *The Third NIST Workshop on Block Cipher Modes of Operation, October 3-4, 2023.* https://csrc.nist.gov/csrc/media/Events/2023/third-workshop-on-block-cipher-modes-of-operation/documents/accepted-papers/TheLandscapeofCommittingAuthenticatedEncryption.pdf.

[12] Mihir Bellare, Viet Tung Hoang, and Cong Wu. The landscape of committing authenticated encryption. *The Third*

NIST Workshop on Block Cipher Modes of Operation, October 3-4, 2023. https://csrc.nist.gov/Presentations/2023/landscape-of-committing-authenticated-encryption.

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

[14] Mihir Bellare, Ruth Ng, and Björn Tackmann. Nonces are noticed: AEAD revisited. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 235–265. Springer, 2019.

[15] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993*, pages 62–73. ACM, 1993.

[16] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In Alfredo De Santis, editor, *Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111. Springer, 1994.

[17] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures - how to sign with RSA and rabin. In Ueli M. Maurer, editor, *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416. Springer, 1996.

[18] Mihir Bellare and Phillip Rogaway. Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient cryptography. In Tatsuaki Okamoto, editor, *Advances in Cryptology - ASIACRYPT 2000, 6th International Conference on the Theory and Application of Cryptology*

and Information Security, Kyoto, Japan, December 3-7, 2000, Proceedings, volume 1976 of *Lecture Notes in Computer Science*, pages 317–330. Springer, 2000.

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

[20] 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.

[21] Francesco Berti, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. On leakage-resilient authenticated encryption with decryption leakages. *IACR Trans. Symmetric Cryptol.*, 2017(3):271–293, 2017.

[22] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Sponge functions. In *ECRYPT Hash Workshop 2007. Proceedings*, 2007.

[23] 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.

[24] 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.

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

[26] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Keccak. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*, pages 313–314. Springer, 2013.

[27] Arghya Bhattacharjee, Ritam Bhaumik, and Chandranan Dhar. Universal context commitment without ciphertext expansion. *IACR Cryptol. ePrint Arch.*, page 1382, 2024.

[28] Eli Biham and Adi Shamir. Differential cryptanalysis of des-like cryptosystems. In Alfred Menezes and Scott A. Vanstone, editors, *Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings*, volume 537 of *Lecture Notes in Computer Science*, pages 2–21. Springer, 1990.

[29] John Black, Martin Cochran, and Thomas Shrimpton. On the impossibility of highly-efficient blockcipher-based hash functions. In *EUROCRYPT*, volume 3494 of *Lecture Notes in Computer Science*, pages 526–541. Springer, 2005.

[30] John Black, Phillip Rogaway, and Thomas Shrimpton. Black-box analysis of the block-cipher-based hash-function constructions from PGV. In Moti Yung, editor, *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, volume 2442 of *Lecture Notes in Computer Science*, pages 320–335. Springer, 2002.

[31] Avik Chakraborti, Nilanjan Datta, Mridul Nandi, and Kan Yasuda. Beetle family of lightweight and secure authenticated encryption ciphers. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(2):218–241, 2018.

[32] Avik Chakraborti, Nilanjan Datta, Mridul Nandi, and Kan Yasuda. Beetle family of lightweight and secure authenticated encryption ciphers. *Cryptology ePrint Archive*, 2018.

[33] Bishwajit Chakraborty, Nilanjan Datta, and Mridul Nandi. Designing full-rate sponge based AEAD modes. In Anupam Chattopadhyay, Shivam Bhasin, Stjepan Picek, and Chester Rebeiro, editors, *Progress in Cryptology - INDOCRYPT 2023 - 24th International Conference on Cryptology in India, Goa, India, December 10-13, 2023, Proceedings, Part I*, volume 14459 of *Lecture Notes in Computer Science*, pages 89–110. Springer, 2023.

[34] Bishwajit Chakraborty, Chandranan Dhar, and Mridul Nandi. Exact security analysis of ASCON. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the*

Theory and Application of Cryptology and Information Security, Guangzhou, China, December 4-8, 2023, Proceedings, Part III, volume 14440 of *Lecture Notes in Computer Science*, pages 346–369. Springer, 2023.

[35] Bishwajit Chakraborty, Chandranan Dhar, and Mridul Nandi. Tight multi-user security of ascon and its large key extension. In Tianqing Zhu and Yannan Li, editors, *Information Security and Privacy*, pages 57–76, Singapore, 2024. Springer Nature Singapore.

[36] Bishwajit Chakraborty, Ashwin Jha, and Mridul Nandi. On the security of sponge-type authenticated encryption modes. *IACR Trans. Symmetric Cryptol.*, 2020(2):93–119, 2020.

[37] 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.

[38] Debrup Chakraborty, Cuauhtemoc Mancillas López, and Palash Sarkar. Disk encryption: do we need to preserve length? *J. Cryptogr. Eng.*, 8(1):49–69, 2018.

[39] John Chan and Phillip Rogaway. Anonymous AE. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part II*, volume 11922 of *Lecture Notes in Computer Science*, pages 183–208. Springer, 2019.

[40] John Chan and Phillip Rogaway. On committing authenticated-encryption. In Vijayalakshmi Atluri, Roberto Di Pietro, Christian Damsgaard Jensen, and Weizhi Meng, editors, *Computer Security - ESORICS 2022 - 27th European Symposium on Research in Computer Security, Copenhagen, Denmark, September 26-30, 2022, Proceedings, Part II*, volume 13555 of *Lecture Notes in Computer Science*, pages 275–294. Springer, 2022.

[41] Donghoon Chang, Nilanjan Datta, Avijit Dutta, Bart Mennink, Mridul Nandi, Somitra Sanadhya, and Ferdinand Sibleyras. Release of unverified plaintext: Tight unified model and application to ANYDAE. *IACR Trans. Symmetric Cryptol.*, 2019(4):119–146, 2019.

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

[43] Yu Long Chen, Antonio Flórez-Gutiérrez, Akiko Inoue, Ryoma Ito, Tetsu Iwata, Kazuhiko Minematsu, Nicky Mouha, Yusuke Naito, Ferdinand Sibleyras, and Yosuke Todo. Key committing security of AEZ and more. *IACR Trans. Symmetric Cryptol.*, 2023(4):452–488, 2023.

[44] Wonseok Choi, ByeongHak Lee, Jooyoung Lee, and Yeongmin Lee. Toward a fully secure authenticated encryption scheme from a pseudorandom permutation. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part III*, volume 13092 of *Lecture Notes in Computer Science*, pages 407–434. Springer, 2021.

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

[46] Joan Daemen, Silvia Mella, and Gilles Van Assche. Committing authenticated encryption based on SHAKE. *IACR Cryptol. ePrint Arch.*, page 1494, 2023.

[47] 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.

[48] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.

[49] Ivan Damgård. A design principle for hash functions. In *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 416–427. Springer, 1989.

[50] Jean Paul Degabriele, Marc Fischlin, and Jérôme Govinden. The indifferentiability of the duplex and its practical applications. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, December 4-8, 2023, Proceedings,*

Part VIII, volume 14445 of *Lecture Notes in Computer Science*, pages 237–269. Springer, 2023.

[51] Anand Desai. The security of all-or-nothing encryption: Protecting against exhaustive key search. In Mihir Bellare, editor, *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings*, volume 1880 of *Lecture Notes in Computer Science*, pages 359–375. Springer, 2000.

[52] Chandranan Dhar, Yevgeniy Dodis, and Mridul Nandi. Revisiting collision and local opening analysis of ABR hash. In Dana Dachman-Soled, editor, *3rd Conference on Information-Theoretic Cryptography, ITC 2022, July 5-7, 2022, Cambridge, MA, USA*, volume 230 of *LIPIcs*, pages 11:1–11:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[53] Chandranan Dhar, Jordan Ethan, Ravindra Jejurikar, Mustafa Khairallah, Eik List, and Sougata Mandal. Context-committing security of leveled leakage-resilient aead. *IACR Trans. Symmetric Cryptol.*, 2024(2):348–370, 2024.

[54] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theory*, 22(6):644–654, 1976.

[55] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2: Lightweight authenticated encryption and hashing. *J. Cryptol.*, 34(3):33, 2021.

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

[57] Yevgeniy Dodis, Paul Grubbs, Thomas Ristenpart, and Joanne Woodage. Fast message franking: From invisible salamanders to encryptment. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 155–186. Springer, 2018.

[58] Yevgeniy Dodis, Dmitry Khovratovich, Nicky Mouha, and Mridul Nandi. T5: Hashing five inputs with three compression calls. In *2nd Conference*

on Information-Theoretic Cryptography, ITC 2021, July 23-26, 2021, Virtual Conference, pages 24:1–24:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[59] 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.

[60] Morris Dworkin. Recommendation for block cipher modes of operation: Methods and techniques. NIST Special Publication 800-38A, December 2001.

[61] Morris Dworkin. Recommendation for block cipher modes of operation: Galois/counter mode (gcm) and gmac. NIST Special Publication 800-38D, November 2007.

[62] Morris Dworkin. Sha-3 standard: Permutation-based hash and extendable-output functions. Technical Report FIPS PUB 202, National Institute of Standards and Technology, 2015.

[63] Shimon Even and Yishay Mansour. A construction of a cioher from a single pseudorandom permutation. In Hideki Imai, Ronald L. Rivest, and Tsutomu Matsumoto, editors, *Advances in Cryptology - ASIACRYPT '91, International Conference on the Theory and Applications of Cryptology, Fujiyoshida, Japan, November 11-14, 1991, Proceedings*, volume 739 of *Lecture Notes in Computer Science*, pages 210–224. Springer, 1991.

[64] Pooya Farshim, Benoît Libert, Kenneth G. Paterson, and Elizabeth A. Quaglia. Robust encryption, revisited. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *Public-Key Cryptography - PKC 2013 - 16th International Conference on Practice and Theory in Public-Key Cryptography, Nara, Japan, February 26 - March 1, 2013. Proceedings*, volume 7778 of *Lecture Notes in Computer Science*, pages 352–368. Springer, 2013.

[65] Pooya Farshim, Claudio Orlandi, and Razvan Rosie. Security of symmetric primitives under incorrect usage of keys. *IACR Trans. Symmetric Cryptol.*, 2017(1):449–473, 2017.

[66] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor,

Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

[67] Christian Forler, Eik List, Stefan Lucks, and Jakob Wenzel. Efficient beyond-birthday-bound-secure deterministic authenticated encryption with minimal stretch. In Joseph K. Liu and Ron Steinfeld, editors, *Information Security and Privacy - 21st Australasian Conference, ACISP 2016, Melbourne, VIC, Australia, July 4-6, 2016, Proceedings, Part II*, volume 9723 of *Lecture Notes in Computer Science*, pages 317–332. Springer, 2016.

[68] Yitchak Gertner and Amir Herzberg. Committing encryption and publicly-verifiable signcryption. *IACR Cryptol. ePrint Arch.*, page 254, 2003.

[69] 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.

[70] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. On the cryptographic applications of random functions. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, volume 196 of *Lecture Notes in Computer Science*, pages 276–288. Springer, 1984.

[71] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

[72] Robert Granger, Philipp Jovanovic, Bart Mennink, and Samuel Neves. Improved masking for tweakable blockciphers with applications to authenticated encryption. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 263–293. Springer, 2016.

[73] Paul Grubbs, Jiahui Lu, and Thomas Ristenpart. Message franking via committing authenticated encryption. In Jonathan Katz and Hovav Shacham,

editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III*, volume 10403 of *Lecture Notes in Computer Science*, pages 66–97. Springer, 2017.

[74] Shay Gueron and Yehuda Lindell. GCM-SIV: full nonce misuse-resistant authenticated encryption at under one cycle per byte. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pages 109–119. ACM, 2015.

[75] Shoichi Hirose. Compactly committing authenticated encryption using tweakable block cipher. In Miroslaw Kutylowski, Jun Zhang, and Chao Chen, editors, *Network and System Security - 14th International Conference, NSS 2020, Melbourne, VIC, Australia, November 25-27, 2020, Proceedings*, volume 12570 of *Lecture Notes in Computer Science*, pages 187–206. Springer, 2020.

[76] Shoichi Hirose and Kazuhiko Minematsu. Compactly committing authenticated encryption using encryptment and tweakable block cipher. *IACR Cryptol. ePrint Arch.*, page 1670, 2022.

[77] Shoichi Hirose and Kazuhiko Minematsu. A formal treatment of envelope encryption. *IACR Cryptol. ePrint Arch.*, page 1727, 2023.

[78] Viet Tung Hoang, Ted Krovetz, and Phillip Rogaway. Robust authenticated-encryption AEZ and the problem that it solves. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 15–44. Springer, 2015.

[79] Thomas Holenstein, Robin Künzler, and Stefano Tessaro. The equivalence of the random oracle model and the ideal cipher model, revisited. In Lance Fortnow and Salil P. Vadhan, editors, *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 89–98. ACM, 2011.

[80] Tetsu Iwata and Kazuhiko Minematsu. Stronger security variants of GCM-SIV. *IACR Trans. Symmetric Cryptol.*, 2016(1):134–157, 2016.

[81] Tetsu Iwata and Kan Yasuda. BTM: A single-key, inverse-cipher-free mode for deterministic authenticated encryption. In Michael J. Jacobson Jr., Vincent Rijmen, and Reihaneh Safavi-Naini, editors, *Selected Areas in Cryptography, 16th Annual International Workshop, SAC 2009, Calgary, Alberta, Canada, August 13-14, 2009, Revised Selected Papers*, volume 5867 of *Lecture Notes in Computer Science*, pages 313–330. Springer, 2009.

[82] Jérémy Jean, Ivica Nikolic, and Thomas Peyrin. Joltik v1.3. *Submission to CAESAR Competition*, 2013 - 2019. https://competitions.cr.yp.to/round2/joltikv13.pdf.

[83] Jérémy Jean, Ivica Nikolic, Thomas Peyrin, and Yannick Seurin. Deoxys v1.41. *Submission to CAESAR Competition*, 2013 - 2019. https://competitions.cr.yp.to/round3/deoxysv141.pdf.

[84] Jérémy Jean, Ivica Nikolic, Thomas Peyrin, and Yannick Seurin. The deoxys AEAD family. *J. Cryptol.*, 34(3):31, 2021.

[85] 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.

[86] 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.

[87] Joe Kilian and Phillip Rogaway. How to protect DES against exhaustive key search (an analysis of DESX). *J. Cryptol.*, 14(1):17–35, 2001.

[88] Lars R. Knudsen and David A. Wagner. Integral cryptanalysis. In Joan Daemen and Vincent Rijmen, editors, *Fast Software Encryption, 9th International Workshop, FSE 2002, Leuven, Belgium, February 4-6, 2002, Revised Papers*, volume 2365 of *Lecture Notes in Computer Science*, pages 112–127. Springer, 2002.

[89] Juliane Krämer, Patrick Struck, and Maximiliane Weishäupl. Committing authenticated encryption: Sponges vs. block-ciphers in the case of the NIST LWC finalists. *IACR Cryptol. ePrint Arch.*, page 1525, 2023.

[90] Julia Len, Paul Grubbs, and Thomas Ristenpart. Partitioning oracle attacks. In Michael D. Bailey and Rachel Greenstadt, editors, *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, pages 195–212. USENIX Association, 2021.

[91] Moses D. Liskov, Ronald L. Rivest, and David A. Wagner. Tweakable block ciphers. *J. Cryptol.*, 24(3):588–613, 2011.

[92] Mitsuru Matsui. Linear cryptanalysis method for DES cipher. In Tor Helleseth, editor, *Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer, 1993.

[93] Sanketh Menda, Julia Len, Paul Grubbs, and Thomas Ristenpart. Context discovery and commitment attacks - how to break ccm, eax, siv, and more. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part IV*, volume 14007 of *Lecture Notes in Computer Science*, pages 379–407. Springer, 2023.

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

[95] 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.

[96] Bart Mennink and Bart Preneel. Efficient parallelizable hashing using small non-compressing primitives. *Int. J. Inf. Sec.*, 15(3):285–300, 2016.

[97] Ralph C. Merkle. Protocols for public key cryptosystems. In *IEEE Symposium on Security and Privacy*, pages 122–134. IEEE Computer Society, 1980.

[98] Ralph C. Merkle. One way hash functions and DES. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 428–446. Springer, 1989.

[99] Yusuke Naito, Yu Sasaki, and Takeshi Sugawara. Kivr: Context-committing authenticated encryption using plaintext redundancy and application to gcm and variants. *The Third NIST Workshop on Block Cipher Modes of Operation, October 3-4, 2023.* https://csrc.nist.gov/csrc/media/Events/2023/third-workshop-on-block-cipher-modes-of-operation/documents/accepted-papers/KIVRContextCommittingAuthenticatedEncryption.pdf.

[100] Yusuke Naito, Yu Sasaki, and Takeshi Sugawara. Kivr: Context-committing authenticated encryption using plaintext redundancy and application to gcm and variants. *The Third NIST Workshop on Block Cipher Modes of Operation, October 3-4, 2023.* https://csrc.nist.gov/Presentations/2023/kivr.

[101] Yusuke Naito, Yu Sasaki, and Takeshi Sugawara. LM-DAE: low-memory deterministic authenticated encryption for 128-bit security. *IACR Trans. Symmetric Cryptol.*, 2020(4):1–38, 2020.

[102] NIST. Submission requirements and evaluation criteria for the Lightweight Cryptography Standardisation Process, 2018. https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/final-lwc-submission-requirements-august2018.pdf.

[103] 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.

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

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

[106] Goutam Paul and Subhamoy Maitra. *RC4 Stream Cipher and Its Variants.* CRC Press, Inc., USA, 1st edition, 2011.

[107] Olivier Pereira, François-Xavier Standaert, and Srinivas Vivek. Leakage-resilient authentication and encryption from symmetric cryptographic primitives. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pages 96–108. ACM, 2015.

[108] Thomas Peyrin and Yannick Seurin. Counter-in-tweak: Authenticated encryption modes for tweakable block ciphers. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 33–63. Springer, 2016.

[109] Phillip Rogaway. Nonce-based symmetric encryption. In Bimal K. Roy and Willi Meier, editors, *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers*, volume 3017 of *Lecture Notes in Computer Science*, pages 348–359. Springer, 2004.

[110] Phillip Rogaway and Thomas Shrimpton. A provable-security treatment of the key-wrap problem. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, volume 4004 of *Lecture Notes in Computer Science*, pages 373–390. Springer, 2006.

[111] Phillip Rogaway and John P. Steinberger. Constructing cryptographic hash functions from fixed-key blockciphers. In *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 433–450. Springer, 2008.

[112] Phillip Rogaway and John P. Steinberger. Security/efficiency tradeoffs for permutation-based hashing. In *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 220–236. Springer, 2008.

[113] Claude E. Shannon. Communication theory of secrecy systems. *Bell Syst. Tech. J.*, 28(4):656–715, 1949.

[114] Thomas Shrimpton and Martijn Stam. Building a collision-resistant compression function from non-compressing primitives. In *ICALP (2)*, volume 5126 of *Lecture Notes in Computer Science*, pages 643–654. Springer, 2008.

[115] Martijn Stam. Beyond uniformity: Better security/efficiency tradeoffs for compression functions. In *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, pages 397–412. Springer, 2008.

[116] John P. Steinberger. Stam's collision resistance conjecture. In *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 597–615. Springer, 2010.

[117] John P. Steinberger, Xiaoming Sun, and Zhe Yang. Stam's conjecture and threshold phenomena in collision resistance. In *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 384–405. Springer, 2012.

[118] Patrick Struck and Maximiliane Weishäupl. Constructing committing and leakage-resilient authenticated encryption. Cryptology ePrint Archive, Paper 2024/190, 2024. https://eprint.iacr.org/2024/190.

[119] David Wagner. A Generalized Birthday Problem. In Moti Yung, editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 288–303. Springer, 2002.