INDIAN STATISTICAL INSTITUTE KOLKATA

Design and Analysis of Authenticated Encryption Modes

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

by

Arghya Bhattacharjee

Supervised By

Mridul Nandi

September 11, 2024

Statement

I declare that this thesis titled "Design and Analysis of Authenticated Encryption Modes" and the work presented in it are my own, and were produced by my own original research. I confirm that this research was done wholly while in candidature for a doctoral degree at Indian Statistical Institute; that no part of this thesis has previously been submitted for a degree or any other qualification at this institute or any other institution; that wherever I have used or developed on the published work of others, this is always clearly attributed; that wherever I have quoted from the work of others, the source is always clearly cited; that with the exception of such quotations, this thesis is entirely my own work; that I have acknowledged all main sources of help; and that wherever the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself, and have obtained explicit permission from the others to use the joint work in this thesis.

Arghya Bhattacharjee

September 11, 2024

"Remember this, my friends: there are no such things as bad plants or bad men. There are only bad cultivators."

Victor Hugo, Les Misérables

Abstract

This thesis proposes and analyses the security of a few symmetric key modes. The first three of them are NAEAD modes, named Oribatida, ISAP+ and OCB+. Oribatida is lightweight, sponge-based, INT-RUP secure and achieves better than the default PRF security of a keyed sponge. ISAP+ is an instance of a generic EtHM involving a PRF and a hash, a generalisation of ISAP-type modes. The generic sponge hash of ISAP is replaced with a feed-forward variant of it in ISAP+, which results in better security. OCB+ uses OTBC-3 (a nonce-respecting BBB secure offset-based tweakable block-cipher) in an OCB-like mode to achieve BBB privacy. We conclude with a BBB secure NE mode named CENCPP*, which is a public permutation-based variant of the block-cipher-based mode CENC as well as a variable output length version of SoEM. All the relevant security proofs have been done using a method named Coefficients H Technique.

List of Publications

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

[1] Bhattacharjee, Arghya, López, Cuauhtemoc Mancillas, List, Eik and Nandi, Mridul. "The Oribatida v1.3 Family of Lightweight Authenticated Encryption Schemes" Journal of Mathematical Cryptology, vol. 15, no. 1, 2021, pp. 305-344. https://doi.org/10.1515/jmc-2020-0018

[2] Bhattacharjee, A., Chakraborti, A., Datta, N., Mancillas-López, C., Nandi, M. (2022). ISAP+: ISAP with Fast Authentication. In: Isobe, T., Sarkar, S. (eds) Progress in Cryptology – INDOCRYPT 2022. INDOCRYPT 2022. Lecture Notes in Computer Science, vol 13774. Springer, Cham. https://doi.org/10.1007/978-3-031-22912-1_9

[3] Bhattacharjee, A., Bhaumik, R., Nandi, M. (2022). Offset-Based BBB-Secure Tweakable Block-ciphers with Updatable Caches. In: Isobe, T., Sarkar, S. (eds) Progress in Cryptology – INDOCRYPT 2022. INDOCRYPT 2022. Lecture Notes in Computer Science, vol 13774. Springer, Cham. https://doi.org/10.1007/978-3-031-22912-1_8

[4] Bhattacharjee, A., Dutta, A., List, E., Nandi, M. CENCPP*: beyond-birthday-secure encryption from public permutations. Des. Codes Cryptogr. 90, 1381–1425 (2022). https://doi.org/10.1007/s10623-022-01045-z

Acknowledgment

I begin by taking this opportunity to acknowledge the contribution of my parents in my life. They are the fundamental reason behind all my achievements, however humble or grand these might be. I also thank the other members of my loving family—my younger brother, my maternal uncle, my (late) grandfather, my (late) grandmother, my (late) maternal grandfather and my maternal grandmother—for their unconditional support and uncompromising hard work thanks to which my life so far has been a delightfully happy ride.

Next, I extend my sincere gratitude to my supervisor Mridul Nandi. Since the day I first met him, he has been relentlessly supportive and encouraging as my true friend, philosopher and guide; I could not have imagined researching and writing this thesis without him.

I further thank all my teachers, seniors, peers, and juniors in our institute for creating a disarmingly friendly and productive atmosphere around me where I never had to hesitate before approaching anyone at any time, however odd, with any questions, however silly.

I remain deeply indebted to all the office staff and other workers in our institute, who work behind the scenes day in and day out, with sincerity and diligence, to keep the academic machinery running smoothly, so we can go about our research work in peace. They are surely the heroes we need, even if we do little to deserve them.

I end with a shout out to my other family members and relatives, my friends and neighbours, and all the people I know or don't know, who have, through their kind words and honest deeds, touched my life and made me the person I am today.

Arghya Bhattacharjee

September 11, 2024

Contents

List of Figures

List of Tables

List of Algorithms

Dedicated to my beloved parents

Chapter 1

Introduction

In this chapter, we try to put the thesis to context. We start with Section 1.1 which tries to motivate the work of the thesis. After that, Section 1.2 introduces the basic terminologies. Section 1.3 peeks into the history of the subject. Section 1.4 discusses some basic elements of symmetric-key cryptography. Section 1.5 discusses a little about the recently concluded NIST lightweight cryptography project. Finally, Section 1.6 gives an outline of the fundamental contribution of this thesis.

1.1 Motivation of the Thesis

The major underlying theme of this thesis is authenticated encryption (AE). Chapters 3, 4 and 5 propose three new AE modes and analyse their security. In the following, we discuss why we are interested in authenticated encryption in general. (Chapter 6 diverts from this theme and proposes an encryption mode.)

When someone wants to send a message to someone else over an insecure channel, they may want to achieve something more than just confidentiality of the message because the adversary can try to do something else as well, apart from trying to extract information from the intercepted message. They can try to tamper with the message as well. In that case, the sender may rightfully want to let the receiver know about the authenticity of the message. An "authentic" message is a message that has not been modified by anyone who doesn't have the secret key.

An encryption mode only provides data privacy (or confidentiality), whereas a Message Authentication Code (MAC) only provides data authenticity (or integrity). It should be apparent by now that an AE mode is used to provide both securities to the message.

A natural question at this point should be why one should use an AE mode in place of using an encryption mode and a MAC. The reason is that an AE uses a single key and is more robust than the combination of an encryption mode and a MAC in the sense that one needs to keep fewer things in mind while performing authenticated encryption. Usually, the only thing that one should follow is that it should use a "non-repeating" IV (or in one word, a nonce) for each encryption. On the other hand, the combination of an encryption mode and a MAC uses two keys, one for each module, and there are many ways one can get it wrong if one simply combines an encryption mode and a MAC in any arbitrary way. Here, we'd like to mention that the first strong interest in designing AE modes was sparked by the work of Charanjit S. Jutla [1, 2].

Now, as the sender transforms the message or the plaintext to its corresponding ciphertext using an AE mode, it may want to send some additional information with the ciphertext. When using an AE mode to secure a network protocol, for example, this information can include addresses, ports, sequence numbers, protocol version numbers, and other fields that indicate how the plaintext or the ciphertext should be handled, forwarded or processed. Though this information does not need to be kept confidential, in many situations, it is desirable to authenticate it. This additional information is called associated data, and an AEAD (Authenticated Encryption with Associated Data) mode is used to provide data confidentiality to the message and data authenticity to both the message and the associated data. At this point, it is worth mentioning that the need to handle associated data when using an AE mode was first pointed out by Burt Kaliski to Phillip Rogaway [3] over personal communication.

It has been a long time since those early days, and the AEAD schemes are used in a wide range of applications. Recently, a group of international cryptologic researchers organised a competition named CAESAR (Competition for Authenticated Encryption: Security, Applicability, and Robustness) to encourage the design of AEAD schemes [4]. Also, the National Institute of Standards and Technology (NIST) conducted a project to standardise a lightweight AEAD scheme [5]. This thesis primarily explores this research area, and in doing so, it proposes a new AEAD scheme and new variations of a couple of existing AEAD schemes. We discuss the NIST lightweight cryptography project in more detail in Section 1.5 since two of our contributory chapters are related to that.

1.2 Understanding the Terminologies

"Cryptography" (from Ancient Greek: κρυπτός, Romanised: kryptós, English Meaning: "hidden" or "secret", and Ancient Greek: γράφειν, Romanised: gráphein, English Meaning: "to write") is the study of the techniques by which information can be concealed that can later be recovered by legitimate users and is either impossible or computationally infeasible for unauthorised persons to do so. On the other hand, "Cryptanalysis" (from kryptós, and Ancient Greek: ανάλυση, Romanised: analýein, English Meaning: "to analyse") is the study of techniques by which unauthorised persons can recover cryptographically concealed information. "Cryptology" (from kryptós, and Ancient Greek: λογία, Romanised: logia, English Meaning: "study") is often and mistakenly considered a synonym for "Cryptography" and occasionally for "Cryptanalysis", but specialists in the field have for years adopted the convention that "Cryptology" is the more inclusive term, encompassing both "Cryptography" and "Cryptanalysis". Moreover, RFC 2828 advises that "Steganography" (from Ancient Greek: στεγανός, Romanised: steganós, English Meaning: "covered" or "concealed", and Ancient Greek: γραφή, Romanised: graphia, English Meaning: "writing") (the practice of representing information within another message or physical object, in such a manner that the presence of the information is not evident to human inspection) is sometimes included in "Cryptology". The first use of the term "cryptograph" dates back to the 19th century - originating from "The Gold-Bug", a story by Edgar Allan Poe, published in 1843.

1.3 A Brief History

1.3.1 Antiquity, Medieval, Pre World War Cryptography[1]

The earliest known use of cryptography is found in non-standard hieroglyphs carved into the wall of a tomb from the Old Kingdom of Egypt circa 1900 BCE. Subsequent instances of application of cryptography include some clay tablets from Mesopotamia, one dated near 1500 BCE, simple monoalphabetic substitution ciphers (such as the Atbash cipher) used by the Hebrew scholars beginning perhaps around 600 to 500 BCE, Mlecchita Vikalpa or "the art of understanding writing in cypher, and the writing of words in a peculiar way" documented in the Kama Sutra in India around 400 BCE to 200 CE, parts of the Egyptian demotic Greek

[1] https://en.wikipedia.org/wiki/History_of_cryptography

Magical Papyri dated from the 100s BCE to the 400s CE, scytale transposition cipher used by the Spartan military, another Greek method developed by Polybius (now called the "Polybius Square") and the Caesar cipher and its variations by the Romans.

David Kahn notes in The Codebreakers that modern cryptology originated among the Arabs, the first people to systematically document cryptanalytic methods. Some of the supporting pieces of evidence are the Book of Cryptographic Messages written by Al-Khalil (717 – 786 CE), the invention of the frequency analysis technique for breaking monoalphabetic substitution ciphers and a book on cryptography entitled Risalah fi Istikhraj al-Mu'amma (Manuscript for the Deciphering Cryptographic Messages) by Al-Kindi (801 - 873 CE), an Arab mathematician, and an important contribution of Ibn Adlan (1187 – 1268 CE) on sample size for use of frequency analysis. Ahmad al-Qalqashandi (1355 – 1418 CE) wrote the Subh al-a 'sha, a 14-volume encyclopedia which included a section on cryptology. This information was attributed to Ibn al-Durayhim (1312 - 1361 CE) who is also credited with an exposition on and a worked example of cryptanalysis, including the use of tables of letter frequencies and sets of letters which cannot occur together in one word.

In early medieval England between the years 800–1100 CE, substitution ciphers were frequently used by scribes as a playful and clever way to encipher notes, solutions to riddles, and colophons. This period saw vital and significant cryptographic experimentation in the West. The earliest example of the homophonic substitution cipher is the one used by the Duke of Mantua in the early 1400s CE. The polyalphabetic cipher was most clearly explained by Leon Battista Alberti around 1467 CE, for which he was called the "father of Western cryptology". Johannes Trithemius, in his work Poligraphia, invented the tabula recta, a critical component of the Vigenère cipher. Trithemius also wrote the Steganographia. The French cryptographer Blaise de Vigenère devised a practical polyalphabetic system which bears his name, the Vigenère cipher.

Cryptography, cryptanalysis, and secret-agent/courier betrayal featured in the Babington plot during the reign of Queen Elizabeth I which led to the execution of Mary, Queen of Scots. Robert Hooke suggested in the chapter Of Dr. Dee's Book of Spirits, that John Dee made use of Trithemian steganography, to conceal his communication with Queen Elizabeth I. The chief cryptographer of King Louis XIV of France was Antoine Rossignol; he and his family created what is known as the Great Cipher because it remained unsolved from its initial use until 1890

CE, when French military cryptanalyst, Étienne Bazeries solved it. An encrypted message from the time of the Man in the Iron Mask (decrypted just prior to 1900 CE by Étienne Bazeries) has shed some, regrettably non-definitive, light on the identity of that real, if legendary and unfortunate, prisoner.

An example of something more than ad hoc approaches to cryptanalysis is Charles Babbage's Crimean War era work on mathematical cryptanalysis of polyalphabetic ciphers, redeveloped and published somewhat later by the Prussian Friedrich Kasiski. Understanding of cryptography at this time typically consisted of hard-won rules of thumb; see, for example, Auguste Kerckhoffs' cryptographic writings in the latter 19th century. Edgar Allan Poe used systematic methods to solve ciphers in the 1840s. In particular, he placed a notice of his abilities in the Philadelphia paper Alexander's Weekly (Express) Messenger, inviting submissions of ciphers, most of which he proceeded to solve. His success created a public stir for some months. He later wrote an essay on methods of cryptography, which proved useful as an introduction for novice British cryptanalysts attempting to break German codes and ciphers during World War I, and a famous story, The Gold-Bug, in which cryptanalysis was a prominent element.

Cryptography and its misuse were involved in the execution of Mata Hari and in Dreyfus' conviction and imprisonment, both in the early 20th century. Cryptographers were also involved in exposing the machinations which had led to the Dreyfus affair; Mata Hari, in contrast, was shot.

1.3.2 World War I Cryptography[2]

In World War I the Admiralty's Room 40 broke German naval codes and played an important role in several naval engagements during the war, notably in detecting major German sorties into the North Sea that led to the battles of Dogger Bank and Jutland as the British fleet was sent out to intercept them. However, its most important contribution was probably in decrypting the Zimmermann Telegram, a cable from the German Foreign Office sent via Washington to its ambassador Heinrich von Eckardt in Mexico which played a major part in bringing the United States into the war.

In 1917, Gilbert Vernam proposed a teleprinter cipher in which a previously prepared key, kept on paper tape, is combined character by character with the plaintext message to produce the cyphertext. This led to the development of electromechanical devices as cipher machines, and to the only unbreakable cipher, the

[2] https://en.wikipedia.org/wiki/World_War_I_cryptography

one-time pad.
During the 1920s, Polish naval officers assisted the Japanese military with code and cipher development.
Mathematical methods proliferated in the period prior to World War II (notably in William F. Friedman's application of statistical techniques to cryptanalysis and cipher development and in Marian Rejewski's initial break into the German Army's version of the Enigma system in 1932).

1.3.3 World War II Cryptography[3]

Cryptography was used extensively during World War II because of the importance of radio communication and the ease of radio interception. The nations involved fielded a plethora of code and cipher systems, many of the latter using rotor machines. As a result, the theoretical and practical aspects of cryptanalysis, or codebreaking, were much advanced.
Possibly the most important codebreaking event of the war was the successful decryption by the Allies of the German "Enigma" Cipher. The first break into Enigma was accomplished by the Polish Cipher Bureau around 1932; the techniques and insights used were passed to the French and British Allies just before the outbreak of the war in 1939. They were substantially improved by British efforts at Bletchley Park during the war. The decryption of the Enigma Cipher allowed the Allies to read important parts of German radio traffic on important networks and was an invaluable source of military intelligence throughout the war. Intelligence from this source and other high-level sources, such as Cryptanalysis of the Lorenz cipher, was eventually called Ultra.
A similar break into the most secure Japanese diplomatic cipher, designated Purple by the US Army Signals Intelligence Service, started before the US entered the war. The product from this source was called Magic.
On the other side, German code breaking in World War II achieved some notable successes cracking British naval and other ciphers.

1.3.4 Modern Cryptography[4]

Encryption in modern times is achieved by using algorithms that have a key to encrypt and decrypt information. There are two main types of cryptosystems:

[3] https://en.wikipedia.org/wiki/World_War_II_cryptography
[4] https://en.wikipedia.org/wiki/Cryptography

symmetric and asymmetric. In symmetric systems, the only ones known until the 1970s, the same secret key encrypts and decrypts a message. Data manipulation in symmetric systems is significantly faster than in asymmetric systems. Asymmetric systems use a "public key" to encrypt a message and a related "private key" to decrypt it. The advantage of asymmetric systems is that the public key can be freely published, allowing parties to establish secure communication without having a shared secret key. In practice, asymmetric systems are used to first exchange a secret key, and then secure communication proceeds via a more efficient symmetric system using that key. Examples of asymmetric systems include Diffie–Hellman key exchange, RSA (Rivest–Shamir–Adleman), ECC (Elliptic Curve Cryptography), and Post-quantum cryptography. Secure symmetric algorithms include the commonly used AES (Advanced Encryption Standard) which replaced the older DES (Data Encryption Standard).

Poor designs and implementations are sometimes adopted, and there have been important cryptanalytic breaks of deployed cryptosystems in recent years. Notable examples of broken crypto designs include the first Wi-Fi encryption scheme WEP, the Content Scrambling System used for encrypting and controlling DVD use, the A5/1 and A5/2 ciphers used in GSM cell phones, and the CRYPTO1 cipher used in the widely deployed MIFARE Classic smart cards from NXP Semiconductors, a spun-off division of Philips Electronics. All of these are symmetric ciphers. Thus far, not one of the mathematical ideas underlying public key cryptography has been proven to be "unbreakable", and so some future mathematical analysis advances might render systems relying on them insecure. While few informed observers foresee such a breakthrough, the key size recommended for security as best practice keeps increasing as increased computing power required for breaking codes becomes cheaper and more available. Quantum computers, if ever constructed with enough capacity, could break existing public key algorithms and efforts are underway to develop and standardise post-quantum cryptography. Even without breaking encryption in the traditional sense, side-channel attacks can be mounted that exploit information gained from the way a computer system is implemented, such as cache memory usage, timing information, power consumption, electromagnetic leaks or even sounds emitted. Newer cryptographic algorithms are being developed that make such attacks more difficult.

1.4 Symmetric-Key Cryptography

As the works in the thesis are restricted to the symmetric-key cryptography setup, we'd like to discuss some basic elements of the same before going into the original contribution of the thesis. In symmetric-key cryptography, the same cryptographic keys are used for both the encryption of plaintext and the decryption of ciphertext. Symmetric-key encryption uses either block-ciphers or stream-ciphers as primitives and builds different cryptographic modes on top of those primitives to achieve different security goals.

- **Block-ciphers** take a number of bits and encrypt them in a single unit, padding the plaintext to achieve a multiple of the block size. A block-cipher consists of two algorithms, one for encryption, and the other for decryption, which must be the inverse of the encryption algorithm. Each algorithm accepts two inputs: an input block and a key, and yields an output block of length equal to that of the input block. For each key, the block-cipher is a permutation (a bijective mapping) over the set of input blocks. Each key selects one permutation from the set of $(2^n)!$ possible permutations where n is the block size in bits. The AES (Advanced Encryption Standard) algorithm, approved by NIST in December 2001, uses 128-bit blocks.

  A more generalised version of block-ciphers is called **tweakable block-ciphers**. A tweakable block-cipher accepts a third input called the tweak along with the key and the input block. The tweak, along with the key, selects the permutation computed by the tweakable block-cipher. At times, changing the tweak is sufficiently lightweight compared to a usually fairly expensive key setup operation.

- **Stream-ciphers** encrypt the digits (typically bytes), or letters (in substitution ciphers) of a message one at a time. An example is ChaCha20. Note that none of the symmetric key modes proposed and analysed in the thesis uses a stream-cipher as the underlying primitive.

Some of the basic security goals of some common cryptographic modes are as follows. Note that there are other security goals as well, but we haven't dealt with them in the thesis.

- An encryption mode ensures privacy, i.e., it's tough to distinguish the ciphertext corresponding to the plaintext from a random bit string of equal length.

- An authentication mode ensures authenticity, i.e., it's easy to detect any attempt to change the ciphertext by an adversary at the receiver end. MACs are used to achieve this goal.
- An authenticated encryption mode ensures both privacy and authenticity.
- A cryptographic hash function ensures the following three security goals.
  - Collision resistance, i.e., it's tough to find two inputs with equal hash value.
  - Pre-image Resistance, i.e., given a hash value, it's tough to find an input with that hash value.
  - Second Pre-image Resistance, i.e., given an input, it's tough to find another input with a hash value equal to that of the given input.

1.5 NIST Lightweight Cryptography Project

As it's relevant to one of our proposed authenticated encryption modes, we'd like to talk a little about the NIST lightweight cryptography project. In August 2018, NIST published a call for algorithms (test vector generation code) to be considered for lightweight cryptographic standards with AEAD and optional hashing functionalities. The AEAD requirements were as follows.

- The submitters are allowed to submit a family of AEAD algorithms, where members of the family may vary in external parameters (e.g., key length, nonce length), or in internal parameters (e.g., number of rounds, or state size).
- The family shall include at most 10 members.
- An AEAD algorithm shall not specify key lengths that are smaller than 128 bits.
- Cryptanalytic attacks on an AEAD algorithm shall require at least 2^{112} computations on a classical computer in a single-key setting.
- The family shall include one primary member with the following properties.
  - It will have a key length of at least 128 bits.
  - It will have a nonce length of at least 96 bits.

- It will have a tag length of at least 64 bits.
- The limits on the input sizes (plaintext, associated data, and the amount of data that can be processed under one key) for this member shall not be smaller than $2^{50} - 1$ bytes.

If the family supports a key size larger than 128 bits, it is recommended that at least one member has the following properties.

- It will have a key size of 256 bits.
- Cryptanalytic attacks on this member shall require at least 2^{224} computations on a classical computer in a single-key setting.

NIST received 57 submissions to be considered for standardisation. After the initial review of the submissions, 56 were selected as Round 1 candidates. Of the 56 Round 1 candidates, 32 were selected to advance to Round 2. On February 7, 2023, NIST announced the selection of the Ascon family [6] for lightweight cryptography standardisation.

1.6 Contributions

In the following, we briefly give an overview of the contributory chapters of this thesis. Note that we might have used terms which might need further explanation. Please refer to Chapter 2 and/or the chapter corresponding to the particular work for every such explanation.

1.6.1 Oribatida

The third chapter contains the first contribution of this thesis, which is a sponge-based NAEAD (Nonce-based Authenticated Encryption with Associated Data) mode named Oribatida. Oribatida is lightweight as well as both NAEAD and INT-RUP secure. The security bounds are shown to be tight. The NAEAD security of Oribatida is better than the traditional security of sponge-based NAEAD modes, which allows it to be instantiated with lighter permutations. The INT-RUP security of Oribatida depends only on the number of online or construction queries which eliminates the dependency on the number of offline or primitive queries. This is desirable in real life as the number of offline queries is usually much greater than the number of online queries. In this chapter, we instantiate Oribatida with a new lightweight permutation named SimP. Oribatida was selected as a round 2

candidate of the NIST lightweight cryptography project. This chapter is based on the publication [7].

1.6.2 ISAP+

The next chapter contains the second contribution of this thesis, which is another sponge-based NAEAD mode named ISAP+. Initially, we propose a permutation-based generic EtHM (Encrypt then Hash based MAC) type NAEAD mode using a PRF (Pseudo Random Function) (say F) and a hash function (say H) which is essentially a generalisation of ISAP [8, 9] type constructions. ISAP was one of the finalists of the NIST lightweight cryptography project. We show that the NAEAD security of EtHM can be expressed in terms of the PRF security of F and the 2PI+ security of H. The 2PI+ security is a new security notion defined in the chapter. Then, we show that the generic sponge hash (used in ISAP) can be replaced by a feed-forward variant of it (used in ISAP+) with better 2PI+ security. This results in an improved throughput at the ciphertext absorption phase of the hash. An older version of ISAP+ (named FEASP) has been selected as the second runner-up of the Light-Weight Cipher Design Challenge 2020, organised by the National Centre of Excellence (CoE) in collaboration with R. C. Bose Centre for Cryptology and Security - ISI Kolkata. This chapter is based on the publication [10], and the full version is available at [11].

1.6.3 OCB+

The next chapter contains the third contribution of this thesis, which is another NAEAD mode named OCB+. We observe that a nonce-respecting tweakable block-cipher is the building block for OCB [12], which is an NAEAD mode with birthday-bound privacy security and beyond-birthday-bound (BBB) authenticity security [13]. We propose a nonce-respecting BBB-secure tweakable block-cipher named OTBC-3 and use it in an OCB like mode named OCB+. We show that OCB+ has both BBB privacy security and BBB authenticity security. This chapter is based on the publication [14], and the full version is available at [15].

1.6.4 CENCPP*

The next chapter contains the fourth contribution of this thesis, which is a BBB-secure NE (Nonce based Encryption) mode named CENCPP*, which is a public permutation based variant of the block-cipher based mode CENC [16] as well as a

variable output length version of Sum-of-Even-Mansour (SoEM) [17]. CENCPP*$[w]$ uses two independent secret keys with a general key-scheduling matrix and $(w+1)$ independent permutations. We instantiate the key-scheduling matrix in a particular way and call that instance CENCPP$[w]$. We also propose a domain-separated single permutation variant of CENCPP$[w]$ and call that variant DS-CENCPP$[w]$. This chapter is based on the publication [18], and the full version is available at [19].

Chapter 2

Preliminaries

In this chapter, we try to build the foundation for the rest of the thesis by setting up the notions and notations that we'll use from time to time in the following chapters. Section 2.1 introduces the general notations. The rest of the sections in this chapter are dedicated to the notions relevant to this thesis. Sections 2.2, 2.3, and 2.4 discuss the distinguishing advantage of an adversary and the security notions of NE and NAEAD, respectively. Finally, Section 2.5 mentions the famous Coefficients H Technique that'll be used multiple times in the rest of the thesis.

2.1 General Notations

We usually use lowercase letters (e.g., x, y) for integers and indices, uppercase letters (e.g., X, Y) for binary strings and functions, and calligraphic uppercase letters (e.g., $\mathcal{X}$, $\mathcal{Y}$) for sets and spaces. We use $\mathbb{N}$ and $\mathbb{Z}$ to denote the set of natural numbers and the set of integers respectively. We use 0^x and 1^y to denote the sequence of x 0's and y 1's respectively. We use $\{0,1\}^x$, $\{0,1\}^{\geq x}$ and $\{0,1\}^*$ to denote the set of binary strings of length x, the set of binary strings of length at least x and the set of all binary strings respectively. We use $\mathbb{F}_2$ to denote the field of characteristic 2 and $\mathbb{F}_2^n = \{0,1\}^n$ to denote the set of n-element vectors over $\mathbb{F}_2$. For any $X \in \{0,1\}^*$, we use $|X|$, $\|X\|$ and X_i to denote the number of bits, the number of blocks and the i-th block of the binary string X respectively, where the definition of one "block" is always clear from the context. For two binary strings X and Y, we use $X\|Y$ to denote the concatenation of X and Y. For any $X \in \{0,1\}^*$, we define the parsing of X into r-bit blocks as $X_1 \cdots X_x \xleftarrow{}_r X$, where $|X_i| = r$ for all $i < x$ and $1 \leq |X_x| \leq r$ such that $X = X_1\| \cdots \|X_x$. For any $X \in \{0,1\}^*$, $X_1 \cdots X_x \twoheadleftarrow_r X$ does the work of $X_1 \cdots X_x \xleftarrow{}_r X$, and follows it by the compulsory

10^* padding. Given any sequence $X = X_1 \cdots X_x$ and $1 \leq a \leq b \leq x$, we denote the subsequence $X_a \cdots X_b$ by $X[a \cdots b]$. For any $X \in \{0,1\}^n$, we use $X[i]$ to denote the i-th bit of X, and we define the bit order by $X = X[n-1] \| \cdots \| X[1] \| X[0]$. For any $X \in \{0,1\}^x$, we write $(X_1, X_2, \cdots, X_m) \xleftarrow{x_1, x_2, \cdots, x_m} X$ for the splitting of X into $X_1 = X[x-1 \cdots x-x_1]$, $X_2 = X[x-x_1-1 \cdots x-x_1-x_2]$, $\cdots$, $X_m = X[x_m-1 \cdots 0]$, where $x = x_1 + x_2 + \cdots + x_m$ holds. For integers $a \leq b$, we write $[a \cdots b]$ for the set $\{a, \cdots, b\}$, and for integer $a \geq 1$, we write $[a]$ for the set $\{1, \cdots, a\}$. We use the notations $\lceil x \rceil$ and $\lfloor x \rfloor$ to denote the decimal ceiling and floor function on the integer x respectively, and similarly, $\lceil X \rceil_r$ and $\lfloor X \rfloor_r$, to denote the most significant r bits and the least significant r bits of the binary string X respectively. We often denote 2^n by N. We write $\emptyset$ for the empty set, ε for the empty string and $\perp$ for the invalid symbol. For an event E, we denote its complementary event by $\overline{\mathsf{E}}$ and the probability of E by $\Pr[\mathsf{E}]$. For a given set $\mathcal{X}$ and non-negative integer x, we write $\mathcal{X}^{\leq x}$ for the union set $\cup_{i=0}^{x} \mathcal{X}^i$. For a non-negative $x < 2^n$, we write $\langle x \rangle_n$ for its conversion to an n-bit binary string with the most significant bit at the left-most position, e.g., $\langle 135 \rangle_8 = 10000111$. We may omit n if it's clear from the context. For a finite set S, we use $|S|$ to denote its size. Thus,

$$|\{0,1\}^m| = 2^m.$$

For a finite set S and a random variable X, we say X is *uniformly sampled* from S, denoted by $X \xleftarrow{\$} S$, if for each $x \in S$,

$$\Pr[X = x] = \frac{1}{|S|}.$$

Thus, when a binary string of length m is uniformly sampled, every string is picked with a probability $1/2^m$. A *random function* $f : S \to \{0,1\}^m$ samples $f(x)$ uniformly from $\{0,1\}^m$ for each $x \in S$. A function $f : S_1 \to S_2$ is called *injective* if for any distinct $x_1, x_2 \in S_1$, $f(x_1) \neq f(x_2)$. An injective function from S to S is called a *permutation* over S. We use $\mathsf{Func}(\mathcal{X}, \mathcal{Y})$ for the set of all functions $F : \mathcal{X} \to \mathcal{Y}$ and $\mathsf{Perm}(\mathcal{X})$ for the set of all permutations $P : \mathcal{X} \to \mathcal{X}$. We use $X \leftarrow x$ to denote the assignment of the value x to the variable X. Matrices are denoted with boldface letters, and for a matrix $\mathbf{H}$, we use $|\mathbf{H}|$ to denote its determinant. We use the Pochhammer falling factorial power notation

$$(a)_b := a(a-1)\cdots(a-b+1).$$

For ease of notation, we write + to denote field addition (bitwise XOR) when used between two or more field elements. Field multiplication in $\mathbb{GF}(2^n)$ is denoted with a bold dot ($\cdot$). Given a vector space $\mathcal{V} \subseteq \mathbb{F}$, and an element $\alpha \in \mathcal{K}$, we define the space $\alpha \cdot \mathcal{V} \stackrel{\text{def}}{=} \{\alpha \cdot V : V \in \mathcal{V}\}$. We write $\alpha\mathcal{V}$ or $\alpha \cdot \mathcal{V}$ when the operation is clear from the context. Given two spaces $\mathcal{V}, \mathcal{W} \subseteq \mathbb{F}$, we define $\mathcal{V} + \mathcal{W} \stackrel{\text{def}}{=} \{V \in \mathcal{V}, W \in \mathcal{W} : V + W\}$, where addition is in $\mathbb{F}$. In particular, given two binary strings X and Y, we denote their bitwise XOR by $X \oplus Y$ when $|X| = |Y|$. For two positive integers x and y with $x > y$ and two bit strings $X \in \mathbb{F}_2^x$ and $Y \in \mathbb{F}_2^y$, we define $X \oplus_y Y \stackrel{\text{def}}{=} X \oplus (0^{x-y} \| Y)$.

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

$$\mathbf{Adv}^{\mathcal{O}_1,\mathcal{O}_0}(\mathcal{A}) \stackrel{\text{def}}{=} \left| \Pr_{\mathcal{O}_0}[\mathcal{A} \Rightarrow 1] - \Pr_{\mathcal{O}_1}[\mathcal{A} \Rightarrow 1] \right|,$$

where the subscript of Pr denotes the oracle with which $\mathcal{A}$ is playing. All probabilities are defined over the random coins of the oracles and those of $\mathcal{A}$, if applicable. Here, we consider information-theoretic adversary $\mathcal{A}$ whose resources are bounded only in terms of its maximum numbers of queries and blocks that it can ask to its available oracles. One can derive its computation-theoretic counterpart straightforwardly.

$\mathcal{O}_0$ 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. The world in which $\mathcal{A}$ interacts with the real (or ideal) oracle is called the real (or ideal respectively) world. 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 subscript, 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 one or more of $F, F^*, \mathcal{G}$ and $\mathcal{A}$ from the notation of the distinguishing advantage when they are clear from the context.

2.2.1 PRF Advantage

Given two non-empty sets or spaces $\mathcal{X}, \mathcal{Y}$, let $F : \mathcal{X} \to \mathcal{Y}$ be a function, and $\rho \xleftarrow{\$} \mathsf{Func}(\mathcal{X}, \mathcal{Y})$. Then, the PRF advantage of $\mathcal{A}$ is defined as

$$\mathbf{Adv}^{F}_{\mathsf{PRF}}(\mathcal{A}) \stackrel{\text{def}}{=} \mathbf{Adv}^{F,\rho}(\mathcal{A}) .$$

We call $\mathcal{A}$ a PRF adversary.

2.2.2 PRP Advantage

Given a non-empty set or space $\mathcal{X}$, let $F : \mathcal{X} \to \mathcal{X}$ be a function, and $P \xleftarrow{\$} \mathsf{Perm}(\mathcal{X})$. Then, the PRP advantage of $\mathcal{A}$ is defined as

$$\mathbf{Adv}^{F}_{\mathsf{PRP}}(\mathcal{A}) \stackrel{\text{def}}{=} \mathbf{Adv}^{F,P}(\mathcal{A}) .$$

We call $\mathcal{A}$ a PRP adversary.

We can also parameterise $\mathcal{A}$ in terms of the resources it can use. We write $\mathbf{Adv}^{\mathcal{E}}_{\mathrm{X}}(q_c, \sigma, m)$ to denote the maximum distinguishing advantage of an X adversary for $\mathcal{E}$, where $X \in \{\mathsf{PRF}, \mathsf{PRP}\}$, that asks $\leq q_c$ encryption queries of $\leq \sigma$ blocks in total to its oracle such that the maximum number of message blocks in a query is at most m. Note that the definition of one "block" is always clear from the context. In the ideal permutation model, we write $\mathbf{Adv}^{\mathcal{E}}_{\mathrm{X}}(q_p, q_c, \sigma, m)$ to denote the maximum distinguishing advantage of an X adversary for $\mathcal{E}$, where $X \in \{\mathsf{PRF}, \mathsf{PRP}\}$, that asks $\leq q_p$ queries to the underlying permutation, $\leq q_c$ encryption queries of $\leq \sigma$ blocks in total to its oracle such that the maximum number of message blocks in a query is at most m. Since we consider information-theoretic adversary, we will not consider their time parameter t in defining their advantage. However, based on contexts, we may omit some resources in defining the adversarial advantage.

2.3 NE and Its Security Notion

A *Nonce-based Encryption* (NE) is interpreted as a combination of a key-space $\mathcal{K}$, a nonce-space $\mathcal{N}$ and a message space $\mathcal{M}$ along with two functions $\mathsf{Enc} : \mathcal{K} \times \mathcal{N} \times$

$\mathcal{M} \to \mathcal{M}$ (called the Encryption Function) and $\mathsf{Dec} : \mathcal{K} \times \mathcal{N} \times \mathcal{M} \to \mathcal{M}$ (called the Decryption Function) with the correctness condition that for any $K \in \mathcal{K}, N \in \mathcal{N}$ and $M \in \mathcal{M}$, it holds that if

$$\mathsf{Dec}(K, N, \mathsf{Enc}(K, N, M)) = M\,.$$

2.3.1 NE Security

The NE security game is played between the real oracle Enc described above and an ideal oracle Enc^* where $\mathsf{Enc}^* : \mathcal{K} \times \mathcal{N} \times \mathcal{M} \to \mathcal{M}$ is an ideal random function. The adversary $\mathcal{A}$ can make encryption queries to the oracle. In addition, we assume the following restrictions:

- $\mathcal{A}$ should be nonce-respecting, i.e., should not repeat a nonce in more than one query; and
- $\mathcal{A}$ should not make *pointless* queries, i.e., should not repeat the same query multiple times.

The distinguishing advantage of $\mathcal{A}$ for an NE scheme $\mathcal{E}$ will be denoted by $\mathbf{Adv}^{\mathcal{E}}_{\mathrm{NE}}(\mathcal{A})$. The following security notion is captured in this advantage.

- Privacy, i.e., $\mathcal{A}$ should not be able to distinguish the real oracle from the ideal oracle.

In the ideal permutation model, $\mathcal{A}$ has one additional oracle $P^{\pm}$ that provides access to the permutation P in forward and backward directions. We write $\mathcal{E}[P], \mathsf{Enc}[P]$ and $\mathsf{Dec}[P]$ to indicate that P is the underlying permutation of $\mathcal{E}, \mathsf{Enc}$ and Dec respectively.

2.4 NAEAD and Its Security Notion

A *Nonce-based Authenticated Encryption with Associated Data* (NAEAD) is interpreted as a combination of a key-space $\mathcal{K}$, a nonce-space $\mathcal{N}$, an associated-data-space $\mathcal{AD}$, a message space $\mathcal{M}$ and a tag space $\mathcal{T}$ along with three functions $\mathsf{Enc} : \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{M} \to \mathcal{M} \times \mathcal{T}$ (called the Encryption Function), $\mathsf{Dec} : \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{M} \to \mathcal{M}$ (called the Decryption Function) and $\mathsf{Ver} : \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{M} \times \mathcal{T} \to \{\top, \bot\}$ (called the Verification Function) with the correctness condition that for any $K \in \mathcal{K}, N \in \mathcal{N}, A \in \mathcal{AD}, M, C \in \mathcal{M}$ and $T \in \mathcal{T}$, it holds that if

$$\mathsf{Enc}(K, N, A, M) = (C, T)\,,$$

then

$$\mathsf{Dec}(K, N, A, C) = M\,,$$

and

$$\mathsf{Ver}(K, N, A, C, T) = \top\,.$$

Note that the actual implementation of a particular NAEAD scheme may differ from the above-mentioned definition. For example, the last two functions can be merged into one single function which outputs $\mathcal{M}$ if the output of the third function is $\top$, and $\perp$ otherwise.

2.4.1 NAEAD Security

The PRIV (Privacy) security game is played between the real oracle Enc described above and an ideal oracle Enc^* where $\mathsf{Enc}^* : \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{M} \to \mathcal{M} \times \mathcal{T}$ is an ideal random function. The adversary $\mathcal{A}$ can make encryption queries to the oracle. In addition, we assume the following restrictions:

- $\mathcal{A}$ should be nonce-respecting, i.e., should not repeat a nonce in more than one query; and
- $\mathcal{A}$ should not make *pointless* queries, i.e., should not repeat the same query multiple times.

The distinguishing advantage of $\mathcal{A}$ for an NAEAD scheme $\mathcal{E}$ will be denoted by $\mathbf{Adv}^{\mathcal{E}}_{\mathrm{PRIV}}(\mathcal{A})$. The following security notion is captured in this advantage.

- Privacy, i.e., $\mathcal{A}$ should not be able to distinguish the real oracle from the ideal oracle.

The AUTH (Authenticity) security game is played between the real oracle $(\mathsf{Enc}, \mathsf{Ver})$ described above and an ideal oracle $(\mathsf{Enc}^*, \mathsf{Ver}^*)$ where $\mathsf{Enc}^* : \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{M} \to \mathcal{M} \times \mathcal{T}$ is an ideal random function and $\mathsf{Ver}^* : \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{M} \times \mathcal{T} \to \{\perp\}$ is a constant function. The adversary $\mathcal{A}$ can make encryption and verification queries to the oracle. In addition, we assume the following restrictions:

1. $\mathcal{A}$ should be nonce-respecting, i.e., should not repeat a nonce in more than one encryption query; and

2. $\mathcal{A}$ should not make *pointless* queries, i.e., should not make the verification query (K, N, A, C, T) if it has already made an encryption query (K, N, A, M) and received (C, T) in response.

The distinguishing advantage of $\mathcal{A}$ for an NAEAD scheme $\mathcal{E}$ will be denoted by $\mathbf{Adv}^{\mathcal{E}}_{\mathrm{AUTH}}(\mathcal{A})$. The following security notion is captured in this advantage.

- Authenticity, i.e., $\mathcal{A}$ should not be able to forge the real oracle. In other words, $\mathcal{A}$ should not be able to make a verification query to $\mathsf{Ver}/\mathsf{Ver}^*$ to which the response isn't $\perp$.

The NAEAD security game is a combination of the PRIV and the AUTH security games which is played between the real oracle $(\mathsf{Enc}, \mathsf{Ver})$ described above and an ideal oracle $(\mathsf{Enc}^*, \mathsf{Ver}^*)$ where $\mathsf{Enc}^* : \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{M} \to \mathcal{M} \times \mathcal{T}$ is an ideal random function and $\mathsf{Ver}^* : \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{M} \times \mathcal{T} \to \{\perp\}$ is a constant function. The adversary $\mathcal{A}$ can make encryption and verification queries to the oracle. In addition, we assume the following restrictions:

1. $\mathcal{A}$ should be nonce-respecting, i.e., should not repeat a nonce in more than one encryption query; and

2. $\mathcal{A}$ should not make *pointless* queries, i.e., should not repeat the same encryption query multiple times or should not make the verification query (K, N, A, C, T) if it has already made an encryption query (K, N, A, M) and received (C, T) in response.

The distinguishing advantage of $\mathcal{A}$ for an NAEAD scheme $\mathcal{E}$ will be denoted by $\mathbf{Adv}^{\mathcal{E}}_{\mathrm{NAEAD}}(\mathcal{A})$. The following two security notions are captured in this advantage.

1. Privacy, i.e., $\mathcal{A}$ should not be able to distinguish the real oracle from the ideal oracle.

2. Authenticity, i.e., $\mathcal{A}$ should not be able to forge the real oracle. In other words, $\mathcal{A}$ should not be able to make a verification query to $\mathsf{Ver}/\mathsf{Ver}^*$ to which the response isn't $\perp$.

2.4.2 RUP Security

The PRIV-PA1/PA2 (Privacy) security game is played between the real oracle $(\mathsf{Enc}, \mathsf{Dec})$ described above and an ideal oracle $(\mathsf{Enc}^*, \mathsf{Dec}^*)$ where $\mathsf{Enc}^* : \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{M} \rightarrow \mathcal{M} \times \mathcal{T}$ is an ideal random function and Dec^* is a simulator or extractor that mimics the outputs of Dec. There are two types of PRIV security games. In the PA1 (Plaintext Awareness 1) security game, the simulator has insight into the queries that the adversary $\mathcal{A}$ makes to Enc^*. In the PA2 (Plaintext Awareness 2) security game, the simulator does not have insight into the queries that $\mathcal{A}$ makes to Enc^*. $\mathcal{A}$ can make encryption and decryption queries to the oracle. In addition, we assume the following restrictions:

1. $\mathcal{A}$ should be nonce-respecting, i.e., should not repeat a nonce in more than one encryption query; and
2. $\mathcal{A}$ should not make *pointless* queries, i.e., should not repeat the same query multiple times, should not make the decryption query (K, N, A, C) if it has already made an encryption query (K, N, A, M) and received (C, T) in response or should not make the encryption query (K, N, A, M) if it has already made a decryption query (K, N, A, C) and received M in response.

The distinguishing advantage of $\mathcal{A}$ for an NAEAD scheme $\mathcal{E}$ will be denoted by $\mathbf{Adv}^{\mathcal{E}}_{\text{PRIV-PA1/PA2}}(\mathcal{A})$. The following security notion is captured in this advantage.

- Privacy, i.e., $\mathcal{A}$ should not be able to distinguish the real oracle from the ideal oracle.

The AUTH-PA1/PA2 (Authenticity) security game is played between the real oracle $(\mathsf{Enc}, \mathsf{Dec}, \mathsf{Ver})$ described above and an ideal oracle $(\mathsf{Enc}^*, \mathsf{Dec}^*, \mathsf{Ver}^*)$ where $\mathsf{Enc}^* : \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{M} \rightarrow \mathcal{M} \times \mathcal{T}$ is an ideal random function, Dec^* is a simulator or extractor that mimics the outputs of Dec and $\mathsf{Ver}^* : \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{M} \times \mathcal{T} \rightarrow \{\perp\}$ is a constant function. The adversary $\mathcal{A}$ can make encryption, decryption and verification queries to the oracle. In addition, we assume the following restrictions:

1. $\mathcal{A}$ should be nonce-respecting, i.e., should not repeat a nonce in more than one encryption query; and
2. $\mathcal{A}$ should not make *pointless* queries, i.e., should not repeat the same query multiple times, should not make the decryption query (K, N, A, C) if it has

already made an encryption query (K, N, A, M) and received (C, T) in response or should not make the encryption query (K, N, A, M) if it has already made a decryption query (K, N, A, C) and received M in response.

The distinguishing advantage of $\mathcal{A}$ for an NAEAD scheme $\mathcal{E}$ will be denoted by $\mathbf{Adv}^{\mathcal{E}}_{\text{AUTH-PA1/PA2}}(\mathcal{A})$. The following security notion is captured in this advantage.

- Authenticity, i.e., $\mathcal{A}$ should not be able to forge the real oracle. In other words, $\mathcal{A}$ should not be able to make a verification query to $\mathsf{Ver}/\mathsf{Ver}^*$ to which the response isn't $\perp$.

The INT-RUP-PA1/PA2 (Integrity under Release of Unverified Plaintext - PA1 / PA2) security game is a combination of the PRIV-PA1/PA2 and the AUTH-PA1/PA2 security games which is played between the real oracle $(\mathsf{Enc}, \mathsf{Dec}, \mathsf{Ver})$ described above and an ideal oracle $(\mathsf{Enc}^*, \mathsf{Dec}^*, \mathsf{Ver}^*)$ where $\mathsf{Enc}^* : \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{M} \to \mathcal{M} \times \mathcal{T}$ is an ideal random function, Dec^* is a simulator or extractor that mimics the outputs of Dec and $\mathsf{Ver}^* : \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{M} \times \mathcal{T} \to \{\perp\}$ is a constant function. The adversary $\mathcal{A}$ can make encryption, decryption and verification queries to the oracle. In addition, we assume the following restrictions:

1. $\mathcal{A}$ should be nonce-respecting, i.e., should not repeat a nonce in more than one encryption query; and
2. $\mathcal{A}$ should not make *pointless* queries, i.e., should not repeat the same query multiple times, should not make the decryption query (K, N, A, C) if it has already made an encryption query (K, N, A, M) and received (C, T) in response or should not make the encryption query (K, N, A, M) if it has already made a decryption query (K, N, A, C) and received M in response.

The distinguishing advantage of $\mathcal{A}$ for an NAEAD scheme $\mathcal{E}$ will be denoted by $\mathbf{Adv}^{\mathcal{E}}_{\text{INT-RUP-PA1/PA2}}(\mathcal{A})$. The following two security notions are captured in this advantage.

1. Privacy, i.e., $\mathcal{A}$ should not be able to distinguish the real oracle from the ideal oracle.
2. Authenticity, i.e., $\mathcal{A}$ should not be able to forge the real oracle. In other words, $\mathcal{A}$ should not be able to make a verification query to $\mathsf{Ver}/\mathsf{Ver}^*$ to which the response isn't $\perp$.

The INT-RUP security notion was introduced in [20].

2.5 Coefficients H Technique

The Coefficients H Technique is a proof method by Patarin [21] that was modernised by Chen and Steinberger [22, 23]. A distinguisher $\mathcal{A}$ interacts with and obtains outputs from a real oracle $\mathcal{O}_1$ or an ideal oracle $\mathcal{O}_0$. (The oracle could be a sequence of multiple oracles.) The results of its interaction are collected in a transcript τ. The oracle can sample random coins before the experiment (often a key or an ideal primitive that is sampled beforehand) and is then deterministic. A transcript τ is attainable if $\mathcal{A}$ can observe τ with non-zero probability in the ideal world.
The Fundamental Theorem of the Coefficients H Technique, whose proof can be found, e.g., in [21–23], states the following:

Theorem 2.1 ([21])**.** Assume, there exist $\epsilon_1, \epsilon_2 \geq 0$ such that

$$\Pr_{\mathcal{O}_0}[\mathsf{bad}] \leq \epsilon_1,$$

and for any attainable transcript τ obtained without encountering bad,

$$\frac{\Pr_{\mathcal{O}_1}[\tau]}{\Pr_{\mathcal{O}_0}[\tau]} \geq 1 - \epsilon_2.$$

Then, for all adversaries $\mathcal{A}$, it holds that $\mathbf{Adv}^{\mathcal{O}_0,\mathcal{O}_1}(\mathcal{A}) \leq \epsilon_1 + \epsilon_2$.

The technique has been generalised by Hoang and Tessaro [24] in their expectation method, which allowed them to derive the Fundamental Theorem as a corollary. Since we only consider bad events in the ideal world, we will write $\Pr_{\mathcal{O}_0}[\mathsf{bad}]$ simply as $\Pr[\mathsf{bad}]$ when there is no scope for confusion; the same notation is used when the event bad is broken down into further sub-events.

Chapter 3

Oribatida

Permutation-based modes have been established for lightweight authenticated encryption, apparent from the high interest in them in the NIST lightweight cryptography project. In permutation-based modes, there are two parameters, r and c, with $r + c$ being (almost) the same as the size of the permutation. The first parameter r (a public parameter called "rate") usually determines the speed of the mode, and the second parameter c (a secret parameter called "capacity") usually determines the security of the same. As the sum of these two parameters is fixed once the designer fixes the underlying permutation, there is always this trade-off between speed and security for such modes. However, the security is usually upper bounded by $O(\sigma^2/2^c)$ bits, where σ is the number of calls. The development of more schemes that provide higher security bounds led to the CHES'18 proposal Beetle that raised this bound to $O(r\sigma/2^c)$.

While authenticated encryption can be performed in an online manner, authenticated decryption assumes that the resulting plaintext is buffered and never released if the corresponding tag is incorrect. Since lightweight devices may lack the resources for buffering, additional robustness guarantees, such as integrity under release of unverified plaintexts (INT-RUP), are desirable. In this stronger setting, the security of the established schemes, including Beetle, is limited by $O(q_p q_d/2^c)$, where q_d is the maximum number of decryption queries, and q_p that of offline or primitive queries, which motivates novel approaches.

This chapter proposes Oribatida, a permutation-based AE scheme that derives s-bit masks from previous permutation outputs to mask ciphertext blocks. Oribatida can provide a security bound of $O(r\sigma^2/2^{c+s})$, which allows smaller permutations for the same level of security. It provides a security level dominated by $O(\sigma_d^2/2^c)$ under INT-RUP adversaries, which eliminates the dependency on primitive queries.

We prove its security under nonce-respecting and INT-RUP adversaries. We show that our INT-RUP bound is tight and show general attacks on previous constructions.

3.1 Introduction

3.1.1 Permutation-based Modes

Permutation-based modes have been established for various applications of symmetric key cryptography during the previous decade. Keyless modes have been standardised as the hash function SHA-3 and its derivative SHAKE for the extendable output functions [25]. Keyed modes are used for authentication [26] or encryption [27]. Moreover, permutation-based schemes have found widespread adoption for authenticated encryption, as the CAESAR co-selection Ascon [28], or many more candidates have shown, e.g., PRIMATES [29], NORX [30], Ketje [31], or Keyak [32].

The sponge [33] and duplex [34] modes transform an internal n-bit state iteratively with a public permutation. Both modes absorb an input stream block-wise to generate a pseudo-random output stream. While sponges separate the input (absorption) and output (squeezing) phases, the duplex mode generates the i-th output block directly after absorbing the i-th input block. In both modes, an n-bit permutation absorbs the data in r-bit chunks $r < n$, called the rate.

Keyed Sponge Variants were introduced by Bertoni et al. [35] and can be categorised into inner-keyed, outer-keyed, and full-keyed variants (cf. [36]; recently, Dobraunig and Mennink added the suffix-keyed sponge [37]. The inner-keyed sponge [38] initialises the inner part with the key, $(0 \,\|\, K)$, whereas the outer-keyed sponge [35] (so-dubbed by [39]) concatenates key and message $K \,\|\, M$ for the output. The full-keyed sponge [40] employs the full state in the absorption phase; the suffix-keyed sponge uses a keyed function only at the end.

Permutation-based modes are analysed mostly in the ideal-permutation model. For authenticated encryption, an adversary $\mathcal{A}$ shall distinguish between two worlds consisting of two oracles: each world has (1) a construction oracle that $\mathcal{A}$ can ask encryption and verification queries to and (2) a primitive oracle that provides access to the internal permutation. The former oracle represents online queries, whereas the latter represents offline queries. $\mathcal{A}$ can ask q_e encryption queries, q_v

TABLE 3.1: Existing PRF bounds for keyed sponges. FKS/IKS/OKS = full-/inner-/outer-keyed sponge, IP = ideal permutation, indiff. = indifferentiability

Scheme			Model	Bound	Work
FKS	IKS	OKS			
–	–	•	IP (Indiff.)	$O(\frac{\sigma^2+\sigma}{2^c})$	[45]
–	•	•	IP	$O(\frac{\sigma^2}{2^c})$	[38]
–	–	•	IP	$O(\frac{\sigma^2+\sigma q_p}{2^c})$	[39]
–	•	–	IP	$O(\frac{\sigma^2+\sigma q_p}{2^c}+\frac{q_p}{2^n})$	[39]
NORX-like			IP	$O(\min(\frac{\sigma^2}{2^n}, \frac{\sigma}{2^c}, \frac{q}{2^k}))$	[47]
–	–	•	IP	$O(\frac{q_c^2+\sigma q_c+q_c q_p}{2^c})$	[42]
•	–	–	IP	$O(\frac{\sigma^2}{2^n}+\frac{q_c^2 m}{2^c}+\frac{\sigma_c q_p}{2^k})$	[43]
–	•	•	IP	$O(\frac{q_c^2+q_c q_p}{2^c})$	[44]
•	–	–	IP	$O(\frac{q_c q_p+q_c^2}{2^c}+\frac{q_p}{2^k})$	[41]
•	–	–	IP	$O(\frac{q_c^2+q_c q_p}{2^c}+\frac{q_p}{2^k})$	[36]
–	–	•	IP	$O(\frac{q_c^2+q_c q_p}{2^c}+\frac{c^{k/r} q_p}{2^k})$	[36]

verification queries, and q_p construction queries; σ and k usually denote the number of blocks over all construction queries and the secret key size, respectively.

Sponge modes for authenticated encryption started with the Duplex construction and the AE scheme SpongeWrap [34] and MonkeyDuplex [40], and led to a considerable corpus of analysis, e.g., [39, 41–44]. Early, Bertoni et al. [45] showed that the sponge is indifferentiable from a random oracle [46] for up to $O(2^{c/2})$ calls to the permutation. Their follow-up work [35] improved the bounds for the unkeyed sponge to $O(\frac{q_p\sigma}{2^c}+\frac{q_c}{2^k})$ if $\sigma \ll 2^{c/2}$. For SpongeWrap, Bertoni et al. [34] had shown a privacy bound of $O(\frac{q}{2^k}+\frac{\sigma^2}{2^c})$ and an authenticity bound of $O(\frac{q}{2^k}+\frac{\sigma^2}{2^c}+\frac{q}{2^\tau})$. Jovanovic et al. [47] improved the asymptotic authenticity bound, although under the limitation of at most $\sigma \ll 2^{c/2}$ decryption queries. Summarising many previous results, Mennink [36] showed that keyed sponges achieve PRF security of around $O(\frac{q_c^2+q_c q_p}{2^c}) + \mathbf{Adv}^{\mathrm{KP}}_{\Pi}$. He coined the final term the key-prediction security, e.g., in $O(\frac{q_p}{2^k})$ for full-keyed sponges and $k < n$. The recent duplex-based AE scheme Beetle [48] added a transform to the output so that the plaintext input that is added to the inner part and to the ciphertext output block differ. As a result, Beetle offered a bound of $O(\frac{rq_p+r\sigma}{2^c}+\frac{q_v+q_p}{2^r}+\frac{\sigma^2+q_p^2}{2^n})$. Table 3.1 summarises some of the most noteworthy results in the past. Improvements to those general bounds appear hard, which motivates the search for novel constructions.

Correct authenticated decryption requires the entire plaintext to be buffered until the tag has been verified. On certain architectures (e.g., the memory-constrained devices in the Internet of Things (IoT), real-time streaming protocols etc.), this requirement can exceed the available storage and induce unacceptable latency. Andreeva et al. [20] introduced notions for privacy and authenticity under the release of unverified plaintext material. Security guarantees such as INT-RUP represent valuable additional levels of robustness.

3.1.2 Research Gap

While Beetle improved the bound for nonce-based authenticated encryption, this bound does not hold in the INT-RUP setting: as we will outline, there exists an attack with an advantage of $\Omega(\frac{q_p q_v}{2^c})$. The naturally arising question is whether one can achieve higher INT-RUP security. This is motivated by a practical demand: while the primary advantage of sponges is simplicity, they are also useful for lightweight applications in resource-constrained environments. In such contexts, buffering multi-block decryption outputs is likely infeasible, which renders INT-RUP security advantageous. For example, the NIST lightweight cryptography project requested 112-bit integrity security for AE schemes and support of at least $2^{50} - 1$ encrypted bytes of data. Inserting them into Mennink's bound [36] of $\frac{q_c^2 + q_c q_p}{2^c}$, the NIST requirements would imply permutation sizes of at least 172 bits plus a plausible rate. A higher INT-RUP security could lead to smaller permutations, reducing area, and energy consumption simultaneously.

3.1.3 Contributions

This chapter contains three contributions. First, it answers the question above in the affirmative by showing a sponge with dynamic s-bit masks that achieves NAEAD security of $O(\frac{q_p}{2^k} + \frac{q_d}{2^\tau} + \frac{q_p \sigma}{2^{c+s}} + \frac{\sigma^2}{2^n})$. Replacing the terms by the NIST requirements of $k \geq 128$, $\tau \geq 64$, $q_p \leq 2^{112}$ and $\sigma \geq 2^{50}$, a trade-off could use $c+s \geq 162$, or better $c+s \geq 192$ bits to be secure for up to $\sigma \in O(2^{64})$ blocks and $q_p \leq 2^{128}$ primitive queries. Thus, the rate could be reduced considerably compared to the bound from [36]. We prove that the INT-RUP advantage is at most $O(q_d^2/2^c)$, i.e., depends solely on the number of online (i.e., construction) queries, contrasting the bound of $O(q_d q_p/2^c)$ for the generic duplex and previous constructions, as illustrated in Table 3.2. The difference may seem minor. However, eliminating the dependency on offline (i.e., primitive) queries is a valuable gain. Because in real life, the number of offline queries is usually much greater than the number of online queries. As a result, one can use a smaller permutation or higher throughput depending on requirements. We propose a novel permutation-based AE scheme Oribatida that applies dynamic masks to the outputs and provides the NIST security bounds with permutation sizes of only 192 bits. As our second contribution, we show that the bound provided by our proposal is tight with an attack that matches the proved bound. Moreover, we show that it also applies to other duplex-based designs in general if masks would be added.

TABLE 3.2: Comparison of the security bounds for INT-RUP attacks on previous permutation-based AE schemes and our construction

	Bound	
Scheme	Unmasked	Masked
Generic Duplex [34]	$O(q_d q_p/2^c)$	$O(q_d^2/2^c)$
Beetle [48]	$O(q_d q_p/2^c)$	$O(q_d^2/2^c)$
SPoC [51]	$O(q_d q_p/2^c)$	$O(q_d^2/2^r)$
Oribatida [This Chapter]	–	$O(q_d^2/2^c)$

Remark 3.1. Finally, we acknowledge an observation by Rohit and Sarkar on the NIST mailing list [49]. We note that our proposal here is a slightly updated variant of the NIST submission [50], that addresses their observation by masking the authentication tag. We call it Oribatida v1.3, but use Oribatida hereafter. We will discuss the effect of the slight update later.

3.1.4 Outline

Section 3.2 motivates our proposal by showing INT-RUP attacks on the duplex mode and other existing schemes. Section 3.3 describes Oribatida in general. We close the parenthesis of INT-RUP attacks on Oribatida and other duplex-based modes when in Section 3.4. We analyse the security on Oribatida for the standard nonce-based AE setting in Section 3.5 and in the INT-RUP setting in Section 3.6. Next, Section 3.7 compares it with those second-round NIST candidates that claim INT-RUP security. Section 3.8 discusses the slight update from [50] and the associated improvement. Subsequently, Section 3.9 specifies an instance with a Simon-based permutation, whose security is discussed in Section 3.10 from previous works. Section 3.11 reports on the result of a hardware implementation of Oribatida before Section 3.12 concludes this chapter.

3.2 INT-RUP Attacks on Existing AE Schemes

This section shows attacks under INT-RUP adversaries on the duplex mode, as well as on recent more secure AE schemes Beetle or SPoC. For each construction, we briefly recall the necessary parts of their definition. As summarised in Table 3.2, the proposed attacks possess an advantage of $\Omega(\frac{q_d q_p}{2^c})$ on the previous constructions. Thus, the improved bounds of Beetle or SPoC do not carry over to the INT-RUP setting. For all attacks, we consider a random permutation $\pi \xleftarrow{\$} \mathsf{Perm}(\mathcal{B})$ and a

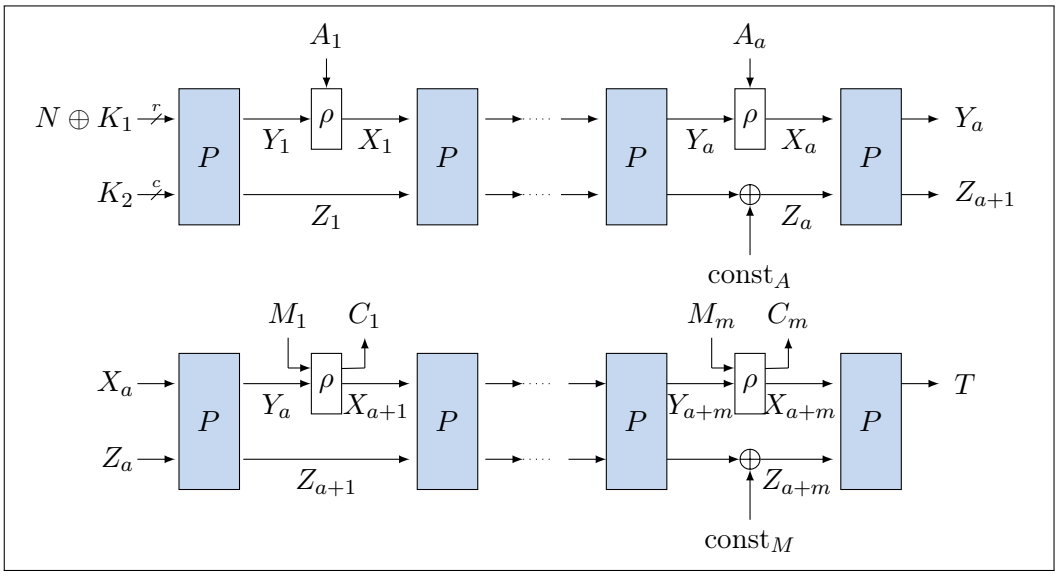


FIGURE 3.1: The Beetle authenticated encryption scheme

randomly chosen secret key $K \xleftarrow{\$} \mathcal{K}$. We denote by $\mathcal{A}$ a nonce-respecting INT-RUP adversary against the individual schemes.

The main idea of all the attacks in this section is as follows: $\mathcal{A}$ asks q_d decryption queries s. t. any predetermined r bits (e.g., the first r bits) of the input to one of the permutations of the construction are fixed and known (say X). The remaining $n - r = c$ bits may vary. Next, $\mathcal{A}$ asks q_p primitive queries Q^1, Q^2, ... , Q^{q_p} with the first r bits fixed to X, but with pairwise distinct c bits, and receives R^1, R^2, ... , R^{q_p}. When $q_d \cdot q_p \approx 2^c$, $\mathcal{A}$ can expect a state collision between an online input to the permutation and an (offline) permutation query. This collision can be detected from the first r bits of the outputs of the corresponding construction queries, which will be equal for the colliding inputs. Once $\mathcal{A}$ knows the full state at the input to the permutation of the construction, it can revert the permutation calls in the construction and finally recover the key. We adapt this strategy for the duplex mode and Beetle before we consider the differences in SPoC and hybrids.

3.2.1 INT-RUP Attack on The Duplex Mode

Let us consider the Sponge-Wrap mode [34].

1. The adversary $\mathcal{A}$ asks q_d decryption queries (N, A^1, C), (N, A^2, C), ... , (N, A^{q_d}, C), and receives M^1, M^2, ..., M^{q_d}. The associated data A^i consist of a single block, the ciphertexts $C = (C_1, C_2)$ are fixed to the same two blocks for each query.

2. Now $\mathcal{A}$ can follow the generic idea to complete the attack.

The advantage of the attack is $\Omega(\frac{q_d q_p}{2^c})$.

3.2.2 INT-RUP Attack on Beetle

Beetle [48] is a recent permutation-based light-weight AE scheme. From now onward, we consider the updated variant [52] that fixed the proof and details (such as no double call to the permutation). An overview of the encryption process is provided in Figure 3.1. The map $\rho : \{0,1\}^r \times \{0,1\}^r \to \{0,1\}^r \times \{0,1\}^r$ computes $\rho(I_1, I_2) \stackrel{\text{def}}{=} (\mathsf{shuffle}(I_1) \oplus I_2, I_1 \oplus I_2)$ for all inputs $I_1, I_2 \in \{0,1\}^r$, where $\mathsf{shuffle}(x) \stackrel{\text{def}}{=} (\lfloor x \rfloor_{r/2} \,\|\, \lfloor x \rfloor_{r/2} \oplus \lceil x \rceil_{r/2})$. The results of ρ are ordered as $(X_{a+i}, C_i) \leftarrow \rho(Y_{a+i}, M_i)$ and $(Y_{a+i}, M_i) \leftarrow \rho^{-1}(X_{a+i}, C_i)$, respectively.

1. The adversary $\mathcal{A}$ asks q_d encryption queries (N^1, A^1, M), (N^2, A^2, M), $\ldots$, (N^{q_d}, A^{q_d}, M) to the encryption oracle, and receives C^1, C^2, $\ldots$, C^{q_d}. Size of M and A^i is one block for each i.

2. Then, $\mathcal{A}$ asks q_d decryption queries $(N^1, A^1, C^{1'})$, $(N^2, A^2, C^{2'})$, $\ldots$, $(N^{q_d}, A^{q_d}, C^{q'_d})$ to the decryption oracle where $C^{i'} \leftarrow Y_2^i \oplus \mathsf{shuffle}(Y_2^i)$. This ensures that the first r bits of the input to the third permutation always equal zero.

3. Now $\mathcal{A}$ can follow the generic idea to complete the attack.

The advantage of the attack is again $\Omega(\frac{q_d q_p}{2^c})$.

3.2.3 INT-RUP Attack on SPoC

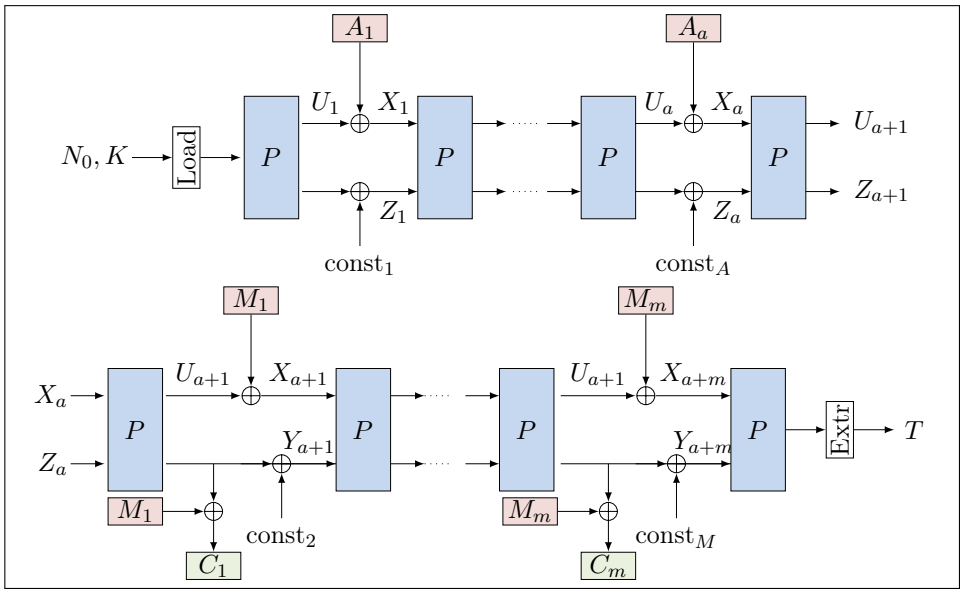
SPoC (see Figure 3.2a) is a permutation-based NIST candidate [51] that uses the capacity to derive ciphertext outputs, while it still absorbs the message in the rate. In SPoC, the adversary cannot fix any part of the state in contrast to SpongeWrap and Beetle– though, there is a similar attack:

1. $\mathcal{A}$ asks q_d queries (N, A, C^1), (N, A, C^2), $\ldots$, (N, A, C^{q_d}) to the decryption oracle and receives $M^1, M^2, \ldots, M^{q_d}$. The associated data A and ciphertext C^i consist of a single block for every i. This ensures that the first r bits of the input to the third permutation always equal $M^1 \oplus C^1$.

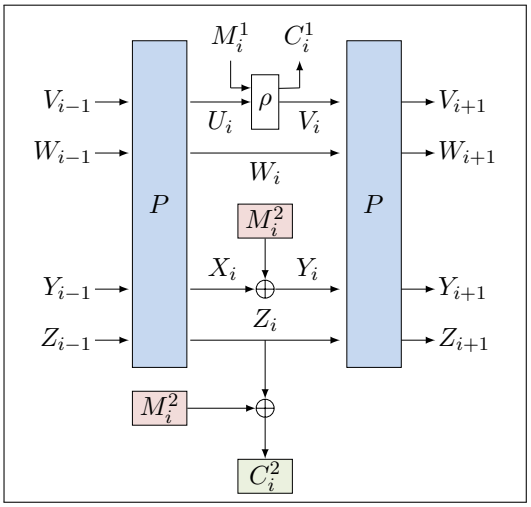
2. Now $\mathcal{A}$ can follow the generic idea to complete the attack.

The advantage of the attack is again $\Omega(\frac{q_d q_p}{2^c})$.

FIGURE 3.2: SPoC and a hybrid of Beetle and SPoC



(A) The SPoC authenticated encryption scheme



(B) A hybrid of Beetle and SPoC

3.2.4 INT-RUP Attack on A Hybrid of Beetle and SPoC

We can generalise our attacks to hybrid modes of Beetle and SPoC as well. Such a hybrid would use both modes Beetle and SPoC in parallel to process the queries. We illustrate it in Figure 3.2b. Each message block (say M) is parsed into two sections (say M^1 and M^2), where $|M^1| = r_1$ and $|M^2| = r_2$. M^1 is processed with Beetle to a ciphertext block C^1; M^2 with SPoC to a ciphertext block C^2; The final ciphertext block becomes $C \leftarrow C^1 \,\|\, C^2$, and the associated data blocks and the ciphertext blocks for decryption are treated in a similar manner. Note that the hybrid mode is parameterised by r_1, r_2 and c with the condition $c \geq r_2$. The size of rate and capacity of the Beetle part are r_1 and $c - r_2$; the size of both rate and

capacity of the SPoC part is r_2. As a result, the size of rate and capacity of the hybrid mode is $r = r_1 + r_2$ and c. When $r_2 = 0$, the hybrid mode translates to the Beetle mode. Similarly, when $r_1 = 0$, the hybrid mode is equivalent to the SPoC mode.

An INT-RUP attack on such modes could be defined as follows:

1. $\mathcal{A}$ asks q_d decryption queries (N, A^1, C), (N, A^2, C), ..., (N, A^{q_d}, C) to the decryption oracle and receives $M^1, M^2, \ldots, M^{q_d}$. The ciphertext C and associated data A^i consist of a single block for every i.

2. There exists at least one value of the last r_2 bits of the input to the third permutation which remains same for at least $q = \frac{q_d}{2^{r_2}}$ queries. Suppose q such queries are $(N, A^{1'}, C)$, $(N, A^{2'}, C)$, ..., $(N, A^{q'}, C)$.

3. $\mathcal{A}$ can detect the previous step as it knows the value of the last r_2 bits of the input to the third permutation because that will be equal to the last r_2 bits of $C \oplus M^i$.

4. $\mathcal{A}$ retains those q queries and discards the rest.

5. For each of the above queries, $\mathcal{A}$ updates the value of the first r_1 bits of the ciphertext to $Y_2 \oplus \mathsf{shuffle}(Y_2)$ and varies the remaining r_2 bits. This ensures that the first r_1 bits of the input to the third permutation always equal zero.

6. In the way mentioned above, $\mathcal{A}$ can ask q_d more decryption queries to the decryption oracle. This time, a total of r bits (first r_1 bits and last r_2 bits) of the input to the third permutation are fixed and known to $\mathcal{A}$.

7. Then, $\mathcal{A}$ can follow the generic idea to complete the attack.

The advantage of the attack is again $\Omega(\frac{q_d q_p}{2^c})$.

3.2.5 Discussion

As a takeaway from this section, the unmasked sponge-based AE schemes allow INT-RUP attacks whose advantage can depend linearly on the number of offline primitive queries. We think that any AE construction which uses linear feedback is vulnerable to such an attack ($\Omega(\frac{q_d q_p}{2^c})$) unless it uses more state. The next section defines Oribatida that masks its ciphertexts for higher INT-RUP resistance. After its definition, we will get back to attacks on it and on strengthened versions of the schemes sketched here to show that they can be similarly extended by a ciphertext masking.

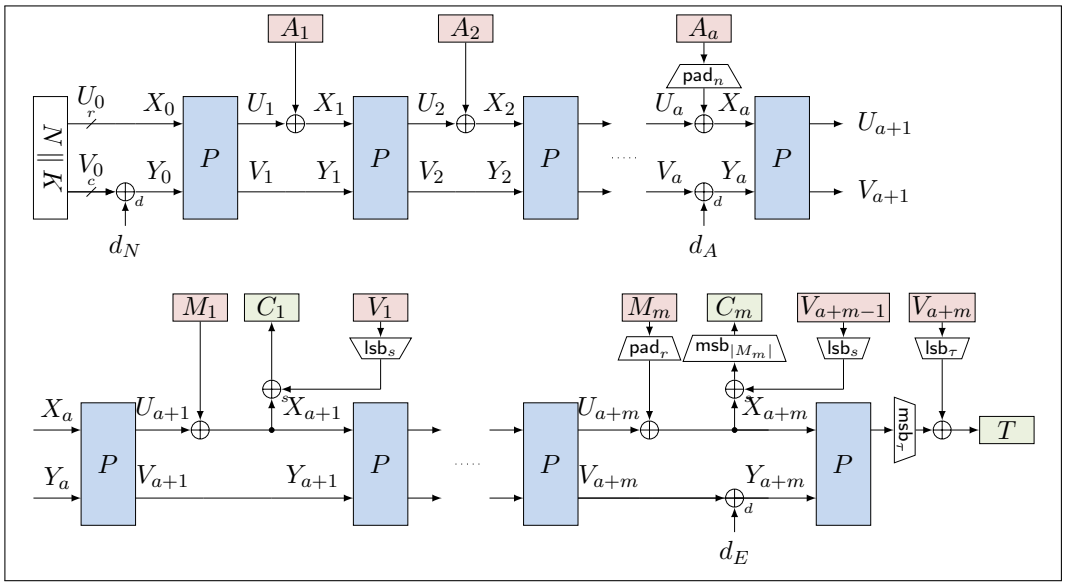


FIGURE 3.3: Authenticated encryption of a-block associated data A and m-block message M with Oribatida.

3.3 Specification of Oribatida

At its core, Oribatida is a variant of the monkey-wrap design [40], but adds a ciphertext masking. This section considers a slightly updated version of Oribatida. Section 3.8 discusses the update from Oribatida v1.2 from [50] to Oribatida v1.3 in this chapter.

In the following, let $P \in \mathsf{Perm}(\mathcal{B})$ be permutations. We denote by (X_i, Y_i) the inputs and by (U_i, V_i) the outputs of the primitive(s). As in the classical sponge, Oribatida considers the state $S_i = (U_i \,\|\, V_i)$ as a rate U_i of r bits, where inputs are XORed to, and a capacity V_i of $c = n - r$ bits. Unlike the usual sponge, an s-bit part of the capacity is used to mask the subsequent ciphertext block. The complete specification is given in Algorithm 1 and a pictorial depiction of the same is given in Figure 3.3. We assume that the key size is at most the capacity, $k \leq c$, and the tag size is at most $\tau \leq r$ bits.

3.3.1 Initialisation

Each variant of Oribatida uses a fixed-size nonce N, whose length ν is such that $k + \nu = n$ bits. N is concatenated with the key K to initialise the state: $N \,\|\, K$: $(X_0, Y_0) \leftarrow (N \,\|\, K) \oplus_d \langle d_N \rangle_d$. The domain d_N is XORed to the d least significant bits of the initial state. The first value S_1 results from $S_1 \leftarrow P(U_0 \,\|\, V_0)$; V_1 is stored for masking the first block of ciphertext later.

Algorithm 1 Specification of Oribatida. The domain-encoding functions GetDomainForN, GetDomainForA, and GetDomainForE are instantiation-specific. They are defined in Algorithm 2.

101: **function** $\mathcal{E}_K^{N,A}(M)$
102: $\quad \ell_A \leftarrow |A|$
103: $\quad \ell_E \leftarrow |M|$
104: $\quad d_N \leftarrow \textsc{GetDomainForN}(\ell_A, \ell_E)$
105: $\quad d_A \leftarrow \textsc{GetDomainForA}(\ell_A, \ell_E)$
106: $\quad d_E \leftarrow \textsc{GetDomainForE}(\ell_E)$
107: $\quad$ **if** $|A|= 0$ **then** $A \leftarrow 1 \,\|\, 0^{r-1}$
108: $\quad A \leftarrow \mathsf{pad}_r(A)$
109: $\quad M \leftarrow \mathsf{pad}_r(M)$
110: $\quad (S_1, V_1) \leftarrow \textsc{Init}(K, N, d_N, \ell_A)$
111: $\quad S_{a+1} \leftarrow \textsc{ProcessAD}(S_1, A, d_A)$
112: $\quad (C, T) \leftarrow \textsc{Encrypt}(S_{a+1}, M, V_1, d_E, \ell_E)$
113: $\quad$ **return** (C, T)

121: **function** $\textsc{Encrypt}(S_{a+1}, M, V_1, d_E, \ell_E)$
122: $\quad x \leftarrow \ell_E \bmod r$
123: $\quad M_1 \cdots M_m \leftarrow_r M$
124: $\quad V \leftarrow V_1$
125: $\quad$ **for** $i = 1 \cdots m$ **do**
126: $\quad\quad (U_{a+i}, V_{a+i}) \xleftarrow{r,c} S_{a+i}$
127: $\quad\quad X_{a+i} \leftarrow M_i \oplus U_{a+i}$
128: $\quad\quad C_i \leftarrow X_{a+i} \oplus_s \mathsf{lsb}_s(V)$
129: $\quad\quad Y_{a+i} \leftarrow V_{a+i}$
130: $\quad\quad$ **if** $i = m$ **then**
131: $\quad\quad\quad Y_{a+i} \leftarrow Y_{a+i} \oplus_d d_E$
132: $\quad\quad\quad C_m \leftarrow \mathsf{msb}_x(C_m)$
133: $\quad\quad V \leftarrow V_{a+i}$
134: $\quad\quad S_{a+i+1} \leftarrow P(X_{a+i} \,\|\, Y_{a+i})$
135: $\quad C \leftarrow (C_1 \,\|\, C_2 \,\|\, \cdots \,\|\, C_m)$
136: $\quad T \leftarrow \mathsf{msb}_\tau(S_{a+m+1}) \oplus_s \mathsf{lsb}_s(V)$
137: $\quad$ **return** (C, T)

141: **function** $\textsc{Init}(K, N, d_N, \ell_A)$
142: $\quad V_0 \leftarrow \mathsf{lsb}_s(N \,\|\, K)$
143: $\quad S_1 \leftarrow P((N \,\|\, K) \oplus_d d_N)$
144: $\quad V_1 \leftarrow \mathsf{lsb}_s(S_1)$
145: $\quad$ **return** (S_1, V_1)

151: **function** $\textsc{ProcessAD}(S_1, A, d_A)$
152: $\quad A_1 \cdots A_a \leftarrow_r A$
153: $\quad$ **for** $i = 1 \cdots a-1$ **do**
154: $\quad\quad S_{i+1} \leftarrow P(S_i \oplus (A_i \,\|\, 0^c))$
155: $\quad S_{a+1} \leftarrow P(S_a \oplus (A_a \,\|\, 0^c) \oplus_d d_A)$
156: $\quad$ **return** S_{a+1}

201: **function** $\mathcal{D}_K^{N,A}(C, T)$
202: $\quad \ell_A \leftarrow |A|$
203: $\quad \ell_E \leftarrow |C|$
204: $\quad d_N \leftarrow \textsc{GetDomainForN}(\ell_A, \ell_E)$
205: $\quad d_A \leftarrow \textsc{GetDomainForA}(\ell_A, \ell_E)$
206: $\quad d_E \leftarrow \textsc{GetDomainForE}(\ell_E)$
207: $\quad$ **if** $|A|= 0$ **then** $A \leftarrow 1 \,\|\, 0^{r-1}$
208: $\quad A \leftarrow \mathsf{pad}_r(A)$
209: $\quad C \leftarrow \mathsf{pad}_r(C)$
210: $\quad (S_1, V_1) \leftarrow \textsc{Init}(K, N, d_N, \ell_A)$
211: $\quad S_{a+1} \leftarrow \textsc{ProcessAD}(S_1, A, d_A)$
212: $\quad (M, T') \leftarrow \textsc{Decrypt}(S_{a+1}, C, V_1, d_E, \ell_E)$
213: $\quad$ **if** $T = T'$ **then return** M
214: $\quad$ **else return** $\perp$

221: **function** $\textsc{Decrypt}(S_{a+1}, C, V_1, d_E, \ell_E)$
222: $\quad x \leftarrow \ell_E \bmod r$
223: $\quad V \leftarrow V_1$
224: $\quad$ **if** $\ell_E = 0$ **then**
225: $\quad\quad T' \leftarrow \mathsf{msb}_\tau(S_{a+1}) \oplus_s \mathsf{lsb}_s(V)$
226: $\quad\quad$ **return** (ε, T')
227: $\quad C_1 \cdots C_m \leftarrow_r C$
228: $\quad$ **for** $i = 1 \cdots m$ **do**
229: $\quad\quad (U_{a+i}, V_{a+i}) \xleftarrow{r,c} S_{a+i}$
230: $\quad\quad X_{a+i} \leftarrow C_i \oplus_s \mathsf{lsb}_s(V)$
231: $\quad\quad Y_{a+i} \leftarrow V_{a+i}$
232: $\quad\quad M_i \leftarrow U_{a+i} \oplus X_{a+i}$
233: $\quad\quad$ **if** $i = m$ **then**
234: $\quad\quad\quad Y_{a+i} \leftarrow Y_{a+i} \oplus_d d_E$
235: $\quad\quad\quad M_m \leftarrow \mathsf{msb}_x(M_m)$
236: $\quad\quad V \leftarrow V_{a+i}$
237: $\quad\quad S_{a+i+1} \leftarrow P(X_{a+i} \,\|\, Y_{a+i})$
238: $\quad M \leftarrow (M_1 \,\|\, M_2 \,\|\, \cdots \,\|\, M_m)$
239: $\quad T' \leftarrow \mathsf{msb}_\tau(S_{a+m+1}) \oplus_s \mathsf{lsb}_s(V)$
240: $\quad$ **return** (M, T')

241: **function** $\textsc{pad}_x(X)$
242: $\quad$ **if** $|X| \bmod x = 0$ **then return** X
243: $\quad$ **return** $X \,\|\, 1 \,\|\, 0^{x-(|X| \bmod x)-1}$

251: **function** $\textsc{lsb}_x(X)$
252: $\quad$ **if** $|X| \leq x$ **then return** X
253: $\quad$ **return** $X[(|X|-x-1) \cdots 0]$

261: **function** $\textsc{msb}_x(X)$
262: $\quad$ **if** $|X| \leq x$ **then return** X
263: $\quad$ **return** $X[(|X|-1) \cdots (|X|-x)]$

3.3.2 Processing Associated Data

After the initialisation, the associated data A is split into r-bit blocks and is absorbed in the rate. If its length is not a multiple of r bits, A is padded with a 10^*-padding if $|A| \bmod r \not\equiv 0$ such that its length becomes the next highest multiple of r bits. If the associated data is empty, it is padded to one full block 10^{r-1}. In this case, we denote the length of the padded associated data A in blocks also as $a = 1$. The padded A is split into r-bit blocks $(A_1, \cdots, A_a)$. Given $(U_i, V_i) \xleftarrow{r,c} S_i$, A_i is XORed to the rate of the state: $X_i \leftarrow U_i \oplus A_i$, for $1 \leq i < a$. For all non-final

blocks of A, the capacity of the permutation output, V_i, is simply forwarded to that of the subsequent input to the permutation P: $Y_i \leftarrow V_i$. The state is updated with P afterwards, for all $1 < i < a$ (that is, except the final a-th block of A): $S_{i+1} \leftarrow P(X_i \,\|\, Y_i)$. When the final block A_a is processed, a domain d_A is XORed to the least significant byte of the capacity.

3.3.3 Encryption

After A has been processed, the message M is encrypted. Similarly as for the associated data, if its length is not a multiple of r bits, M is padded with a 10^*-padding such that its length after padding becomes the next highest multiple of r bits. An empty message $M = \varepsilon$ will not be padded.
After M is split into r-bit blocks $(M_1, \cdots, M_m)$ (after padding if necessary), the blocks M_i are processed one after the other. Given the state value $(U_{a+i}, V_{a+i}) \xleftarrow{r,c} S_{a+i}$, the current block M_i is XORed to the rate U_{a+i}: $X_{a+i} \leftarrow M_i \oplus U_{a+i}$. The capacity is simply forwarded: $Y_{a+i} \leftarrow V_{a+i}$. Then, $(X_{a+i} \,\|\, Y_{a+i})$ is used as input to a call to P to derive the next state value $S_{a+i+1} \leftarrow P(X_{a+i} \,\|\, Y_{a+i})$.
The ciphertext blocks C_i are computed from a sum of the current rate, the current plaintext block, and a (partial) earlier mask value from the capacity. The first ciphertext block is computed from $C_1 \leftarrow X_{a+i} \oplus_s \mathsf{lsb}_s(V_1)$. If C_1 is the final ciphertext block, it is computed as $C_1 \leftarrow \mathsf{msb}_{\ell_E}(X_{a+i} \oplus_s \mathsf{lsb}_s(V_1))$, where ℓ_E denotes the length of M before padding. Non-final ciphertext blocks C_i, $1 < i < m$ are computed from $C_i \leftarrow X_{a+i} \oplus_s \mathsf{lsb}_s(V_{a+i-1})$, for $1 < i < m$. If $m > 1$, the final ciphertext block results from $C_m \leftarrow \mathsf{msb}_{\ell_E \bmod r}(X_{a+m} \oplus_s \mathsf{lsb}_s(V_{a+m-1}))$. For the final message block, a domain d_E is XORed to the least significant byte of the capacity: $Y_{a+m} \leftarrow V_{a+m} \oplus_d \langle d_E \rangle_d$. Similar as for A, d_E uses three pairwise distinct domains depending on whether M was empty, non-empty and required no padding, or non-empty and has been padded. P is called another time to derive $S_{a+m+1} \leftarrow P(X_{a+m} \,\|\, Y_{a+m})$. Its rate is XORed with the most significant τ bits of the key V_{a+m}, and – truncated to τ bits if necessary – is released as the authentication tag: $T \leftarrow \mathsf{msb}_\tau(S_{a+m+1}) \oplus_s \mathsf{lsb}_s(V_{a+m})$. Note that, for $s = \tau$ as for our instantiations, the tag is masked as the ciphertext output blocks, which unifies this process.

Algorithm 2 Instantiated domains

```
11: function GetDomainForN(ℓ_A, ℓ_E)
12:     if ℓ_A = 0 ∧ ℓ_E = 0 then return ⟨9⟩_d
13:     return ⟨5⟩_d
21: function GetDomainForA(ℓ_A, ℓ_E)
22:     if ℓ_A = 0 ∧ ℓ_E = 0 then return ⟨0⟩_d
23:     if ℓ_A = 0 ∧ ℓ_E > 0 then return ⟨7⟩_d
24:     if ℓ_E > 0 ∧ ℓ_A mod r ≡ 0 then return ⟨4⟩_d
25:     if ℓ_E > 0 ∧ ℓ_A mod r ≢ 0 then return ⟨6⟩_d
26:     if ℓ_E = 0 ∧ ℓ_A mod r ≡ 0 then return ⟨12⟩_d
27:     if ℓ_E = 0 ∧ ℓ_A mod r ≢ 0 then return ⟨14⟩_d
31: function GetDomainForE(ℓ_E)
32:     if ℓ_E mod r ≡ 0 then return ⟨13⟩_d
33:     if ℓ_E mod r ≢ 0 then return ⟨15⟩_d
```

TABLE 3.3: Instantiated domains of Oribatida. • = yes, – = no

$\lvert A\rvert$		$\lvert M\rvert$		Domains		
>0	$\bmod r \equiv 0$	>0	$\bmod r \equiv 0$	d_N	d_A	d_E
–	–	–	–	$\langle 9\rangle_d$	$\langle 0\rangle_d$	–
–	–	•	–	$\langle 5\rangle_d$	$\langle 7\rangle_d$	$\langle 15\rangle_d$
–	–	•	•	$\langle 5\rangle_d$	$\langle 7\rangle_d$	$\langle 13\rangle_d$
•	–	–	–	$\langle 5\rangle_d$	$\langle 14\rangle_d$	–
•	–	•	–	$\langle 5\rangle_d$	$\langle 6\rangle_d$	$\langle 15\rangle_d$
•	–	•	•	$\langle 5\rangle_d$	$\langle 6\rangle_d$	$\langle 13\rangle_d$
•	•	–	–	$\langle 5\rangle_d$	$\langle 12\rangle_d$	–
•	•	•	–	$\langle 5\rangle_d$	$\langle 4\rangle_d$	$\langle 15\rangle_d$
•	•	•	•	$\langle 5\rangle_d$	$\langle 4\rangle_d$	$\langle 13\rangle_d$

3.3.4 Decryption

The decryption takes a tuple (K, N, A, C, T). The initialisation with K and N as well as the processing of the associated data A is performed in the same manner as for encryption. If $|C| \bmod r \neq 0$, the decryption pads C with a 10^*-padding to the next multiple of r bits. In all cases, it splits C into r-bit blocks $(C_1, \cdots, C_{m-1})$ plus a final block C_m. If $m > 1$, the plaintext block is computed as $X_{a+i} \leftarrow C_i \oplus_s \mathsf{lsb}_s(V)$ and $M_i \leftarrow (U_{a+i} \oplus X_{a+i})$, where $V = V_1$ for $i = 1$, and $V = V_{a+i-1}$ otherwise. The capacity is simply forwarded to the next call of the permutation: $Y_{a+i} \leftarrow V_{a+i}$. The subsequent state is then $(U_{a+i+1} \,\|\, V_{a+i+1}) \leftarrow S_{a+i+1} \leftarrow P(X_{a+i} \,\|\, Y_{a+i})$.
The final plaintext block is computed from the padded ciphertext block C_m as $X_{a+m} \leftarrow C_m \oplus_s \mathsf{lsb}_s(V)$ and $M_m \leftarrow \mathsf{lsb}_x(U_{a+m} \oplus X_{a+m})$, where $x \leftarrow \ell_E \bmod r$. For the final block, the domain d_E is XORed to the least significant byte of the capacity: $Y_{a+m} \leftarrow V_{a+m} \oplus_d \langle d_E \rangle_d$. The would-be tag T' is derived by computing $(T' \,\|\, Z) \leftarrow P(X_{a+m} \,\|\, Y_{a+m}) \oplus_s \mathsf{lsb}_s(V_{a+m})$, and using only its most significant τ bits: $T' \leftarrow \mathsf{msb}_\tau(T' \,\|\, Z)$ as for the encryption If $T = T'$, the ciphertext is considered valid, and $M = (M_1 \,\|\, \cdots \,\|\, M_m)$ is released as plaintext. Otherwise, the ciphertext is deemed invalid, and $\perp$ is returned.

3.3.5 Domain Separation

For domain separation, Oribatida defines constants d_N, d_A and d_E. The domains are XORed with the least significant byte of the state at three stages. Domains are encoded as d-bit strings, where $d = 4$ bits suffice in practice. The value depends on the presence of A and M and whether their final blocks are absent, partial, or integral to prevent trivial collisions of inputs to P among blocks of A and M.

The constants are determined by four bits (t_3, t_2, t_1, t_0) that reflect inputs in the hardware API, similar to, e.g., [48]:

- **EOI:** t_3 is the **end-of-input** control bit. This bit is set to 1 if the current data block is the final block of the input. Note that if the associated data is empty, then the created 10^{r-1}-block is never treated as the final block of the input.

- **EOT:** t_2 is the **end-of-type** control bit. This bit is set to 1 if the current data block is the final block of the same type, i.e., it is the last block of the nonce/associated data/message. Note that if both the associated data and the message are empty, then neither the nonce nor the created 10^{r-1}-block of the associated data is considered as the final block of its type.

- **Partial:** t_1 is the **partial-control** bit. This bit is set to 1 if the size of the current block is less than the block size. Note that if the associated data is empty and the message is non-empty, then the created 10^{r-1}-block is treated as a partial block.

- **Type:** t_0 is the **type-control** bit, identifying the type of the current block. For the nonce and the final message block, $t_0 = 1$. If the associated data is empty and the message is non-empty, then $t_0 = 1$ for the created 10^{r-1}-block of the associated data. For all other cases, $t_0 = 0$.

While processing a data block, the domains are set as the integer representation of $t_3 \,\|\, t_2 \,\|\, t_1 \,\|\, t_0$. For example, processing the nonce (which is always a complete r-bit block) with empty associated data and non-empty message yields $d_N = (t_3 t_2 t_1 t_0) = (0101)_2 = 5$. Details are provided in Algorithm 2; ℓ_A denotes the length of A and ℓ_E that of M in bits before padding. An overview is given in Table 3.3.

3.4 INT-RUP Attacks on Schemes with Masked Ciphertexts

The approach of Oribatida is to employ (a portion of) the capacity of the previous permutation output to mask the ciphertext outputs. This strategy is generic enough to apply to other modes, such as Beetle or SPoC. We can informally define a masked variant of Beetle and SPoC. The masked beetle uses Z_{i-1} and XORs $\mathsf{lsb}_s(Z_{i-1})$ to the s rightmost bits of the ciphertext block C_i, for $i > 1$ and $s \leq c$.

If no associated data is present, we define $Z_0 = K_2$. A masked variant of SPoC would employ the rate (since it is the hidden part). Thus, for the masked SPoC, we define $s \leq r$ and define that $\mathsf{lsb}_s(U_{i-1})$ is XORed to C_i for $i > 0$. If no associated data is present, we define U_0 for the rate of the initial input to the permutation P.
For Oribatida, the masked Beetle or the masked SPoC, the attacks in Section 3.2 do not apply directly. However, there exist attacks on each of them with advantage $\Omega(q_d^2/2^c)$.

3.4.1 The Generic INT-RUP Attack on Oribatida (Masked Duplex)

Here, we consider an attack on Oribatida that shows that our INT-RUP bound of $O(q_d^2/2^c)$ is tight, i.e., the advantage of the attack is $\Omega(q_d^2/2^c)$.

1. $\mathcal{A}$ asks q_d decryption queries (N^i, A^i, C^i, T^i), where N^i and C^i is static for all queries. We assume that the associated data A^i are pairwise distinct and consist of a single block for all queries. $\mathcal{A}$ obtains M^i from the encryption oracle, for $1 \leq i \leq q_d$. The rate $X_2^i \leftarrow C_1^i \oplus_s \mathsf{lsb}_s(V_1^i)$ is constant for all queries.

2. For $q_d \approx O(2^{c/2})$, $\mathcal{A}$ can expect a collision in the capacity of the input: $V_2^i \leftarrow V_2^j$ for some distinct $i \neq j$, $i, j \in [1 \cdots q_d]$. Then, this collision leads to a full-state collision that can be detected when $M_1^i = M_1^j$.

3. $\mathcal{A}$ asks encryption queries (N, A^i, M^i) and obtains (C^i, T) for some tag T.

4. (N, A^j, C^i, T) is a valid forgery and yields M^j.

3.4.2 INT-RUP Attack on The Masked Beetle

1. $\mathcal{A}$ asks q_d encryption queries (N^1, A^1, M), (N^2, A^2, M), $\ldots$, (N^{q_d}, A^{q_d}, M) to the encryption oracle, and receives C^1, C^2, $\ldots$, C^{q_d}. The associated data A^i consists of a single block for each i; the message M contains $\lceil \frac{c}{r} \rceil$ blocks.

2. $\mathcal{A}$ asks $q_d - 1$ decryption queries, one for each encryption query except the first encryption query, to the decryption oracle. The decryption query of the i-th encryption query is $(N^i, A^i, C^{i'})$, where $C^{i'} = Y_2^1 \oplus Y_2^i \oplus \mathsf{shuffle}(Y_2^1) \oplus \mathsf{shuffle}(Y_2^i)$. $\mathcal{A}$ can calculate the r.h.s. as $\mathsf{shuffle}(Y_2^1) \oplus \mathsf{shuffle}(Y_2^i) = C^1 \oplus C^i$,

and $Y_2^1 \oplus Y_2^i$ directly from $\mathsf{shuffle}(Y_2^1) \oplus \mathsf{shuffle}(Y_2^i)$ with the definition of $\mathsf{shuffle}$. So, the first r bits of the input to the third permutation call always equal those of the first encryption query.

3. Afterwards, $\mathcal{A}$ repeats Step 2 to 6 from Section 3.4.1 to complete the attack.

The advantage of the attack is $\Omega(q_d^2/2^c)$. However, the attack strategy differs for SPoC.

3.4.3 INT-RUP Attack on The Masked SPoC

Here, $\mathcal{A}$ has to perform the attack in two stages.

1. First, $\mathcal{A}$ asks q_d decryption queries (N, A^1, C), (N, A^2, C), ... , (N, A^{q_d}, C) to the decryption oracle, and receives M^1, M^2, ..., M^{q_d}. The associated data A^i consists of a single block for each i; the ciphertext C consists of two blocks.

2. When $q_d \approx \mathcal{O}(2^r)$, $\mathcal{A}$ expects a collision in the first r bits of the input to the third permutation call. $\mathcal{A}$ can detect this collision by looking at the first message block because it will be equal only for the two colliding queries.

3. Suppose the associated data of the two colliding queries are A^i and A^j.

4. $\mathcal{A}$ makes q_1 queries (N, A^i, C^1), (N, A^i, C^2), ... , (N, A^i, C^{q_1}), and q_2 queries (N, A^j, C^1), (N, A^j, C^2), ... , (N, A^j, C^{q_2}) to the decryption oracle.

5. When $q_1 \cdot q_2 \approx \mathcal{O}(2^r)$, $\mathcal{A}$ expects a full state collision at the input to the third permutation, between one query with associated data A^i and another query with associated data A^j.

6. Suppose the two ciphertexts corresponding to the two colliding queries are C^p and C^q, and the corresponding messages are M^p and M^q.

7. $\mathcal{A}$ identifies those pairs of queries $(N, A^i, C^x), (N, A^j, C^y), 1 \leq x \leq q_1$ and $1 \leq y \leq q_2$, for which the sum of the second message blocks equals that of the first message blocks. For each such pair, $\mathcal{A}$ updates C^x and C^y by appending $\lceil \frac{c}{r} \rceil - 1$ ciphertext blocks to each s.t., $C_2^x = C_2^y$, ..., $C_{\lceil \frac{c}{r} \rceil}^x = C_{\lceil \frac{c}{r} \rceil}^y$, and makes decryption queries with the updated ciphertexts. For $k > 2$, M_k^p will equal M_k^q.

8. Next, $\mathcal{A}$ asks (N, A^i, M^p) to the encryption oracle; suppose, the tag is T.

9. Then, $\mathcal{A}$ successfully forges with the query (N, A^j, C^q, T).

Again, the advantage of the attack is $\Omega(q_d^2/2^c)$.

3.5 NAEAD Security Analysis

This section analyses the NAEAD security of Oribatida. In the following, let $K \xleftarrow{\$} \mathcal{K}$ and $\pi \xleftarrow{\$} \mathsf{Perm}(\mathcal{B})$. We use $\Pi[\pi,\pi]_K = \Pi[\pi]_K$ as short form of Oribatida, instantiated with π for P, and keyed by K. Let $\mathcal{A}$ be a nonce-respecting NAEAD adversary w.r.t. $\Pi[\pi]_K$. We denote by q_p, q_f, q_b, q_c, q_e, q_d, σ_c, σ_e, σ_d the number of primitive queries, forward primitive queries, backward primitive queries, construction queries, encryption queries, decryption queries, blocks summed over all construction queries, blocks summed over only all encryption queries, and blocks summed over all decryption queries, respectively. It holds that $q_p = q_f + q_b$, $q_c = q_e + q_d$, and $\sigma_c = \sigma_e + \sigma_d$. For simplicity, we define a function ρ as

$$\rho(i,j) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } j = 1 \\ a^i + j - 1 & \text{otherwise}\,. \end{cases}$$

So, $V^i_{\rho(i,j)}$ denotes the used block for masking the ciphertext block C^i_j.
Recall the notion of a longest common prefix from [53]. Let $Q = (N, A, M, C, T)$ be a query of $\mathcal{A}$ with the response. Let $\mathcal{Q}$ denote a set of queries without Q, i.e., $Q \notin \mathcal{Q}$. We define the length of the longest common prefix of M and another message M' as $\mathsf{LCP}(M, M') \stackrel{\text{def}}{=} \max_i \left\{1 \le j \le i : M_j = M'_j\right\}$. Given $\mathcal{Q}$ and M, we overload the notation by considering the longest common prefix of M with the queries in $Q' = (N', A', M', C', T') \in \mathcal{Q}$: $\mathsf{LCP}(M, \mathcal{Q}) \stackrel{\text{def}}{=} \max_{M' \in \mathcal{Q}} \{\mathsf{LCP}(M, M')\}$. We also define

$$\mathsf{LCP}^{N,A}(M, \mathcal{Q}) \stackrel{\text{def}}{=} \max_{\substack{(N',A',M',C',T') \in \mathcal{Q} \\ N'=N \wedge A'=A}} \{\mathsf{LCP}(M, M')\}\,.$$

Collisions of chaining values are trivial if in the longest common prefix and non-trivial otherwise.

Theorem 3.2 (NAEAD Security of Oribatida)**.** Let $\mathcal{A}$ be a nonce-respecting adversary w.r.t. $\Pi[\pi]_K$. Then, $\mathbf{Adv}^{\Pi[\pi]_K}_{\text{NAEAD}}(\mathcal{A})$ is upper bounded by

$$\frac{\binom{\sigma}{r} + 2\binom{q_p}{r}}{2^{r(r-1)}} + \frac{\sigma^2}{2^n} + \frac{3q_p}{2^k} + \frac{r(q_d + \sigma_d) + 2\sigma_e q_p + q_p q_c + q_d(\sigma_e + q_p) + 2rq_p}{2^{c+s}} + \frac{q_d}{2^\tau}\,.$$

Proof. We follow the strategy of the NAEAD proof of Beetle [48]. The queries by $\mathcal{A}$ and their corresponding answers are collected in a transcript $\tau = (\tau_e, \tau_d, \tau_p)$. In that transcript, the encryption construction queries are stored as tuples $\tau_e = \{(N^i, A^i, M^i, C^i, T^i)\}$, for $1 \leq i \leq q_e$, the decryption construction queries are stored as tuples $\tau_d = \{(N^i, A^i, M^i, C^i, T^i)\}$, for $1 \leq i \leq q_d$, and primitive queries are stored as tuples $\tau_p = \{(Q^i, R^i)\}$, where $\pi(Q^i) = R^i$, for $1 \leq i \leq q_p$.

Sampling. We define the ideal oracle to consist of an online and an offline phase. In the online phase, the ideal oracle samples the responses (C^i, T^i) uniformly at random from the bit strings of expected lengths for encryption queries. For decryption queries, it always outputs $\perp$. For forward primitive queries Q^i, it forwards the result of $\pi(Q^i)$ to $\mathcal{A}$; for backward primitive queries R^i, it forwards the result of $\pi^{-1}(R^i)$.
In the offline phase, the ideal oracle samples the internal chaining values $V^i_j \xleftarrow{\$} \{0,1\}^c$ and $U^i_j \xleftarrow{\$} \{0,1\}^r$ uniformly at random for all construction queries in encryption direction. It derives the analogous internal chaining values V^i_j, for $1 \leq j \leq k$ for all construction queries in decryption direction (N^i, A^i, C^i, T^i) that share $N^i, A^i = N^{i'}, A^{i'}$, and for which $C^i_1 = C^{i'}_1$, …, $C^i_k = C^{i'}_k$ holds for some i'-th construction query, where $i \neq i'$. Moreover, for construction queries whose plaintext or ciphertext length is not a multiple of r bits, the oracle samples exactly the missing bits C^i_m uniformly independently at random that are not fixed from previous queries, at most $r - |C^i_{m^i}|$ bits at a time. The so-sampled values for the final blocks $C^i_{m^i}$ are stored also in the transcript. Moreover, the random key K is revealed to $\mathcal{A}$ after the offline phase.

Bad Events. We define the following bad events. If any of them occurs, the adversary aborts, and we define that it wins in this case.

- bad_1: Multi-collision on the rate X in encryption construction queries. For some $w \geq r$, $\exists$ indices (i_1, j_1), (i_2, j_2), …, (i_w, j_w) with i_1, i_2, …, $i_w \in [1 \cdots q_e]$, and $j_1 \in [1 \cdots a^{i_1}]$, $j_2 \in [1 \cdots a^{i_2}]$, etc. s. t. $X^{i_1}_{j_1} = X^{i_2}_{j_2} = \ldots = X^{i_w}_{j_w}$.
- bad_2: Collision of permutation inputs in encryption construction queries: $\exists$ indices $(i, j) \neq (i', j')$ with $i, i' \in [1 \cdots q_e]$, $j \in [1 \cdots m^i]$, and $j' \in [1 \cdots m^{i'}]$ s. t. $\left(X^i_j \,\|\, Y^i_j\right) = \left(X^{i'}_{j'} \,\|\, Y^{i'}_{j'}\right)$.

- bad_3: Collision of permutation outputs in encryption construction queries: $\exists$ indices $(i,j) \neq (i',j')$ with $i,i' \in [1 \cdots q_e]$, $j \in [1 \cdots m^i]$, and $j' \in [1 \cdots m^{i'}]$ s. t. $\left(U_j^i \,\|\, V_j^i\right) = \left(U_{j'}^{i'} \,\|\, V_{j'}^{i'}\right)$.
- bad_4: Collision of permutation inputs between a construction and a primitive query: $\exists$ indices (i,j,i') with $i \in [1 \cdots q_e]$, $j \in [1 \cdots m^i]$, and $i' \in [1 \cdots q_p]$ s. t. $\left(X_j^i \,\|\, Y_j^i\right) = Q^{i'}$.
- bad_5: Collision of permutation outputs between a construction and a primitive query: $\exists$ indices (i,j,i') with $i \in [1 \cdots q_e]$, $j \in [1 \cdots m^i]$, and $i' \in [1 \cdots q_p]$ s. t. $\left(U_j^i \,\|\, V_j^i\right) = R^{i'}$.
- bad_6: Initial-state collision with a primitive query: $\exists$ indices (i,i') with $i \in [1 \cdots q_e]$ and $i' \in [1 \cdots q_p]$ s. t. $(X_0^i \,\|\, Y_0^i) = Q^{i'}$.
- bad_7: Multi-collision in the rate of w outputs of forward primitive queries: for some $w \geq r$, $\exists$ $i_1, i_2, \ldots, i_w \in [1 \cdots q_p]$ s. t. $\mathsf{msb}_r(R^{i_1}) = \mathsf{msb}_r(R^{i_2}) = \cdots = \mathsf{msb}_r(R^{i_w})$.
- bad_8: Multi-collision in the rate of w outputs of backward primitive queries: for some $w \geq r$, $\exists$ $i_1, i_2, \ldots, i_w \in [1 \cdots q_p]$ s. t. $\mathsf{msb}_r(Q^{i_1}) = \mathsf{msb}_r(Q^{i_2}) = \cdots = \mathsf{msb}_r(Q^{i_w})$.

We define that the adversary is provided with all internal chaining values V_j^i and U_j^i after its interaction, but before it outputs its decision bit, which only strengthens the adversary. We denote by Θ_{real} and Θ_{ideal} random variables that represent the distribution of transcripts in the real and the ideal world, respectively. We define the set of bad transcripts BADT, to contain exactly those attainable transcripts τ for which at least one of the bad events occurred. It holds that $\Pr[\Theta_{\text{ideal}} \in \text{BADT}] \leq \sum_{i=1}^{8} \Pr[\mathsf{bad}_i]$. The probability of bad transcripts in the ideal world is treated in the proof of Lemma 3.3. The ratio of good transcripts is bounded in Lemma 3.4. Our bound in Theorem 3.2 follows from them and the fundamental Lemma of the H-coefficient Technique [54], using $w = r$ in the bounds.

Lemma 3.3. Let $w \geq r$ be a positive integer. It holds that

$$\Pr[\Theta_{\text{ideal}} \in \text{BADT}] \leq \frac{\binom{\sigma}{w} + 2\binom{q_p}{w}}{2^{r(w-1)}} + \frac{\sigma^2}{2^n} + \frac{3q_p}{2^k} + \frac{2\sigma_e q_p + q_p q_c + 2w \cdot q_p}{2^{c+s}}\,.$$

Proof. In the following, we upper bound the probabilities of the individual bad events.

bad_1: Multi-collision on X in encryption construction queries. In the ideal world, the ciphertext blocks are sampled independently and uniformly at random from the strings of expected length. The internal values X^i_j can be computed by $\mathcal{A}$ once it is given the transcript including the internal chaining values V^i_j. It must hold that $X^i_j \leftarrow C^i_{j-a^i} \oplus_s \mathsf{lsb}_s(V^i_{\rho(i,j-a^i)})$. The random sampling of C implies that the probability of the values X^i_j to take any specific r-bit value is $1/2^r$. Note that in the case of a padded ciphertext block, each padded bit of $C^i_{m^i}$ is also sampled once randomly and given in the transcript. Hence, the probability for $X^i_{a^i+m^i}$ to take any r-bit value is also 2^{-r} in the ideal world. For fixed indices (i_1, j_1), (i_2, j_2), …, (i_w, j_w), it holds that

$$\Pr\left[X^{i_1}_{j_1} = X^{i_2}_{j_2} = \ldots = X^{i_w}_{j_w}\right] \leq 2^{-r(w-1)}\,.$$

Over all queries and blocks of τ_e, it follows that

$$\Pr\left[\mathsf{bad}_1\right] \leq \frac{\binom{\sigma}{w}}{2^{r(w-1)}}\,.$$

bad_2: Collision of two permutation inputs in encryption construction queries. Here, we consider

$$\Pr\left[\left(X^i_j \,\|\, Y^i_j\right) = \left(X^{i'}_{j'} \,\|\, Y^{i'}_{j'}\right)\right]\,.$$

All ciphertext blocks and the internal chaining values $V^i_{\rho(i,j-a^i)}$, $j > a^i$ are sampled independently and uniformly at random. Moreover, padded bits of ciphertexts are sampled also independently and uniformly at random. Though, we have to consider two cases;

- $j = 0 \wedge j' = 0$: since X^i_0 and $X^{i'}_0$ contain nonces, and since we assume $\mathcal{A}$ to be nonce-respecting, the probability for a collision is zero in this case.
- $j > 0$: In this case, $Y^i_j = V^i_j \oplus \text{const}$, where const $\in \{d_N, d_A, d_E\}$ is a public constant. Moreover, X^i_j is derived from C^i_j; so, both X^i_j and Y^i_j are chosen independently and uniformly at random, and the probability for a collision in this case is at most 2^{-n}.

Therefore, for fixed indices $(i, j) \neq (i', j')$, the probability is

$$\Pr\left[\left(X^i_j \,\|\, Y^i_j\right) = \left(X^{i'}_{j'} \,\|\, Y^{i'}_{j'}\right)\right] \leq 2^{-n}\,.$$

Over all combinations of indices, it follows that

$$\Pr\left[\mathsf{bad}_2\right] \leq \frac{\binom{\sigma}{2}}{2^n}\,.$$

bad_3: Collision of two permutation outputs in encryption construction queries. This case is analogous to bad_2. The permutation outputs V_j^i are sampled randomly. In all cases, it holds that

$$\Pr\left[\left(U_j^i \,\|\, V_j^i\right) = \left(U_{j'}^{i'} \,\|\, V_{j'}^{i'}\right)\right] \leq 2^{-n}\,.$$

Over all combinations of indices, it follows that

$$\Pr\left[\mathsf{bad}_3\right] \leq \frac{\binom{\sigma}{2}}{2^n}\,.$$

bad_4: Collision of permutation inputs between a construction and a primitive query. Again, we consider $X_j^i \leftarrow C_{j-a^i}^i \oplus_s \mathsf{lsb}_s(V_{\rho(i,j-a^i)}^i)$ and $Y_j^i \leftarrow V_j^i \oplus \mathrm{const}$, where const is a public constant. The values $C_{j-a^i}^i$ and $V_{\rho(i,j-a^i)}^i$ are sampled randomly, the values $(X_j^i \,\|\, Y_j^i)$ take any value with probability at most 2^{-n}.

1. Assume, the primitive query was asked before the construction query. If the construction query was in encryption direction, the collision probability for fixed queries is at most 2^{-n}, for $q_p \cdot q_c$ combinations.

2. If the primitive query was asked after an encryption query, then, the latter one produced a tag. If the primitive query starts at any other block, $\mathcal{A}$ can see $r - s$ bits. Hence, the probability is at most $2^{-(c+s)}$ for $q_p \cdot \sigma_e$ combinations. If the primitive query starts from the tag, the adversary sees $c + s$ unmasked bits. Assuming $\overline{\mathsf{bad}}_1$, there are at most w equal tags over all encryption queries. So, the probability for a collision is $2^{-(n-\tau)}$, for at most $w \cdot q_p$ combinations.

Over all combinations of indices, it follows that

$$\Pr\left[\mathsf{bad}_4 | \overline{\mathsf{bad}}_1\right] \leq \max\left(\frac{\sigma_e \cdot q_p}{2^n}, \frac{\sigma_e \cdot q_p}{2^{c+s}}\right) + \frac{w \cdot q_p}{2^{c+s}} \leq \frac{\sigma_e \cdot q_p}{2^{c+s}} + \frac{w \cdot q_p}{2^{c+s}}\,.$$

bad_5: Collision of permutation outputs between an encryption construction query and a primitive query. Again, $U_{j+a^i}^i$ can be derived from

$M_j^i \oplus C_j^i \oplus_s \mathsf{lsb}_s(V_{\rho(i,j)}^i)$ and the values C_j^i and $V_{\rho(i,j)}^i$ are sampled randomly. So, the values $U_{j+a^i}^i \,\|\, V_{j+a^i}^i$ take any value with probability at most 2^{-n}. If the primitive query starts at any other block, $\mathcal{A}$ can see $r-s$ bits. Hence, the probability is at most $2^{-(c+s)}$ for $q_p \cdot \sigma_e$ combinations. Following a similar argument as for bounding bad_4 and excluding bad_1, we obtain over all combinations of indices that

$$\Pr\left[\mathsf{bad}_5 | \overline{\mathsf{bad}_1}\right] \leq \frac{\sigma \cdot q_p}{2^{c+s}} + \frac{w \cdot q_p}{2^{c+s}}\,.$$

bad_6: Initial-state collision with a primitive query. Here, we know that the key is chosen uniformly at random. We distinguish between collisions depending on whether the primitive query was a forward query or a backward query.

1. If the primitive query was a forward query, it must hit the correct value of $K \oplus_d d_N$. So, the probability is at most $q_p/2^k$ to collide with encryption construction queries. Considering also decryption queries, a nonce can repeat but change d_N. Since there exist at most three distinct values for d_N, the probability is at most $3q_p/2^k$ to collide.

2. If the primitive query was in backward direction, its response must hit any initial state of a construction query. If the construction query was asked before the primitive, $\mathcal{A}$ sees at best $r-s$ bits of C_1. Then, the probability is at most $q_c \cdot q_p/2^{c+s}$.

3. If the primitive query was asked before the construction query, $\mathcal{A}$ can use the nonce part of the primitive query's result as a nonce. Though, a collision needs the key part to be correct, which holds with probability at most $3q_p/2^k$.

Over all possible options, we obtain

$$\Pr\left[\mathsf{bad}_6\right] \leq \frac{3q_p}{2^k} + \frac{q_c \cdot q_p}{2^{c+s}}\,.$$

bad_7: Multi-collision in the rate of w outputs of forward primitive queries. Since π is chosen randomly from the set of all permutations, the outputs are chosen randomly from a set of size $2^n - (i-1)$ for the i-th primitive query. So, the probability for w distinct queries to collide in their rate is at most $1/2^{r(w-1)}$ as for bad_7 in the NAEAD proof. Over all queries, the probability is upper bounded by

$$\Pr\left[\mathsf{bad}_7\right] \leq \frac{\binom{q_p}{w}}{2^{r(w-1)}}\,.$$

bad_8: Multi-collision in the rate of w outputs of backward primitive queries. Following a similar argumentation as for bad_7, we obtain

$$\Pr\left[\mathsf{bad}_8\right] \leq \frac{\binom{q_p}{w}}{2^{r(w-1)}} .$$

Our bound in Lemma 3.3 follows from summing up all probabilities.

Lemma 3.4. Let $\tau \in \textsc{GoodT}$. Then

$$\frac{\Pr[\Theta_{\text{real}} = \tau]}{\Pr[\Theta_{\text{ideal}} = \tau]} \leq 1 - \left(\frac{q_d}{2^\tau} + \frac{q_d(q_p + \sigma_e)}{2^{c+s}} + \frac{(\sigma_d + q_d) \cdot w}{2^{c+s}}\right) .$$

Proof. It remains to lower bound the ratio of real and ideal probability of obtaining a good transcript τ. Let $\tau = (\tau_e, \tau_d, \tau_p)$ be an attainable transcript, where $\tau_d = \bot_{\text{all}}$ contains only $\bot$ for all responses. Since all ciphertext-block outputs and all internal chaining values in encryption queries are sampled independently and uniformly at random, their probability is $1/2$ per bit. We define $\sigma_{\mathsf{distinct}}$ for the number of distinct calls to the permutation over all encryption and decryption queries. In the ideal world, it holds that

$$\begin{aligned}\Pr\left[\Theta_{\text{ideal}} = \tau\right] &= \Pr\left[K\right] \cdot \Pr\left[\tau_e \wedge \tau_p \wedge \tau_d\right] \\ &= \Pr\left[K\right] \cdot \Pr\left[\tau_e\right] \cdot \Pr[\tau_p] \cdot \Pr[\tau_d] = \frac{1}{2^k} \cdot \frac{1}{(2^n)^{\sigma_{\mathsf{distinct}}}} \cdot \frac{1}{(2^n)_{q_p}} \cdot 1\end{aligned}$$

since the outputs from encryption queries are sampled uniformly at random; so, the encryption and decryption transcripts τ_e and τ_d are independent from τ_p.
In the real world, the probabilities for choosing K as key and π as permutation are equal to those of the ideal world. We can separate the probability into

$$\Pr\left[\Theta_{\text{real}} = \tau\right] = \Pr\left[K\right] \cdot \Pr\left[\tau_e \wedge \tau_p \wedge \tau_d\right] = \frac{1}{2^k} \cdot \Pr\left[\tau_e, \tau_d | \tau_p\right] \cdot \Pr\left[\tau_p\right]$$

since the encryption and the decryption transcript depend on the choice of the permutation π. Let $\top_i$ denote that the i-th decryption query was a valid forgery. We can upper bound

$$\begin{aligned}\Pr\left[\tau_e, \tau_d | \tau_p\right] \cdot \Pr\left[\tau_p\right] &\leq \Pr\left[\tau_e | \tau_p\right] \cdot \Pr\left[\tau_p\right] - \left(\sum_{i=1}^{q_d} \Pr\left[\tau_e \wedge \top_i | \tau_p\right] \cdot \Pr\left[\tau_p\right]\right) \\ &= \Pr\left[\tau_e | \tau_p\right] \cdot \Pr\left[\tau_p\right] - \Pr\left[\tau_e | \tau_p\right] \cdot \Pr\left[\tau_p\right] \cdot \left(\sum_{i=1}^{q_d} \Pr\left[\top_i | \tau_e | \tau_p\right] \cdot \Pr\left[\tau_p\right]\right)\end{aligned}$$

$$= (\Pr[\tau_e|\tau_p] \cdot \Pr[\tau_p]) \cdot (1-\epsilon) \;, \tag{3.1}$$

where we define

$$\epsilon \stackrel{\text{def}}{=} \sum_{i=1}^{q_d} \epsilon_i \qquad \text{and} \qquad \epsilon_i \stackrel{\text{def}}{=} \Pr[\top_i|\tau_e|\tau_p] \cdot \Pr[\tau_p] \;.$$

The probability of primitive queries is given by the fraction of all permutations π that would produce τ_p, which is

$$\Pr[\tau_p] = \frac{1}{(2^n)_{q_p}} \,,$$

as in the ideal world. The ciphertext blocks C_j^i from encryption queries as well as the chaining values V_j^i are results from the permutation π and hence, depend on the permutation. Since τ is a good transcript, there are no undesired collisions, e.g., between primitive and construction queries. Hence, all internally computed values $(U_j^i \,\|\, V_j^i)$ – note that U_j^i can be derived from $C_{j-a^i}^i \oplus_s \mathsf{lsb}_s(V_{\rho(i,j-a^i)}^i) \oplus M_{j-a^i}^i$ by the adversary – are results of fresh values or predefined in decryption queries from the result of previous encryption queries. Then, the probabilities for the outputs of π in construction queries are given by $1/(2^n)_{\sigma_{\mathsf{distinct}}}$. It is not difficult to see that for positive $\sigma_{\mathsf{distinct}}$, the ratio of the interpolation probabilities from Equation (3.1) can be bounded by

$$\frac{(\Pr[\tau_e|\tau_p|\Theta_{\text{real}}] \cdot \Pr[\tau_p|\Theta_{\text{real}}])}{(\Pr[\tau_e|\Theta_{\text{ideal}}])} = \frac{\frac{1}{(2^n)_{\sigma_{\mathsf{distinct}}}}}{\frac{1}{(2^n)^{\sigma_{\mathsf{distinct}}}}} = \frac{(2^n)^{\sigma_{\mathsf{distinct}}}}{(2^n)_{\sigma_{\mathsf{distinct}}}} \geq 1 \,.$$

It remains to upper bound ϵ. For this purpose, we upper bound the values ϵ_i for transcripts that contain forgeries. Since τ is a good transcript, we assume that bad events do not hold. Hence, either $\top_i$ does not hold, which yields $\epsilon_i = 0$; in the opposite case, we have to consider a few mutually exclusive cases in the following. We assume that there exists a decryption query (N^i, A^i, C^i, T^i) s. t. T^i is valid. In all cases, the tag can simply be guessed correctly if the block $(X_{a^i+m^i}^i \,\|\, Y_{a^i+m^i}^i)$ is fresh. Then, the probability for the tag to be correct is $2^{-\tau}$. So, we can concentrate on the cases where it is non-fresh in the following. Prior, we define $(X_{i_1}, X_{i_2}, \ldots, X_{i_{w+1}})$ as a w-chain if there exist $(Y_{i_1}, Y_{i_2}, \ldots, Y_{i_{w+1}})$ s. t. the following chain has been obtained from primitive queries:

$$\pi(X_{i_1} \,\|\, Y_{i_1}) = (U_{i_2} \,\|\, V_{i_2}) = (U_{i_2} \,\|\, Y_{i_2}) \;,$$

$$\begin{aligned}\pi\left(X_{i_2} \,\|\, Y_{i_2}\right) &= \left(U_{i_3} \,\|\, V_{i_3}\right) = \left(U_{i_3} \,\|\, Y_{i_3}\right) \,, \\ &\vdots \\ \pi\left(X_{i_w} \,\|\, Y_{i_w}\right) &= \left(U_{i_{w+1}} \,\|\, V_{i_{w+1}}\right) = \left(U_{i_{w+1}} \,\|\, Y_{i_{w+1}}\right) \,.\end{aligned}$$

The cases are:

- **Case (A):** N^i is fresh; so, there is no earlier construction query $i' \neq i$ s. t. $N^i = N^{i'}$.
- **Case (B):** N^i is old, but (N^i, A^i) is fresh, i.e., there exists no earlier construction query $i' \neq i$ with $(N^i, A^i) = (N^{i'}, A^{i'})$.
- **Case (C):** (N^i, A^i) is old, but (N^i, A^i, C^i) is fresh, i.e., there exists no earlier construction query $i' \neq i$ with $(N^i, A^i, C^i) = (N^{i'}, A^{i'}, C^{i'})$, and no w-chain of primitive queries is hit.
- **Case (D):** (N^i, A^i, C^i) is old; (N^i, A^i, C^i) a prefix of another construction query.
- **Case (E):** (N^i, A^i) is old and there exists a w-chain of primitive queries that is hit.

Clearly, the cases cover all possible options. We assume that no previous bad events occur, in particular, no w-multi-collisions or collisions with the primitive queries occurred.

Case (A). We excluded bad_6 in this case. The probability that $(N^i \,\|\, K) \oplus_d d_N$ hits any block $(X_j^{i'} \,\|\, Y_j^{i'})$ from another construction query so that the final block is old is at most

$$\Pr\left[\left(\left(N^i \,\|\, K\right) \oplus_d d_N\right) = \left(X_j^{i'} \,\|\, Y_j^{i'}\right)\right] \leq \frac{\sigma_e}{2^{c+s}} \,.$$

Case (B). Let $p \leq a^i + m^i$ denote the length of the longest common prefix of the i-th query with all other queries. In Case (B), the probability that any block $(X_j^i \,\|\, Y_j^i)$ with $j \geq p+1$ matches the permutation input of any other encryption-query block or primitive query can be upper bounded by

$$\Pr\left[\left(X_j^i \,\|\, Y_j^i\right) = \left(X_{j'}^{i'} \,\|\, Y_{j'}^{i'}\right)\right] \leq \frac{\sigma_e}{2^{c+s}} + \frac{q_p}{2^{c+s}} \,.$$

Case (C). A similar argument as for Case (B) can be applied in Case (C). The probability that there exists $i' \neq i$, s. t. for some block indices, it holds that (j, j'): $(X_j^i \,\|\, Y_j^i) = (X_{j'}^{i'} \,\|\, Y_{j'}^{i'})$ is at most $1/2^{c+s}$. So, it holds that

$$\Pr\left[\left(X_j^i \,\|\, Y_j^i\right) = \left(X_{j'}^{i'} \,\|\, Y_{j'}^{i'}\right)\right] \leq \frac{\sigma_e}{2^{c+s}} + \frac{q_p}{2^{c+s}}\,.$$

Case (D). This case needs that $(X_{a^i+m^i+1}^i \,\|\, Y_{a^i+m^i+1}^i)$ matches the permutation input of any other encryption-query block or primitive query. The probability can be upper bounded by

$$\Pr\left[\left(X_j^i \,\|\, Y_j^i\right) = \left(X_{j'}^{i'} \,\|\, Y_{j'}^{i'}\right)\right] \leq \frac{\sigma_e}{2^{c+s}} + \frac{q_p}{2^{c+s}}\,.$$

Case (E). Assume that $(X_{p+1}^i \,\|\, Y_{p+1}^i)$ hits a w-chain of primitive queries. Under the assumption that no other bad events occurred, the probability is at most

$$\Pr\left[\left(X_{p+1}^i \,\|\, Y_{p+1}^i\right) \,\middle|\, \bigwedge_{i=1}^{8} \overline{\mathsf{bad}_i}\right] \leq \frac{(m^i+1)\cdot w}{2^{c+s}}\,.$$

Over all decryption queries, we obtain

$$\epsilon \leq \sum_{i=1}^{q_d}\left(\frac{1}{2^\tau} + \frac{\sigma_e}{2^{c+s}} + \frac{q_p}{2^{c+s}} + \frac{(m^i+1)\cdot w}{2^{c+s}}\right) \leq \frac{q_d}{2^\tau} + \frac{q_d(q_p+\sigma_e)}{2^{c+s}} + \frac{(\sigma_d+q_d)\cdot w}{2^{c+s}}\,.$$

Our claim in Lemma 3.4 follows.

3.6 INT-RUP Analysis

We use the same notations as in Section 3.5, but add some. Let q_d and σ_d be the number of decryption queries and blocks over decryption queries, respectively, and q_v and σ_v the analogs for verification queries. We replace $\pi \xleftarrow{\$} \mathsf{Perm}(\mathcal{B})$, assume $K \xleftarrow{\$} \mathcal{K}$, and denote $\Pi[\pi]_K$ for $\mathsf{Oribatida}$ with π and K.

Theorem 3.5 (INT-RUP Security of $\mathsf{Oribatida}$)**.** Let $\mathcal{A}$ be a nonce-respecting adversary w.r.t. $\Pi[\pi]_K$. Then

$$\begin{aligned}\mathbf{Adv}^{\Pi[\pi]_K}_{\text{INT-RUP}}(\mathcal{A}) \leq\ & \frac{\sigma_e^2}{2^n} + \frac{4\sigma_e\sigma_d + 4\sigma q_p + q_c q_p + q_p + r(\sigma_d+q_d) + 3rq_p}{2^{c+s}} \\ & + \frac{q_d^2 + \binom{q_d+q_v}{2}}{2^c} + \frac{\binom{q_e}{r}}{2^{\tau(r-1)}} + \frac{3q_p}{2^k} + \frac{2\binom{q_p}{r}}{2^{r(r-1)}} + \frac{2q_v}{2^\tau}\,.\end{aligned}$$

Proof. The INT-RUP analysis of Oribatida follows a similar strategy as our NAEAD analysis. However, this time, the adversary has access to three oracles for encryption, decryption and verification. Moreover, the encryption and decryption oracles are the same in both the real *and* the ideal world. Both worlds differ only in the verification oracle. To alleviate the task, we replace the oracles for encryption and decryption $\widetilde{\mathcal{E}}[\pi]_K$, $\widetilde{\mathcal{D}}[\pi]_K$ with a pair of **consistent** pseudo-random oracles $\$_{\widetilde{\mathcal{E}}}[\pi]$ and $\$_{\widetilde{\mathcal{D}}}[\pi]$ (we define our intent of consistency for encryption and decryption in a moment). The advantage between both settings can be upper bounded by $\Delta_{\mathcal{A}_1}\left(\widetilde{\mathcal{E}}[\pi]_K, \widetilde{\mathcal{D}}[\pi]_K, \widetilde{\mathcal{V}}[\pi]_K, \pi^{\pm}; \$_{\widetilde{\mathcal{E}}}, \$_{\widetilde{\mathcal{D}}}, \bot, \pi^{\pm}\right)$. Note that the oracles $\$_{\widetilde{\mathcal{E}}}$ and $\$_{\widetilde{\mathcal{D}}}$ differ from the *independent* random oracles in the stronger RUPAE notion. In the RUPAE notion, they sample **independently** from each other without considering common prefixes between queries, which would be impossible to achieve for an online AE scheme. Again, we consider the H-coefficient approach. So, we define several bad events and bad as well good transcripts. If any of the bad events occur, the adversary aborts and is defined to win. Next, we consider the probability of forgeries under those idealised oracles. So, we can exclude the previous bad events and study the probability of forgeries. Finally, we study the ratio of interpolation probabilities for good transcripts.

Sampling Consistently in the Online Phase. This online phase contains much from the offline phase of the NAEAD analysis. We define the ideal encryption oracle as in the NAEAD proof: it samples the responses (C^i, T^i) uniformly at random from all bit strings of expected lengths for encryption queries. The ideal decryption oracle, however, must sample plaintext outputs consistently. For this purpose, the ideal encryption oracle has to sample also the internal chaining values $V^i_j \xleftarrow{\$} \{0,1\}^c$ and $U^i_j \xleftarrow{\$} \{0,1\}^r$ uniformly at random for all construction queries already in the online phase. It stores the values of C^i_j, V^i_j, and U^i_j also internally, but does not release U^i_j and V^i_j in this phase.
On each input (N^i, A^i, C^i, T^i), the ideal decryption oracle looks up the length of the longest common prefix of the query $p \leftarrow \mathsf{LCP}^{N^i,A^i}(C^i, \mathcal{Q})$ with all previous queries $\mathcal{Q}$. For all blocks in the common prefix $1 \le j \le p$, it uses the same outputs M^i_j that have been fixed from previous queries. Since the oracle has sampled V^i_{p+1} for the $(p+1)$-th block, it can deduce all bits not fixed from previous query outputs. Assume, $i \neq i'$, $(N^i, A^i) = (N^{i'}, A^{i'})$, and $p = \mathsf{LCP}^{N^i,A^i}(C^i, C^{i'})$ where $p < m^i, m^{i'}$. Then, $C^i_{p+1} = C^i_{p+1} \oplus \Delta$ is the block directly after the common prefix. Sampling

V_j^i and deriving U_j^i ensures consistent sampling, i.e., $M_{p+1}^i = M_{p+1}^{i'} \oplus \Delta$ for all such queries with non-empty common prefix.
Starting from the $(p+2)$-th block, the ideal decryption oracle samples the responses $M_j^i \xleftarrow{\$} \{0,1\}^r$, $V_j^i \xleftarrow{\$} \{0,1\}^c$, and $U_j^i \xleftarrow{\$} \{0,1\}^r$ uniformly and independently at random from the bit strings of expected lengths, for $p+2 \leq j \leq a^i + m^i$. Note that queries whose ciphertext lengths are not multiples of r bits are answered consistently since the oracle samples V_j^i, and all bits fixed from previous queries are used. For verification queries, the ideal verification oracle always outputs $\perp$. For forward primitive queries Q^i, the ideal oracle forwards $\pi(Q^i)$; for backward primitive queries R^i, it returns $\pi^{-1}(R^i)$.

Offline phase. Here, the ideal oracle releases the internal chaining values (U_j^i, V_j^i), after the considered adversary made all queries, but before outputting the decision bit. The ideal oracle also reveals a random key $K \xleftarrow{\$} \mathcal{K}$ then.

Bad Events. Whenever we consider a non-trivial collision between blocks or chaining values at block indices j, j' of two messages, we assume that at least one of them exceeds the longest common prefix.

- bad_1: Non-trivial collision of permutation inputs in construction queries: $\exists$ $(i,j) \neq (i',j')$ with $i, i' \in [1 \cdots q_c]$, $j \in [1 \cdots m^i]$, and $j' \in [1 \cdots m^{i'}]$ s.t. $(X_j^i \,\|\, Y_j^i) = (X_{j'}^{i'} \,\|\, Y_{j'}^{i'})$.

- bad_2: Non-trivial collision of permutation outputs in construction queries: $\exists$ $(i,j) \neq (i',j')$ with $i, i' \in [1 \cdots q_c]$, $j \in [1 \cdots m^i]$, and $j' \in [1 \cdots m^{i'}]$ s.t. $(U_j^i \,\|\, V_j^i) = (U_{j'}^{i'} \,\|\, V_{j'}^{i'})$.

- bad_3: Multi-collision between w tags. For some $w \geq r$, there exist i_1, i_2, …, i_w with i_1, i_2, …, $i_w \in [1 \cdots q_e]$, s.t. $T^{i_1} = T^{i_2} = \ldots = T^{i_w}$.

- bad_4: Non-trivial collision of permutation inputs between construction and primitive query: $\exists$ (i,j,i') with $i \in [1 \cdots q_c]$, $j \in [1 \cdots m^i]$, and $i' \in [1 \cdots q_p]$ s.t. $(X_j^i \,\|\, Y_j^i) = Q^{i'}$.

- bad_5: Non-trivial collision of permutation outputs between construction and primitive query: $\exists$ $i \in [1 \cdots q_c]$, $j \in [1 \cdots a^i + m^i]$ and $i' \in [1 \cdots q_p]$ s.t. $(U_j^i \,\|\, V_j^i) = R^{i'}$.

- bad_6: Initial-state collision with a primitive query: $\exists$ $i \in [1 \cdots q_c]$ and $i' \in [1 \cdots q_p]$ s.t. $(X_0^i \,\|\, Y_0^i) = Q^{i'}$.

- bad_7: Multi-collision in the rate of w outputs of forward primitive queries: for some $w \geq r$, $\exists\ i_1, i_2, \ldots, i_w \in [1 \cdots q_p]$ s.t. $\mathsf{msb}_r(R^{i_1}) = \cdots = \mathsf{msb}_r(R^{i_w})$.
- bad_8: Multi-collision in the rate of w outputs of backward primitive queries: for some $w \geq r$, $\exists\ i_1, i_2, \ldots, i_w \in [1 \cdots q_p]$ s.t. $\mathsf{msb}_r(Q^{i_1}) = \cdots = \mathsf{msb}_r(Q^{i_w})$.
- bad_9: Forgery in decryption queries if all blocks are old: There exists some $i \in [1 \cdots q_d]$ s.t. for all blocks $0 \leq j \leq a^i + m^i$, there exist indices i', j' with $i' \in [1 \cdots q_c]$, $j' \in [1 \cdots m^{i'}]$ or $i' \in [1 \cdots q_p]$ s.t. $(X_j^i \,\|\, Y_j^i) = (X_{j'}^{i'} \,\|\, Y_{j'}^{i'})$ or $(X_j^i \,\|\, Y_j^i) = Q^{i'}$ and the tag is valid: $\mathsf{msb}_\tau\left(\pi\left(X_{a^i+m^i}^i \,\|\, Y_{a^i+m^i}^i\right)\right) = T^i$.

We denote by Θ_{real} and Θ_{ideal} random variables that represent the distribution of transcripts in the real and the ideal world, respectively. We define BadT to contain exactly the attainable transcripts τ for which at least one of the bad events occurred. All other attainable transcripts are in GoodT. Then $\Pr[\Theta_{\text{ideal}} \in \text{BadT}] \leq \sum_{i=1}^{8} \Pr[\mathsf{bad}_i]$. The probability of bad transcripts in the ideal world is treated in the proof of Lemma 3.6. The ratio of obtaining a good transcript is bounded in Lemma 3.7. Our bound in Theorem 3.2 follows from those and the fundamental Lemma of the H-coefficient Technique [54]. We apply $w = r$ in the bound of Lemma 3.6.

Lemma 3.6. Let $w \geq r$ be a positive integer. It holds that

$$\Pr[\Theta_{\text{ideal}} \in \text{BadT}] \leq \frac{\sigma_e^2}{2^n} + \frac{3\sigma_e\sigma_d + 3\sigma q_p + q_c q_p + q_p + w(\sigma_d + q_d) + 3wq_p}{2^{c+s}} + \frac{q_d^2 + \binom{q_d+q_v}{2}}{2^c} + \frac{\binom{q_e}{w}}{2^{\tau(w-1)}} + \frac{3q_p}{2^k} + \frac{2\binom{q_p}{w}}{2^r(w-1)} + \frac{q_v}{2^\tau}.$$

Proof. In the following, we upper bound the probabilities of the individual bad events. For most of them, we differentiate between encryption and decryption queries.

bad_1: Collision of two permutation inputs in construction queries.

1. Among encryption queries only: Here, it holds that

$$\Pr\left[\left(X_j^i \,\|\, Y_j^i\right) = \left(X_{j'}^{i'} \,\|\, Y_{j'}^{i'}\right)\right] \leq 2^{-n}.$$

   Since there exist $\binom{\sigma_e}{2}$ block combinations, we obtain $\binom{\sigma_e}{2}/2^n$.

2. Dec-then-Enc: If we consider an encryption query block to collide with a block from a previous decryption query, the probability is at most $2^{-(c+s)}$

since $\mathcal{A}$ can see $r - s$ bits that it can use as the nonce. We have $q_e\sigma_d$ combinations of such blocks. For the remaining $\sigma_e\sigma_d$ blocks, the probability is 2^{-n}.

3. Among decryption queries only: w.l.o.g., we consider the first such collision. If $\mathcal{A}$ modifies the nonce in the later following query, the bound is the same as for encryption-only queries. So, we assume in the remainder of that the later query is a decryption query. Let $j-1$ be the first modified block and assume it is in the message-processing part. If the block indices differ $j \neq j'$, the probability is $2^{-(c+s)}$. Otherwise, assume $j = j'$ and $A^i = A^{i'}$. Then, the permutation output $(U_j^{i'} \parallel V_j^{i'})$ is sampled randomly in the ideal world. If $\mathcal{A}$ leaves $C^i_{j-a^i} = C^{i'}_{j-a^i}$, it automatically holds that

$$X_j^i = C^i_{j-a^i} \oplus \mathsf{lsb}_s\left(V^i_{\rho(i,j-a^i)}\right) = X_j^{i'} = C^i_{j-a^i} \oplus \mathsf{lsb}_s\left(V^i_{\rho(i,j-a^i)}\right) .$$

So, $X_j^i = X_j^{i'}$. With probability 2^{-c}, it also holds for the capacity $V_{j+1}^i = V_{j+1}^{i'}$ and thus $Y_{j+1}^i = Y_{j+1}^{i'}$. Note that this approach holds only for the first differing block, for which which yields a term of $\binom{q_d}{2}/2^c$. If the collision does not hold, the masks beginning for the $(j+2)$-th block will differ and the probability decreases to $2^{-(c+s)}$, which produces a term of $\binom{\sigma_d}{2}/2^{c+s}$.

4. Enc-then-Dec: It remains to consider collisions between an encryption query, followed by a decryption query. If the block indices $j \neq j'$ differ, the probability is again $2^{-(c+s)}$, for at most $\sigma_e \cdot \sigma_d$ combinations. Otherwise, if $j = j'$, $\mathcal{A}$ can apply the strategy above for a collision. Then, the probability is 2^{-c}; though, the q_d queries can collide at most with one encryption query each since we consider the first collision, producing a term of $q_d/2^c$.

Over all cases, we obtain

$$\Pr\left[\mathsf{bad}_1\right] \leq \frac{\binom{\sigma_e}{2}}{2^n} + \max\left(\frac{q_e\sigma_d}{2^{c+s}} + \frac{\sigma_e\sigma_d}{2^{c+s}} + \frac{q_d}{2^c}\right) + \frac{\binom{\sigma_d}{2}}{2^{c+s}} + \frac{\binom{q_d}{2}}{2^c} \leq \frac{\binom{\sigma_e}{2}}{2^n} + \frac{\sigma_e\sigma_d}{2^{c+s}} + \frac{\binom{q_d}{2}}{2^c} .$$

bad_2: Collision of two permutation outputs in encryption construction queries. This case is analogous to bad_1. Over all combinations of indices, it follows that

$$\Pr\left[\mathsf{bad}_2\right] \leq \frac{\binom{\sigma_e}{2}}{2^n} + \frac{\sigma_e\sigma_d}{2^{c+s}} + \frac{\binom{q_d}{2}}{2^c} .$$

bad_3: Multi-collision on w tags from encryption queries. Since the tags are sampled uniformly and independently at random in the ideal world, it holds that

$$\Pr\left[T^{i_1}_{j_1} = T^{i_2}_{j_2} = \ldots = T^{i_w}_{j_w}\right] \leq \frac{\binom{q_e}{w}}{2^{\tau(w-1)}}\,.$$

bad_4: Collision of permutation inputs between a construction and a primitive query. Again, we consider $X^i_j \leftarrow C^i_{j-a^i} \oplus_s \mathsf{lsb}_s(V^i_{\rho(i,j-a^i)})$ and $Y^i_j \leftarrow V^i_j \oplus \mathrm{const}$, where const is a public constant. The values $C^i_{j-a^i}$ and $V^i_{\rho(i,j-a^i)}$ are sampled randomly, the values $(X^i_j \,\|\, Y^i_j)$ take any value with probability at most 2^{-n}.

1. Assume, the primitive query was asked before the construction query. If the construction query was in encryption direction, the collision probability for fixed queries is at most 2^{-n}, for $q_p \cdot q_c$ combinations.

2. Otherwise, if the construction query was a decryption query, $\mathcal{A}$ can see $r-s$ bits. Hence, the probability is at most $2^{-(c+s)}$, for $q_p \cdot q_c$ combinations.

3. The same argument can be applied in the case when the primitive query was asked after a decryption query. Then, the adversary can see $r-s$ unmasked bits of the rate from C^i_j. Again, the probability is at most $2^{-(c+s)}$ and we have $q_p \cdot q_c$ combinations.

4. If the primitive query was asked after an encryption query, then the latter produced a tag. If the primitive query targets any other block, the argument is the same as in Case 3. If the primitive query starts from the tag, the adversary sees $\tau - s$ unmasked bits. Assuming $\overline{\mathsf{bad}}_3$, there are at most w equal tags over all encryption queries. So, the probability for a collision is $2^{-(c+s)}$, for $w \cdot q_p$ combinations.

Over all combinations of indices, it follows that

$$\Pr\left[\mathsf{bad}_4 | \overline{\mathsf{bad}}_3\right] \leq \max\left(\frac{\sigma_e \cdot q_p}{2^n}, \frac{\sigma_e \cdot q_p}{2^{c+s}}\right) + \frac{\sigma_d \cdot q_p}{2^{c+s}} + \frac{w \cdot q_p}{2^{c+s}} \leq \frac{\sigma \cdot q_p}{2^{c+s}} + \frac{w \cdot q_p}{2^{c+s}}\,.$$

bad_5: Collision of permutation outputs between a construction and a primitive query. Again, $U^i_{j+a^i}$ can be derived from $M^i_j \oplus C^i_j \oplus_s \mathsf{lsb}_s(V^i_{\rho(i,j)})$; the values C^i_j and $V^i_{\rho(i,j)}$ are sampled randomly. This case is similar as bad_4. Over all

index combinations

$$\Pr\left[\mathsf{bad}_5|\overline{\mathsf{bad}_3}\right] \leq \max\left(\frac{\sigma_e \cdot q_p}{2^n}, \frac{\sigma_e \cdot q_p}{2^{c+s}}\right) + \frac{\sigma_d \cdot q_p}{2^{c+s}} + \frac{w \cdot q_p}{2^{c+s}} \leq \frac{\sigma \cdot q_p}{2^{c+s}} + \frac{w \cdot q_p}{2^{c+s}}\,.$$

bad_6: Initial-state collision with a primitive query. Here, we distinguish between the cases whether the construction query was asked before or after the primitive query and whether the primitive query was in forward or backward direction.

1. Assume, the primitive query was asked after the construction query. If the primitive query was a forward query, it must hit the correct value of $K \oplus_d d_N$. This probability is at most $q_p/2^k$ to collide when considering encryption construction queries. Considering also decryption queries, a nonce can repeat often; though, the initial state can take three different values for the same nonce, namely if the decryption query changes the length of associated data and message, affecting d_N. Since there exist at most three distinct values for d_N, the probability to collide is at most $3q_p/2^k$.

2. If the primitive query was in backward direction, its response must hit any initial state of a construction query. If the construction query was asked before the primitive, $\mathcal{A}$ sees at best $r - s$ bits of C_1. Then, the probability is at most $q_c \cdot q_p/2^{c+s}$.

3. If the primitive query was asked before the construction query, $\mathcal{A}$ can use the nonce part of the primitive query's result as the nonce. However, a collision must hit the key part, which holds with probability at most $3q_p/2^k$.

4. If the primitive query was in backward direction, $\mathcal{A}$ sees at best $r - s$ bits of C_1. Then, there is at most one starting state, assuming $\overline{\mathsf{bad}_1}$, which yields $q_p/2^{c+s}$.

Over all possible options, we obtain

$$\Pr\left[\mathsf{bad}_6|\overline{\mathsf{bad}_1}\right] \leq \frac{3q_p}{2^k} + \max\left(\frac{q_c \cdot q_p}{2^{c+s}}, \frac{q_p}{2^{c+s}}\right) \leq \frac{3q_p}{2^k} + \frac{q_c \cdot q_p}{2^{c+s}} + \frac{q_p}{2^{c+s}}\,.$$

bad_7: Multi-collision in the rate of w outputs of forward primitive queries. Since π is chosen randomly from the set of all permutations, the outputs are sampled uniformly at random from a set of size at least $2^n - (i-1)$ for

the i-th query. So, the probability for w distinct queries to collide in their rate is upper bounded by

$$\frac{2^c-1}{2^n-1}\cdot\frac{2^c-2}{2^n-2}\cdot\dots\cdot\frac{2^c-(w-1)}{2^n-(w-1)}=\prod_{i=1}^{w-1}\frac{2^c-i}{2^n-i}\leq\left(\frac{2^c}{2^n}\right)^{w-1}=2^{-r(w-1)}\,.$$

Over all primitive query indices, it holds that

$$\Pr\left[\mathsf{bad}_7\right]\leq\frac{\binom{q_p}{w}}{2^{r(w-1)}}\,.$$

bad_8: Multi-collision in the rate of w outputs of backward primitive queries. Using a similar argumentation as for bad_7, we obtain

$$\Pr\left[\mathsf{bad}_8\right]\leq\frac{\binom{q_p}{w}}{2^{r(w-1)}}\,.$$

bad_9: Forgeries if all blocks are old. It remains to bound the probability of a successful forgery of a verification query (N^i, A^i, C^i, T^i) s. t. T^i is valid and where each block is old.
We consider the same five mutually exclusive cases as in the NAEAD proof. In all cases, the tag can simply be guessed correctly if the block $(X^{a^i+m^i} \,\|\, Y^{a^i+m^i})$ is fresh. Then, the probability for the tag to be correct is upper bounded by $2^{-\tau}$. We adopt the cases and the notions from the NAEAD proof and assume that no previous bad events occur, in particular no w-multi-collisions described earlier or collisions with primitive queries.

- **Case (A):** N^i is fresh; so, there is no earlier construction query $i' \neq i$ s. t. $N^i = N^{i'}$.
- **Case (B):** N^i is old, but (N^i, A^i) is fresh, i.e., there exists no earlier construction query $i' \neq i$ with $(N^i, A^i) = (N^{i'}, A^{i'})$.
- **Case (C):** (N^i, A^i) is old, but (N^i, A^i, C^i) is fresh, i.e., there exists no earlier construction query $i' \neq i$ with $(N^i, A^i, C^i) = (N^{i'}, A^{i'}, C^{i'})$, and no w-chain of primitive queries is hit.
- **Case (D):** (N^i, A^i, C^i) is old; (N^i, A^i, C^i) is a prefix of another construction query.
- **Case (E):** (N^i, A^i) is old and there exists a w-chain of primitive queries that is hit.

Clearly, the cases cover all possible options. We assume that no previous bad events occur, in particular no w-multi-collisions described earlier or collisions with primitive queries.

Case (A). We excluded bad_4, i.e., collisions of permutation inputs between construction and primitive queries in this case. The probability that $(N^i \,\|\, K) \oplus_d d_N$ hits any block $(X_j^{i'} \,\|\, Y_j^{i'})$ from another construction query so that the final block is old is at most

$$\Pr\left[\left((N^i \,\|\, K) \oplus_d d_N\right) = \left(X_j^{i'} \,\|\, Y_j^{i'}\right)\right] \leq \frac{\sigma + q_p}{2^{c+s}}\,.$$

Cases (B)–(D). Let $p \leq a^i + m^i$ denote the length of the longest common prefix of the i-th query with all other queries. The probability that any block $(X_j^i \,\|\, Y_j^i)$ with $j \geq p+1$ matches the permutation input of any other query block or primitive query can be upper bounded analogously as bad_1 and bad_4:

$$\Pr\left[\left(X_j^i \,\|\, Y_j^i\right) = \left(X_{j'}^{i'} \,\|\, Y_{j'}^{i'}\right)\right] \leq \frac{\sigma_e + q_d}{2^{c+s}} + \frac{\sigma_e \cdot \sigma_d}{2^{c+s}} + \frac{\binom{q_d}{2}}{2^c} + \frac{\sigma \cdot q_p}{2^{c+s}} + \frac{w \cdot q_p}{2^{c+s}}\,.$$

Over all verification queries, we obtain

$$\Pr\left[\mathsf{bad}_9 \,\middle|\, \bigwedge_{i=1}^{8} \overline{\mathsf{bad}_i}\right] \leq \frac{q_v}{2^\tau} + \frac{\sigma_e \cdot \sigma_d}{2^{c+s}} + \frac{\binom{q_d+q_v}{2}}{2^c} + \frac{\sigma \cdot q_p}{2^{c+s}} + \frac{w \cdot q_p}{2^{c+s}} + \frac{w \cdot (\sigma_d + q_d)}{2^{c+s}}\,.$$

Case (E). Assume that $(X_{p+1}^i \,\|\, Y_{p+1}^i)$ hits a w-chain of primitive queries. Under the assumption that no other bad events occurred, the probability is at most

$$\Pr\left[\left(X_{p+1}^i \,\|\, Y_{p+1}^i\right)\right] \leq \frac{(m^i + 1) \cdot w}{2^{c+s}}\,.$$

Over all verification queries, we obtain

$$\Pr\left[\mathsf{bad}_9 \,\middle|\, \bigwedge_{i=1}^{8} \overline{\mathsf{bad}_i}\right] \leq \frac{q_v}{2^\tau} + \frac{\sigma_e \cdot \sigma_d}{2^{c+s}} + \frac{\binom{q_d+q_v}{2}}{2^c} + \frac{\sigma \cdot q_p}{2^{c+s}} + \frac{w \cdot q_p}{2^{n-\tau}} + \frac{w \cdot (\sigma_d + q_d)}{2^{c+s}}\,.$$

Our bound in Lemma 3.6 follows from summing up all probabilities.

Lemma 3.7. Let $\tau \in \textsc{GoodT}$. Then

$$\frac{\Pr[\Theta_{\text{real}} = \tau]}{\Pr[\Theta_{\text{ideal}} = \tau]} \leq 1 - \left(\frac{q_d}{2^\tau} + \frac{\sigma_d(\sigma_e + q_p)}{2^{c+s}}\right)\,.$$

Proof. It remains to bound the ratio of the probabilities for obtaining a good transcript τ in the real and the ideal world, respectively. The bound is similar to that of Lemma 3.4. The difference to the NAEAD proof is that the ideal decryption oracle also generates pseudorandom output blocks M_j^i beyond the longest common prefix. The NAEAD transcript also contained the sampled internal values, as does the transcript τ here. Since we assume that no bad events have occurred, we revisit the following cases for forgeries:

- **Case (A):** The final input to π, $(X^i_{a^i+m^i} \,\|\, Y^i_{a^i+m^i})$ is fresh, i.e., has not occurred before. Then, the probability that the authentication tag τ^i is valid is at most $1/2^\tau$.
- **Case (B):** The final input to π, $(X^i_{a^i+m^i} \,\|\, Y^i_{a^i+m^i})$ is old, but there exists some block index $j \in [1 \cdots a^i + m^i]$ s. t. $(X^i_j \,\|\, Y^i_j)$ is fresh. Since the input is old, the probability that the result of any of the next blocks is old is at most $\frac{(\sigma+q_p)}{2^{c+s}}$.
- **Case (C):** There exists no $j \in [1 \cdots a^i + m^i]$ s. t. $(X^i_j \,\|\, Y^i_j)$ is fresh. The probability that all of those blocks are old is at most $\frac{m^i(\sigma_e+q_p)}{2^{c+s}}$.

It follows that

$$\epsilon_i \leq \frac{1}{2^\tau} + \frac{\sigma_e + q_p}{2^{c+s}} + \frac{m^i(\sigma_e + q_p)}{2^{c+s}} \,.$$

Over all indices $i \in [1 \cdots q_d]$, it follows that

$$\epsilon \leq \sum_{i=1}^{q_d} \epsilon_i \leq \frac{q_d}{2^\tau} + \frac{\sigma_d(\sigma_e + q_p)}{2^{c+s}} \,.$$

Our claim in Lemma 3.7 follows. □

3.7 Comparison with Lightweight INT-RUP-secure Schemes

Among the submissions to the NIST lightweight cryptography project, ESTATE [55], LAEM [56], LOTUS-AEAD and LOCUS-AEAD [57] claimed security in the INT-RUP model. Among these modes, ESTATE, LOTUS-AEAD, and LOCUS-AEAD were elected into the second round. This section compares our proposal to those; Table 3.4 gives a summary.

TABLE 3.4: Comparison of Oribatida with other INT-RUP-security claiming submissions to the NIST lightweight cryptography project. n/t = block/tweak length of the primitive, m = #message segments, sec. = security, IF = inverse-free, •/– = feature is present/absent

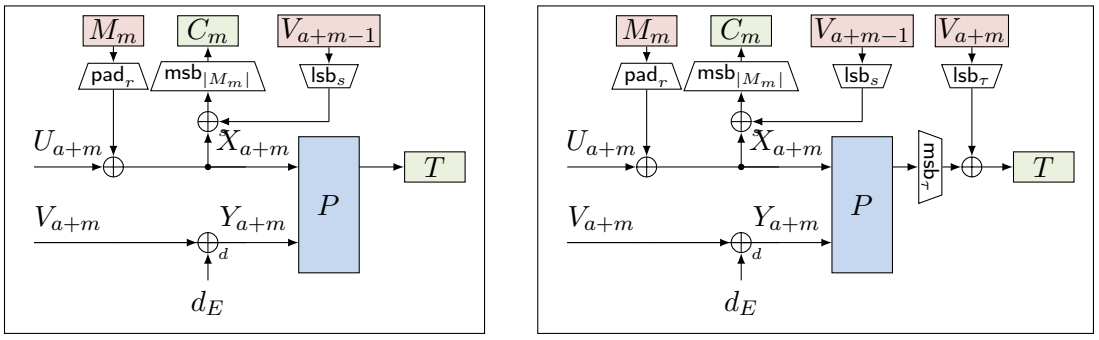
	Sizes (bits)							Security		Features	
Construction	$\|N\|$	$\|K\|$	$\|T\|$	n	t	State	Rate	NAEAD	INT-RUP	1-pass	IF
Oribatida-192 (v1.2) [50]	64	128	96	192	0	288	96	89	48	•	•
Oribatida-256 (v1.2) [50]	128	128	128	256	0	320	128	121	64	•	•
Oribatida v1.3-192 **[This work]**	64	128	96	192	0	288	96	121	48	•	•
Oribatida v1.3-256 **[This work]**	128	128	128	256	0	320	128	121	64	•	•
ESTATE [55]	128	128	128	128	4	260	64	64	64	–	•
LOCUS-AEAD [57]	128	128	64	64	4	324	32	64	64	•	–
LOTUS-AEAD [57]	128	128	64	64	4	384	32	64	64	•	•

3.7.1 Brief Description

ESTATE follows SIV [58]: the associated data and message are authenticated using a variant of CBC-MAC with a tweakable block-cipher before the tag is used as an initial vector of CBC-like encryption. The intermediate values are used as keystream and added to the message blocks. LOCUS-AEAD and LOTUS-AEAD employ a variant of PMAC [59] to process the associated data with the tweakable block-cipher. For encryption, LOTUS-AEAD uses a variant of OTR [60], a two-round, two-branch Feistel structure to process the message in double blocks. LOCUS-AEAD employs an encryption similar to OCB [61] and EME/EME* [62]. Both LOCUS-AEAD and LOTUS-AEAD employ a single pass over the message for encryption, but two calls to the primitive per message block. The intermediate values are summed to the associated data hash and the final message block; the encrypted sum yields the tag.

3.7.2 Efficiency

Oribatida processes 96- or 128-bit message blocks per primitive call, whereas the size of the message processed in one primitive call is 64 bits for ESTATE and 32 for LOTUS-AEAD and LOCUS-AEAD. Thus, Oribatida offers higher throughput; moreover, the state size of Oribatida (288 and 320 bits, respectively) is smaller than those of LOTUS-AEAD (388 bits) and LOCUS-AEAD (324 bits). ESTATE has a state size of 260 bits; all three must process the message with two calls to the primitive. LOCUS-AEAD requires the inverse operation of the underlying block-cipher to be available for the decryption. In sum, Oribatida possesses a smaller



(A) Oribatida v1.2 [50].

(B) Oribatida v1.3.

FIGURE 3.4: Tag generation of Oribatida

state size than LOCUS-AEAD and LOTUS-AEAD, and higher NAEAD security, as well as a higher rate, compared to its INT-RUP-secure competitors.

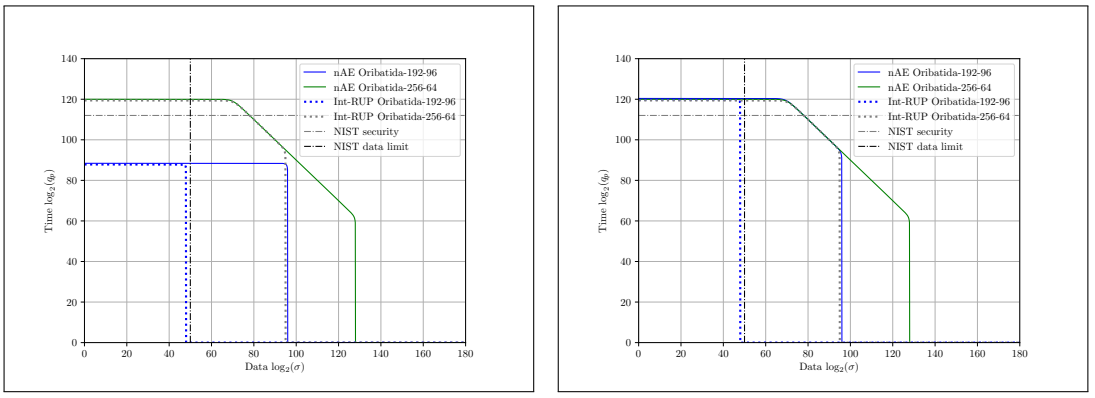
3.7.3 Security

All three competitors are based on tweakable block-ciphers, with INT-RUP claims limited by the birthday bound of the internal primitive. ESTATE inherits INT-RUP security until the birthday bound from SIV, which has been considered in [20, Section 6.2]. While LOCUS-AEAD and LOTUS-AEAD share similarities to OCB and OTR, they use intermediate checksums as in EME designs in the tag-generation process. Informally, modifying any message block will result in new pseudorandom internal values and therefore a pseudorandom input to the tag computation.

3.8 Discussion of the Updated Variant Oribatida v1.3

This section discusses the update from Oribatida (v1.2) from [50] to Oribatida v1.3 in this chapter, that addresses the observation by Rohit and Sarkar [49] in a straight-forward manner. Here, we briefly discuss only the differences:

1. Oribatida v1.2 released the tag without masking. As a consequence, the adversary has seen the full rate and had to guess only the $n-\tau$-bit hidden part to be able to invert the encryption process. To avoid this attack, Oribatida v1.3 masks the tag such that the adversary sees $\tau - s$ bits if $s \leq \tau$, which restores the complexity from $q/2^{n-\tau}$ to $q/2^{c+s}$. Figure 3.4 illustrates both



(A) Oribatida v1.2 [50] (B) Oribatida v1.3

FIGURE 3.5: Security of Oribatida using $q_c = 2^{50}$

tag-generation processes for comparison. The masking of the authentication tag is performed exactly as for ciphertext blocks, which streamlines this process.

2. Oribatida v1.2 employed two permutations P and P', where the latter was intended to be a more efficient variant of the former. In practice, P' was instantiated with a round-reduced version of P, which was only used for processing intermediate blocks of associated data. This was fine since an upper bound on the probability of differentials was sufficient for security and not pseudo-randomness. Oribatida v1.3 unified the process and uses P at every location.

3. Oribatida v1.2 used a different starting value V_0 for masking the ciphertexts when the associated data was empty and V_1 otherwise. The reason was simply efficiency since empty associated data did not yield a value V_1. In contrast, Oribatida v1.3 always pads the associated data such that there always exists an intermediate value V_1 that is not used as capacity in the message-processing step. This decision implies a slightly lower throughput for empty associated data but adds unification.

4. As a result of Aspect (3), Oribatida v1.3 uses slightly different and more domains to properly address also the additional case when the associated data was empty.

Among all changes, only Aspect (1) is crucial. All further aspects helped unify the design. The security effect of the additional tag masking is illustrated in Figure 3.5 for the maximum number of $q_c = 2^{50}$ construction queries as in the

NIST guidelines. One can observe that it salvages the NAEAD security of the 192-bit version of Oribatida v1.3. Note that the figure cannot illustrate that many primitive (offline) queries to the permutation are in practice much easier to obtain than construction queries.

3.9 Instantiation of Oribatida

This section specifies the permutation SimP. From a high-level view, SimP is a variant of the domain extender Ψ_r by Coron et al. [63]. We define SimP to use a round-reduced variant of the Simon [64] block-cipher and its key schedule through four such steps. We briefly recall Ψ_r before we describe the details of Simon, provide an overview of existing cryptanalysis, and close with a discussion of the implications on SimP.

3.9.1 The Ψ_r Domain Extender

The Ψ_r family is a two-branch Feistel-like network that consists of r calls to (pairwise independent) block-ciphers. An illustration of Ψ_4 is given at the top of Figure 3.6. Let $\mathsf{BlockCipher}(\mathcal{K}, \mathcal{B})$ denote the set of all block-ciphers over $\mathcal{B}$ with key space $\mathcal{K}$. For Ψ_r, $\pi_1, \pi_2, \ldots, \pi_r \in \mathsf{BlockCipher}(\mathbb{F}_2^n, \mathbb{F}_2^n)$ are pairwise independent block-ciphers which use one branch R^i as state input, and the other one, L^i, as secret key. Coron et al. provide statements on the indifferentiability of their constructions.

Theorem 3.8 ([63])**.** Let $\pi_1, \pi_2, \pi_3 \xleftarrow{\$} \mathsf{BlockCipher}(\mathbb{F}_2^n, \mathbb{F}_2^n)$ be pairwise independent block-ciphers. The three-step construction Ψ_3 with an ideal block-cipher is (t_D, t_S, q, ϵ)-indifferentiable from an ideal cipher with $t_S = O(qn)$ and $\epsilon = 5q^2/2^n$.

Intuitively, it follows that a four-step construction with a fourth independent block-cipher $\pi_4 \xleftarrow{\$} \mathsf{BlockCipher}(\mathbb{F}_2^n, \mathbb{F}_2^n)$ inherits at least the security of the three-step construction.

3.9.2 Φ_r: A Variant of Ψ_r That Includes The Key Schedule

The Ψ_r construction has to store the state that is transformed through the block-cipher π_i's state transformation, plus the key of the current step. Internally, the block-ciphers π_i also have to expand the secret key to subkeys that add to the total memory requirement. We propose a variant that avoids the need to store the current secret key input. For this purpose, we define the key-schedule permutation

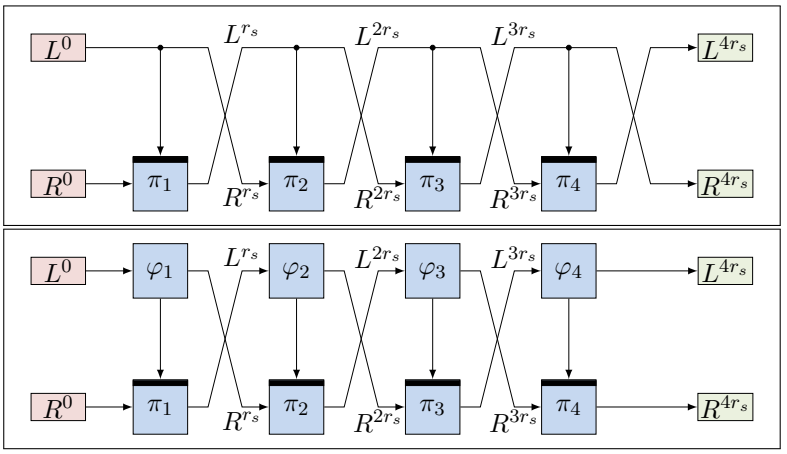


FIGURE 3.6: **Top:** The construction Ψ_4 [63]. The blocks π_i denote block-ciphers over $\mathbb{F}_2^n$ with key space $\mathbb{F}_2^n$. **Bottom:** High-level view of the construction Φ_4 as a variant of Ψ_4. The blocks φ_i represent the key schedules that produce the subkeys and which are externalised from the block-ciphers π_i in Φ_4. φ_i feeds the subkeys to π_i and outputs the final subkey K^{r_s} to become the next value R^{ir_s}.

$\varphi_i : \mathbb{F}_2^n \to \mathbb{F}_2^n$ that takes an initial key K as input and outputs the subkeys $K^0, \ldots, K^{r_s}$ for fixed number of rounds r_s of π_i. An illustration is given at the bottom of Figure 3.6. Hereafter, we call the construction Φ_r when it consists of r steps in total. Note that Φ_r omits the final swap of the halves for simplicity and since it does not affect the security.

3.9.3 Simon

The Simon family of block-ciphers [64] belongs to the lightest block-ciphers in terms of hardware area and energy efficiency. Its round function consists of only an XOR, three bit-wise rotations, and a bit-wise AND, which renders it particularly lightweight and flexible. Moreover, Simon has been analysed intensively since its proposal; among others, e.g., [65–69] studied the security of Simon-96-96 and Simon-128-128. Considerably, more works targeted the smaller-state variants of Simon, which has recently been standardised as part of ISO/IEC 29167-21:2018 [70]. For concreteness, Simon-96-96 uses a word size $w = 48$ bits and employs 52 rounds, whereas Simon-128-128 uses $w = 64$ bits and 68 rounds.

3.9.4 The SimP-n-θ Family of Permutations

SimP is an instantiation of Φ_4 that tries to adhere to the standard as close as possible, SimP-192 employs the round-reduced Simon-96-96 as π and its key schedule as φ. To form a 256-bit permutation, SimP-256 uses Simon-128-128 with its key

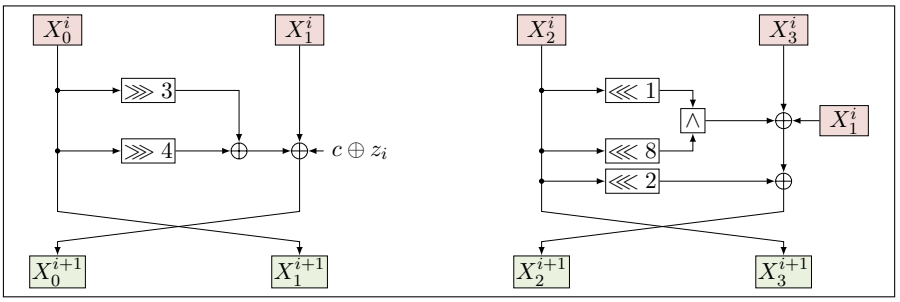


FIGURE 3.7: One iteration of the round function of SimP, which is equivalent to the key-update function (left) and the state-update function (right) of Simon-$2w/2w$, where w is the word size

schedule. One iteration of the round function of Simon-$2w$-$2w$ and its key-update function side by side, as is used in SimP-n, is illustrated in Figure 3.7. Internally, the state of SimP-n-θ consists of four w-bit words $(X_0^i, X_1^i, X_2^i, X_3^i)$, where the superscript index i indicates the state after Round i. We denote by r_s the number of rounds per step, and index the steps from 1 to θ, and the rounds from 1 to $\theta \cdot r_s$. The plaintext is denoted as $(X_0^0, X_1^0, X_2^0, X_3^0)$; the ciphertext is given as $(X_0^{\theta r_s}, X_1^{\theta r_s}, X_2^{\theta r_s}, X_3^{\theta r_s})$.
After Round r_s, the state halves $(X_0^{r_s}, X_1^{r_s})$ and $(X_2^{r_s}, X_3^{r_s})$ are swapped; similarly, they are swapped also after Round $2r_s, \ldots, \theta r_s$. One round of the permutation is illustrated in Figure 3.7. Thus, SimP-192-θ uses Simon-96-96 and consists of four 48-bit words. SimP-256-θ employs the round function and the key-update function of Simon-128-128 as a 256-bit permutation. For SimP-256-θ, the state consists of four 64-bit words.

3.9.4.1 Round Function

Let w be a positive integer for the word size. for SimP-192, $w = 48$ bits; for SimP-256, $w = 64$ bits. Let $f : \mathbb{F}_{2^w} \to \mathbb{F}_{2^w}$ and $g : \mathbb{F}_{2^w} \to \mathbb{F}_{2^w}$ be defined as

$$f(x) \stackrel{\text{def}}{=} (x \lll 8) \wedge ((x \lll 1) \oplus (x \lll 2)) \text{ and}$$
$$g(x) \stackrel{\text{def}}{=} (x \ggg 3) \oplus (x \ggg 4)\,.$$

3.9.4.2 Key-update Function

Let $\varphi_j : (\mathbb{F}_{2^w})^2 \to (\mathbb{F}_{2^w})^2$, for $1 \le j \le \theta$ be key-update functions. Let $\ell = (j-1) \cdot r_s$. On input (X_0^ℓ, X_1^ℓ), it derives r_s keys $(X_0^{\ell+i}, X_1^{\ell+i})$, for $1 \le i \le r_s$, as

$$X_0^{\ell+i} \leftarrow X_1^{\ell+i-1} \oplus g(X_0^{\ell+i-1}) \oplus c \oplus z_{\ell+i-1} \qquad \text{and} \qquad X_1^{\ell+i} \leftarrow X_0^{\ell+i-1}\,,$$

for $1 \leq i \leq r_s$. Note that $c =$ `0xff...ffc` is a w-bit constant.

3.9.4.3 State-update Function

We define the state-update function as $\pi : (\mathbb{F}_{2^w})^{r_s} \times (\mathbb{F}_{2^w})^2 \to (\mathbb{F}_{2^w})^2$, where the first input considers the expanded subkeys. Let $\ell = (j-1) \cdot r_s$. It takes r_s round keys $(X_0^\ell, \ldots, X_{r_s-1}^\ell)$ as key input, as well as (X_2^ℓ, X_3^ℓ) as state input, and computes $(X_2^{\ell+r_s}, X_3^{\ell+r_s})$ recursively as:

$$X_2^{\ell+i} \leftarrow f(X_2^{\ell+i-1}) \oplus X_3^{\ell+i-1} \oplus X_1^{\ell+i-1} \qquad \text{and} \qquad X_3^{\ell+i} \leftarrow X_2^{\ell+i-1}\,,$$

for $1 \leq i \leq r_s$.

3.9.4.4 Step Function

Let $\rho_j : \mathbb{F}_{2^w}^4 \to \mathbb{F}_{2^w}^4$ denote the step function, for $1 \leq j \leq \theta$. Define $L^i = (X_0^i, X_1^i)$ and $R^i = (X_2^i, X_3^i)$. The step transforms $(L^i, R^i) = (X_0^i, X_1^i, X_2^i, X_3^i)$ into $(X_0^{i+r_s}, X_1^{i+r_s}, X_2^{i+r_s}, X_3^{i+r_s})$ as

$$\begin{aligned}(L^{r_s}, R^{r_s}) &= (X_0^{i+r_s}, X_1^{i+r_s}, X_2^{i+r_s}, X_3^{i+r_s}) \\ \rho_j(X_0^i, X_1^i, X_2^i, X_3^i) &\stackrel{\text{def}}{=} (\pi_j(X_2^i, X_3^i), \varphi_j(X_0^i, X_1^i)),\end{aligned}$$

for $1 \leq j < \theta$. One exception is the final step ρ_θ, which omits the final swap of the halves:

$$\rho_\theta(X_0^i, X_1^i, X_2^i, X_3^i) \stackrel{\text{def}}{=} (\varphi_\theta(X_0^i, X_1^i), \pi_\theta(X_2^i, X_3^i)).$$

SimP-n-θ takes a plaintext $(X_0^0, X_1^0, X_2^0, X_3^0)$ as input and outputs $(L^r, R^r) = (X_0^r, X_1^r, X_2^r, X_3^r)$, with $r = \theta r_s$ as ciphertext.

3.9.4.5 Round Constants

The round constants are those of Simon-96-96 and Simon-128-128 [64], respectively. It holds that $c =$ `0xff...ffc`, i.e., all w bits except for the least significant two bits are 1. More precisely, for $w = 48$, it holds that

$$c = (1111\,1111\,1111\,1111\,1111\,1111\,1111\,1111\,1111\,1111\,1111\,1100)_2.$$

TABLE 3.5: Parameters of SimP

Variant	Word size (w)	#Steps (θ)	#Rounds/Step (r_s)
SimP-192-4	48	4	26
SimP-256-4	64	4	34

For $w = 64$, it holds that $c =$

$(1111\,1111\,1111\,1111\,1111\,1111\,1111\,1111\,1111\,1111\,1111\,1111\,1111\,1111\,1111\,1100)_2$.

For both SimP-192 and SimP-256, the constants $z = z_0 z_1 \dots z_{61}$ are defined as

$z = (10\,1011\,1101\,1100\,0000\,1101\,0010\,0110\,0010\,1000\,0100\,0111\,1110\,0101\,1011\,0011)_2$.

The sequence has a period of 62, so $z_i = z_{i \bmod 62}$, for non-negative integers i. Note that the order of the bits z_i is reversed.

3.9.4.6 Number of Steps θ

We consider only the choice of $\theta = 4$ everywhere in our proposed construction.

3.9.4.7 Number of Rounds

SimP-192-4 consists of $r_s = 26$ rounds for each step, and therefore performs $r = 4 \cdot r_s = 104$ rounds in total. SimP-256-4 consists of $r_s = 34$ rounds for each block, and therefore performs $r = 4 \cdot r_s = 136$ rounds in total. For simplicity, we also denote SimP-n-4 as SimP-n. The algorithm of SimP-n-θ is given in Algorithm 3 and the parameters of SimP-n-θ are given in Table 3.5.

3.9.4.8 The Byte Order in Oribatida

For the sake of clarity, Figure 3.8 visualises the byte and word order of the inputs. Let `SB` denote the state S in bytes; for more clarity, we further write this ordering in type-writer font. The rate consists of the first $r/8$ bytes of the state: `SB[0], ..., SB[r/8 - 1]`. The capacity represents the last $c/8$ bytes `SB[r/8], ..., SB[n/8 - 1]`. Similarly, the rate of the state consists of the first words of the permutation input. If the state is interpreted as an n-bit value, the initial Byte 0 contains the most significant eight bits: `SB[0]` $= (S[n-1], S[n-$

Algorithm 3 Specification of SimP-n-θ

101: **function** SimP-n-$\theta(M)$
102: $X_0^0 X_1^0 X_2^0 X_3^0 \leftarrow_w M$
103: **for** $i \leftarrow 1 \cdots \theta$ **do**
104: **for** $j \leftarrow 1 \cdots r_s$ **do**
105: $\ell \leftarrow (i-1) \cdot r_s + j$
106: $X_0^\ell \leftarrow X_1^{\ell-1} \oplus g(X_0^{\ell-1}) \oplus c \oplus z_{\ell-1}$
107: $X_1^\ell \leftarrow X_0^{\ell-1}$
108: $X_2^\ell \leftarrow X_3^{\ell-1} \oplus f(X_2^{\ell-1}) \oplus X_1^{\ell-1}$
109: $X_3^\ell \leftarrow X_2^{\ell-1}$
110: **if** $i \neq \theta$ **then**
111: $(X_0^\ell, X_1^\ell, X_2^\ell, X_3^\ell) \leftarrow \mathsf{swap}(X_0^\ell, X_1^\ell, X_2^\ell, X_3^\ell)$
112: $C \leftarrow (X_0^{\theta r_s} \| X_1^{\theta r_s} \| X_2^{\theta r_s} \| X_3^{\theta r_s})$
113: **return** C

121: **function** $f(X)$
122: **return** $((X \lll 1) \wedge (X \lll 8)) \oplus (X \lll 2)$

131: **function** $g(X)$
132: **return** $(X \ggg 3) \oplus (X \ggg 4)$

141: **function** swap(X_0, X_1, X_2, X_3)
142: **return** (X_2, X_3, X_0, X_1)

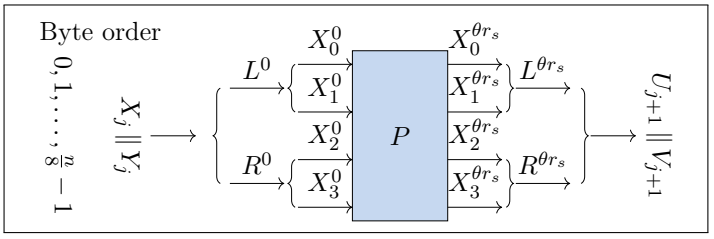


FIGURE 3.8: Byte and word orientation of inputs into and outputs from SimP as used in Oribatida

2], ..., $S[n-8]$). On the other side, the least significant eight bits are stored in Byte `SB[n/8 - 1]`: `SB[n/8 - 1]` = $(S[7], S[6], \ldots, S[0])$.
So, the rate is used first as input to the key-update function; the capacity is used as input to the state-update function.

Remark 3.9. Instantiating a scheme proven in an idealised model such as indifferentiability with a symmetric-key primitive is almost always a heuristic: there simply exist few provably secure instantiations. Using the full Simon-$2w$-$2w$ for each step would be an option for a more secure, but considerably less performant scheme. Concerning SimP, our approach follows the prove-then-prune strategy from AEZ [71]. However, after replacing each step by at least half of the number of rounds, and always using four steps, our approach is far less aggressive than it, as outlined above, and seems to provide a sufficient security margin.

3.10 Security of SimP

The number of steps and rounds of SimP was chosen to resist known cryptanalysis techniques. This section provides a rationale for our choices from the existing

works.

3.10.1 Requirements

Oribatida with a random permutation aims at NAEAD security of $O(r\sigma_d/2^{c+s})$ and INT-RUP security of $O(q_d^2/2^c)$ in the ideal-permutation model. The advantage of those bounds should be much higher than the complexity to recover or predict the key. An instantiation of P must be free of distinguishing properties that allow us to distinguish it from a random permutation with non-negligible advantage and $<< 2^n$ queries. This strengthens the adversary compared to the use of P in Oribatida. There, it can inject nonce, associated data, or message blocks only into the rate and can observe ciphertext and tag outputs also only from that part, but masked. Concretely, we require from P the absence of (truncated, higher-order) differential characteristics with probability $\geq 2^{-n}$, linear approximations with squared correlation $\geq 2^{-n}$, or component functions of degree $< n$ in SimP-4. Moreover, we require the absence of impossible-differential, zero-correlation, or integral distinguishers in SimP-4. However, we disregard rebound or other forms of inside-out attacks that are inapplicable in Oribatida, or splice-and-cut attacks when using SimP as a compression function.

3.10.2 Existing Cryptanalysis on Simon

Various works analysed the Simon family of block-ciphers since its proposal.

3.10.2.1 Differential Cryptanalysis

Cryptanalysis that appeared early after the proposal of Simon followed mainly heuristics for differential cryptanalysis: Abed et al. [65] followed a heuristic branch-and-bound approach that yielded differentials for up to 30 rounds of Simon-96. Biryukov et al. [72] studied more efficient heuristics, but considered the small variants with state sizes up to 64 bits. Dinur et al. [73] showed that distinguishers on Simon with k key words can be extended by at least k rounds. Interestingly, boomerangs seemed to be less a threat to Simon-like ciphers than pure differentials.

Kölbl et al. [74] redirected the research focus to the search for optimal characteristics. More recently, Liu et al. [67] employed a variant of Matsui's algorithm [75] to find optimal differential characteristics. They found that characteristics with probability higher than 2^{-96} covered at most 27 rounds. Moreover, they found

at best 31-round differentials with an accumulated probability higher than 2^{-96}, i.e., of probability $2^{-95.34}$. For Simon-128, they showed that optimal differential characteristics covered at most 37 rounds and found 41-round differentials with a cumulative probability of $2^{-123.74}$.

3.10.2.2 Linear Cryptanalysis

Linear cryptanalysis is similarly effective for Simon-like ciphers as its differential counterpart. Alizadeh et al. [76, 77] reported multi-trail linear distinguishers on all variants of Simon. For Simon-96-96, they proposed a distinguisher on up to 31 rounds that could be extended by two rounds in a key-recovery attack. Similarly, they reported a 37-round distinguisher for Simon-128-128 that could be extendable by two rounds. Chen and Wang [66] proposed improved key-recovery attacks with the help of dynamic key guessing. To the best of our knowledge, their attacks are the most effective ones for our considered variants in terms of the number of covered rounds, with up to 37 rounds of Simon-96-96 and up to 49 rounds of Simon-128-128 in theory.
Similar as for differentials, Liu et al. studied also optimal linear approximations [78]. They found that the optimal linear approximations can reach at most 28 rounds for Simon-96, and at most 37 rounds for Simon-128. Moreover, they determined linear hulls with potential of $2^{-93.8}$ for 31 rounds of Simon-96, and $2^{-123.15}$ for 41 rounds of Simon-128.

3.10.2.3 Integral, Impossible-differential, and Zero-correlation Distinguishers

Integral attacks cover at most 22 rounds for Simon-96-96 and 26 rounds of Simon-128-128. Initially, Zhang et al. [80] found integral distinguishers on up to 21 and 25 rounds for Simon-96 and Simon-128. Their results were extended by one round each by Xiang et al. [69], and later by Todo and Morii [81]. The latter could show the absence of integrals for 25-round Simon-96, which was confirmed by Kondo et al. [82].
The maximal number of rounds that impossible-differential and zero-correlation distinguishers can cover is given by at most twice the length of the maximal diffusion. From the results by Kölbl et al. [74], full diffusion is achieved by 11 rounds for Simon-96 and 13 rounds for Simon-128-128. So, impossible-differential and zero-correlation distinguishers can cover at most 22 and 26 rounds in the single-key setting.

TABLE 3.6: Existing results of best distinguishers and best key-recovery attacks on Simon-96 in the single-key setting. – = not given; Pr. = probability; Pot. = linear potential; Deg. = degree

Type	#Rounds	Time	Data	Pr./Pot./Deg.	Ref
Simon-96-96 Distinguishers					
Algebraic	14	–	20 CPs	–	[68]
Integral	22	2^{95}	2^{95} CP	95	[69]
Differential	30	–	–	92.2	[65]
Differential	31	–	–	95.34	[67]
Linear	31	–	–	93.8	[78]
Simon-96-96 Key-recovery Attacks					
Multiple Linear	33	$2^{94.42}$	$2^{94.42}$ KP	94.42	[79]
Linear Hull	37	$2^{88.0}$	$2^{95.2}$ KP	95.2	[66]
Simon-128-128 Distinguishers					
Algebraic	16	–	20 CP	–	[68]
Integral	26	2^{126}	2^{126} CP	126	[69]
Linear	37	–	–	128	[79]
Differential	41	–	–	123.74	[67]
Linear Hull	41	–	–	123.15	[78]
Simon-128-128 Key-recovery Attack					
Linear Hull	49	$2^{127.6}$	$2^{127.6}$ CP	126.6	[66]

3.10.2.4 Related-key Distinguishers

Kondo et al. [82] searched for iterative key differences in Simon. This allowed them to extend previous results by four to 15 rounds. For Simon-96-96, the authors found iterative key differentials for up to 20 rounds. It remains unclear if this yields an impossible differential; in the best case, a key-iterated 20-round distinguisher could be extended by $2 + 2 + 2$ wrapping rounds: two more blank rounds where one key word is not used, plus two rounds where the key difference can be canceled by the state differences, plus two outermost rounds since the result of the non-linear function is independent of the key and therefore predictable in Simon. So, an impossible-differential distinguisher could cover up to 26 rounds. Though, such an upper bound has not been formulated to an attack on the here-considered versions by Kondo et al.; therefore, it is not contained in the overview in Table 3.6.

TABLE 3.7: Probabilities of optimal related-key differential characteristics for round-reduced variants of Simon-96-96 and Simon-128-128. p denotes the probability; SK = single-key model, RK = related-key model

	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
#Rounds	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	
Simon-96-96																		
$-\log_2(p)$ (SK)	4	6	8	12	14	18	20	26	30	36	38	44	48	54	56	62	64	66
	68	72	74	78	80	86	90	96										
$-\log_2(p)$ (RK)	0	2	4	10	12	18	20	26										
Simon-128-128																		
$-\log_2(p)$ (SK)	4	6	8	12	14	18	20	26	30	36	38	44	48	54	56	62	64	66
	68	72	74	78	80	86	90	96	98	104	108	114	116	122	124	126	128	
$-\log_2(p)$ (RK)	0	2	4	10	12	18	20	26										

3.10.2.5 Algebraic Cryptanalysis

Algebraic attacks are unlikely to be a threat to Simon-like constructions for sufficiently many rounds. Raddum [68] pointed out that the large number of rounds is necessary. He demonstrated that the equation systems for up to 14 rounds of Simon-96-96 and up to 16 rounds of Simon-128 can be solved efficiently in a few minutes on an off-the-shelf laptop. Extensions to considerably more rounds are still unknown.

3.10.2.6 Meet-in-the-Middle Attacks

Meet-in-the-middle attacks are successful primarily on primitives that do not use parts of the key in sequences of several rounds. The Simon-$2w$-$2w$ versions use every key bit in each sequence of two subsequent rounds, which limits the chances of meet-in-the-middle attacks drastically. Considering 3-subset meet-in-the-middle attacks, together with an initial structure and partial matching, the length of an attack is limited to roughly that of twice the full diffusion plus four rounds plus the maximal length of an initial structure plus two rounds for a splice-and-cut part, which yields 30 rounds as a rough upper bound. It is unlikely that such attacks cover 30 or more rounds on Simon-$2w$-$2w$.

3.10.2.7 Correlated Sequences

An interesting recent direction may be correlated sequences introduced by Rohit and Gong in [83]. Their technique requires only very few texts and claims to break 27 rounds of Simon-32 and SIMECK-32; thus, it might outperform all previous attacks by at least three rounds. Though, that approach needs further investigation and has seen application only to Simon-32-64 until now.

3.10.3 Implications to SimP

Since the key schedule of Simon is fully linear, the two state words that are transformed by the key schedule allow the prediction of differences, linear and algebraic properties through a full step. In any case, SimP transforms each input word through at least $2r_s$ rounds of Simon.

3.10.3.1 Related-key Differential Cryptanalysis

SimP needs cryptanalysis of related-key differential and linear characteristics. Existing methods such as the exhaustive search in [67] or SAT solvers [74], render such studies difficult due to the large state size since the known tools cannot scale appropriately. There exist peer-reviewed related-key results on Simon, e.g., by Wang et al. [84]. For the sake of feasibility, they restricted their search to related-key trails for the small variants, i.e., Simon-32, Simon-48, and Simon-64.
We conducted experiments using the SAT-based approach from [74] as well as with the branch-and-bound approach from [67] to search for optimal differential characteristics on SimP. Though, the related-key analysis of Simon-like constructions is computationally difficult because of the large state size. We obtained improved trails for only up to seven rounds of Simon-96; starting from eight rounds, the best characteristics found possessed a zero key difference for up to 10 rounds, which suggests that differences in the few key words do not improve the best single-key characteristics. It seems that the probabilities of the existing optimal differential characteristics and linear trails for Simon-96-96 and Simon-128-128 also hold for SimP-192-1 and SimP-256-1 beyond that point. Table 3.7 compares the probabilities of optimal single- and related-key differential characteristics.

3.10.3.2 Differential Distinguishers

Differential distinguishers are not expected to cover more than two steps. For two steps, we can sketch the high-level idea for a potential distinguisher. Let

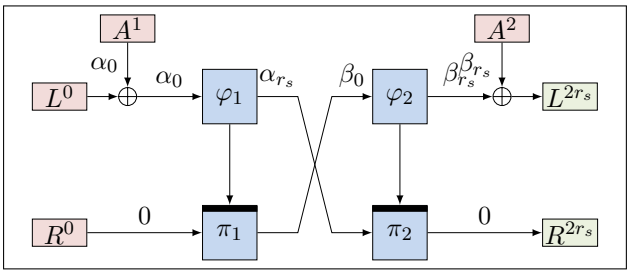


FIGURE 3.9: Setting of a differential attack with the step-reduced instance of SimP

$\alpha_0, \alpha_{r_s}, \beta_0, \beta_{r_s} \in (\mathbb{F}_2^w)^2$. Let $(\alpha_0, 0) \to (\beta_0, \alpha_{r_s})$ be a differential through one step of SimP. $\alpha_0 \to \alpha_{r_s}$ is a probability-one differential through r_s rounds of the key-update function φ_1, and $0 \to \beta_0$ the related-key differential through r_s rounds of the state-update function π_1, keyed with K and $K \oplus \alpha_0$, respectively. Assume, $p =^{\text{def}} \Pr[(\alpha_0, 0) \to (\beta_0, \alpha_{r_s})] > 2^{-2w}$.
In the second step, the difference β_0 is transformed to β_{r_s} linearly, i.e., $\Pr[\beta_0 \to \beta_{r_s}] = 1$. We can assume that $\beta_0 \neq 0$ and (β_0, α_{r_s}) will be transformed to an output difference $(\beta_{r_s}, 0)$ after the second step with probability $q \approx 2^{-2w}$ by approximating π_2 by a random permutation and using the Markov assumption and the random-key hypothesis. In this case, it holds that

$$\Pr\left[(\alpha_0, 0) \xrightarrow{p} (\beta_0, \alpha_{r_s}) \xrightarrow{2^{-2w}} (\beta_{r_s}, 0)\right] = p \cdot 2^{-2w} > 2^{-4w}\,.$$

Since π_1 is a round-reduced variant of Simon with 26 or 34 rounds, it is possible to have such trails. This setting is illustrated in Figure 3.9. However, an adversary would have to find a manyfold of related-key characteristics.

3.10.3.3 Integral and impossible-differential Distinguishers

Integral and impossible-differential distinguishers are possible for up to two steps of SimP in general. We study an integral distinguisher in the following: Consider a structure of texts (L_i^0, R_i^0) that use pairwise distinct values R_i^0 and leave L_i^0 constant. After the first step, the value $R_i^{r_s}$ is also constant for all texts and all values $L_i^{r_s}$ are pairwise distinct. This property is preserved through the linear key schedule to $L_i^{2r_s}$. However, the values of R^{2r_s} are, in general, unknown. Note that there is no word swap after the second step. The third step destroys that knowledge.

Impossible-differential distinguishers can use a similar strategy, by setting $\Delta L^0 = 0$ and testing if $\Delta L^{2r_s} = 0$, which is impossible. Again, note that there is no word swap after the second step.

3.10.3.4 Cube-like Distinguishers

Cube (and integral) distinguishers exploit that the degree of some output-bit component functions is lower than the state size. As discussed above, the degree of each bit is at least w after more than 22 rounds for Simon-96 and more than 26 rounds for Simon-128. SimP transforms each input bit through at least two steps of Simon, that consist of 26 rounds for Simon-96 and 34 rounds for Simon-128. While two out of four steps do not increase the degree in the part that is used as the key-update function, each bit is transformed through full-round Simon-96 or Simon-192. Thus, cube-like and integral distinguishers are not expected to threaten the security of SimP.

3.10.3.5 Number of Steps and Rounds of SimP

SimP benefits from the intensive existing cryptanalysis of Simon. The usage of the key-update function of Simon seems to not promote considerably more effective differential or linear distinguishers compared to the single-key results on Simon. The usage of the $2w$-word key appears not exploitable either by differentials and linear characteristics or by techniques that try to benefit from a larger state, such as meet-in-the-middle distinguishers. The reason seems to be mainly the diffusion in the key schedule together with the relatively large number of rounds.
The number of steps and the number of rounds in our employed instantiations of SimP have been chosen very conservatively, using the number of rounds per step r_s as half the number of rounds in Simon. This choice guarantees that each bit passes at least once through the full-round cipher, and therefore is expected to possess at least the algebraic degree of the full-round cipher. Moreover, the diffusion properties of Simon render impossible-differential, zero-correlation, or integral distinguishers implausible.
The design of SimP is very close to the original design of Simon. So, any considerable improvement in the cryptanalysis on SimP would most likely also be a higher threat on Simon-$2w$-$2w$. While such results are not impossible, the higher number of rounds in SimP provides an additional security margin.

TABLE 3.8: Implementation results for SimP-256 and Oribatida-256-64 encryption/decryption and only encryption on *Virtex 7 FPGA*. LUTs = lookup tables; Enc. = encryption; Dec. = decryption

	LUTs	FF	#Slices	Frequency (MHz)	Cycles	Throughput (Mb/s)
SimP						
SimP-256	495	340	148	580.51	137	542.37
SimP-192	383	259	122	581.98	105	532.10
Oribatida-256-64						
Enc. and Dec.	940	599	298	554.16	138	514.00
Enc. only	805	595	253	560.71	138	520.08

3.11 FPGA Implementations

This section reports on FPGA implementations of SimP and Oribatida.

3.11.1 SimP

SimP is lightweight since its transformations are exactly the round function and the key-update function of Simon-96-96 or Simon-128-128, respectively. Both transformations are based on simple operations such as rotations, XORs, and ANDs that consume only routing resources and bit-wise logical operations. The area in GEs is approximately that of Simon-96 plus some overhead, which is caused by the need for additional input to both transformations due to the swapping after r_s rounds.

Unprotected implementations of Simon are vulnerable against differential power analysis attacks using the leakage generated by the transitions in the state register; the Hamming-distance model captures such leakage. Masking – in particular, Boolean masking (XORing a random value to the output of the round function) – is one countermeasure that can be applied to Simon easily. The simple structure of Simon components allows exploring other countermeasures such as unrolling rounds to achieve higher-order side-channel resistance.

SimP can be implemented in different levels of serialisation, from fully serial implementations that update only a single bit per cycle up to round-based implementations that update the full state in one clock cycle. Depending on the choice, there is a broad implementation spectrum with a trade-off between throughput and area.

3.11.2 Oribatida

Hardware implementations of our proposed instance of Oribatida are relatively straight-forward. It can be implemented efficiently with little extra cost compared to the duplex sponge. Additional costs result from the use of a module to generate the constants for the domain separation, which can be held in ROM. In modern FPGAs, this module takes only four look-up tables (LUTs). For domain separation, only a four-bit XOR is necessary at the input to the capacity of the permutation. An additional 64-bit register to store a mask and a 64-bit XOR to add the mask to the ciphertext are required.
The use of SimP as its main building block allows us to directly transfer the same strategy of using different data-path sizes to Oribatida. Thus, the implementer can choose among various trade-offs between throughput, latency, area, and power consumption.
In terms of side-channel resistance, the same aspects that hold for SimP also hold for Oribatida. Thus, Oribatida does not introduce additional weaknesses of side channels. Table 3.8 lists our implementations results obtained from Xilinx Vivado 2018 optimising for the area. All results represent measurements after the place-and-route process.
In Table 3.8, we list two columns for the number of clock cycles and throughput; the former represents the results for the processing of associated data (with the step-reduced SimP), whereas the latter denotes the results for processing the message (with the non-reduced SimP). Our results still leave room for further improvements in the near future.

3.12 Conclusion

This chapter presented Oribatida, a permutation-based NAEAD scheme that masks the ciphertexts from preceding permutation outputs. As a result, the adversary cannot deduce the masked part of the internal state. Therefore, Oribatida can achieve $O(q_d^2/2^c)$ security against forgeries under the release of unverified plaintexts. Oribatida improves the best known INT-RUP security bound while the permutation can be kept as small as $64 + 128$ bits for 128-bit AE and 64-bit INT-RUP security.
We showed that even recent previous proposals with high AE security guarantees, such as Beetle or SPoC, succumb to attacks with complexity $O(\frac{q_d q_p}{2^c})$. In contrast, the security bound of Oribatida does not depend primarily on the number of

primitive queries. We showed that our bound is tight with a matching INT-RUP attack, generalised our masking approach by applying it also to Beetle or the NIST submission SPoC, and demonstrated its application with similar attacks on their masked variants.

As we have already seen, the other NIST candidates that claim INT-RUP security are all TBC-based. As the performance of an NAEAD mode depends not only on the rate but also on the speed of the underlying primitive, a natural follow-up to this work should be to implement them as well as Oribatida for a fair comparison of their performances.

Chapter 4

ISAP+

This chapter analyses the lightweight, sponge-based NAEAD mode ISAP, one of the finalists of the NIST lightweight cryptography project, that achieves high throughput with inherent protection against differential power analysis (DPA). We observe that ISAP requires 256-bit capacity in the authentication module to satisfy the NIST security criteria. In this chapter, we study the analysis carefully and observe that this is primarily due to the collision in the associated data part of the hash function which can be used in the forgery of the mode. However, the same is not applicable to the ciphertext part of the hash function because a collision in the ciphertext part does not always lead to a forgery. In this context, we define a new security notion, named 2PI+ security, which is a strictly stronger notion than the collision security, and show that the security of a class of encrypt-then-hash based MAC type of authenticated encryptions, that includes ISAP, reduces to the 2PI+ security of the underlying hash function used in the authentication module. Next we investigate and observe that a feed-forward variant of the generic sponge hash achieves better 2PI+ security as compared to the generic sponge hash. We use this fact to present a close variant of ISAP, named ISAP+, which is structurally similar to ISAP, except that it uses the feed-forward variant of the generic sponge hash in the authentication module. This improves the overall security of the mode, and hence we can set the capacity of the ciphertext part to 192 bits (to achieve a higher throughput) and yet satisfy the NIST security criteria.

4.1 Introduction

The emergence of side-channel and fault attacks [85–88] has made it clear that cryptographic implementations may not always behave like a black box. Instead,

they might behave like a grey box where the attacker has physical access to the device executing a cryptographic task. As a result, designers have started to design side-channel countermeasures such as masking [89, 90]. However, cryptographic primitives like block-ciphers (for example, AES [91] and ARX-based designs [92]) are costly to be mask-protected against side-channel attacks. Consequently, designing primitives or modes with inherent side-channel protection is becoming an essential and popular design goal. In this line of design, several block-ciphers (e.g., Noekeon [93], PICARO [94], Zorro [95]) and permutations (e.g., `ASCON`-p [96], `KECCAK`-p [97, 98]) have been proposed with dedicated structures to reduce the resource requirements for masking. In addition, a few NAEAD modes, such as `ASCON` [96, 99], PRIMATES [100], SCREAM [101], KETJE/KEYAK [102] have been proposed and submitted to the CAESAR [4] competition with the same goal in mind. However, they still lead to significant overheads.

In [103], Medwed et al. have proposed a new technique called fresh *re-keying* to have inherent side-channel protection. Following their work, a series of works [104, 105, 105] have been proposed that use this novel concept. This technique requires a side-channel resistant fresh key computation function with the nonce and the master key as the inputs. The main idea behind these designs is to ensure that different session keys are used for different nonces. Hence, a new nonce should be used to generate a fresh session key.

4.1.1 ISAP and Its Variants

In [8], Dobraunig et al. proposed a new authenticated encryption, dubbed ISAP v1, following the re-keying strategy. It is a sponge-based design [45, 106] that follows the Encrypt-then-MAC paradigm [107, 108]. We'll traditionally use the terms *rate* and *capacity* to represent the exposed part and the hidden part of the state of the sponge construction respectively. ISAP v1 claims to offer higher-order differential power analysis (DPA) protection provided by an inherent design strategy that combines a sponge-based stream-cipher for the encryption module with a sponge-based MAC (suffix keyed) for the authentication module. Both the modules compute fresh session keys using a GGM [109] tree-like function to strengthen the key computation against side-channel attacks.

Later, ISAP v1 was improved to ISAP v2 [9, 110], and was submitted to the NIST lightweight cryptography project where it was selected as one of the finalists. ISAP v2 is equipped with several promising features. It recommends two variants,

instantiated with the lightweight permutations ASCON-p and KECCAK-p[400]. Precisely, ISAP v2 retains all the inherent DPA resistance properties of ISAP v1 along with a better resistance against other implementation based attacks. In addition, ISAP v2 is even more efficient in hardware resources than the first version. ISAP v2 has been highly praised by the cryptography community, and to the best of our knowledge, it is the only inherently DPA protected NAEAD mode that aims to be implemented on lightweight platforms. The ISAP mode is flexible and can be instantiated with any sufficiently large permutation. Precisely, the security claims made by the designers depict that ISAP needs around 256-bit capacity to satisfy the NIST security criteria and hence needs a permutation with the state size larger than 256-bits. Thus it is highly desirable to analyse the mode further to understand whether it can be designed with a smaller capacity and hence a higher rate, that directly impacts the throughput.

4.1.2 Improving the Throughput of ISAP

ISAP v2 proposes four instances with the ASCON-p and the KECCAK-p permutations. ISAP v2 with ASCON-p (a 320-bit permutation) is designed with a 64 bit rate. However, it is better to achieve a higher rate design for a higher throughput. The observation is similar for the other instances as well. A potential direction can be to design an algorithm with improved security bound over the capacity to increase the rate without compromising the security level. An increased security bound can also help the designers to achieve the same security with a lower state size. This, in turn, may reduce the register size by using a permutation with a smaller state. A potential choice can be to analyse ISAP with a focus on the BBB (Beyond Birthday Bound) NAEAD security.

We observe that ISAP adopts an efficient approach of applying an unkeyed hash function on the nonce, the associated data, and the ciphertext, and then uses a PRF (Pseudo Random Function) on the hash value. In this case, the security of the NAEAD mode boils down to the collision security of the underlying hash function. This mode can bypass the requirement of storing the master key and can get rid of the key register. However, the hash collision results in a relatively low security bound that may not always be acceptable in ultra-lightweight applications as low security bound forces the designs to adopt primitives with high state size. Thus, an increase in the security bound has the full potential to increase the hardware performance of the design significantly. Motivated by this issue, we aim to study the tightness of the security bound to optimise the throughput and

the hardware footprint. Note that, ISAP is already an efficient construction and has been reviewed rigorously by various research groups. Hence, more detailed mode analysis and any possible mode optimisation can further strengthen the construction.

The security proof in [8] by Dobraunig et al. showed that ISAP achieves security up to the birthday bound on the capacity, i.e. of $O(T^2/2^c)$, where T is the time complexity (or, the number of offline queries to the underlying public permutation), and c is the capacity size (in bits). We observe that this factor arises due to the simple sponge-type hash applied on the nonce, the associated data, and the random ciphertext. It is obvious to get a collision in the nonce and the associated data that can be trivially used by an adversary to mount a forgery. However, it is not evident how a collision in the random ciphertexts can lead to such an attack. In this regard, we investigate the amount of the ciphertext-bit that can be injected per permutation call during the hash and ask the question:

"Can we increase the rate of absorption of the ciphertext blocks in the hash?"

We believe that a positive answer to this question will not only result in a more efficient construction but, more importantly, contribute significantly to the direction of NAEAD mode analysis.

4.1.3 Contributions

In this chapter, we study a simple variant of ISAP that achieves higher throughput keeping all the primary features of ISAP intact. The contribution of this chapter is four-fold:

1. First, in Section 4.3, we propose a permutation-based generic EtHM (Encrypt then Hash based MAC) type NAEAD mode using a PRF and an unkeyed hash function. This is essentially a generalisation of ISAP type constructions. Note that this generic mode does not guarantee any side-channel resistance; only proper instantiation of the PRF ensures that. In Section 4.3.2, we show that the NAEAD security of EtHM can be expressed in terms of the PRF security of a fixed input length, variable output length keyed function F and the 2PI+ security of H. Intuitively, the 2PI+ notion demands that given a challenge random message of some length chosen by the adversary, it is difficult for an adversary to compute the second pre-image of the random message. We introduce the notion in Section 4.2.2. This is in contrast with the traditional collision security that was used for the analysis of ISAP.

2. Next, in Section 4.4, we first show that for generic sponge hash, a collision attack can be extended to a 2PI+ attack. Thus, the generic sponge hash achieves 2PI+ security of $\Omega(T^2/2^c)$, where T is the time complexity and c is the capacity size of the sponge hash (in bits). Next, we consider a feed-forward variant of the sponge hash that uses (i) generic sponge hash to process the nonce and the associated data and (ii) a feed-forward variant of the sponge hash to process the message. We show that the feed-forward property ensures that a collision attack can not be extended to mount a 2PI+ attack. In fact, we prove that this variant of sponge hash obtains an improved security of $O(DT/2^c)$. Note that D and T are data (or, the number of online queries to the construction) and time complexity respectively, and we typically allow $T \approx D^2$. Hence, feed-forward-based sponge achieves a better 2PI+ security as we consider security in terms of D and T instead of traditional one-parameter security (i.e., in terms of T only).

3. Next, in Section 4.5, we consider a simple variant of ISAP with minimal changes, named ISAP+, which is a particular instantiation of the generic EtHM mode. To be specific, the differences between ISAP+ and ISAP are as follows:

   3.1 Instead of using the generic sponge hash as used in ISAP, we use the feed-forward variant of sponge hash as discussed above.

   3.2 In the authentication module of ISAP+, we use the capacity of c' bits for nonce, associated data and first block of ciphertext processing. For rest of the ciphertext blocks, we use capacity of c bits.

   3.3 We make a separation among the messages depending on whether its length is less than r' bits or not. The domain separation is performed by adding 0 or 1 to the capacity part before the final permutation call.

   This modification ensures that ISAP+ achieves improved security of $O(T^2/2^{c'} + DT/2^c)$, where n is the state-size or the size of the permutation (in bits), $c = n - r$, $c' = n - r'$). This security boost allows the designer to choose c' and $c(< c')$ effectively to obtain better throughput.

4. Our primary proposal is denoted as ISAP+-A-128, which is instantiated with ASCON-p. For a fair comparison with ISAP, we show the implementation results of three more variants of ISAP+ in Section 4.6, comparable with the corresponding ISAP instances. We instantiate these variants with ASCON-p

and KECCAK-p[400] permutations. These variants are denoted as ISAP+-A-128, ISAP+-A-128A, ISAP+-K-128 and ISAP+-K-128A. The first two use full and round reduced variants of ASCON-p and the other two use full and round reduced variants of KECCAK-p[400] respectively. Note that all these instances follow the ISAP+ mode, and they only differ in the choice of the underlying permutation. Also note that these four ISAP+ instances use ASCON-p or KECCAK-p[400] with the same number of rounds as their ISAP counterparts for a fair comparison. Finally, we provide a detailed hardware implementation in Section 4.6 for all the ISAP+ and ISAP instances.

Note that we propose only one instance of ISAP+ to stick to the security assumption that we use one single permutation in our construction and only the ISAP+-A-128 variant achieves the same. All the other three variants use different numbers of rounds at different stages of our construction. We have implemented them for a fair comparison with ISAP and to showcase the efficiency of the ISAP+ mode in throughput.

4.1.4 Relevance of the Work

To understand the relevance of the improved security, let us consider the instantiation of ISAP with ASCON and KECCAK and compare them with ISAP+. To satisfy the NIST requirements, ISAP+ can use $c = 192, c' = 256$, and hence, it has an injection rate of $r = 128$-bits for the ciphertexts and an injection rate of $r = 64$ bits for associated data. Table 4.1 demonstrates a comparative study of ISAP and ISAP+ in terms of the number of permutation calls required for the authentication module.

TABLE 4.1: Comparative study of ISAP+ and ISAP on the no. of permutation calls in the authentication module for associated data of length a bits and message of length m bits

Mode	Permutation	Parameters	# permutation calls
ISAP	ASCON	$r = 64$	$\lceil \frac{a+m+1}{64} \rceil$
ISAP+	ASCON	$r = 128, r' = 64$	$\lceil \frac{a+1}{64} \rceil + \lceil \frac{m}{128} \rceil$
ISAP	KECCAK	$r = 144$	$\lceil \frac{a+m+1}{144} \rceil$
ISAP+	KECCAK	$r = 208, r' = 144$	$\lceil \frac{a+1}{144} \rceil + \lceil \frac{m}{208} \rceil$

This result demonstrates that for applications that require long message processing, ISAP+ performs better than ISAP in terms of throughput and speed. Let us consider a concrete example. Consider encrypting a message of length 1 MB with

an associate data of length 1 KB using ASCON permutation. With ISAP, the authentication module requires around $1,31,201$ many primitive calls. On the other hand, with ISAP+ this requires only $65,665$ many primitive calls, which is almost half as compared to ISAP.

This chapter depicts the robustness of the mode ISAP, and how one can increase the throughput of the mode at the cost of some hardware area preserving all the inherent security features. This result seems relevant to the cryptography community in the sense that ISAP is also a finalist of the NIST lightweight cryptography project.

4.1.5 Interpretation of Hardware Implementation Result

Both ISAP+ and ISAP have been implemented using VHDL and mapped on Virtex 7XC7V585T (Vivado 2020.2), and the results are summarised in Table 4.2. The result depicts that all the ISAP+ instances are better in throughput and throughput/area than the ISAP instances with a small compromise in the hardware area. Blue-coloured entry is our primary recommendation.

TABLE 4.2: FPGA results of ISAP+ and ISAP

Instances	Slice Registers	LUTs	Slices	Frequency (MHZ)	Encryption Throughput (Gbps)	Authentication Throughput (Gbps)
ISAP-A-128	818	1550	451	400	2.13	2.13
ISAP+-A-128	1081	1742	507	384.16	2.05	3.07
ISAP-K-128	1143	1932	582	300	3.60	2.16
ISAP+-K-128	1423	2245	613	300	3.60	2.64
ISAP-A-128A	810	1539	449	400	4.27	2.13
ISAP+-A-128A	1070	1728	501	384.16	4.09	3.07
ISAP-K-128A	1131	1850	574	300	5.40	2.70
ISAP+-K-128A	1412	2231	605	300	5.40	4.40

4.2 Preliminaries

4.2.1 Fixed Input - Variable Output PRFs with Prefix Property

A fixed input variable output function (FIL-VOL) is a keyed function $F_K : \{0,1\}^\star \times \{0,1\} \times \mathbb{N} \to \{0,1\}^\star$ that takes as input an input a string $I \in \{0,1\}^\star$, a flag $b \in \{0,1\}$ as input, a positive integer $\ell \in \mathbb{N}$, and outputs a string $O \in \{0,1\}^\ell$, i.e., $O := F_K(I, b; \ell)$. We call such a keyed function a FIL-VOL pseudo random function maintaining the prefix-property if

- for all inputs $(I, b; \ell), (I', b'; \ell')$ with $(I, b) \neq (I', b')$, $F_K(I, b; \ell)$, $F_K(I', b'; \ell)$ are distributed uniformly at random, and
- for all inputs $(I, b; \ell), (I, b; \ell')$, with $\ell' > \ell$, $\lceil F_K(I', b'; \ell') \rceil_\ell = F_K(I', b'; \ell)$.

More formally,

$$\mathbf{Adv}^F_{\mathsf{PRF}}(\mathcal{A}) := |\Pr[\mathcal{A}^{F_K} = 1] - \Pr[\mathcal{A}^f = 1]|,$$

where f is a random function from same domain and range maintaining the prefix property.

4.2.2 Multi-Target 2nd Pre-Image with Associated Data

In this section, we discuss the notion of multi-target 2nd pre-image security of permutation-based hash functions.
In this setting, an adversary (say $\mathcal{A}$) chooses q (nonce, associated data, length)-tuples to the challenger $\mathcal{C}$, say $(N_i, A_i, \ell_i)_{i=1..q}$. The challenger in turn returns q uniformly random messages of specified lengths respectively, say $C_1, \ldots, C_q$. The queries $(N_i, A_i, C_i)_{i=1..q}$ are called *challenge queries*. The goal of $\mathcal{A}$ is to return q' many $(N'_j, A'_j, C'_j)_{j=1..q'}$ (called response queries) tuples such that at least one of the hash values of (N'_j, A'_j, C'_j) matches with the hash value of any one of the (N_i, A_i, C_i). Note that the adversaries are allowed to set some challenge queries as response queries: $(N'_j, A'_j, C'_j) = (N_i, A_i, C_i)$, for some i, j. However, for the winning event the challenge and response queries should be distinct. The adversary can make up to q_p queries to p or p^{-1}. Formally, the advantage of $\mathcal{A}$ is defined as

$$\begin{aligned} \mathbf{Adv}^{\mathrm{H}}_{\mathsf{2PI+}}(\mathcal{A}) \;:=\; & \Pr[\exists i, j, H^p(N'_j, A'_j, C'_j) = H^p(N_i, A_i, C_i), \\ & (N'_j, A'_j, C'_j) \neq (N_i, A_i, C_i)]. \end{aligned}$$

where H is an IV-based hash function. Note that the adversary is allowed to make hash queries before, after, or in between its interaction with the challenger to obtain the challenge message(s). Also, note that the 2PI+ security does not depend on the message length. The fact that the adversary submits a length ℓ_i to the challenger to obtain each message before the submission of the challenge message is merely because the 2PI+ security notion enables its adversary to obtain messages of whatever lengths it pleases.

4.3 An EtHM Paradigm for NAEAD

This section introduces an efficient generalised Encrypt-then-Hash based MAC (EtHM) paradigm for NAEAD modes. This is a generalised paradigm for constructing side-channel resilient modes such as ISAP.

4.3.1 Specification

Let n, k and τ be positive integers such that $n > \tau$. The construction takes as input a plaintext M, a nonce N, an associated data A, and outputs a ciphertext C and a tag T. Given a permutation based FIL-VOL keyed-function with prefix property F_K^p, and a permutation based un-keyed hash function $H^p : \{0,1\}^* \to \{0,1\}^n$ the mode works as follows.

$$C = M \oplus F_K^p(N, 0; |M|),$$
$$T \| D = p(F_K^p(X, 1; |X|) \| Z), \text{ where } X \| Z = H^p(N, A, C)\,.$$

The authenticated encryption module is pictorially depicted in Figure 4.1. Note that T denotes the most significant τ bits of the output of the permutation call. The least significant $(n-\tau)$ bits is denoted by D. Note that we do not need D from the construction point of view; however, we require it during the security analysis. Notations F, H and F_K^p, H^p have been used in this chapter interchangeably for convenience and aren't supposed to create any confusion. From time to time, we'll address this paradigm as EtHM only.

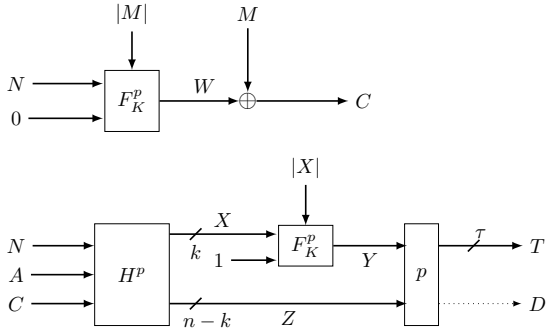


FIGURE 4.1: Authenticated encryption module of the EtHM paradigm

4.3.2 Security of EtHM

In this subsection, we analyse the NAEAD security of EtHM with F as the underlying function and H as the multi-target IV-respecting second pre-image resistant hash function. Formally, we prove the following theorem.

Theorem 4.1 (NAEAD Security of EtHM)**.** *Consider EtHM based on a function F and a hash function H. For all deterministic nonce-respecting non-repeating query making adversary $\mathcal{A}$ which can make at most q_e encryption queries, q_v decryption queries and q_p primitive queries to p and its inverse and assuming $q = q_e + q_v$, there exists two adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$ such that the NAEAD advantage of $\mathcal{A}$ can be bounded by*

$$\begin{aligned}\mathbf{Adv}^{EtHM}_{NAEAD}(\mathcal{A}) \leq \mathbf{Adv}^{F}_{PRF}(\mathcal{B}_1) + 2\mathbf{Adv}^{\lfloor H \rfloor_{n-k}}_{2PI+}(\mathcal{B}_2) + \frac{qq_p}{2^n} + \frac{2kq_v}{2^k} + \frac{q_v}{2^\tau} \\ + \frac{\binom{q_p}{k}}{2^{\tau(k-1)}} + \frac{\binom{q_p}{k}}{2^{(n-k)(k-1)}},\end{aligned}$$

where $\mathcal{B}_1$ can make $2q$ PRF queries and $\mathcal{B}_2$ can make q challenge queries, q response queries and q_p primitive queries to p and its inverse.

Proof. Let $\mathbf{Enc}^{F^p_K,p}$ and $\mathbf{Dec}^{F^p_K,p}$ be the encryption and the decryption function of EtHM respectively. Let us call its oracle $\mathcal{O}_1 = (\mathbf{Enc}^{F^p_K,p}, \mathbf{Dec}^{F^p_K,p}, p)$. We have to upper-bound the distinguishing advantage of $\mathcal{A}$ interacting with $\mathcal{O}_1$ or $\mathcal{O}_3 = (\$, \perp, p)$. For our purpose, we define an intermediate oracle by replacing F^p_K in $\mathcal{O}_1$ by a random function $\$$. Let us call this new intermediate oracle $\mathcal{O}_2 = (\mathbf{Enc}^{\$,p}, \mathbf{Dec}^{\$,p}, p)$. We will employ a standard reduction proof. We break down the distinguishing game of $\mathcal{A}$ using the triangle inequality as follows.

$$\begin{aligned}\mathbf{Adv}^{\mathsf{EtHM}}_{\mathrm{NAEAD}}(\mathcal{A}) &= |\Pr[\mathcal{A}^{\mathcal{O}_1} = 1] - \Pr[\mathcal{A}^{\mathcal{O}_3} = 1]| \\ &\leq |\Pr[\mathcal{A}^{\mathcal{O}_1} = 1] - \Pr[\mathcal{A}^{\mathcal{O}_2} = 1]| \\ &+ |\Pr[\mathcal{A}^{\mathcal{O}_2} = 1] - \Pr[\mathcal{A}^{\mathcal{O}_3} = 1]|\,. \qquad (4.1)\end{aligned}$$

Now, we bound each of the two terms.

Bounding $|\mathbf{Pr}[\mathcal{A}^{\mathcal{O}_1} = 1] - \mathbf{Pr}[\mathcal{A}^{\mathcal{O}_2} = 1]|$. We bound this term by the PRF advantage of F. For that, let us consider the following adversary $\mathcal{B}_1$ that runs $\mathcal{A}$ (any distinguisher of $\mathcal{O}_1$ and $\mathcal{O}_2$) as follows.

- Whenever $\mathcal{A}$ submits an encryption query (N, A, M), $\mathcal{B}_1$ submits $(N, |M|, 0)$ to its challenger. Suppose the challenger returns C. $\mathcal{B}_1$ calculates $X\|Z = H^p(N, A, C)$ with $|X| = k$ and $|Z| = n - k$ and submits $(X, 1; k)$ to its challenger. Suppose the challenger returns Y. $\mathcal{B}_1$ returns $(C, p(Y\|Z))$ to $\mathcal{A}$.

- Similarly, whenever $\mathcal{A}$ submits a decryption query $(\hat{N}, \hat{A}, \hat{C}, \hat{T})$, $\mathcal{B}_1$ submits $(\hat{N}, 0; |\hat{C}|)$ to its challenger. Suppose the challenger returns $\hat{M}$. $\mathcal{B}_1$ calculates $\hat{X}\|\hat{Z} = H^p(\hat{N}, \hat{A}, \hat{C})$ with $|\hat{X}| = k$ and $|\hat{Z}| = n - k$ and submits $(\hat{X}, 1; k)$ to its challenger. Suppose the challenger returns $\hat{Y}$. $\mathcal{B}_1$ calculates $\lceil p(\hat{Y}\|\hat{Z})\rceil_\tau$ If $T = \hat{T}$, then $\mathcal{B}_1$ returns $\hat{M}$ to $\mathcal{A}$. Otherwise it returns $\perp$.

- At the end of the game, $\mathcal{A}$ submits the decision bit to $\mathcal{B}_1$ which it forwards to its challenger. Note that when $\mathcal{A}$ supposedly interacts with $\mathcal{O}_1$ or $\mathcal{B}_1$ supposedly interacts with F_K^p, they submit $b = 1$. Otherwise, they submit $b = 0$.

It is easy to see $\Pr[\mathcal{A}^{\mathcal{O}_1} = 1] = \Pr[\mathcal{B}_1^{F_K^p} = 1]$ and $\Pr[\mathcal{A}^{\mathcal{O}_2} = 1] = \Pr[\mathcal{B}_1^{\$} = 1]$, and hence we obtain the following.

$$|\Pr[\mathcal{A}^{\mathcal{O}_1} = 1] - \Pr[\mathcal{A}^{\mathcal{O}_2} = 1]| = \mathbf{Adv}^F_{\mathsf{PRF}}(\mathcal{B}_1). \tag{4.2}$$

Bounding $|\mathbf{Pr}[\mathcal{A}^{\mathcal{O}_2} = 1] - \mathbf{Pr}[\mathcal{A}^{\mathcal{O}_3} = 1]|$. This bound follows from the lemma given below, the proof of which is deferred to the next section.

Lemma 4.2. *Let $\mathcal{A}$ be a deterministic nonce-respecting non-repeating query making adversary interacting with oracle $\mathcal{O}_2$ or $\mathcal{O}_3$ which can make at most q_e encryption queries, q_v decryption queries and q_p primitive queries to p and its inverse. Assuming $q = q_e + q_v$, there exists an adversary $\mathcal{B}_2$ such that the NAEAD advantage of $\mathcal{A}$ can be bounded by*

$$|\Pr[\mathcal{A}^{\mathcal{O}_2} = 1] - \Pr[\mathcal{A}^{\mathcal{O}_3} = 1]| \leq 2\mathbf{Adv}^{\lfloor H\rfloor_{n-k}}_{\mathit{2PI+}}(\mathcal{B}_2) + \frac{qq_p}{2^n} + \frac{2kq_v}{2^k} + \frac{q_v}{2^\tau} + \frac{\binom{q_p}{k}}{2^{\tau(k-1)}} + \frac{\binom{q_p}{k}}{2^{(n-k)(k-1)}},$$

where $\mathcal{B}_2$ can make q challenge queries, q response queries and q_p primitive queries to p and its inverse.

The proof of the theorem follows from Equation 5.1, Equation 5.2 and Lemma 4.2.

□

4.3.3 Proof of Lemma 4.2

Now we'll prove Lemma 4.2 using Coefficients H Technique step by step.

Step I: Sampling of the Ideal Oracle and Defining the Bad Events. We start with sampling of the ideal oracle and go on mentioning the bad events whenever they occur. Note that whenever we mention a bad event, even if it's not explicitly mentioned, it's implicitly understood that the previous bad events haven't occurred.

In the online phase, the adversary interacts with the oracles and receives the corresponding responses. In this phase, it can make any construction or permutation query. The i-th encryption query is (N^i, A^i, M^i), the i-th decryption query is $(\hat{N}^i, \hat{A}^i, \hat{C}^i, \hat{T}^i)$, $H(\hat{N}^i, \hat{A}^i, \hat{C}^i) = \hat{X}^i \| \hat{Z}^i$ with $|\hat{X}^i| = k$ and $|\hat{Z}^i| = n - k$, the i-th permutation query is U^i if it's a forward query (i.e., p query), and V^i if it's a backward query (i.e., p^{-1} query).

1. Return $(C^i, T^i), \forall i \in [q_e]$, where $C^i \xleftarrow{\$} \{0,1\}^{|M^i|}, T^i \xleftarrow{\$} \{0,1\}^{\tau}$.
2. Return $\perp, \forall i \in [q_v]$.
3. Return the true output values of the permutation queries.
4. Set $X^i := \lceil H(N^i, A^i, C^i) \rceil_k, \hat{X}^i := \lceil H(\hat{N}^i, \hat{A}^i, \hat{C}^i) \rceil_k$,
   $Z^i := \lfloor H(N^i, A^i, C^i) \rfloor_{n-k}, \hat{Z}^i := \lfloor H(\hat{N}^i, \hat{A}^i, \hat{C}^i) \rfloor_{n-k}$

The adversary aborts if the following (bad) event occurs.

- bad1: $\exists i \in [q_e]$ and $j \in [q_v]$ with $i \neq j$ and $(N^i, A^i, C^i) \neq (\hat{N}^j, \hat{A}^j, \hat{C}^j)$ such that $Z^i = \hat{Z}^j$.
- bad2: $\exists i, j \in [q_e]$ with $i \neq j$ such that $Z^i = Z^j$.

In the offline phase, the adversary can no longer interact with any oracle, but the challenger may release some additional information to the adversary before it submits its decision.

1. $Y^i \xleftarrow{\$} \{0,1\}^k, \forall i \in [q_e]$ and $j \in [i-1]$ with $X^i \neq X^j$.
2. $\hat{Y}^i \xleftarrow{\$} \{0,1\}^k, \forall i \in [q_v], j \in [i-1]$ and $\ell \in [q_e]$ with $\hat{X}^i \neq \hat{X}^j$ and $\hat{X}^i \neq X^\ell$.

Again, the adversary aborts if any of the following (bad) events occur.

- bad3: $\exists i \in [q_e]$ and $j \in [q_p]$ such that $Y^i \| Z^i = U^j$.

- bad4: There is a k-multi-collision at the τ most significant bits of the output of the forward permutation queries.
- bad5: There is a k-multi-collision at the $(n-k)$ least significant bits of the output of the backward permutation queries.
- bad6: $\exists i \in [q_v]$ and $j \in [q_p]$ such that $\hat{Y}^i \| \hat{Z}^i = U^j$.

If none of the bad events occur, then

1. $D^i \xleftarrow{\$} \{0,1\}^{n-\tau}, \forall i \in [q_e]$,
2. $\hat{T'}^i \| \hat{D}^i \xleftarrow{\$} \{0,1\}^n, \forall i \in [q_v]$.

Again, the adversary aborts if the following (bad) event occurs.

- bad7: $\exists i \in [q_v]$ such that $\hat{T}^i = \hat{T'}^i$.

Step II: Bounding the Probability of the Bad Events. Now we'll upper bound the probabilities of the bad events.

- bad1: This event says that the capacity part of the hash of an encryption query matches with the capacity part of the hash of a forging query. This is nothing but computing a second pre-image corresponding to a challenge (N, A, C), where C is chosen uniformly at random. Thus, the probability of this event is bounded by the 2PI+ security of H.

  $$\Pr[\mathsf{bad1}] \leq \mathbf{Adv}^{\lfloor H \rfloor_{n-k}}_{\mathsf{2PI+}}(\mathcal{B}_2),$$

  where $\mathcal{B}_2$ can make q challenge queries, q response queries and q_p primitive queries to p and its inverse.

- bad2: This event says that the capacity part of the hash of an encryption query matches the capacity part of the hash of another encryption query. This is again nothing but computing a second pre-image corresponding to a challenge (N, A, C), where C is chosen uniformly at random. Thus, the probability of this event is bounded by the 2PI+ security of H.

  $$\Pr[\mathsf{bad2}] \leq \mathbf{Adv}^{\lfloor H \rfloor_{n-k}}_{\mathsf{2PI+}}(\mathcal{B}_2),$$

  where $\mathcal{B}_2$ can make q challenge queries, q response queries and q_p primitive queries to p and its inverse.

- bad3: For a fixed encryption query and a fixed permutation query, the probability of this event comes out to be equal to $1/2^n$ due to the randomness of U^j. Applying union bound over all possible choices, we obtain

$$\Pr[\mathsf{bad3}] \leq \frac{q_e q_p}{2^n} .$$

- bad4: For a fixed k-tuple of forward permutation queries, the probability of this event comes out to be equal to $1/2^{\tau(k-1)}$ due to the randomness of the permutation output. Applying union bound over all possible choices, we obtain

$$\Pr[\mathsf{bad4}] \leq \frac{\binom{q_p}{k}}{2^{\tau(k-1)}} .$$

- bad5: For a fixed k-tuple of backward permutation queries, the probability of this event comes out to be equal to $1/2^{(n-k)(k-1)}$ due to the randomness of the permutation output. Applying union bound over all possible choices, we obtain

$$\Pr[\mathsf{bad5}] \leq \frac{\binom{q_p}{k}}{2^{(n-k)(k-1)}} .$$

- bad6: We analyse this bad event in the three following sub-cases.

  – In this case, the number of multi-collision at the τ most significant bits of the output of the forward permutation queries is at most k. So the adversary can make a hash query (N, A, C) to obtain $X \| Z$, fix Z as the least significant bits and vary the rest of the bits to obtain the multi-collision. Suppose the multi-collision happens at the value T. In that case, if the adversary makes the decryption query (N, A, C, T), then the probability of bad6 comes out to be equal to $k/2^k$. For q_v decryption queries, this probability comes out to be equal to $kq_v/2^k$.

  – In this case, the number of multi-collision at the $(n-k)$ least significant bits of the output of the backward permutation queries is at most k. So the adversary can fix the τ most significant bits (say T) and vary the rest of the bits to obtain the multi-collisions. Suppose the multi collisions happen at the values $Z_1, Z_2, \cdots, Z_m$. Also suppose that the adversary has q_1 hash pre-images of Z_1, q_2 hash pre-images of Z_2, $\cdots$, q_m hash

pre-images of Z_m, where $q_1+q_2+\cdots+q_m = q_v$. For $i \in [r]$, suppose the adversary has a pre-image (N, A, C) of Z_i. In that case, if the adversary makes the decryption query (N, A, C, T), then the probability of bad6 comes out to be equal to $k/2^k$. For q_v pre-images, this probability comes out to be equal to $kq_v/2^k$.

– If the previous two cases don't occur, i.e., there is no multi-collision, then for a fixed decryption query and a fixed permutation query, the probability of bad6 comes out to be equal to $1/2^n$ due to the randomness of U^j. For q_v decryption queries and q_p permutation queries, this probability comes out to be equal to $q_v q_p/2^n$.

Combining all three cases, we obtain

$$\Pr[\mathsf{bad6}|(\overline{\mathsf{bad3}} \wedge \overline{\mathsf{bad4}} \wedge \overline{\mathsf{bad5}})] \leq \frac{2kq_v}{2^k} + \frac{q_v q_p}{2^n}\,.$$

- bad7: For a fixed decryption query, the probability of this event comes out to be equal to $1/2^\tau$ due to the randomness of $\hat{T'}^i$. Applying union bound over all possible choices, we obtain

$$\Pr[\mathsf{bad7}] \leq \frac{q_v}{2^\tau}\,.$$

Combining everything, we obtain

$$\begin{aligned}\epsilon_{\mathsf{bad}} := \Pr[\mathsf{bad}] &\leq \Pr[\mathsf{bad1} \vee \mathsf{bad2} \vee \cdots \vee \mathsf{bad7}] \\ &\leq 2\mathbf{Adv}^{\lfloor H \rfloor_{n-k}}_{\mathsf{2PI+}}(\mathcal{B}_2) + \frac{qq_p}{2^n} + \frac{2kq_v}{2^k} + \frac{q_v}{2^\tau} \\ &\quad + \frac{\binom{q_p}{k}}{2^{\tau(k-1)}} + \frac{\binom{q_p}{k}}{2^{(n-k)(k-1)}}. \end{aligned} \tag{4.3}$$

Step III: Ratio of Good Interpolation Probabilities. We recall that to obtain oracle $\mathcal{O}_2$, we replace the function F of $\mathcal{O}_1$ with a random function \$. All the remaining specification of $\mathcal{O}_2$ are similar to $\mathcal{O}_1$ (see Section 4.3.1). Let q_x be the number of construction queries with distinct X^i's and $\hat{X}^i$'s and q' be the number of construction queries with distinct (N^i, A^i, C^i)'s and $(\hat{N}^i, \hat{A}^i, \hat{C}^i)$'s. For any good transcript τ, we get

$$\Pr_{\mathcal{O}_2}[\tau] = \frac{1}{2^{n\sigma_e}} \frac{1}{2^{kq_x}} \frac{1}{(2^n)_{q'+q_p}}\,.$$

The first term corresponds to the number of choices for W^i. The second term corresponds to the number of choices for Y^i. The third term corresponds to the number of choices for the outputs of the distinct permutation calls. We also get

$$\Pr_{\mathcal{O}_3}[\tau] = \frac{1}{2^{n\sigma_e}} \frac{1}{2^{nq'}} \frac{1}{2^{kq_x}} \frac{1}{(2^n)_{q_p}} .$$

The first term corresponds to the number of choices for C^i. The second term corresponds to the number of choices for $T^i \| D^i$. The third term corresponds to the number of choices for Y^i. The fourth term corresponds to the number of choices for the outputs of the distinct permutation calls. Thus we finally obtain

$$\frac{\Pr_{\mathcal{O}_2}[\tau]}{\Pr_{\mathcal{O}_3}[\tau]} \geq 1 \,, \text{ i.e., } \epsilon_{\mathsf{good}} = 0. \tag{4.4}$$

Step IV: Final Calculation. The Lemma follows as we use Equation 4.3 and Equation 4.4 in Theorem 2.1.

4.4 Multi-Target 2nd Pre-Image Security of Sponge Based Hashes

This section analyses the 2PI+ security of the sponge hash and some of its variants.

4.4.1 Sponge Hash and Its 2PI+ Security

First, we briefly revisit the sponge hash. Consider the initial state to be $N \| IV$ for some fixed IV. Let $p \in \mathsf{Perm}$ where Perm is the set of all permutations on $\{0,1\}^n$. We call the r most significant bits of the state as rate and the c' least significant bits of the state as capacity. The associated data A and the message C are absorbed in r'-bit blocks by subsequent p-calls, and the output of the last p-call is the hash output T. Figure 4.2 illustrates the sponge hash. Now let us look at its 2PI+ security.

The following attack demonstrates that the sponge hash is vulnerable to a meet-in-the-middle attack as follows.

- Suppose an adversary (say $\mathcal{A}$) submits $(N, A, 2)$ and receives the random message $C_1 \| C_2$ from its challenger where $|C_1| = |C_2| = r'$.
- $\mathcal{A}$ computes the hash as $H = p(p(S_1 \oplus C_1 \| S_2 \oplus 0^\star 1) \oplus (C_2 \| 0^c))$. Suppose $H = p(Y_2 \| Z_2)$ where $|Y_2| = r'$ and $|Z_2| = c$.

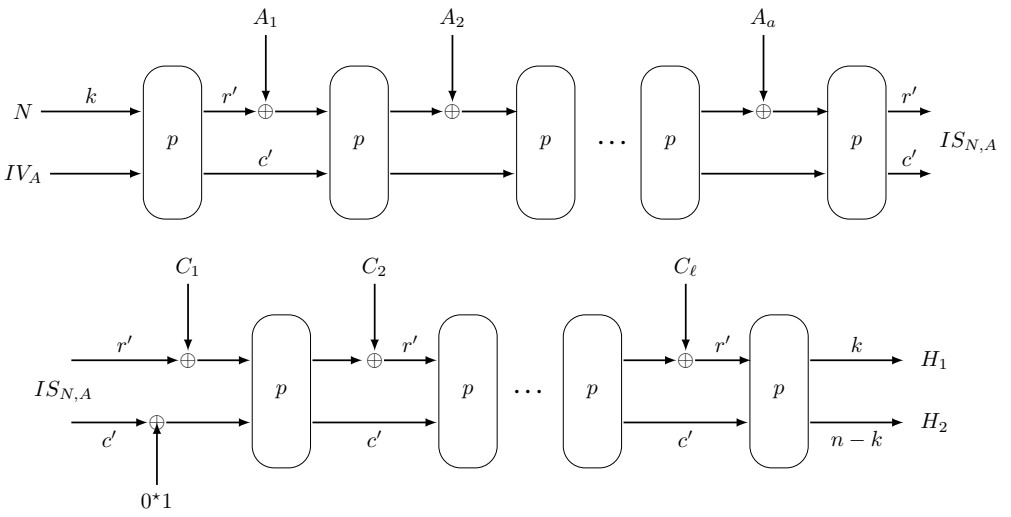


FIGURE 4.2: Sponge hash with ℓ message blocks

- $\mathcal{A}$ makes some p-queries of the form $\star \| IV$ and some p^{-1}-queries of the form $\star \| Z_2$, and stores the p-query outputs in the list $\mathcal{L}_1$ and the p^{-1}-query outputs in the list $\mathcal{L}_2$.
- Suppose the capacity of one entry in $\mathcal{L}_1$ (say $Y_1 \| Z_1$ where $|Y_1| = r'$ and $|Z_1| = c'$) matches with the capacity of one entry in $\mathcal{L}_2$ (say $Y_1^\star \| Z_1$). Suppose $p(N' \| IV) = Y_1 \| Z_1$ and $p^{-1}(Y_2^\star \| Z_2) = Y_1^\star \| Z_1$.
- $\mathcal{A}$ returns $(N', \epsilon, (Y_1 \oplus Y_1^\star) \| (Y_2 \oplus Y_2^\star))$ to its challenger as the second pre-image of the random message $(N, C_1 \| C_2)$.

It is easy to see that the attack succeeds with probability $\frac{|\mathcal{L}_1 \| \mathcal{L}_2|}{2^{c'}}$. In other words, if the adversary is able to make around $2^{c'/2}$ p-queries and p^{-1}-queries each, it would be able to mount this 2PI+ attack with very high probability. Thus, for the sponge hash, the 2PI+ security reduces to the collision security due to the above meet-in-the-middle attack, and the 2PI+ security for sponge hash is $\Omega(q_p^2/2^{c'})$. Now, we are more interested in other hash functions where such a collision attack doesn't induce a 2PI+ attack.

4.4.2 Feed Forward Based Sponge Hash and Its 2PI+ Security

Now, we consider a feed forward variant of the sponge hash. The nonce and associated data processing remain as it is. However, the following modifications during the random message processing:

- The capacity part of the output of the i-th permutation is xored with the previous state capacity to obtain the updated i-th state capacity.
- The message injection rate for the first block of random ciphertext remains r' bits, and for all successive blocks the rate is r bits, where $r \geq r'$. To make things compatible, the capacity part before the first p-call is chopped to the least significant c-bits while feed-forwarding.
- We use a domain separation before the final permutation call depending on the size of the random message. If the size is less than or equal to r', we xor 1 in the capacity.

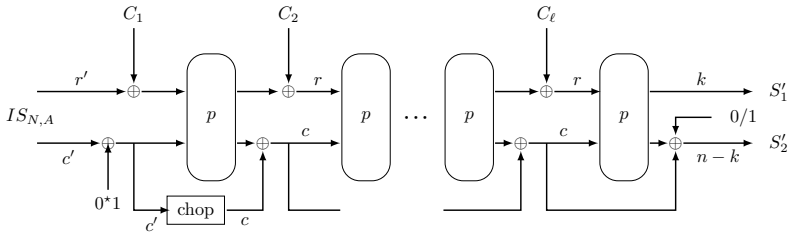


FIGURE 4.3: The feed forward variant of the sponge hash. The initial state $IS_{N,A}$ is generated identically as in the sponge hash, depicted in Fig. 4.2.

Figure 4.3 illustrates the feed forward variant of the sponge hash with n-bit hash value $H_1 \| H_2$. It is easy to see that the attack on the sponge hash can not be extended to this hash due to the feed-forward functionality of this hash. Now let us look at its 2PI+ security. Formally, we state the following lemma:

Lemma 4.3. *Let H be the feed-forward based sponge hash as defined above. The 2PI+ security of the construction is given by*

$$\mathbf{Adv}^{\lfloor H \rfloor_{n-k}}_{\mathsf{2PI+}}(\mathcal{A}) \leq \frac{q_p^2}{2^{c'}} + \frac{(q_p + \sigma_v)\sigma_e}{2^c} + \frac{\sigma_e^2}{2^c} + \frac{q_v}{2^{n-k}},$$

where $\mathcal{A}$ makes at most q_e challenge queries with an aggregate of σ_e blocks, q_v forging attempt queries with an aggregate of σ_v blocks, and q_p many permutation queries.

Proof. First, let us consider the scenario for all challenge queries with $\ell \geq r'$. Suppose at the i-th step, the adversary (say $\mathcal{A}$) submits the i-th message length and receives the random message C^i from its challenger. $\mathcal{A}$ makes successive queries to p to derive the hash value corresponding to the fixed IV and C^i. Moreover, the adversary makes several additional queries to p or p^{-1}.

Graph based Representation. Now, we draw a graph corresponding to all the challenge, permutation queries and forging attempts made by the adversary $\mathcal{A}$. A node of the graph is an n-bit state value. For a challenge or response query, we consider all the permutation inputs as nodes. Suppose the $(i-1)^{th}$ and i^{th} permutation inputs are X_{i-1}, and X_i respectively, then we draw an edge from node X_{i-1} to X_i with edge labelled as C_i, where C_i is i^{th} message injected. The starting vertex for each query (N, A, C) is defined as $IS_{N,A} \oplus (C_1 \| 0)$. Now we consider the direct permutation queries. suppose $\mathcal{A}$ makes a p query with the input X, and the output is Y (i.e., $Y = p(X)$), then we draw an edge from vertex X to vertex $Y \oplus (0 \| \lfloor X \rfloor_c)$. Similarly, if $\mathcal{A}$ makes a p^{-1} query with input $Y^\star$, and the output is $X^\star$ (i.e., $p^{-1}(Y^\star) = X^\star$), we draw an edge from $X^\star$ to $Y^\star \oplus (0 \| \lfloor X^\star \rfloor_c)$ with label 0. Essentially, the p^{-1} queries behave similar to the p queries, and we obtain a directed edge-labeled graph. This is depicted in Fig. 4.4. Thus, overall we have a graph corresponding to all the queries. All the nodes computed during the hash computation (corresponding to the challenge queries) are called "H"-nodes and all the other nodes are called "P"-nodes. So, by definition, the number of H-nodes is σ_e, the total number of primitive calls required for the hash computation of all the challenge messages. The total number of P-nodes is bounded by $(q_p + \sigma_v)$, q_p being the total number of direct p and p^{-1} calls, and σ_v being the number of p calls used in the hash computation for the verification queries.

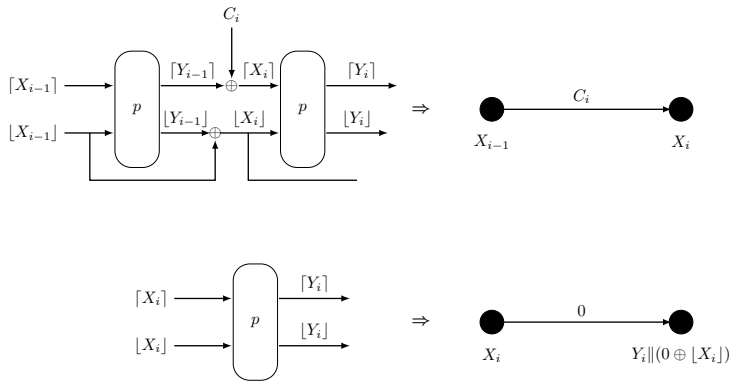


FIGURE 4.4: The graph representation: challenge and forging queries (top), direct permutation queries (bottom)

Definition and Bounding the Probability of a Bad Graph. We call a collision occurs in two nodes if their capacity values are same. Now we call such a graph *bad* if there is a collision (i) among two starting "H" nodes, or (ii) due to a "H" node and a "P" node, (iii) between two "H" nodes. Now let us try to

bound the probability that a graph is bad. For the first case, the initial state collision will reduce to a simple collision attack. This is due to the fact that the nonce and associated data are chosen by the adversary. Hence, this probability can be bounded by $q_p^2/2^{c'}$. For case (ii) and (iii), such a collision will occur with probability at most $\frac{1}{2^c - q_p}$, and the number of possible choices of H nodes and P nodes are σ_e and $(q_p + \sigma_v)$ respectively. Thus, the probability that a graph is *bad* can be bounded by $(\frac{q_p^2}{2^{c'}} + \frac{(q_p+\sigma_v)\sigma_e}{2^c} + \frac{\sigma_e^2}{2^c})$.

Bounding 2PI+ Security for A Good Graph. It is easy to see that if a graph is not *bad*, then we do not have any forgeries, except for random hash value matching, which can be bounded by $\frac{q_v}{2^{n-k}}$.

Combining everything together, the lemma follows.

□

Note that to extend the analysis for shorter challenge queries with $\ell \geq r'$ we need a domain separator at the end (adding 1 at the capacity). This is to resist an attack by guessing the random ciphertext and transferring a collision attack into a 2PI+ attack.

Interpretation of Lemma 4.3. As we can see, the dominating terms in the bound are $q_p^2/2^{c'}$ and $\sigma_e^2/2^c$. Now, as the value of q_p is usually much greater than the value of σ_e, one has the leverage of using $c < c'$ until $q_p^2/2^{c'} = \sigma_e^2/2^c$, which in turn implies $r > r'$, or in other words, greater absorption rate for the ciphertext.

4.5 ISAP+: A Throughput-Efficient Variant of ISAP

In this section, we describe the ISAP+ family of NAEAD mode by instantiating EtHM with a sponge based PRF and the hybrid sponge hash and ultimately come up with the complete specification details of ISAP+.

4.5.1 Specification of ISAP+

Let n, k, r, r' and r_0 be five positive integers satisfying $1 < r, r', r_0 < n$, and IV_{KE}, IV_{KA} and IV_A be three $(n-k)$-bit binary numbers. We call the last three numbers as the initialisation vectors. Let $c = n - r, c' = n - r'$ and $c_0 = n - r_0$. Let p be an n-bit public permutation. The authenticated encryption module of ISAP+ uses a secret key $K \in \{0,1\}^k$, receives a nonce $N \in \{0,1\}^k$, an associated data $A \in \{0,1\}^*$ and a message $M \in \{0,1\}^*$ as inputs, and returns a ciphertext $C \in \{0,1\}^{|M|}$ and a tag $T \in \{0,1\}^k$. The verified decryption module uses the same

Algorithm 4 Formal specification of the authenticated encryption and the verified decryption algorithms of ISAP+

Algorithm ISAP+.AE$_K(N, A, M)$

1. $C \leftarrow$ ISAP+.Enc/Dec(K, N, M)
2. $T \leftarrow$ ISAP+.Auth(K, N, A, C)
3. **return** (C, T)

Algorithm ISAP+.Auth(K, N, A, C)

1. $A_1 \cdots A_a \twoheadleftarrow_{r'} A$
2. **if** $|C| < r'$ **then**
3. $\quad C_1 \leftarrow C \| 10^{r'-|C|}$
4. **else if** $|C| = r'$ **then**
5. $\quad C_1 \leftarrow C$
6. $\quad C_2 \leftarrow 10^r$
7. **else**
8. $\quad C_1 \leftarrow \lceil C \rceil_{r'}$
9. $\quad C_2 \cdots C_\ell \twoheadleftarrow_r \lfloor C \rfloor_{|C|-r'}$
10. $S \leftarrow N \| IV_A$
11. **for** $i = 1$ **to** a
12. $\quad S \leftarrow p(S) \oplus (A_i \| 0^{c'})$
13. $S \leftarrow p(S) \oplus (C_1 \| 0^{c'}) \oplus 0^{n-1}1$
14. **for** $i = 2$ **to** ℓ
15. $\quad S \leftarrow p(S) \oplus (C_i \| 0^c) \oplus 0^r \| \lfloor S \rfloor_c$
16. $S \leftarrow p(S) \oplus 0^r \| \lfloor S \rfloor_c$
17. $S \leftarrow$ (ISAP+.RK$(K, \lceil S \rceil_k, 0, k)) \| \lfloor S \rfloor_{n-k}$
18. **if** $|C| < r'$ **then**
19. $\quad S \leftarrow S \oplus (0^{n-1} \| 1)$
20. **return** $T \leftarrow \lceil p(S) \rceil_k$

Algorithm ISAP+.VD$_K(N, A, C, T)$

1. $T' \leftarrow$ ISAP+.Auth(K, N, A, C)
2. **if** $T = T'$ **then**
3. $\quad$ **return** ISAP+.Enc/Dec(K, N, C)
4. **else**
5. $\quad$ **return** $\perp$

Algorithm ISAP+.Enc/Dec(K, N, X)

1. $X_1 \cdots X_\ell \twoheadleftarrow_{r'} X$
2. $S \leftarrow N \| ($ISAP+.RK$(K, N, 1, n-k))$
3. **for** $i = 1$ **to** ℓ
4. $\quad S \leftarrow p(S)$
5. $\quad Y_i \leftarrow \lceil S \rceil_{r'} \oplus X_i$
6. $Y \leftarrow \lceil Y_1 \| \cdots \| Y_\ell \rceil_{|X|}$
7. **return** Y

Algorithm ISAP+.RK$(K, X, flag, z)$

1. $IV \leftarrow (flag = 1)?\ IV_{KE} : IV_{KA}$
2. $X_1 \cdots X_w \leftarrow_{r_0} X$
3. $S \leftarrow p(K \| IV)$
4. **for** $i = 1$ **to** $(w - 1)$
5. $\quad S \leftarrow p((\lceil S \rceil_{r_0} \oplus X_i) \| \lfloor S \rfloor_{n-r_0})$
6. $S \leftarrow p((\lceil S \rceil_{|X_w|} \oplus X_w) \| \lfloor S \rfloor_{n-|X_w|})$
7. **return** $\lceil S \rceil_z$

secret key K and receives a nonce $N \in \{0,1\}^k$, an associated data $A \in \{0,1\}^*$, a ciphertext $C \in \{0,1\}^*$ and a tag $T \in \{0,1\}^\tau$ as inputs. In case of successful verification, it returns a message $M \in \{0,1\}^{|C|}$. In case the verification fails, it returns $\perp$. Both modules use a sub-module named re-keying (RK). The complete specification of ISAP+ is provided in Algorithm 4. The pictorial representation of the same is provided in Figures 4.5, 4.6, and 4.7.

Viewing ISAP+ as an Instantiation of EtHM. It is easy to that ISAP+ can be viewed as an instantiation of EtHM where the hash function H^p is given by the

feed-forward variant of sponge hash as depicted in Figure 4.7 and the FIL-VOL keyed function F_k^p is described as follows:

- When $flag = 1$ (i.e., inside encryption module), F_k^p involves the rekeying function with $(n-k)$-bit output followed by p calls as depicted in Figures 4.5 and 4.6. The inputs are the nonce N, $flag = 1$ and a parameter ℓ which represents the message length. The number of p calls is equal to the number of r-bit message blocks.
- When $flag = 0$ (i.e., inside authentication module), F_k^p involves only the rekeying function with k-bit output as depicted in Figure 4.5. The inputs are the k most significant bits of the hash output, $flag = 0$ and a parameter $\ell = k$.

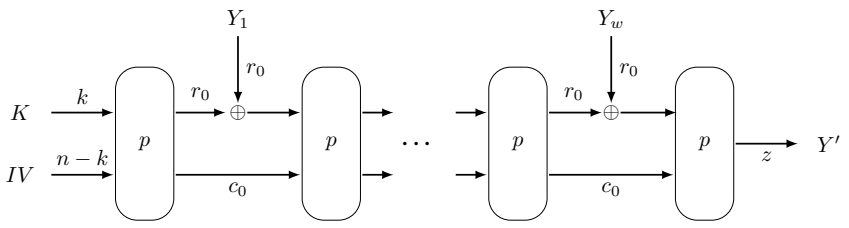


FIGURE 4.5: Re-keying module of ISAP+ on a w-bit input Y

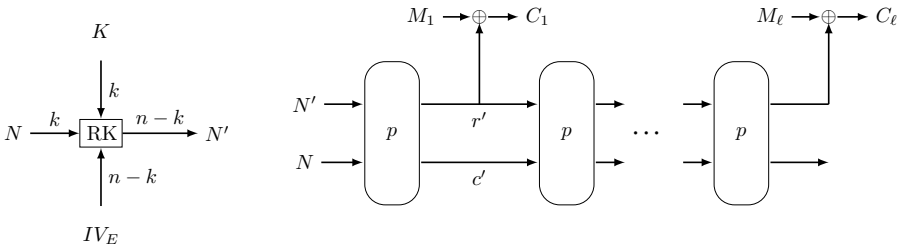


FIGURE 4.6: Encryption module of ISAP+ for ℓ block message

4.5.2 Design Rationale

In this section, we'll try to highlight and explain the main points regarding what motivated the design of EtHM, and in particular, ISAP+.

Improved Rate for Ciphertext Processing in the Hash. As we move from collision security to 2PI+ security at the ciphertext absorption phase of the authentication module, we achieve the same security with a smaller capacity size, which allows us to use a larger rate size for ciphertext absorption.

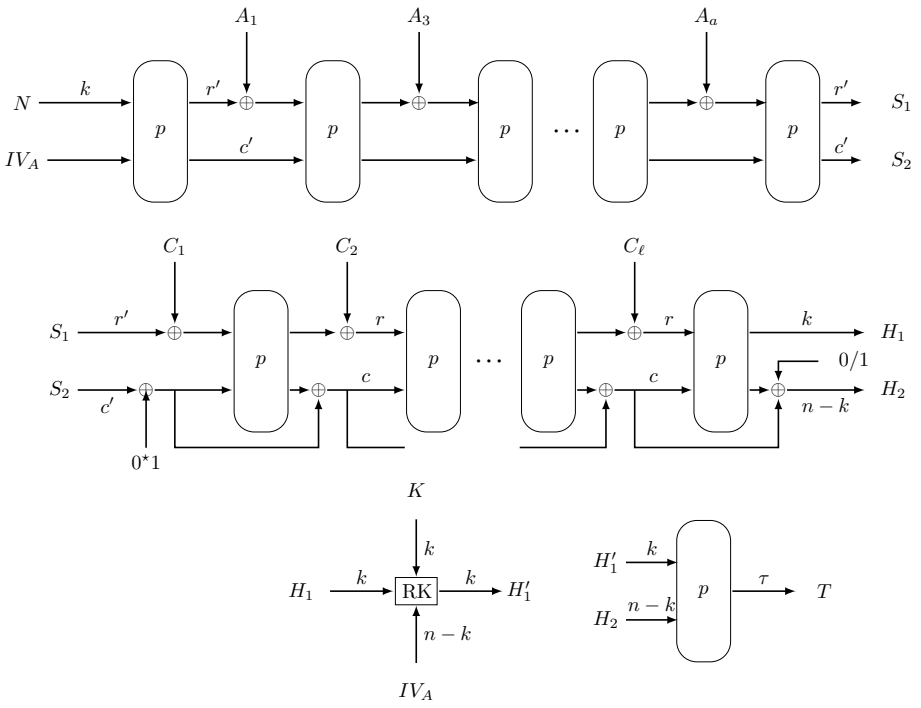


FIGURE 4.7: Authentication module of ISAP+ for a block associated data and ℓ block message

Last Domain Separator. The last domain separator is crucial to domain separate the short and long messages. Without this domain separator, we can have a forgery with one encryption query which consists of a message that is less than one block in length and the corresponding forging attempt which consists of more than one ciphertext block. As a result, a separator bit, applied to the capacity just before the last permutation call, allows us to differentiate these two cases, and ensure that the input to the last permutation is distinct for each of the two queries, which in turn prevents the attack.

4.5.3 Recommended Instantiations

We recommend one primary instance of ISAP+, denoted by ISAP+-A-128, which is instantiated with 12-round ASCON-p for all the permutation calls (i.e., each of s_H, s_B, s_E, and s_K is equal to 12). To showcase the efficiency of ISAP+, we've also implemented three other instances, comparable with the corresponding ISAP instances, as mentioned below in Table 4.3. Blue coloured entry is our primary recommendation.

TABLE 4.3: Recommended parameter values of ISAP+

Instances	Permutation	Bit Size of				No. of Rounds			
		n	r'	r	r_0	s_H	s_B	s_E	s_K
ISAP+-A-128	ASCON-p	320	64	128	1	12	12	12	12
ISAP+-A-128A	ASCON-p	320	64	128	1	12	1	6	12
ISAP+-K-128	KECCAK-p[400]	400	144	208	1	20	12	12	12
ISAP+-K-128A	KECCAK-p[400]	400	144	208	1	16	1	8	8

4.5.4 Security of ISAP+

In this subsection, we analyse the NAEAD security of ISAP+. We show that our design follows that paradigm, and hence we can adapt its security result, and the security of ISAP+ follows. Formally, we prove the following theorem.

Theorem 4.4 (NAEAD Security of ISAP+)**.** *For all deterministic nonce-respecting non-repeating query making adversary $\mathcal{A}$ of ISAP+ which can make at most q_e encryption queries of a total of maximum σ_e blocks, q_v forging queries and q_p primitive queries to p and its inverse, the NAEAD advantage of $\mathcal{A}$ can be bounded by*

$$\mathbf{Adv}^{\mathsf{ISAP+}}_{NAEAD}(\mathcal{A}) \leq \frac{\sigma_e^2 + \sigma_e q_p + q_p^2}{2^{c'}} + \frac{\sigma_e^2 + \sigma_e q_p + \sigma_e \sigma_v}{2^c} + \frac{\binom{q_p}{k}}{2^{\tau(k-1)}} + \frac{\binom{q_p}{k}}{2^{(n-k)(k-1)}} + \frac{q q_p}{2^n} + \frac{2k q_v + q_p}{2^k} + \frac{q_p + q_v}{2^{n-k}} + \frac{q_v}{2^\tau}.$$

Proof. The proof follows directly from Theorem 4.1, as we bound the two terms $\mathbf{Adv}^F_{\mathsf{PRF}}(\mathcal{B}_1)$ and $\mathbf{Adv}^{\lfloor H \rfloor_c}_{\mathsf{2PI+}}(\mathcal{B}_2)$ for ISAP+.

- To bound the first term, we observe that the key can be randomly guessed with probability $\frac{q_p}{2^k}$. Also, the state after re-keying might match with an offline query with probability $\frac{q_p}{2^{n-k}}$, as the capacity part can be controlled. Otherwise the inputs of the outer sponge construction are fresh, and a collision can happen at some stage only if two construction queries have a full state collision, or a construction query has a full state collision with a primitive query. The probability of the first case can be bounded by $\sigma_e^2/2^{c'}$ and the probability of the second case can be bounded by $\sigma_e q_p/2^{c'}$. Hence we achieve the overall PRF security of F as $\left(\frac{\sigma_e q_p + \sigma_e^2}{2^{c'}} + \frac{q_p}{2^k} + \frac{q_p}{2^{n-k}}\right)$. Further details can be found in [39].

- The second term, i.e., the 2PI+ security of the feed-forward based sponge hash can be bounded by $\left(\frac{q_p^2}{2^{c'}} + \frac{(q_p+\sigma_v)\sigma_e}{2^c} + \frac{\sigma_e^2}{2^c} + \frac{q_v}{2^{n-k}}\right)$ (See Lemma 4.3, Section 4.4.2).

□

Side-Channel Resistance. ISAP+ inherits its security against side-channel leakage directly from ISAP. In [111], the authors have clearly mentioned that "There are no requirements on the implementation of the hash function H, since it processes only publicly known data." Following their argument, ISAP+ achieves the same leakage resilience as it modifies only the hash function of ISAP and retains the rest of the design as it is. Accordingly, ISAP+ will provide a similar result on the leakage resilience bound as given in [9, Theorem 1].

4.6 Hardware Implementation Details

In this section, we provide the FPGA implementation detail of the ISAP+ family. All the hardware implementation codes are written in VHDL and are mapped on Virtex 7XC7V585T using Vivado 2020.2 as the underlying tool. The detail results are provided below.

4.6.1 Round Based Implementation of ASCON-p and KECCAK-p

In this section, we describe our round-based implementation of ASCON-p and adopted implementation of KECCAK-p[400]. The implementations are simply basic round based with n-bit datapath (n is the permutation size in bits). In case of ASCON-p, p receives a 320-bit (40-byte) input and processes it in 12 cycles (12 rounds in 12 clock cycles). Hence *cpb* for p is $40/12 = 3.33$. The circuit maintains a 320-bit internal state register which is initialised with the input and then gets updated after every round. It also maintains an additional 4-bit register **Round** to store the current round number. **Round** is initialised by 0 and is incremented by one after every round. After all the rounds are executed, **Round** is again initialised to 0. The architecture executes one round of p in one clock cycle and each round consists of 3 sequential sub modules AC, $SBox$, and Lin. The state is divided into 64 5-bit nibbles. The row representation of the state divides the state into five rows (each of the rows is known as a $Word$), each consisting of 64 bits. AC adds a round specific 1-byte constant to the third row of the intermediate state. It takes two inputs, the intermediate state and the current round number. Next, the $SBox$

module applies a non-linear 5-bit sbox to each of the nibbles of the state. The *Lin* module is used for linear diffusion of the state. This module adds different rotated copies of each $Word$ (horizontally, within each $Word$) to the corresponding $Word$. We directly adopt the KECCAK-p[400] round function implementation from [112] and implement the round based architecture on our own. The Virtex 7 results of both implementations are provided in Table 4.4.

TABLE 4.4: FPGA results of ASCON-p and KECCAK-p[400]

	Slice registers	**LUTs**	**Slices**	**Clock Cycle Time (nS)**	**Frequency (MHZ)**	**Throughput (Gbps)**
ASCON-p	328	937	282	2.5	416.67	11.1
KECCAK-p[400]	415	980	283	2.6	384.61	8.55

4.6.2 Comparison Between ISAP+ and ISAP Virtex 7 Results

We compare the hardware implementation results of ISAP+ and ISAP versions using our own implementation. We would like to point out that all our implementations follow a similar architecture and ignore the overheads associated with the NIST hardware API. Also note that the throughputs for all the versions have been computed for sufficiently long inputs and the clock cycle overheads have been ignored.

Below, we provide FPGA comparison results of ISAP+ and ISAP versions. We implement both ISAP+ and ISAP using VHDL and mapped the implementation on Virtex 7XC7V585T (Vivado 2020.2). We use the same RTL approach and a basic iterative type architecture. We use a common hardware architecture for all the versions of ISAP and ISAP+. Note that we provide our own implementation for ASCON-p (as our main recommendation is based on ASCON-p), and we use the reference round function implementation of KECCAK-p[400] from [112].

The hardware implementation results of the four versions of ISAP+ are presented in Table 4.5. We also report our hardware implementation results for the four versions of ISAP. We implement them on the same platform using the same approach. The detailed results are described in Table 4.6 below. The result depicts that all the ISAP+ versions are better in throughput than the corresponding ISAP versions which in turn depicts that ISAP+ versions remain significantly better than ISAP versions with respect to throughput/area metric. Blue coloured entry in Table 4.5 is our primary recommendation.

TABLE 4.5: FPGA results of ISAP+ versions

Versions	Slice Registers	LUTs	Slices	Frequency (MHZ)	Encryption Throughput (Gbps)	Authentication Throughput (Gbps)
ISAP+-A-128	1081	1742	507	384.16	2.05	3.07
ISAP+-K-128	1423	2245	613	300	3.60	2.64
ISAP+-A-128A	1070	1728	501	384.16	4.09	3.07
ISAP+-K-128A	1412	2231	605	300	5.40	4.40

TABLE 4.6: FPGA results of ISAP versions

Versions	Slice Registers	LUTs	Slices	Frequency (MHZ)	Encryption Throughput (Gbps)	Authentication Throughput (Gbps)
ISAP-A-128	818	1550	451	400	2.13	2.13
ISAP-K-128	1143	1932	582	300	3.60	2.16
ISAP-A-128A	810	1539	449	400	4.27	2.13
ISAP-K-128A	1131	1850	574	300	5.40	2.70

4.7 Conclusion

This chapter proposes a generic framework for a permutation-based EtHM type NAEAD mode using a PRF and an unkeyed hash function with 2PI+ security. We have shown that ISAP follows the framework EtHM, and hence, its security boils down to the 2PI+ security of the underlying hash function. We propose a feed-forward variant of the sponge hash function with improved security and use it to design a new variant of ISAP that achieves improved security and that, in turn, improves the throughput of the construction. Designing new hash functions with better 2PI+ security (using either the same feed-forward technique we have used, or some other novel ideas) and applying them to obtain modes with improved security or throughput seems to be a challenging open problem.

Chapter 5

OCB+

A nonce-respecting tweakable block-cipher is the building-block for the OCB authenticated encryption mode. An XOR-Encrypt-XOR (XEX)-based TBC is used to process each block in OCB. However, XEX can provide at most birthday bound privacy security, whereas in Asiacrypt 2017, beyond-birthday-bound (BBB) forging security of OCB3 was shown in [13]. In this chapter we study how at a small cost we can construct a nonce-respecting BBB-secure tweakable block-cipher. We propose the OTBC-3 construction, which maintains a cache that can be easily updated when used in an OCB-like mode. We show how this can be used in a BBB-secure variant of OCB with some additional keys and a few extra block-cipher calls but roughly the same amortised rate. [1]

5.1 Introduction

Beginning with the formalisation by Katz and Yung [114] and Bellare and Namprempre [107, 108], and the constructions by Jutla [1, 2], the practical significance of NAEAD modes has been accepted in the community, and over the last decade or so the design and analysis of NAEAD modes has been a very active area of research in symmetric-key cryptography. NAEAD is commonly built as a mode of operation of a block-cipher. However, there is often an inherent limitation on the security caused by the birthday paradox on the input or output of a block-cipher, which ensures only $(n/2)$-bit security of NAEAD if a block-cipher with n-bit blocks is used. The $(n/2)$-bit security is commonly referred to as birthday-bound (BB)

[1] When this work was in the process of being published, a concurrently published work [113] proposed a construction named XOCB, which is a modification of OCB3 as well, and achieve BBB privacy at rate one.

security. Possible solutions to break this barrier exist, i.e., NAEAD with beyond-birthday-bound (BBB) security. However, they come with an extra computational cost.

One way to get around this obstacle is to use a tweakable block-cipher (TBC) as the underlying primitive instead of classical block-ciphers. A TBC was formalised by Liskov, Rivest and Wagner [115, 116], and it has an extra t-bit tweak input to provide variability, i.e., it provides a family of 2^t independent block-ciphers indexed by the tweak. Starting from the early Hasty Pudding Cipher [117], many TBC designs have been proposed, including Threefish (in Skein [118]), Deoxys-BC [119], Joltik-BC [120], and KIASU-BC from the TWEAKEY framework [121], and Scream [122], where the last four schemes were submitted to CAESAR (Competition for Authenticated Encryption: Security, Applicability, and Robustness) [123]. We also see other examples including SKINNY [124, 125], QARMA [126], CRAFT [127], the TBCs in the proposals for the NIST lightweight cryptography project, OPP [128] for permutation-based instantiations of OCB3 that uses a (tweakable) Even-Mansour construction, and a construction by Naito [129].

One of the most popular TBC-based NAEAD schemes is OCB. There are three main variants of OCB. The first, now called OCB1 (2001) [130], was motivated by Charanjit Jutla's IAPM [1, 2]. A second version, now called OCB2 (2004) [131, 132], added support for associated data (AD) and redeveloped the mode using the idea of a tweakable block-cipher. Later OCB2 was found to have a disastrous bug [133, 134]. The final version of OCB, called OCB3 (2011) [12], corrected some missteps taken with OCB2 and achieved the best performance yet. OCB3 is simple, parallelisable, efficient, provably secure with BB security, and its security is well analysed [135–137]. It is specified in RFC 7253 [138] and was selected for the CAESAR final portfolio.

In recent times, OCB has been analysed in much detail from various perspectives. A block-cipher-based NAEAD scheme OTR and its TBC-based counterpart $\mathbb{OTR}$ were designed by Minematsu [139] which improve OCB by removing the necessity of the decryption routine of the underlying block-cipher or TBC (this property is often called as the inverse-freeness). Bhaumik and Nandi [13] showed that when the number of encryption query blocks is not more than birthday-bound (an assumption without which the privacy guarantee of OCB3 disappears), even an adversary making forging attempts with the number of blocks in the order of $2^n/\ell_{\text{MAX}}$ (n being the block-size and ℓ_{MAX} being the length of the longest block) may fail to break the integrity of OCB3. Zhang et al. [140, 141] described a new

notion, called plaintext or ciphertext checksum (PCC), which is a generalisation of plaintext checksum (used to generate the tag of OCB), and proved that all authenticated encryption schemes with PCC are insecure in the INT-RUP security model. Then they fixed the weakness of PCC, and described a new approach called intermediate (parity) checksum (I(P)C for short). Based on the I(P)C approach, they provided two modified schemes OCB-IC and OCB-IPC to settle the INT-RUP of OCB in the nonce-misuse setting. They proved that OCB-IC and OCB-IPC are INT-RUP up to the birthday bound in the nonce-misuse setting if the underlying tweakable block-cipher is a secure mixed tweakable pseudorandom permutation (MTPRP). The notion of MTPRP is defined by the authors (Zhang et al. [140, 141]) which we omit in this thesis as we don't need that notion for our purpose. The security bound of OCB-IPC is proved to be tighter than OCB-IC. To improve their speed, they utilised a "prove-then-prune" approach: prove security and instantiate with a scaled-down primitive (e.g., reducing rounds for the underlying primitive invocations). Bao et al. [142] introduced a scheme called XTX*, based on previous tweak extension schemes for TBCs, and defined ZOCB and ZOTR for nonce-based authenticated encryption with associated data. While ΘCB and $\mathbb{OTR}$ have an independent part to process AD, their schemes integrated this process into the encryption part of a plaintext by using the tweak input of the TBC, and thus achieved full absorption and full parallelisability simultaneously. Liénardy et al. [143] showed the vulnerability of both privacy and authenticity of OCB3 against short nonces.

OCB has also found its place in other domains of cryptology like lightweight cryptology and quantum cryptology. Chakraborti et al. [144] proposed a light-weight authenticated encryption (AE) scheme, called Light-OCB, which can be viewed as a lighter variant of OCB as well as a faster variant of LOCUS-AEAD [145] which was a round 2 candidate of the NIST lightweight cryptography project. Bhaumik et al. [146] proposed a new rate-one parallelisable mode named QCB inspired by TAE and OCB and prove its security against quantum superposition queries.

There are two limitations on OCB that we would like to emphasise. The first is that OCB's security crucially depends on the encrypting party not repeating a nonce. The mode should never be used in situations where that can't be assured; one should instead employ a misuse-resistant AE scheme [147]. These include AES-GCM-SIV [148, 149], COLM, and Deoxys-II. A second limitation of OCB is its birthday-bound degradation in provable security. This limitation implies that, given OCB's 128-bit block-size, one must avoid operating on anything near 2^{64}

blocks of data. The RFC on OCB [138] asserts that a given key should be used to encrypt at most 2^{48} blocks (4 petabytes), including the associated data. Practical AE modes that avoid the birthday-bound degradation in security are now known [123, 149–152].

5.1.1 Contributions

In this chapter we explore ways of designing an offset-based tweakable block-cipher that can be used to obtain an OCB-like authenticated encryption mode with better security guarantees. First, we show that when using an n-bit nonce (where n is the width of the block-cipher), it is difficult to go beyond the birthday-bound if we use the same offset to mask the input and the output (OTBC-0). Next, we show that if we take fully independent offsets for masking inputs and outputs for each message, we get full security in the nonce-respecting scenario (OTBC-1); however, this does not fit well in the OCB-like mode, because new additional random-function calls are needed to process each message block.

We proceed to introduce the notion of *updatable offsets*, and explain why TBCs with updatable offsets are well-suited to build an OCB-like mode. Then we build a simple TBC with updatable offsets (OTBC-2), and give a birthday-attack on it that demonstrates that such a construction is not sufficient to get beyond-birthday security for the OCB. Finally, we introduce the notion of offsets that are not updatable by themselves, but are efficiently computable from updatable *caches*. As the most important technical contribution of the chapter, we instantiate a TBC with this property (OTBC-3) and show that it achieves a beyond-birthday TPRP security in the number of nonces queried, as long as the maximum length of each message (i.e., the maximum number of times each block is used) is not very high. Additionally, we also show that OTBC-3 achieves at least security up to the birthday-bound even when the nonce is misused and inverse queries are allowed.

Finally, we use OTBC-3 to design an authenticated encryption mode called OCB+, which is beyond-birthday secure in both privacy and authenticity. We argue how the privacy bound follows from our security proof of OTBC-3, while the authenticity can be proved in the exact same way as in [13]. OCB+ uses nine random function calls for processing each nonce, so its rate is approximately $\sigma/(\sigma + 9q)$, where σ is the total number of blocks including messages and associated data, and q is the number of distinct nonces. When the messages are sufficiently long, this rate comes close to 1, making this as efficient as OCB3, but with a BBB security guarantee.

5.2 Preliminaries

5.2.1 TPRP, TPRP* and TSPRP Security Notions

Given a tweak-space $\mathcal{W}$, let $\mathsf{Perm}(\mathcal{W}, n)$ be the set of all functions $\tilde{\pi} : \mathcal{W} \times \{0,1\}^n \to \{0,1\}^n$ such that for any tweak $W \in \mathcal{W}$, $\tilde{\pi}(W, \cdot)$ is a permutation over $\{0,1\}^n$. Then a $\tilde{\pi}^*$ distributed uniformly at random over $\mathsf{Perm}(\mathcal{W}, n)$ will be called a *tweakable random permutation* (TRP).

Let $\mathcal{K}$ denote a key-space. Then $\tilde{E} : \mathcal{K} \times \mathcal{W} \times \{0,1\}^n \to \{0,1\}^n$ will be called a *tweakable pseudorandom permutation* (TPRP) if for a key K distributed uniformly at random over $\mathcal{K}$ and for any adversary $\mathcal{A}$ trying to distinguish $\tilde{E}_K := \tilde{E}(K, \cdot, \cdot)$ from $\tilde{\pi}^*$, $\mathbf{Adv}^{\tilde{E}_K, \tilde{\pi}^*}(\mathcal{A})$ is small. We call this game the TPRP game and denote the advantage of $\mathcal{A}$ as $\mathbf{Adv}^{\tilde{E}}_{\mathrm{TPRP}}(\mathcal{A})$ in short.

We will be more interested in a modified version of the TPRP game, where $\mathcal{A}$ is under the added restriction that no two queries can be made with the same tweak. We call this the *tweak respecting pseudorandom permutation* (TPRP*) game, and denote the corresponding advantage of $\mathcal{A}$ as $\mathbf{Adv}^{\tilde{E}}_{\mathrm{TPRP}^*}(\mathcal{A})$.

Finally, the *tweakable strong pseudorandom permutation* (TSPRP) game allows $\mathcal{A}$ to make both encryption and decryption queries to the oracle. The advantage term of $\mathcal{A}$ in a TSPRP game will be denoted $\mathbf{Adv}^{\tilde{E}}_{\mathrm{TSPRP}}(\mathcal{A})$.

5.2.2 Mirror Theory

Consider a sequence of n-bit variables $W_1, \ldots, W_t$, subject to r bi-variate equations of the form

$$W_i + W_j = \delta_{ij}.$$

Consider the graph with $W_1, \ldots, W_t$ as vertices and the bi-variate equations as weighted edges with δ_{ij} the weight between W_i and W_j. Suppose we can show that the graph is cycle-free, and that each path has a non-zero sum of weights. Let $\xi_{\max}$ be the size of the largest component of this graph. Then Mirror Theory tells us that as long as $\xi^2_{\max} \leq \sqrt{N}/\log_2 N$ and $t \leq N/12\xi^2_{\max}$, the number of solutions to the system of equations such that W_i's are all distinct is at least $(N)_t/N^r$. [153, 154]

5.3 Finding a Suitable Tweakable Block-cipher

We set out to find an offset-based Tweakable Block-cipher that could give us a beyond-birthday security bound for OCB+. The general structure of this is as follows:

$$C = \pi(M + T) + \widehat{T},$$

where the offsets T and $\widehat{T}$ are functions of the nonce $\mathcal{N}$ and the block-number i.

5.3.1 Attempt with Same Offset

The first question we asked is whether it is possible to achieve this by having $T = \widehat{T}$, i.e., adding the same offset before and after the block-cipher call, like in OCB. The most powerful version of this is to have

$$T = \widehat{T} = f(\mathcal{N}, i)$$

for some $2n$-bit-to-n-bit random function f. This we call OTBC-0, defined as

$$\mathsf{OTBC\text{-}0}(\mathcal{N}, i, M) := \pi(M + f(\mathcal{N}, i)) + f(\mathcal{N}, i).$$

This construction is shown in Figure 5.1.

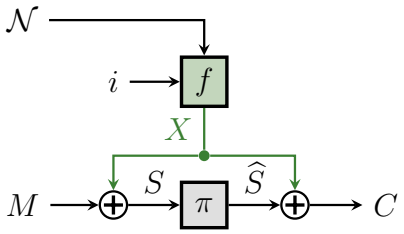


FIGURE 5.1: OTBC-0: Same offset

5.3.1.1 Birthday Attack on OTBC-0.

Unfortunately, OTBC-0 fails to give us beyond birthday-bound security. This is because for two queries with the same message, there is a collision in the ciphertext whenever there is a collision in the output of f; in addition the ciphertext-collision can also happen if the sum of the outputs of π and f collide. This shows that the collision probability at C is roughly double the collision probability in an ideal tweakable block-cipher, which can be detected in the birthday-bound. A more formal description of the attack is given below.

5.3.1.2 Attack on OTBC-0

Let the i-th nonce be N_i, and the corresponding offset and ciphertext for the message M_i be X_i and C_i respectively. We get

$$\pi(M_i + X_i) + X_i = C_i.$$

For two distinct queries (say i-th and j-th query with $i \neq j$), we get

$$\pi(M_i + X_i) + \pi(M_j + X_j) + X_i + X_j = C_i + C_j.$$

So, whenever $M_i + X_i = M_j + X_j$, we have $M_i + C_i = M_j + C_j$. We call this event E and consider it as a distinguishing event. We define an event E_1 in which $M_i + X_i = M_j + X_j$ for some $i \neq j$. From the above discussion we see that

$$E_1 \Rightarrow E.$$

Let $\mathsf{cp}(q, N)$ denote the probability of finding a collision-pair in a uniform random sample of size q from a population of size N. Thus,

$$\Pr_{\mathcal{O}_1}[E_1] = \mathsf{cp}(q, N).$$

We also have

$$\Pr_{\mathcal{O}_1}[E] = \mathsf{cp}(q, N) + \Pr_{\mathcal{O}_1}[E \mid \neg E_1] \cdot (1 - \mathsf{cp}(q, N)).$$

Now, when E_1 does not happen, all inputs of π must be distinct. Let us denote the inputs and output of π for the ith query by S_i and $\widehat{S}_i$ respectively. Then the event E is equivalent to finding a collision-pair among the $S_i + \widehat{S}_i$ values. Given that E_1 does not happen both S_i's and $\widehat{S}_i$'s are sampled as uniformly without replacement and $(S_i)_{i\in[q]}$ is independent from $(\widehat{S}_i)_{i\in[q]}$. By using well known result [155, 156], we know that the sum of independent without-replacement samples is almost identically distributed as a uniform random sample and hence

$$\Pr_{\mathcal{O}_1}[E \mid \neg E_1] \approx \mathsf{cp}(q, N).$$

Thus, we have

$$\Pr_{\mathcal{O}_1}[E] \approx \mathsf{cp}(q, N) + (1 - \mathsf{cp}(q, N)) \cdot \mathsf{cp}(q, N).$$

On the other hand in the ideal world,

$$\Pr_{\mathcal{O}_0}(E) = \mathsf{cp}(q, N),$$

as for every distinct nonce the responses should be uniformly and independently distributed. Hence, the distinguishing advantage is around $\mathsf{cp}(q, N)(1 - \mathsf{cp}(q, N))$. Now we know that $\mathsf{cp}(q, N) = 1/2$ is attained for a $q = O(\sqrt{N})$ and hence for that choice of q, the distinguishing advantage is at least $1/4$.

5.3.2 Independent Offsets

We deduce from the preceding subsection that using the same offset above and below can never give us beyond-birthday TPRP* security for the tweakable block-cipher. We next examine the most powerful version of this possible, where the two offsets on either side of π come from two completely independent $2n$-bit-to-n-bit random functions f_1 and f_2. This we call OTBC-1, defined as

$$\mathsf{OTBC\text{-}1}(\mathcal{N}, i, M) := \pi(M + f_1(\mathcal{N}, i)) + f_2(\mathcal{N}, i).$$

This construction is shown in Figure 5.2.

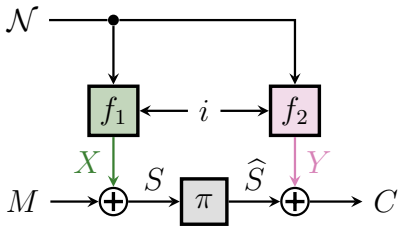


FIGURE 5.2: OTBC-1: Different random offsets

5.3.2.1 Security of OTBC-1.

As it turns out, OTBC-1 trivially achieves full TPRP* security. This is because in a tweak-respecting game, the offsets are always random and independent of all other offsets in the game, making it impossible to glean any information from the oracle responses. We formally state this as the following theorem.

Theorem 5.1. *For any TPRP* adversary $\mathcal{A}$ making q queries, we have*

$$\boldsymbol{Adv}^{\mathsf{OTBC\text{-}1}}_{\mathrm{TPRP}^*}(\mathcal{A}) = 0.$$

Proof. Let's call $(\mathcal{N}, i)$ as $\mathcal{T}$. We'll use Coefficients H Technique to bound the advantage of the adversary.

Transcript Notation. The adversary makes q encryption queries $(\mathcal{T}^1, M^1), \cdots, (\mathcal{T}^q, M^q)$ to the oracle, and receives $C^1, \cdots, C^q$ as the corresponding responses. So the query-response transcript of the adversary initially looks like $\{(\mathcal{T}^1, M^1, C^1), \cdots, (\mathcal{T}^q, M^q, C^q)\}$.

Sampling in the Ideal World. For each encryption query, the ideal oracle samples the output with replacement from $\{0,1\}^n$ uniformly at random. Once the adversary is done with all its queries, the oracle releases some additional information to the adversary. The ideal oracle samples them in the following way:

- For all $j \in [q]$, the ideal oracle samples X^j with replacement from $\{0,1\}^n$ uniformly at random.
- For all $j \in [q]$, the ideal oracle samples $\widehat{S}^j$ without replacement from $\{0,1\}^n$ uniformly at random.

The real oracle releases the corresponding true values in this additional release phase. After the additional release, the extended transcript looks like the following: $\{(\mathcal{T}^1, M^1, C^1, X^1, \widehat{S}^1), \cdots, (\mathcal{T}^q, M^q, C^q, X^q, \widehat{S}^q)\}$.

Advantage of the Adversary. For any attainable transcript τ, we get the real interpolation probability as

$$\Pr_{\mathcal{O}_1}[\tau] = \frac{1}{N^q} \cdot \frac{1}{N^q} \cdot \frac{1}{(N)_q}\,.$$

The first, second and third term in the denominator on the right hand side represents the number of choices for X, Y and $\widehat{S}$ respectively. We also get the ideal interpolation probability as

$$\Pr_{\mathcal{O}_0}[\tau] = \frac{1}{N^q} \cdot \frac{1}{N^q} \cdot \frac{1}{(N)_q}\,.$$

The first, second and third term in the denominator on the right hand side represents the number of choices for C, X and $\widehat{S}$ respectively. Thus we finally get

$$\frac{\Pr_{\mathcal{O}_1}[\tau]}{\Pr_{\mathcal{O}_0}[\tau]} = 1\,.$$

Applying H-Coefficient Technique with $\epsilon_1 = \epsilon_2 = 0$ completes the proof. □

5.3.3 Updatable Offsets

While OTBC-1 is a fully secure tweakable block-cipher, it's not very interesting to us in the context of OCB+. This is because when the same nonce is used with different block-numbers (as we need for OCB+), new calls to f_1 and f_2 are needed for each new block-number. Thus we need three primitive calls to process every block of message, which robs us of the main advantage of an OCB-like design.
This points us to the next desirable feature we need in the offsets: they should be *efficiently updatable* when we keep the nonce same and increment the block-number. We call a $2n$-bit-to-n-bit function h efficiently updatable on the second input if there is an efficiently computable function g (called the *update* function) such that for each i we have

$$h(\mathcal{N}, i+1) = g(i, h(\mathcal{N}, i)).$$

In other words, given $h(\mathcal{N}, i)$ has already been computed, $h(\mathcal{N}, i+1)$ can be computed through the update function g while bypassing a fresh call to h. (For this to make sense, of course, h should be computationally heavy and g should be much faster than h.) Note that the update function may or may not use i as an additional argument; while in this chapter we'll only consider update functions that are *stationary* (i.e., ignore the block-number i, and apply the same function at each block to get the offset for the next block), it is possible to have an update function that varies with i but still satisfies the above-discussed criteria.

5.3.3.1 The simplest updatable design.

The simplest way to design an updatable function is to call a random function f on the nonce $\mathcal{N}$ once, and then use a stationary update function to obtain the offset for each successive block-number. This can be formally defined as follows:

$$\begin{aligned} h(\mathcal{N}, 1) &= g(f(\mathcal{N})), \\ h(\mathcal{N}, i) &= g(h(\mathcal{N}, i-1)) = g^i(f(\mathcal{N})), \qquad i \geq 2. \end{aligned}$$

Using these updatable offsets with two independent random functions f_1 and f_2 for input-masking and output-masking respectively, we can define a tweakable

block-cipher OTBC-g as

$$\mathsf{OTBC\text{-}g}(\mathcal{N}, i, M) = \pi\left(M + g^i(f_1(\mathcal{N}))\right) + g^i(f_2(\mathcal{N})).$$

5.3.3.2 Instantiating OTBC-g.

In commonly used finite fields, there generally exist primitive elements that allow very fast multiplication. As an instantiation of g, we use multiplication with one such fixed primitive α. Concretely, we define the update function as

$$g(f(\mathcal{N})) = \alpha \cdot f(\mathcal{N}).$$

Thus, we use as the updatable offsets

$$T = \alpha^i \cdot f_1(\mathcal{N}), \qquad \widehat{T} = \alpha^i \cdot f_2(\mathcal{N}).$$

This gives us the construction OTBC-2, defined as

$$\mathsf{OTBC\text{-}2}(\mathcal{N}, i, M) = \pi\left(M + \alpha^i \cdot f_1(\mathcal{N})\right) + \alpha^i \cdot f_2(\mathcal{N}).$$

This construction is shown in Figure 5.3.

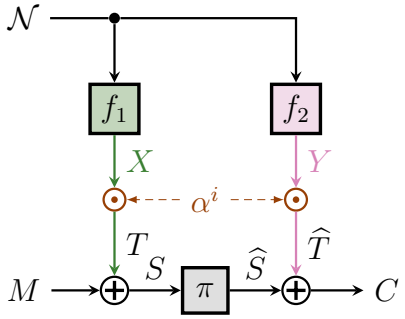


FIGURE 5.3: OTBC-2: Updatable offsets with two independent random-function calls

5.3.3.3 Attack on OTBC-2.

Unfortunately, this simple updatable function is not sufficient to give us beyond-birthday-bound security. This is because since the update function is linear and publicly known, we can make queries such that successive message blocks under the same nonce follow the update relation, which forces the successive S blocks to also conform to the update relation. Thus, one collision on S between two

different nonces ensures that successive blocks also see an S-collision, which can be exploited in a distinguishing attack. This we state as the following theorem.

Theorem 5.2. *There exists a distinguisher $\mathcal{A}$ querying with q nonces and L blocks under each nonce with $L \geq 12$ in a TPRP* game against OTBC-2 such that*

$$\boldsymbol{Adv}^{\mathsf{OTBC\text{-}2}}_{TPRP^*}(\mathcal{A}) \geq \Omega\left(\frac{q^2L^2}{N}\right).$$

Proof. $\mathcal{A}$ picks q distinct nonces $\mathcal{N}^{(1)}, \ldots, \mathcal{N}^{(q)}$, and q distinct starting messages $M_1^{(1)}, \ldots, M_1^{(q)}$. For each $j \in [q]$ it makes L queries $(\mathcal{N}^{(j)}, 1, M_1^{(j)}), \ldots, (\mathcal{N}^{(j)}, L, M_L^{(j)})$, such that for each $i \in [L]$ we have

$$M_i^{(j)} := \alpha^{i-1} \cdot M_1^{(j)}.$$

This ensures that we have

$$\begin{aligned} S_i^{(j)} &:= M_i^{(j)} + \alpha^i \cdot X^{(j)} \\ &= \alpha^{i-1} \cdot M_1^{(j)} + \alpha^i \cdot X^{(j)} \\ &= \alpha^{i-1} \cdot \left(M_1^{(j)} + \alpha \cdot X^{(j)}\right) = \alpha^{i-1} \cdot S_1^{(j)}, \end{aligned}$$

where $X^{(j)} := f_1(\mathcal{N}^{(j)})$ and $S_i^{(j)}$ is the input of π on the query $(\mathcal{N}^{(j)}, i, M_i^{(j)})$.

Input Collision. Suppose we have distinct $j, j' \in [q]$ and some $i, i' \in [L-1]$ (not necessarily distinct), such that $S_i^{(j)} = S_{i'}^{(j')}$. Then we have

$$\begin{aligned} S_{i+1}^{(j)} = \alpha^i \cdot S_1^{(j)} &= \alpha \cdot \left(\alpha^{i-1} \cdot S_1^{(j)}\right) \\ &= \alpha \cdot S_i^{(j)} \\ &= \alpha \cdot S_{i'}^{(j')} \\ &= \alpha \cdot \left(\alpha^{i-1} \cdot S_1^{(j')}\right) = \alpha^i \cdot S_1^{(j')} = S_{i'+1}^{(j')}. \end{aligned}$$

In other words, a collision on two input blocks in two different nonces forces a collision on the next block as well (and, in fact, this dominoes into all successive blocks till one of the block-numbers reach L). $\mathcal{A}$ can use this property to mount the distinguishing attack.

Distinguishing Event. $\mathcal{A}$ searches for a pair of distinct $j, j' \in [q]$ and $i, i' \in [L-2]$ (not necessarily distinct) such that

$$C_{i+2}^{(j)} + C_{i'+2}^{(j')} = \alpha \cdot \left(C_{i+1}^{(j)} + C_{i'+1}^{(j')}\right) = \alpha^2 \cdot \left(C_i^{(j)} + C_{i'}^{(j')}\right).$$

If such j, j', i, i' exist, $\mathcal{A}$ outputs 1, else it outputs 0.
We note that in the real world, whenever $S_i^{(j)} = S_{i'}^{(j')}$, we have $\widehat{S}_i^{(j)} = \widehat{S}_{i'}^{(j')}$, which implies that

$$C_i^{(j)} + C_{i'}^{(j')} = \alpha^i \cdot Y^{(j)} + \alpha^{i'} \cdot Y^{(j')}.$$

From the above discussion, we know that $S_i^{(j)} = S_{i'}^{(j')}$ forces the collisions $S_{i+1}^{(j)} = S_{i'+1}^{(j')}$ and $S_{i+2}^{(j)} = S_{i'+2}^{(j')}$. The first of these implies that

$$\begin{aligned} C_{i+1}^{(j)} + C_{i'+1}^{(j')} &= \alpha^{i+1} \cdot Y^{(j)} + \alpha^{i'+1} \cdot Y^{(j')} \\ &= \alpha \cdot \left(\alpha^i \cdot Y^{(j)} + \alpha^{i'} \cdot Y^{(j')}\right) \\ &= \alpha \cdot \left(C_i^{(j)} + C_{i'}^{(j')}\right), \end{aligned}$$

and similarly the second implies that

$$C_{i+2}^{(j)} + C_{i'+2}^{(j')} = \alpha \cdot \left(C_{i+1}^{(j)} + C_{i'+1}^{(j')}\right) = \alpha^2 \cdot \left(C_i^{(j)} + C_{i'}^{(j')}\right).$$

Thus, the collision $S_i^{(j)} = S_{i'}^{(j')}$ for distinct $j, j' \in [q]$ and $i, i' \in [L-2]$ is enough to trigger the distinguishing event.
In the ideal world, this event requires two collisions, each with probability $1/N$. Since there are $q(q-1)/2$ choices for j, j' and $(L-2)^2$ choices for i, i', we have

$$\Pr_{\mathcal{O}_0}[\mathcal{A} \text{ outputs } 1] \approx \binom{q}{2} \cdot \frac{(L-2)^2}{N^2}.$$

But in the real world, this only requires one collision, as the other is automatically enforced. Thus,

$$\Pr_{\mathcal{O}_1}[\mathcal{A} \text{ outputs } 1] \approx \binom{q}{2} \cdot \frac{(L-2)^2}{N}.$$

This completes the proof of the claimed lower bound on the advantage of $\mathcal{A}$. □

5.3.4 Offsets with Updatable Caches

To get around this problem, we observe that in order to use an offset-based tweakable block-cipher in OCB+, we don't really need it to be updatable; it is enough

for it to maintain a small and updatable hidden state or *cache*, such that the offsets are efficiently computable from the cache. Letting ψ denote the *caching function*, g the update function as before, h the offset-generating function, and φ the cache-to-offset function, we have

$$\psi(\mathcal{N}, i+1) = g(i, \psi(\mathcal{N}, i)), \qquad h(\mathcal{N}, i) = \varphi(\psi(\mathcal{N}, i)).$$

Again, for this to make sense, h and ψ should be computationally heavy when computed from scratch, while g and φ should be much faster.

5.3.4.1 Updatable Caches, Non-updatable Offsets.

To avoid the kind of attack that we found on OTBC-2, we want to design a tweakable block-cipher with offsets which are not themselves updatable, but are efficiently computable from updatable caches. This makes the offsets more independent, while still giving us a means of updating them efficiently at a small additional cost.

One simple way to achieve this is to use two independent random functions f_1 and f_2 on the nonce, put the outputs in the cache as two different *branches*, and use two different update functions g and g' on the two branches; the offset can then be generated as the sum of the two branches. This can be formally defined as follows:

$$\begin{aligned}
\psi(\mathcal{N}, 1) &= (g(f_1(\mathcal{N})), g'(f_2(\mathcal{N}))),\\
\psi(\mathcal{N}, i) &= [g, g'](\psi(\mathcal{N}, i-1)) = (g^i(f_1(\mathcal{N})), g'^i(f_2(\mathcal{N}))), \qquad i \geq 2,\\
\varphi(x, y) &= x + y,\\
h(\mathcal{N}, i) &= g^i(f_1(\mathcal{N})) + g'^i(f_2(\mathcal{N})) = \varphi(\psi(\mathcal{N}, i)),
\end{aligned}$$

where $[g, g']$ denotes the two-input function that applies g to the first input and g' to the second input. Note that $h(\mathcal{N}, i)$ is not efficiently computable from $h(\mathcal{N}, i-1)$ without accessing the cache $\psi(\mathcal{N}, i-1)$, which makes the offsets themselves non-updatable in the absence of the cache. Using these offsets we can define a tweakable block-cipher OTBC-gg' as

$$\mathsf{OTBC\text{-}gg'}(\mathcal{N}, i, M) = \pi\left(M + g^i(f_1(\mathcal{N})) + g'^i(f_2(\mathcal{N}))\right) + f_3(\mathcal{N}) + g^i(\pi(0^n)).$$

where f_3 is a third independent random-function. Note that we do not bother to use the non-updatable updates for masking the output, because $\mathcal{A}$ can make

only encryption queries, and thus cannot exploit the same weakness in the output-masking.

5.3.4.2 Instantiating OTBC-gg'.

As the main contribution of this section, we propose a concrete instantiation of OTBC-gg' and analyse its security. As before we keep the field-multiplication by α as g, and for g' we use field-multiplication by α^2. The resulting tweakable block-cipher, called OTBC-3, is defined as

$$\mathsf{OTBC\text{-}3}(\mathcal{N}, i, M) = \pi\left(M + \alpha^i \cdot f_1(\mathcal{N}) + \alpha^{2i} \cdot f_2(\mathcal{N})\right) + f_3(\mathcal{N}) + \alpha^i \cdot \pi(0^n).$$

This construction is shown in Figure 5.4.

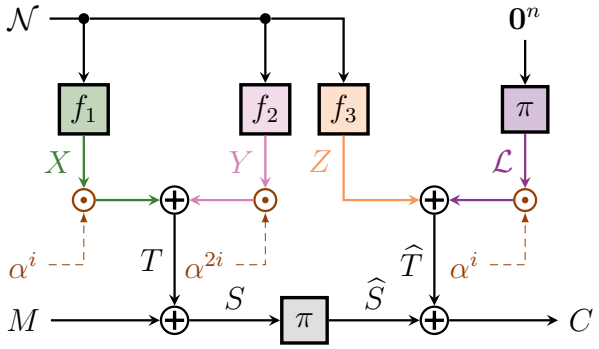


FIGURE 5.4: OTBC-3: Offsets with updatable caches using three independent random-function calls

Algorithm 5 $\mathsf{OTBC\text{-}3}^{f_1,f_2,f_3,\pi}(\mathcal{N}, i, M)$

1: $T \leftarrow \alpha^i f_1(\mathcal{N}) \oplus \alpha^{2i} f_2(\mathcal{N})$
2: $\widehat{T} \leftarrow f_3(\mathcal{N}) \oplus \alpha^i \pi(0^n)$
3: $S \leftarrow M \oplus T$
4: $\widehat{S} \leftarrow \pi(S)$
5: $C \leftarrow \widehat{S} \oplus \widehat{T}$
6: **return** C

5.3.5 TPRP* Security Analysis of OTBC-3

Consider a distinguisher $\mathcal{A}$ making σ encryption queries to OTBC-3 with q distinct nonces and $\ell^{(j)} \leq L$ block-numbers $1, \ldots, \ell^{(j)}$ for the j-th nonce for each $j \in [q]$. Then we have the following result.

Theorem 5.3. *As long as $\sigma \leq N/n^2L^2$, we have*

$$\boldsymbol{Adv}^{\mathsf{OTBC\text{-}3}}_{TPRP^*}(\mathcal{A}) \leq \frac{n\sigma L}{N}.$$

Proof. In this proof, we'll use the following lemma.

Lemma 5.4. *For some $r \geq 2$ and $2r$ numbers $i_1, i'_1, \ldots, i_r, i'_r < N$ such that $i_j \neq i'_j$ for each $j \in [r]$, define*

$$\mathbf{B}_r = \begin{bmatrix} \alpha^{i_1} & \alpha^{2i_1} & \alpha^{i'_2} & \alpha^{2i'_2} & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & \alpha^{i_2} & \alpha^{2i_2} & \alpha^{i'_3} & \alpha^{2i'_3} & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \alpha^{i_3} & \alpha^{2i_3} & \alpha^{i'_4} & \alpha^{2i'_4} & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & \alpha^{i_{r-1}} & \alpha^{2i_{r-1}} & \alpha^{i'_r} & \alpha^{2i'_r} \\ \alpha^{i'_1} & \alpha^{2i'_1} & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & \alpha^{i_r} & \alpha^{2i_r} \end{bmatrix}.$$

Then $\mathbf{B}_r$ is at least of rank r.

Proof. First we observe that $\mathbf{B}_2$ is of rank 2 since, for the leftmost 2×2 submatrix of $\mathbf{B}_2$, we have

$$\begin{vmatrix} \alpha^{i_1} & \alpha^{2i_1} \\ \alpha^{i'_1} & \alpha^{2i'_1} \end{vmatrix} = \alpha^{i_1+2i'_1} + \alpha^{2i_1+i'_1} = \alpha^{i_1+i'_1}(\alpha^{i_1} + \alpha^{i'_1}) \neq 0.$$

(This also holds for the rightmost 2×2 submatrix.) Next we observe that $\mathbf{B}_3$ is of rank 3 since, for the leftmost 3×3 submatrix of $\mathbf{B}_3$, we have

$$\begin{vmatrix} \alpha^{i_1} & \alpha^{2i_1} & \alpha^{i'_2} \\ 0 & 0 & \alpha^{i_2} \\ \alpha^{i'_1} & \alpha^{2i'_1} & 0 \end{vmatrix} = \alpha^{i_2} \begin{vmatrix} \alpha^{i_1} & \alpha^{2i_1} \\ \alpha^{i'_1} & \alpha^{2i'_1} \end{vmatrix} = \alpha^{i_1+i'_1+i_2}(\alpha^{i_1} + \alpha^{i'_1}) \neq 0.$$

(This also holds for any of the three other contiguous 3×3 submatrices of $\mathbf{B}$.) This leaves the case $r \geq 4$. We consider two cases, based on whether r is even or odd. First, suppose $r = 2m$. Then we look at the $2m \times 2m$ submatrix $\mathbf{H}$ of $\mathbf{B}_r$

consisting of the columns $4p-1$ and $4p$ for each $p \in [m]$. Thus,

$$\mathbf{H} = \begin{bmatrix} \alpha^{i'_2} & \alpha^{2i'_2} & 0 & 0 & \cdots & 0 & 0 \\ \alpha^{i_2} & \alpha^{2i_2} & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \alpha^{i'_4} & \alpha^{2i'_4} & \cdots & 0 & 0 \\ 0 & 0 & \alpha^{i_4} & \alpha^{2i_4} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \alpha^{i'_{2m}} & \alpha^{2i'_{2m}} \\ 0 & 0 & 0 & 0 & \cdots & \alpha^{i_{2m}} & \alpha^{2i_{2m}} \end{bmatrix}.$$

We observe that $\mathbf{H}$ is a block-diagonal matrix of the form

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_1 & \mathbf{0}_{2\times 2} & \cdots & \mathbf{0}_{2\times 2} \\ \mathbf{0}_{2\times 2} & \mathbf{H}_2 & \cdots & \mathbf{0}_{2\times 2} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{2\times 2} & \mathbf{0}_{2\times 2} & \cdots & \mathbf{H}_m \end{bmatrix},$$

where for each $p \in [m]$,

$$\mathbf{H}_p = \begin{bmatrix} \alpha^{i'_{2p}} & \alpha^{2i'_{2p}} \\ \alpha^{i_{2p}} & \alpha^{2i_{2p}} \end{bmatrix}.$$

Thus, $|\mathbf{H}_p| = \alpha^{i_{2p}+i'_{2p}}(\alpha^{i_{2p}} + \alpha^{i'_{2p}}) \neq 0$ for each $p \in [m]$, and

$$|\mathbf{H}| = |\mathbf{H}_1| \cdot |\mathbf{H}_2| \cdot \ldots \cdot |\mathbf{H}_m| \neq 0,$$

which shows that $\mathbf{H}$ (and thus $\mathbf{B}_r$) is of rank $2m$. Next suppose $r = 2m+1$. We consider the $(m+1) \times (m+1)$ submatrix $\mathbf{H}$ of $\mathbf{B}_r$ consisting of columns $4p-1$

and $4p$ for each $p \in [m]$, as well as column $4m+1$. Thus,

$$\mathbf{H} = \begin{bmatrix} \alpha^{i'_2} & \alpha^{2i'_2} & 0 & 0 & \cdots & 0 & 0 & 0 \\ \alpha^{i_2} & \alpha^{2i_2} & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & \alpha^{i'_4} & \alpha^{2i'_4} & \cdots & 0 & 0 & 0 \\ 0 & 0 & \alpha^{i_4} & \alpha^{2i_4} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \alpha^{i'_{2m}} & \alpha^{2i'_{2m}} & 0 \\ 0 & 0 & 0 & 0 & \cdots & \alpha^{i_{2m}} & \alpha^{2i_{2m}} & \alpha^{i'_{2m+1}} \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & \alpha^{i_{2m+1}} \end{bmatrix}.$$

Again, we observe that $\mathbf{H}$ is a block-diagonal matrix of the form

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_1 & \mathbf{0}_{2\times 2} & \cdots & \mathbf{0}_{2\times 3} \\ \mathbf{0}_{2\times 2} & \mathbf{H}_2 & \cdots & \mathbf{0}_{2\times 3} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{3\times 2} & \mathbf{0}_{3\times 2} & \cdots & \mathbf{H}_m \end{bmatrix},$$

where for each $p \in [m-1]$,

$$\mathbf{H}_p = \begin{bmatrix} \alpha^{i'_{2p}} & \alpha^{2i'_{2p}} \\ \alpha^{i_{2p}} & \alpha^{2i_{2p}} \end{bmatrix},$$

and

$$\mathbf{H}_m = \begin{bmatrix} \alpha^{i'_{2m}} & \alpha^{2i'_{2m}} & 0 \\ \alpha^{i_{2m}} & \alpha^{2i_{2m}} & \alpha^{i'_{2m+1}} \\ 0 & 0 & \alpha^{i_{2m+1}} \end{bmatrix}.$$

We've already seen that $|\mathbf{H}_p| \neq 0$ for each $p \in [m-1]$. Further, we see that $|\mathbf{H}_m| = \alpha^{i_{2m}+i'_{2m}+i_{2m+1}}(\alpha^{i_{2m}} + \alpha^{i'_{2m}}) \neq 0$. Thus,

$$|\mathbf{H}| = |\mathbf{H}_1| \cdot |\mathbf{H}_2| \cdot \ldots \cdot |\mathbf{H}_m| \neq 0,$$

which shows that $\mathbf{H}$ (and thus $\mathbf{B}_r$) is of rank $2m+1$. □

Label the q nonces $\mathcal{N}^{(1)}, \ldots, \mathcal{N}^{(q)}$. For the j-th nonce, there are $\ell^{(j)}$ queries $(\mathcal{N}^{(j)}, 1, M_1^{(j)}), \ldots, (\mathcal{N}^{(j)}, \ell^{(j)}, M_{\ell^{(j)}}^{(j)})$, with outputs $(C_1^{(j)}, \ldots C_{\ell^{(j)}}^{(j)})$ respectively. For the internal transcript, we have $\mathcal{L}$, the encryption of 0 with π, and for the j-th nonce, we have the three random-function outputs $X^{(j)}, Y^{(j)}, Z^{(j)}$; finally, we have the (input, output) pairs $(S_1^{(j)}, \widehat{S}_1^{(j)}), \ldots, (S_{\ell^{(j)}}^{(j)}, \widehat{S}_{\ell^{(j)}}^{(j)})$ to π, and the (input-offset, output-offset) pairs $(T_1^{(j)}, \widehat{T}_1^{(j)}), \ldots, (T_{\ell^{(j)}}^{(j)}, \widehat{T}_{\ell^{(j)}}^{(j)})$. Then this extended transcript satisfies the following equations for each $j \in [q]$ and each $i \in [\ell^{(j)}]$:

$$S_i^{(j)} = M_i^{(j)} + T_i^{(j)}, \qquad \widehat{S}_i^{(j)} = C_i^{(j)} + \widehat{T}_i^{(j)},$$
$$T_i^{(j)} = \alpha^i \cdot X^{(j)} + \alpha^{2i} \cdot Y^{(j)}, \qquad \widehat{T}_i^{(j)} = Z^{(j)} + \alpha_i \cdot \mathcal{L}.$$

5.3.5.1 Internal Sampling.

Following the query phase of the game, in the ideal world we sample the internal transcript as follows (subject to certain bad events to be defined subsequently):

- Sample $X^{(j)}, Y^{(j)}$ uniformly at random for each $j \in [q]$;
- Check for bad1, bad2, bad3, bad4;
- Sample $\mathcal{L}$ uniformly at random;
- Check for bad5, bad6;
- Let $S_1, \ldots, S_t$ be a labeling of the unique values in $\{S_i^{(j)} \mid j \in [q], i \in [\ell^{(j)}]\}$;
- Sample $\{\widehat{S}_k \mid k \in [t]\}$ directly from good set, subject to the equations $\widehat{S}_i^{(j)} + \widehat{S}_{i'}^{(j)} = C_i^{(j)} + C_{i'}^{(j)} + (\alpha^i + \alpha^{i'}) \cdot \mathcal{L}$ for each $j \in [q]$ and each $i, i' \in [\ell^{(j)}]$.

Before describing the bad events bad1, ..., bad6, we define two graphs on the extended transcript.

5.3.5.2 Transcript Graph.

For distinct $j_1, j_2 \in [q]$, there is an edge (j_1, j_2) in G if we have some $i_1 \in [\ell^{(j_1)}]$ and some $i_2 \in [\ell^{(j_2)}]$ such that $S_{i_1}^{(j_1)} = S_{i_2}^{(j_2)}$.
We will refer to paths of length 2 in G as *links*. A link (j_1, j_2, j_3) formed with the collisions $S_{i_1}^{(j_1)} = S_{i_2}^{(j_2)}$ and $S_{i_2'}^{(j_2)} = S_{i_3}^{(j_3)}$ for some $i_1 \in [\ell^{(j_1)}]$, $i_2, i_2' \in [\ell^{(j_2)}]$ and $i_3 \in [\ell^{(j_3)}]$ is called *degenerate* if $i_2 = i_2'$ and *non-degenerate* otherwise. We observe that the above link being degenerate implies $S_{i_1}^{(j_1)} = S_{i_3}^{(j_3)}$, so (j_1, j_3) is also an edge in G. By *short-circuiting* a degenerate link (j_1, j_2, j_3) we will refer to the operation of replacing it with the edge (j_1, j_3).

A path of length ≥ 3 is called non-degenerate if at least one of its sublinks is non-degenerate. When a non-degenerate path contains a degenerate sublink, we can short-circuit it to obtain a shorter non-degenerate path. We can repeat this operation as long as the path contains degenerate sublinks to end up with a *minimal* non-degenerate path. When the initial path is a cycle, we end up with either a minimal non-degenerate cycle or a *double-collision* edge, i.e., an edge (j_1, j_2) in G such that for distinct $i_1, i_1' \in [\ell^{(j_1)}]$ and distinct $i_2, i_2' \in [\ell^{(j_2)}]$ we have $S_{i_1}^{(j_1)} = S_{i_2}^{(j_2)}$, and $S_{i_1'}^{(j_1)} = S_{i_2'}^{(j_2)}$.

5.3.5.3 Dual Graph (for Mirror Theory).

We also define a second graph H on the transcript, which is something of a dual of the first. This is the graph we need to check for the conditions necessary to apply mirror theory. First consider the graph H' such that the vertices of H' are the distinct values $S_1, \ldots, S_t$, and there is an edge between S_i and $S_{i'}$ in H if they appear in the same nonce, i.e., if there is some $j \in [q]$, $i, i' \in [\ell^{(j)}]$ such that $\widehat{S}_i^{(j)} + \widehat{S}_{i'}^{(j)} = C_i^{(j)} + C_{i'}^{(j)} + (\alpha^i + \alpha^{i'}) \cdot \mathcal{L}$; further, the weight of this edge is then $C_i^{(j)} + C_{i'}^{(j)} + (\alpha^i + \alpha^{i'}) \cdot \mathcal{L}$.

From H' we get H by dropping all *redundant edges*—for each $j \in [\ell^{(j)}]$, out of the fully connected subgraph of G with $\binom{\ell^{(j)}}{2}$ edges, we only keep a spanning tree of $\ell^{(j)} - 1$ edges, and drop the rest. For instance, one way of choosing H could be to just keep the edge between $\widehat{S}_i^{(j)}$ and $\widehat{S}_{i+1}^{(j)}$ for each $i \in [\ell^{(j)} - 1]$. (Note that we assume here that all $\widehat{S}_i^{(j)}$ are distinct within any j, because that is the only use-case we'll need; the notions however easily generalise to graphs with intra-nonce collisions.)

We observe that H is cycle-free as long as G is cycle-free, and that the size $\xi_{\max}$ of the largest component of H is at most LM when M is the size of the largest component of G.

5.3.5.4 Bad Events.

Based on the graphs G and H defined above, we can describe our bad events.

bad1: *We have $j \in [q]$ and distinct $i, i' \in [\ell^{(j)}]$ such that $S_i^{(j)} = S_{i'}^{(j)}$.*

bad2: *There is a double-collision edge in G.*

bad3: *There is a minimal non-degenerate cycle in G.*

bad4: *G has a component of size $> n$.*

bad5: *We have $j \in [q]$ and distinct $i, i' \in [\ell^{(j)}]$ such that $C_i^{(j)} + C_{i'}^{(j)} = (\alpha^i + \alpha^{i'}) \cdot \mathcal{L}$.*

bad6: *We have a path in H on which the edge-weights sum to 0.*

Next we give an upper bound on the probability of at least one bad event happening in the ideal world. Define

$$\mathsf{bad} := \bigcup_{p=1}^{6} \mathsf{bad[p]}.$$

Then we have the following lemma.

Lemma 5.5. *In the ideal world,*

$$\Pr[\mathit{bad}] \leq \frac{n\sigma L}{N}.$$

of Lemma 1. We bound the probability of each of the six bad events one by one below.

bad1: *We have $j \in [q]$ and distinct $i, i' \in [\ell^{(j)}]$ such that $S_i^{(j)} = S_{i'}^{(j)}$.*

For a fixed choice of indices j, i and i', the probability of the event comes out to be $1/N$ due to the randomness of $T_i^{(j)}$ or $T_{i'}^{(j)}$. From union bound over all possible choices of indices, we obtain

$$\Pr[\mathsf{bad1}] \leq \frac{1}{N} \sum_{j=1}^{q} \ell^{(j)2} \leq \frac{L}{N} \sum_{j=1}^{q} \ell^{(j)} \leq \frac{\sigma L}{N}.$$

bad2: *There is a double-collision edge in G.*

This implies that we have distinct $j_1, j_2 \in [q]$, distinct $i_1, i_1' \in [\ell^{(j_1)}]$, and distinct $i_2, i_2' \in [\ell^{(j_2)}]$ such that $S_{i_1}^{(j_1)} = S_{i_2}^{(j_2)}$, and $S_{i_1'}^{(j_1)} = S_{i_2'}^{(j_2)}$. This can be written as $\mathbf{B}_2\mathbf{v} = \mathbf{c}$, where

$$\mathbf{B}_2 = \begin{bmatrix} \alpha^{i_1} & \alpha^{2i_1} & \alpha^{i_2} & \alpha^{2i_2} \\ \alpha^{i_1'} & \alpha^{2i_1'} & \alpha^{i_2'} & \alpha^{2i_2'} \end{bmatrix}, \mathbf{v} = \begin{bmatrix} X^{(j_1)} \\ Y^{(j_1)} \\ X^{(j_2)} \\ Y^{(j_2)} \end{bmatrix}, \mathbf{c} = \begin{bmatrix} M_{i_1}^{(j_1)} + M_{i_2}^{(j_2)} \\ M_{i_1'}^{(j_1)} + M_{i_2'}^{(j_2)} \end{bmatrix}.$$

$\mathbf{B}_2$ is of rank 2 by Lemma 5.4. Thus, when we fix $j_1, j_2, i_1, i_1', i_2, i_2'$, we have

$$\Pr[\mathbf{B}_2\mathbf{v} = \mathbf{c}] \leq \frac{1}{N^2}.$$

Thus,

$$\Pr[\mathsf{bad2}] \le \frac{1}{N^2} \sum_{j_1=1}^{q} \sum_{j_2=1}^{q} \ell^{(j_1)2} \ell^{(j_2)2} \le \frac{L^2}{N^2} \sum_{j_1=1}^{q} \sum_{j_2=1}^{q} \ell^{(j_1)} \ell^{(j_2)} \le \frac{\sigma^2 L^2}{N^2}\,.$$

bad3: *There is a minimal non-degenerate cycle in the transcript graph.*

First, suppose there is a minimal non-degenerate cycle of length 3. Thus, we have distinct $j_1, j_2, j_3 \in [q]$, distinct $i_1, i'_1 \in [\ell^{(j_1)}]$, distinct $i_2, i'_2 \in [\ell^{(j_2)}]$, and distinct $i_3, i'_3 \in [\ell^{(j_3)}]$ such that $S_{i_1}^{(j_1)} = S_{i'_2}^{(j_2)}$, $S_{i_2}^{(j_2)} = S_{i'_3}^{(j_3)}$, and $S_{i_3}^{(j_3)} = S_{i'_1}^{(j_1)}$. (We name the indices like this for symmetry.) As before, this can be written as $\mathbf{B}_3\mathbf{v} = \mathbf{c}$, where

$$\mathbf{B}_3 = \begin{bmatrix} \alpha^{i_1} & \alpha^{2i_1} & \alpha^{i'_2} & \alpha^{2i'_2} & 0 & 0 \\ 0 & 0 & \alpha^{i_2} & \alpha^{2i_2} & \alpha^{i'_3} & \alpha^{2i'_3} \\ \alpha^{i'_1} & \alpha^{2i'_1} & 0 & 0 & \alpha^{i_3} & \alpha^{2i_3} \end{bmatrix}, \mathbf{v} = \begin{bmatrix} X^{(j_1)} \\ Y^{(j_1)} \\ X^{(j_2)} \\ Y^{(j_2)} \\ X^{(j_3)} \\ Y^{(j_3)} \end{bmatrix}, \mathbf{c} = \begin{bmatrix} M_{i_1}^{(j_1)} + M_{i'_2}^{(j_2)} \\ M_{i_2}^{(j_2)} + M_{i'_3}^{(j_3)} \\ M_{i_3}^{(j_3)} + M_{i'_1}^{(j_1)} \end{bmatrix}.$$

$\mathbf{B}_3$ is of rank 3 by Lemma 5.4. Thus, when we fix $j_1, j_2, j_3, i_1, i'_1, i_2, i'_2, i_3, i'_3$, we have

$$\Pr[\mathbf{B}_3\mathbf{v} = \mathbf{c}] \le \frac{1}{N^3}.$$

Next, suppose there is a minimal non-degenerate cycle of length $r \ge 4$. Thus we have distinct $j_1, \ldots, j_r \in [q]$; for $u \in [r-1]$ we have $i_u \in [\ell^{(j_u)}]$ and $i'_{u+1} \in [\ell^{(j_{u+1})}]$ such that $S_{i_u}^{(j_u)} = S_{i'_{u+1}}^{(j_{u+1})}$; and finally, we have $i_r \in [\ell^{(j_r)}]$ and $i'_1 \in [\ell^{(j_1)}]$ such that $S_{i_r}^{(j_r)} = S_{i'_1}^{(j_1)}$; the cycle being minimal non-degenerate

implies that for each $u \in [r]$, $i_u \neq i'_u$. This can be written as $\mathbf{B}_r\mathbf{v} = \mathbf{c}$, where

$$\mathbf{B}_r = \begin{bmatrix} \alpha^{i_1} & \alpha^{2i_1} & \alpha^{i'_2} & \alpha^{2i'_2} & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & \alpha^{i_2} & \alpha^{2i_2} & \alpha^{i'_3} & \alpha^{2i'_3} & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \alpha^{i_3} & \alpha^{2i_3} & \alpha^{i'_4} & \alpha^{2i'_4} & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & \alpha^{i_{r-1}} & \alpha^{2i_{r-1}} & \alpha^{i'_r} & \alpha^{2i'_r} \\ \alpha^{i'_1} & \alpha^{2i'_1} & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & \alpha^{i_r} & \alpha^{2i_r} \end{bmatrix},$$

$$\mathbf{v} = \begin{bmatrix} X^{(j_1)} \\ Y^{(j_1)} \\ X^{(j_2)} \\ Y^{(j_2)} \\ \vdots \\ X^{(j_r)} \\ Y^{(j_r)} \end{bmatrix}, \mathbf{c} = \begin{bmatrix} M_{i_1}^{(j_1)} + M_{i'_2}^{(j_2)} \\ M_{i_2}^{(j_2)} + M_{i'_3}^{(j_3)} \\ M_{i_3}^{(j_3)} + M_{i'_4}^{(j_4)} \\ \vdots \\ M_{i_{r-1}}^{(j_{r-1})} + M_{i'_r}^{(j_r)} \\ M_{i_r}^{(j_r)} + M_{i'_1}^{(j_1)} \end{bmatrix}.$$

$\mathbf{B}_r$ is of rank r by Lemma 5.4. Thus, for each $r \geq 3$, when we fix $j_1, \cdots, j_r, i_1, i'_1, \cdots, i_r, i'_r$, we have

$$\Pr[\mathbf{B}_r\mathbf{v} = \mathbf{c}] \leq \frac{1}{N^r}.$$

Assuming $2\sigma L \leq N$, we have

$$\begin{aligned} \Pr[\mathsf{bad3}] &\leq \sum_{r=3}^{q} \frac{\prod_{u=1}^{r} \ell^{(j_u)2}}{N^r} \\ &\leq \sum_{r=3}^{q} \left(\left(\frac{L}{N}\right)^r \prod_{u=1}^{r} \ell^{(j_u)} \right) \leq \sum_{r=3}^{q} \left(\frac{\sigma L}{N}\right)^r \leq \frac{2\sigma^3 L^3}{N^3}. \end{aligned}$$

bad4: *G has a component of size $> n$.*

For a component of size M, the minimum number of nonces in that component should be $p+1$ where $p = \lceil M/L \rceil - 1$ with p collisions among themselves. In other words, $\exists$ distinct $j_1, j_2, \cdots, j_{p+1} \in [q]$ and $i_1 \in \ell^{(j_1)}$, $i_2, i'_2 \in \ell^{(j_2)}$,

$i_3, i'_3 \in \ell^{(j_3)}$, $\cdots$, $i_p, i'_p \in \ell^{(j_p)}$, $i_{p+1} \in \ell^{(j_{p+1})}$ such that

$$S_{i_1}^{(j_1)} = S_{i_2}^{(j_2)}, S_{i'_2}^{(j_2)} = S_{i_3}^{(j_3)}, \ldots, S_{i'_p}^{(j_p)} = S_{i_{p+1}}^{(j_{p+1})}.$$

For a fixed choice of indices, the probability of the event comes out to be $1/N^p$. The independence assumption comes from the fact that every equation from the system of equations mentioned above introduces a fresh nonce. From union bound over all the possible choices of indices, we obtain

$$\begin{aligned}
\Pr[\mathsf{bad4}] &\leq \frac{1}{N^p} \sum_{j_1=1}^{q} \sum_{j_2=1}^{q} \cdots \sum_{j_{p+1}=1}^{q} \ell^{(j_1)2} \ell^{(j_2)2} \ldots \ell^{(j_{p+1})2} \\
&\leq \frac{L^{p+1}}{N^p} \sum_{j_1=1}^{q} \sum_{j_2=1}^{q} \cdots \sum_{j_{p+1}=1}^{q} \ell^{(j_1)} \ell^{(j_2)} \ldots \ell^{(j_{p+1})} \\
&\leq \frac{\sigma^{p+1} L^{p+1}}{N^p} = \frac{\sigma L}{N} \left(\frac{\sigma^p L^p}{N^{p-1}} \right) .
\end{aligned}$$

Assuming $\sigma L \leq N/2$ and $p = n$, we get

$$\Pr[\mathsf{bad4}] \leq \frac{\sigma L}{N} .$$

bad5: *We have $j \in [q]$ and distinct $i, i' \in [\ell_j]$ such that $C_i^{(j)} + C_{i'}^{(j)} = (\alpha^i + \alpha^{i'}) \cdot \mathcal{L}$.*

For a fixed choice of indices j, i and i', the probability of the event comes out to be $1/N$ due to the randomness of $\mathcal{L}$. From union bound over all possible choices of indices, we obtain

$$\Pr[\mathsf{bad5}] \leq \frac{1}{N} \sum_{j=1}^{q} \ell^{(j)2} \leq \frac{L}{N} \sum_{j=1}^{q} \ell^{(j)} \leq \frac{\sigma L}{N} .$$

bad6: Suppose the first and last vertices on a path inside some component are $\widehat{S}_i^{(j)}$ and $\widehat{S}_{i'}^{(j')}$. Also suppose that the path goes through $x_1, x_2, \cdots, x_y$ vertices of position $i_1, i_2, \cdots, i_y$ respectively. Then this bad event implies

$$C_i^{(j)} + C_{i'}^{(j')} + (\alpha^i + x_1 \alpha^{i_1} + \cdots + x_y \alpha^{i_y} + \alpha^{i'}) \cdot \mathcal{L} = 0 .$$

For a fixed choice of the vertex pair $(\widehat{S}_i^{(j)}, \widehat{S}_{i'}^{(j')})$, the probability of the event comes out to be $1/N$ due to the randomness of $\mathcal{L}$. Applying union bound over all possible vertex pairs, and summing over all components $\mathcal{C}$ of G, we

get

$$\Pr[\mathsf{bad6}] \leq \sum_{\mathcal{C}} \frac{1}{2N} \cdot \left(\sum_{j \in \mathcal{C}} \ell^{(j)} \right)^2 \leq \sum_{\mathcal{C}} \frac{1}{2N} \cdot \xi_{\max} \cdot \sum_{j \in \mathcal{C}} \ell^{(j)} = \frac{\xi_{\max}\sigma}{2N} \leq \frac{n\sigma L}{2N}.$$

Thus, by union-bound, we have

$$\Pr[\mathsf{bad}] \leq \frac{4\sigma L}{N} + \frac{\sigma^2 L^2}{N^2} + \frac{2\sigma^3 L^3}{N^3} + \frac{n\sigma L}{2N} \leq \frac{n\sigma L}{N},$$

which completes the proof of the lemma. □

5.3.5.5 Bounding the Ratio of Good Probabilities.

Let τ be a good transcript. In the real world, there are q distinct inputs to f_1, q distinct inputs to f_2, and t distinct inputs to π. Thus,

$$\Pr_{\mathcal{O}_1}[\tau] = \frac{1}{N^{2q}(N)_t}.$$

In the ideal world, in the online stage, there are σ outputs that are sampled uniformly at random. In the offline stage, q more values are sampled uniformly, and finally t variables are sampled from the good set subject to r non-redundant equations (we calculate r later). Since $\sigma < N/n^2L^2$, and none of the bad events has happened, the conditions for applying mirror theory are fulfilled. Thus, using mirror theory,

$$\Pr_{\mathcal{O}_0}[\tau] \leq \frac{1}{N^{\sigma+q}} \cdot \frac{N^r}{(N)_t} \leq \frac{1}{N^{\sigma+q-r}(N)_t}.$$

To calculate r, we note that every repeated use of a nonce adds a non-redundant equation to the system. Thus, $r = \sigma - q$, giving us

$$\Pr_{\mathcal{O}_0}[\tau] \leq \frac{1}{N^{2q}(N)_t}.$$

Thus, we have

$$\frac{\Pr_{\mathcal{O}_1}[\tau]}{\Pr_{\mathcal{O}_0}[\tau]} \geq 1,$$

Applying the H-Coefficient Technique with $\epsilon_1 = n\sigma L/N$ and $\epsilon_2 = 0$ completes the proof. □

5.3.6 TSPRP Security Analysis of OTBC-3

Let's call $(\mathcal{N}, i)$ as $\mathcal{T}$. We'll use Coefficients H Technique to bound the advantage of the adversary.

Transcript Notation. The adversary makes encryption queries $(\mathcal{T}^{(j)}, M^{(j)})$ to the oracle to receive $C^{(j)}$ and decryption queries $(\mathcal{T}^{(j')}, C^{(j')})$ to the oracle to receive $K^{(j')}$ with $j, j' \in [\sigma]$ and $j \neq j'$. So the query-response transcript of the adversary initially looks like $\{(\mathcal{T}^{(1)}, M^{(1)}, C^{(1)}), \cdots, (\mathcal{T}^{(\sigma)}, M^{(\sigma)}, C^{(\sigma)})\}$.

Sampling in the Ideal World. For each encryption query $(\mathcal{T}^{(j)}, M^{(j)})$, the ideal oracle samples $C^{(j)}$ with replacement from $\{0,1\}^n$ uniformly at random. Similarly, for each decryption query $(\mathcal{T}^{(j')}, C^{(j')})$, the ideal oracle samples $M^{(j')}$ with replacement from $\{0,1\}^n$ uniformly at random. Once the adversary is done with all its queries, the oracle releases some additional information to the adversary. The ideal oracle samples them in the following way:

- The ideal oracle samples $\mathcal{L}$ from $\{0,1\}^n$ uniformly at random.
- For all $j \in [\sigma]$, the ideal oracle samples $X^{(j)}, Y^{(j)}$ and $Z^{(j)}$ with replacement from $\{0,1\}^n$ uniformly at random.

The real oracle releases the corresponding true values in this additional release phase. After the additional release, the extended transcript looks like the following: $\{\mathcal{L}, (\mathcal{T}^{(1)}, M^{(1)}, C^{(1)}, X^{(1)}, Y^{(1)}, Z^{(1)}), \cdots, (\mathcal{T}^{(q)}, M^{(q)}, C^{(q)}, X^{(q)}, Y^{(q)}, Z^{(q)})\}$.

Bad Events and Their Probabilities. We identify the following events as bad.

bad1: $\exists j, j' \in [\sigma]$ with $j \neq j'$ such that $S^{(j)} = S^{(j')}$. The probability of this event can be bounded by (σ^2/N) due to the randomness of X or Y.

bad2: $\exists j, j' \in [\sigma]$ with $j \neq j'$ such that $\widehat{S}^{(j)} = \widehat{S}^{(j')}$. The probability of this event can also be bounded by (σ^2/N) due to the randomness of X or Y.

bad3: $\exists j \in [\sigma]$ such that $S^{(j)} = 0^n$. The probability of this event can be bounded by (σ/N) due to the randomness of X or Y.

bad4: $\exists j \in [\sigma]$ such that $\widehat{S}^{(j)} = \mathcal{L}$. The probability of this event can also be bounded by (σ/N) due to the randomness of X or Y.

Good Interpolation Probabilities and Their Ratio. For any good transcript τ, we get the real interpolation probability as

$$\Pr_{\mathcal{O}_1}[\tau] = \frac{1}{N^\sigma} \cdot \frac{1}{N^\sigma} \cdot \frac{1}{N^\sigma} \cdot \frac{1}{(N)_{\sigma+1}} .$$

The first, second and third term in the denominator on the right hand side represents the number of choices for X, Y and Z respectively, and the fourth term represents the number of choices for distinct permutation calls. We also get the ideal interpolation probability as

$$\Pr_{\mathcal{O}_0}[\tau] = \frac{1}{N^\sigma} \cdot \frac{1}{N^\sigma} \cdot \frac{1}{N^\sigma} \cdot \frac{1}{N^{\sigma+1}} .$$

The first, second and third term in the denominator on the right hand side represents the number of choices for X, Y and Z respectively, and the fourth term represents the number of choices for distinct permutation calls. Thus we finally we get

$$\frac{\Pr_{\mathcal{O}_1}[\tau]}{\Pr_{\mathcal{O}_0}[\tau]} \geq 1 .$$

Advantage of the Adversary. Applying H-Coefficient Technique, we get that the TSPRP advantage of the adversary is bounded above by

$$\epsilon_1 = \frac{2\sigma^2}{N} + \frac{2\sigma}{N} .$$

□

5.4 An Application of OTBC-3

Using the tweakable block-cipher OTBC-3, we define an authenticated encryption scheme OCB+ that is about as efficient as OCB3 while providing a higher degree of privacy guarantee without affecting the authenticity guarantee of OCB3. This is shown in Figure 5.5.

5.4.1 Nonce Handling

OCB+ uses a nonce $\mathcal{N}$ of $n-2$ bits, with the final two bits reserved for domain separation. $\mathcal{N}\|00$ is used for processing the message blocks, $\mathcal{N}\|01$ is used for processing the tag, and $\mathcal{N}\|10$ is used for handling the associated data.

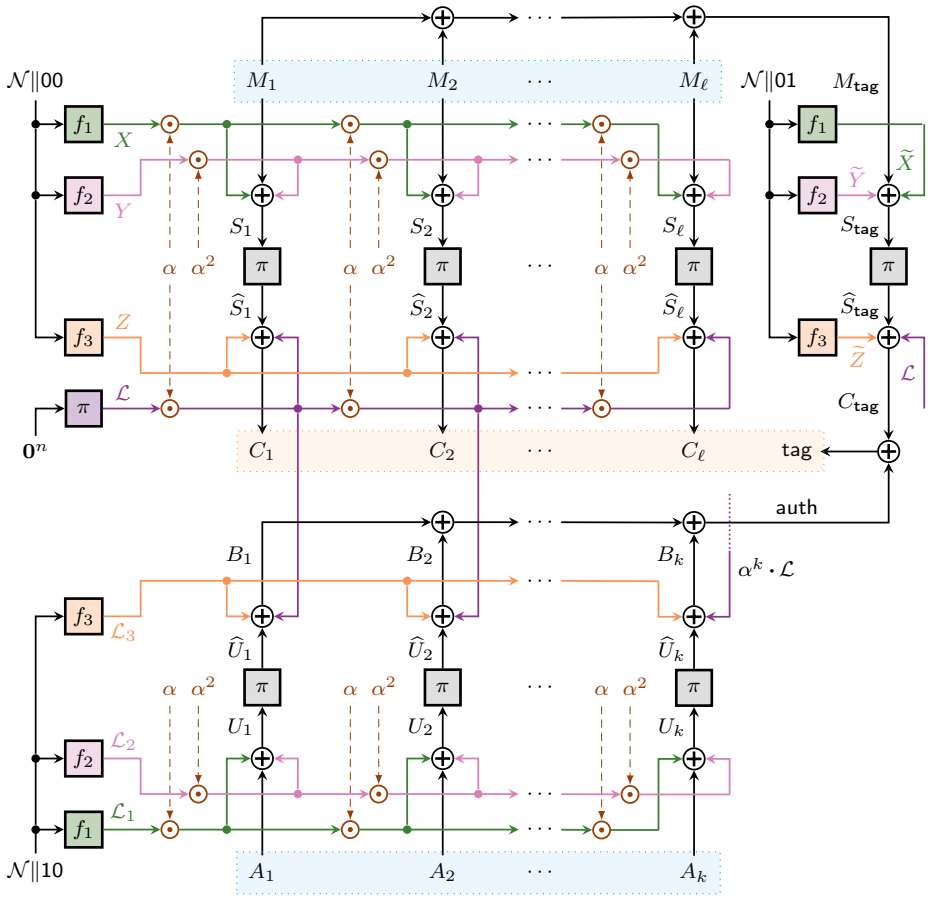


FIGURE 5.5: The OCB+ construction. α is a primitive field-element that allows efficient multiplication.

5.4.2 Handling Incomplete Blocks

Incomplete blocks can be handled in the same way as in OCB3, modifying the masking constants for the incomplete blocks. This does not affect the privacy bound significantly, and since the focus of this chapter is to improve the privacy guarantee of OCB3, we skip giving specific details on how to handle incomplete blocks in OCB+.

5.4.3 Security Claims

We claim that as long as the maximum length L permitted for each message (i.e., the maximum number of blocks encrypted using the same nonce) is small, OCB+ provides both beyond-birthday privacy and beyond-birthday authenticity. Formally we claim the following.

Theorem 5.6. *Consider a distinguisher $\mathcal{A}$ of OCB+ which can make q encryption queries with distinct nonces with σ blocks and q' decryption queries to its*

Algorithm 6 $\mathsf{OCB+}^{f_1,f_2,f_3,\pi}(\mathcal{N}, A, M)$

$M_{\mathsf{tag}} \leftarrow 0^n$
$\mathsf{auth} \leftarrow 0^n$
for $i \leftarrow 1$ to ℓ **do**
    $M_{\mathsf{tag}} \leftarrow M_{\mathsf{tag}} \oplus M_i$
    $C_i \leftarrow \mathsf{OTBC\text{-}3}^{f_1,f_2,f_3,\pi}(\mathcal{N}\|\mathsf{00}, i, M_i)$
$C \leftarrow C_1\|\cdots\|C_\ell$
$C_{\mathsf{tag}} \leftarrow \mathsf{OTBC\text{-}3}^{f_1,f_2,f_3,\pi}(\mathcal{N}\|\mathsf{01}, 0, M_{\mathsf{tag}})$
for $i \leftarrow 1$ to k **do**
    $B_i \leftarrow \mathsf{OTBC\text{-}3}^{f_1,f_2,f_3,\pi}(\mathcal{N}\|\mathsf{10}, i, A_i)$
    $\mathsf{auth} \leftarrow \mathsf{auth} \oplus B_i$
$\mathsf{tag} \leftarrow C_{\mathsf{tag}} \oplus \mathsf{auth}$
$T \leftarrow \mathsf{chop}_\tau(\mathsf{tag})$
return (C, T)

challenger. Suppose the length of the i-th message and the i-th associated data are ℓ_i and k_i respectively, where $\ell_i, k_i \leq L \forall i \in [q_e]$. As long as $\sigma \leq N/n^2L^2$, we have

$$\boldsymbol{Adv}^{\mathsf{OCB+}}_{\mathsf{NAEAD}}(\mathcal{A}) \leq \frac{n\sigma L}{N} + O\left(\frac{q'L}{N}\right).$$

Proof. Suppose there is a distinguisher $\mathcal{B}$ of $\mathsf{OTBC\text{-}3}$ which can make $\sigma + q$ queries to its challenger and which works in the following way. It runs $\mathcal{A}$ to start the game. Whenever $\mathcal{A}$ makes the i-th encryption query $(\mathcal{N}^i, A^i, M^i)$, $\mathcal{B}$ does the following.

- For the j-th message block M^i_j, it makes the encryption query$(\mathcal{N}^i\|\mathsf{00}, j, M^i_j)$ to its challenger. Suppose it receives C^i_j as the response.
- Suppose the length of M^i is ℓ_i blocks. It makes and encryption query $(\mathcal{N}^i\|\mathsf{01}, 0, M^i_1 + \cdots + M^i_{\ell_i})$ to its challenger. Suppose it receives C^i_{tag} as response.
- For the j-th associated data block A^i_j, it makes the encryption query $(\mathcal{N}^i\|\mathsf{10}, j, A^i_j)$ to its challenger. Suppose it receives B^i_j as response.
- Suppose the length of A^i is k_i blocks. It calculates $\mathsf{auth}^i = B^i_1 + \cdots + B^i_{k_i}$.
- Finally it returns $(C^i_1\|\cdots\|C^i_{\ell_i}, \mathsf{chop}_\tau(C^i_{\mathsf{tag}} + \mathsf{auth}^i))$ to $\mathcal{A}$.

Once $\mathcal{A}$ submits its decision bit, $\mathcal{B}$ carries it forward to its challenger as its own decision bit as well. Then we obtain the following privacy advantage of $\mathcal{A}$:

$$\mathbf{Adv}^{\mathsf{OCB+}}_{\mathsf{priv}}(\mathcal{A}) = \mathbf{Adv}^{\mathsf{OTBC\text{-}3}}_{\mathrm{TPRP}^*}(\mathcal{B}).$$

Combining this result with Theorem 5.3, we obtain

$$\mathbf{Adv}^{\mathsf{OCB+}}_{\mathsf{priv}}(\mathcal{A}) \leq \frac{n\sigma L}{N}. \tag{5.1}$$

From the security analysis in Section 4 of [157], we obtain the following authenticity advantage of $\mathcal{A}$.

$$\mathbf{Adv}^{\mathsf{OCB+}}_{\mathsf{auth}}(\mathcal{A}) \leq O\left(\frac{q'L}{N}\right). \tag{5.2}$$

The result of Theorem 5.6 follows directly from (5.1) and (5.2). □

5.5 Conclusion

In this chapter, we have proposed a new nonce-respecting BBB secure offset-based TBC and used it in an OCB-like mode to obtain a new NAEAD mode named OCB+, which has both BBB privacy security as well as BBB authenticity security. A natural follow-up to this work should be to implement OCB+ in a practical setup and benchmark it against OCB to get an idea of how much the design overhead of OCB+ costs in performance for real-life applications.

Chapter 6

CENCPP*

Public permutations have been established as important primitives for the purpose of designing cryptographic schemes. While many such schemes for authentication and encryption have been proposed in the past decade, the birthday bound in terms of the primitive's block length n has been mostly accepted as the standard security goal. Thus, remarkably little research has been conducted yet on permutation-based modes with higher security guarantees. At CRYPTO'19, Chen et al. showed two constructions with higher security based on the sum of two public permutations. Their work has sparked increased interest in this direction by the community. However, since their proposals were domain-preserving, the question of encryption schemes with beyond-birthday-bound security was left open.
This chapter tries to address this gap by proposing CENCPP*, a nonce-based encryption scheme from public permutations. Our proposal is a variant of Iwata's block-cipher-based mode CENC that we adapt for public permutations, thereby generalising Chen et al.'s Sum-of-Even-Mansour construction to a mode with variable output lengths. Like CENC, our proposal enjoys a comfortable rate-security trade-off that needs $w + 1$ calls to the primitive for w primitive outputs. We show a tight security level for up to $O(2^{2n/3}/w^2)$ primitive calls. While the term of $w \geq 1$ can be arbitrary, two independent keys suffice. Beyond our proposal of CENCPP* in a generic setting with $w + 1$ independent permutations, we show that only $\log_2(w + 1)$ bits of the input for domain separation suffice to obtain a single-permutation variant with a security level of up to $O(2^{2n/3}/w^4)$ queries.

6.1 Introduction

Permutation-based cryptography has become an important branch of symmetric-key cryptography. Permutations spare the cryptographer from the task of designing and analysing a secure key schedule. Permutations have a long history in many applications. For example, the eSTREAM candidate Salsa [158] already allowed hashing, expansion, and encryption based on a permutation. After Keccak's selection as the SHA-3 standard [159], the number of proposed permutations and the number of schemes built upon them has surged. Nowadays, various schemes exist, few for hashing and authentication like Chaskey [26], but many more for authenticated encryption, where many AE schemes are based on the Duplex construction [34].

The security of many block-cipher-based modes, such as GCM [160] or OCB3 [161] is limited by the birthday bound of the primitive's state size (usually indicated by n bits). This limitation renders the privacy guarantees void when some internal collision occurs, which happens with non-negligible probability after $O(2^{n/2})$ blocks have been processed under the same key. Modes with higher security guarantees appear helpful for cases where smaller primitives must be used. As a response, the cryptographic community has proposed various modes with higher security over the previous decades, such as CENC [16]. Many more modes have been proposed in the domain of MACs and fixed-output-length PRFs, that includes designs like PMAC$^+$ [162], Sum-ECBC [163], 3kf9 [164], LightMAC_Plus [165], or the Sum of GCM constructions [166], many of which could be generalised under the framework of Double-block-Hash-then-Sum designs [167]. The rise of tweakable block-ciphers (TBCs) [168], that take a tweak as an additional public input, allowed the construction of further modes with enhanced security guarantees, such as ΘCB3 [161] or $\mathbb{OTR}$ [60].

For permutation-based modes, the birthday-bound limitation is often tolerated, e.g. in Farfalle [27], or Elephant [169], or OPP [170] and compensated by the usage of larger permutations. However, birthday-bound-secure permutation-based modes are not useful in practice if the underlying permutation size is small. For example, a birthday-bound-secure permutation-based mode instantiated with PHOTON [171] of state size 100 bits or SPONGENT [172] of state size 88 bits, gives only 50 (resp. 44) bits of security. At CRYPTO'19, Chen et al. [17] initiated a line of research for fixed output-length PRFs with beyond-birthday-bound security. They proposed two designs: the Sum-of-Even-Mansour constructions (SoEM) and the Sum of Key-alternating Ciphers (SoKAC), with proofs for up to $O(2^{2n/3})$ queries.

The single-primitive variants were revisited by Nandi [173] and Chakraborti et al. [174], respectively. More importantly, the latter work proposed PDM-MAC, a version of SoKAC that needed only a single permutation and its inverse, as well as only a single key while maintaining security for up to $O(2^{2n/3})$ queries.
Besides those stateless deterministic constructions, at least two nonce-based PRFs for variable-length inputs with higher security exist. In [174], Chakraborti et al. also proposed PDM*MAC, which extends PDM-MAC to variable-length inputs by adding a polynomial hash of the message in the middle. At Africacrypt 2020, Dutta et al. [175] introduced nEHtM_p, a variant of Enhanced Hash-then-MAC from public permutations. Both PDM*MAC and nEHtM_p are nonce-based $2n/3$-bit-secure MACs. In the long run, however, the question will be to build more secure authenticated encryption schemes. For this purpose, at least equally secure encryption modes are necessary. The constructions above can produce only fixed-length outputs. Modes with security beyond the birthday bound are desirable for settings that are bound to small primitives but need higher security. One can use a fixed-output-length PRF repeatedly by changing the input for every block. However, this would imply a rate of 1/2, e.g. for SoEM or PDM-MAC in counter mode. Therefore, the task of designing a variable-output-length encryption scheme with comparable security and higher efficiency is still open.

6.1.1 Contributions

In this chapter, we propose CENCPP*$[w]$, a mode of operation, built from n-bit permutations with $O(2^{2n/3}/w^2)$ security where w is a small user-adjustable integer that represents a trade-off between security and efficiency. It is a variable-output-length version of SoEM22 that adapts Iwata's block-cipher-based mode CENC [16]. CENCPP* can be instantiated directly with usual permutations and requires only two independent keys for variable sizes. While our generic construction CENCPP*$[w]$ assumes $(w+1)$ independent permutations, we suggest a variant that needs only a single public permutation while sacrificing only $\log_2(w+1)$ bits of the input space for separating domains. We derive domain-separated single-primitive variants of SoEM and CENCPP*, that we call DS-SoEM and DS-CENCPP$[w]$, and show their security. We argue that two independent keys are necessary and sufficient for our security guarantees by providing distinguishers for all constructions in $O(2^{n/2})$ queries if they used a single key or a simple key-scheduling approach. Moreover, we describe distinguishers in $O(2^{2n/3})$ queries to show that the security is effectively tight except for the logarithmic factor in w.

Table 6.1 compares our proposals with beyond-birthday-secure PRFs from the literature that are built on public permutations. Although no standalone parallelisable encryption mode from permutations seems to exist, our mode is not a novum; the encryption procedures inside many permutation-based authenticated encryption schemes, as in Elephant [169], OPP [170], Minalpher [176], etc. can be seen as such. We added them as well as SoEM and PDM-MAC in counter mode for comparison. To compare their state sizes, let k, ν, and c denote the length of keys, nonces, and counters in bits, respectively. Elephant-like modes have $\min(n/2, k)$-bit security and need $2n + \nu + c$ bits of state size: $\nu + c$ bits for the nonce and counter input, n bits for the mask derived from the key, and n bits for the current block. A similar argument holds for Duplex-based constructions, which need only $2n$ bits of state for $\min(n/2, k)$-bit security. In contrast, the previously proposed PRFs with beyond-birthday-bound security need more memory. For example, SoEM and SoKAC21 need $4n$ bits each: $2n$ bits for the keys and $2n$ bits for the state. With a single key, PDM-MAC could reduce the memory to $3n$ bits. CENCPP and DS-CENCPP need a similar amount of memory as SoEM but are nonce-based. Thus, CENCPP needs $2n$ bits for the state, $2n$ bits for two keys, and $\nu + c$ bits for nonce and counter. It would be desirable to further reduce those figures in future work.

Hereafter, Section 6.2 recalls preliminaries before Section 6.3 defines CENCPP*. We employ two different keys for security and show that it is necessary to combine them for most primitive calls. We show that simpler key schedulings would lead to a birthday-bound distinguisher in Section 6.4. Next, we analyse the security of the generic CENCPP* construction in Section 6.5. In Section 6.6, we propose domain-separated variants of SoEM and CENCPP*, called DS-SoEM and DS-CENCPP. We provide a design rationale and distinguishers on weaker variants in Section 6.7. We analyse the security of DS-CENCPP and DS-SoEM in Section 6.8.1 and 6.8.2, respectively. Section 6.9 concludes.

6.2 Preliminaries

In this chapter, for any n-bit string X and non-negative integer $x \leq n$, we'll use $\mathsf{lsb}_x(X)$ and $\mathsf{msb}_x(X)$ to denote the x least significant and most significant bits of X, respectively.

Lemma 6.1. Let $d \geq 0$ be a positive integer and $K_0, K_1 \xleftarrow{\$} \{0,1\}^n$ be two independent n-bit random variables. Let $\mathbf{A}_{2\times 2} = (a_{ij}) \in \{0,1\}^n$ be a non-singular

TABLE 6.1: Comparison with existing PRFs built on public permutations with birthday-bound and beyond-birthday-bound security. Prim. = primitives, IF = inverse-free, n = state size (in bits), w = word parameter, d = domain size, ν = nonce size, c = counter size, $*$ = variable size, security in $O(\cdot)$ bits under n-bit keys, •/– = yes/no, † = rate for the finalisation only, (B)BB = (beyond-)birthday-bound

	Efficiency					Bits		
Construction	#Prim.	#Keys	IF	Nonce	Rate	State size	In	Out	Security
Fixed-length input, fixed-length output									
PDM-MAC [174]	1	1	–	–	1/2	$3n$	n	n	$2n/3$
SoEM22 [17]	2	2	•	–	1/2	$4n$	n	n	$2n/3$
SoKAC22 [17]	2	2	•	–	1/2	$4n$	n	n	$2n/3$
pEDM [177]	1	2	•	–	1/2	$4n$	n	n	$2n/3$
DS-SoEM [**Sect.** 6.6]	1	2	•	–	$(n-d)/2n$	$4n$	$n-d$	n	$2n/3$
Variable-length input, fixed-length output									
nEHtM$_p$ [175]	1	2	•	•	1/2 †	$4n-1$	$*$	n	$2n/3$
PDM*MAC [174]	1	2	–	•	1/2 †	$3n$	$*$	n	$2n/3$
1K-PDM*MAC [174]	1	1	–	•	1/2 †	$3n$	$*$	n	$2n/3$
Variable-length input, variable-length output BB security									
Elephant [169]	1	1	•	•	1	$2n+\nu+c$	$*$	$*$	$\min(k, n/2)$
Minalpher [176]	1	1	•	•	1	$2n+\nu+c$	$*$	$*$	$\min(k, n/2)$
OPP [170]	1	1	•	•	1	$2n+\nu+c$	$*$	$*$	$\min(k, n/2)$
Variable-length input, variable-length output, BBB security									
CTR-SoEM22	2	2	•	–	1/2	$4n+\nu+c$	$*$	$*$	$2n/3$
CTR-PDM-MAC	1	1	–	–	1/2	$3n+\nu+c$	$*$	$*$	$2n/3$
CENCPP* [**Sect.** 6.3]	w+1	2	•	•	$w/(w+1)$	$4n+\nu+c$	$*$	$*$	$2n/3-\log(w^2)$
DS-CENCPP [**Sect.** 6.6]	1	2	•	•	$w(n-d)/((w+1)n)$	$4n+\nu+c$	$*$	$*$	$2n/3-\log(w^4)$

matrix. Then for any $b_1 \in \{0,1\}^{n-d}$ and for any $b_2 \in \{0,1\}^n$

$$\Pr\left[\mathsf{msb}_{n-d}(a_{0,0}\cdot K_0 \oplus a_{0,1}\cdot K_1) = b_1, (a_{1,0}\cdot K_0 \oplus a_{1,1}\cdot K_1) = b_2\right] = \frac{2^d}{2^{2n}}\,.$$

Proof. Let us consider the two equations

$$\begin{cases} a_{0,0}\cdot K_0 \oplus a_{0,1}\cdot K_1 = b_1 \| \langle \alpha \rangle_d \\ a_{1,0}\cdot K_0 \oplus a_{1,1}\cdot K_1 = b_2\,, \end{cases}$$

where $\alpha \in \{0,1\}^d$. Now, the number of solutions to the above system of equations

is 1. Therefore, by varying the last d bits of the constant of the first equations to its all possible choices, we have total 2^d many solutions to the original system of equations and hence the result follows. □

A simple corollary of the above result yields the following:

Lemma 6.2. Let $\mathbf{A}_{2\times 2} = (a_{ij}) \in \{0,1\}^n$ be a non-singular matrix. For any $b_1, b_2 \in \{0,1\}^n$

$$\Pr\left[K_0, K_1 \xleftarrow{\$} \{0,1\}^n : \mathbf{A} \cdot (K_0, K_1)^\top = (b_1, b_2)^\top\right] = 2^{-2n}.$$

Proof. This result simply follows from Lemma 6.1 by setting $d = 0$. □

Lemma 6.3. *Let $0 \leq p_i \leq 1$ for $i = 1, \ldots, n$. Then, we have*

$$\prod_{i=1}^{n}(1-p_i) \leq 1 - \sum_{i=1}^{n} p_i + \sum_{1\leq i<j\leq n} p_i p_j.$$

Proof. We prove the result by induction on n. The result holds true for $n = 1, 2$. Let the result holds true for $n = m$. We prove the result for $n = m+1$. Therefore,

$$\begin{aligned}
\prod_{i=1}^{m+1}(1-p_i) &= \prod_{i=1}^{m}(1-p_i)(1-p_{m+1}) \\
&\leq (1 - \sum_{i=1}^{m} p_i + \sum_{1\leq i<j\leq m} p_i p_j)(1-p_{m+1}) \\
&= (1 - \sum_{i=1}^{m+1} p_i + \sum_{1\leq i<j\leq m+1} p_i p_j) + \sum_{1\leq i<j\leq m} p_i p_j p_{m+1} \\
&\leq (1 - \sum_{i=1}^{m+1} p_i + \sum_{1\leq i<j\leq m+1} p_i p_j),
\end{aligned}$$

which proves the result for $n = m+1$ and hence we prove the result. □

6.3 The CENCPP* Mode

This section defines a generic CENC construction that we call CENCPP*. Standing on the shoulders of existing constructions, we start with the necessary details of SoEM and CENC.

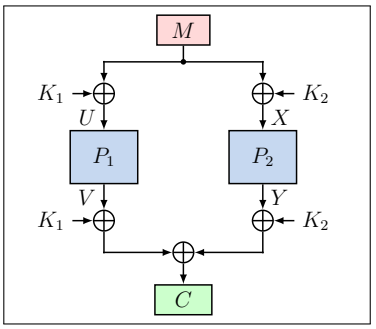


FIGURE 6.1: The construction SoEM22 by Chen et al. [17]

6.3.1 SoEM

At CRYPTO'19, Chen et al. [17] proposed SoEM (Sum of Even-Mansour constructions) and SoKAC (Sum of Key-alternating Ciphers). Both designs represent fixed-length PRFs which they provided analyses for up to $O(2^{2n/3})$ queries for both. An improved analysis that showed subtleties of the proof of SoKAC21 was presented later in [173]. The former sums the results of two single-round Even-Mansour ciphers; the latter is a variant of Encrypted Davies-Meyer [178] from public instead of keyed primitives.

Chen et al. parameterised their constructions as SoEM$\lambda\kappa$ and SoKAC$\lambda\kappa$, where λ denoted the number of permutations, and κ the number of keys. Figure 6.1 illustrates SoEM22, which will be relevant in this chapter. Both modes need two calls to the independent permutations. Moreover, SoEM demanded two independent keys. Chen et al. also studied SoEM12 with a single permutation: $P(M \oplus K_1) \oplus K_1 \oplus P(M \oplus K_2) \oplus K_2$, and SoKAC12 as $P(P(M \oplus K_1) \oplus K_2) \oplus K_1 \oplus P(M \oplus K_1) \oplus K_2$, and showed distinguishers with $O(2^{n/2})$ queries for both. However, Chakraborti et al. [174] showed that the distinguisher on the latter may be incorrect and SoKAC12 could offer a security bound of $\Omega(2^{2n/3})$ (cf. [177]).

6.3.2 CENC

CENC is a nonce-based block-cipher mode that generalises the sum of permutations by Iwata [16]. It uses the nonce concatenated with a counter as block-cipher input, splits each sequence of w message blocks into chunks, and processes them by XORP. In XORP, the message M is split into w blocks of n bits, for a small positive integer w. Let n, ν, μ be integers such that $n = \nu + \mu$ and $w+1 \leq 2^{\mu}$. Let $E : \mathcal{K} \times \mathbb{F}_2^n \to \mathbb{F}_2^n$ be a block-cipher, and let $\mathcal{N} = \mathbb{F}_2^{\nu}$ be a nonce space. The remaining μ input bits are used for a counter. Let $K \in \mathcal{K}$ be a secret key and $N \in \mathcal{N}$ be a nonce. Then,

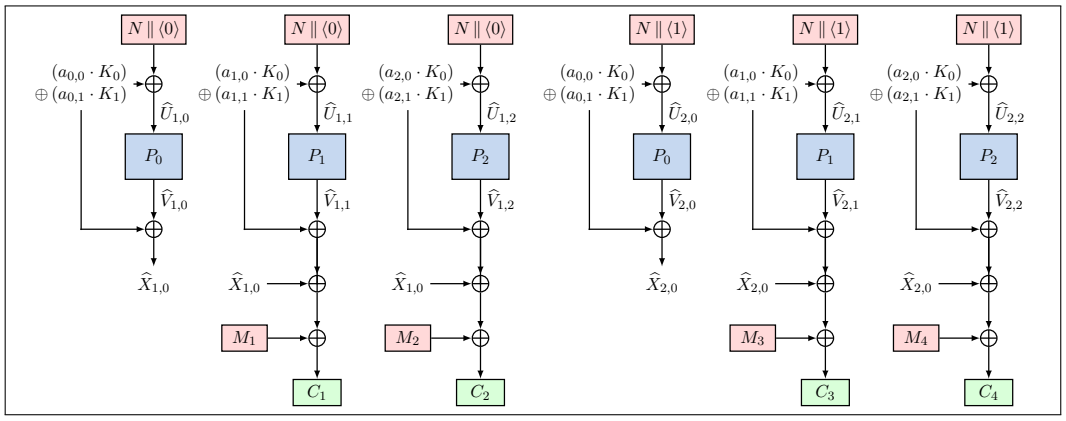


FIGURE 6.2: Encryption of a message $M = (M_1, \ldots, M_4)$ with $\mathsf{CENCPP}^*[(P_0, P_1, P_2), 2]_{K_0,K_1}$. The final chunk is truncated if its length is less than $2n$ bits. N is a nonce, K_0 and K_1 are independent secret keys and P_0, P_1, and P_2 independent permutations. In this figure, $\widehat{U}_{i,j}$ (resp. $\widehat{V}_{i,j}$) denotes the permutation input (resp. output) for the j-th invocation of the permutation in the i-th chunk. For the i-th chunk, $\widehat{X}_{i,0}$ denotes $\widehat{V}_{i,0} \oplus a_{i,0}K_0 \oplus a_{i,1}K_1$.

$\mathsf{XORP}[E_K, w](N, s)$ computes a key stream $S_1 \| \ldots \| S_w$ as

$$S_i \stackrel{\text{def}}{=} E_K(N \| \langle s \rangle_\mu) \oplus E_K(N \| \langle s + i \rangle_\mu), \text{ for } i \in [w].$$

Thus, it makes $w + 1$ block-cipher calls with pairwise distinct inputs, where $E_K(X \| \langle s \rangle_\mu)$ with the starting value s of the counter is XORed to each of the other blocks. $\mathsf{XORP}[E_K, w]$ can be used as a length-restricted encryption scheme by XORing its output to a message M of $|M| \leq n \cdot w$ bits. The final chunk is simply truncated to the length of the final message block. We slightly adapt the definition by [16, 179] to

$$\mathsf{XORP}[E_K, w] : \mathcal{N} \times \mathbb{F}_2^\mu \to (\mathbb{F}_2)^{n \cdot w},$$

where $\mathsf{XORP}[E_K, w](N, i)$ uses $N \| \langle i \rangle_\mu$, $N \| \langle i + 1 \rangle_\mu$, ... as inputs to E_K.
CENC concatenates several instances of $\mathsf{XORP}[E_K, w]$ with pair-wise distinct inputs. Let $M \in \mathbb{F}_2^*$ be a message s. t. $M_1 \cdots M_m \leftarrow_n M$. Let $\ell = \lceil m/w \rceil$ denote the number of chunks. It must hold that $\ell \cdot (w + 1) < 2^\mu$. Then

$$\mathsf{CENC}[E_K, w](N, M) \stackrel{\text{def}}{=} \mathsf{msb}_{|M|}\left(\|_{i=0}^{\ell-1} \mathsf{XORP}[E_K, w]\,(N, i \cdot (w + 1)) \right) \oplus M.$$

6.3.3 CENCPP*

In the following, we adapt CENC to the public-permutation setting. Let $\mathbf{A} = (a_{ij})$ be a $(w+1) \times 2$ dimensional matrix such that each of its elements a_{ij} is an n-bit binary string. Let $P_0, \ldots, P_w \in \mathsf{Perm}(\mathbb{F}_2^n)$ be permutations, and let $K_0, K_1 \in \mathbb{F}_2^n$ be independent secret keys. We define $\mathbf{P} \stackrel{\text{def}}{=} (P_0, \ldots, P_w)$ as shorthand form. Furthermore, $\mathcal{D} \subseteq \mathbb{F}_2^\mu$ be a set of domains, s. t. $n = \nu + \mu$. For brevity, we define a key vector $\mathbf{K} = (K_0, K_1)$. We combine both keys K_0 and K_1 for the individual permutations as $(a_{i,0} \cdot K_0) \oplus (a_{i,1} \cdot K_1)$ to generate the i-th round key K'_i, for all $i \in [0..w]$. In matrix notation, we write this as follows:

$$\mathbf{A} \cdot \mathbf{K} = \begin{bmatrix} a_{0,0} & a_{0,1} \\ a_{1,0} & a_{1,1} \\ \ldots & \ldots \\ a_{w,0} & a_{w,1} \end{bmatrix} \cdot \begin{bmatrix} K_0 \\ K_1 \end{bmatrix} = \begin{bmatrix} K'_0 \\ K'_1 \\ \vdots \\ K'_w \end{bmatrix}.$$

We call $\mathbf{A}$ the *key-scheduling matrix*. We adapt XORP to XORPP* to note that it is based on the XOR of **public** permutations. For a key-scheduling matrix $\mathbf{A}$ of dimension $(w+1)\times 2$, we define XORPP* $[\mathbf{P}, w, \mathbf{A}] : (\mathbb{F}_2^n)^2 \times \mathbb{F}_2^n \to (\mathbb{F}_2^n)^w$, instantiated with $w+1$ permutations $P_0, \ldots, P_w$, a key space $(\mathbb{F}_2^n)^2$ and the key-scheduling matrix $\mathbf{A}$. We write XORPP* as short for XORPP* $[\mathbf{P}, w, \mathbf{A}]$ when w, key-scheduling matrix $\mathbf{A}$ and the permutations $\mathbf{P}$ are clear from the context. Given that the permutations are independent, CENCPP* uses the same input $(N \,\|\, \langle i \rangle_\mu)$ for each permutation in one call of XORPP*. We define encryption and decryption of the nonce-based mode CENCPP* as given in Algorithm 7 and a pictorial depiction of encryption of a four-block message using CENCPP* has been given in Figure 6.2.

6.3.4 Discussion

Further constructions with beyond-birthday security from public permutations are naturally possible. However, our proposal CENCPP* seems efficient. Instantiating CENC with a two-round Even-Mansour construction could be a generic approach that can provide roughly the security of the primitive, i.e., $2n/3$ bits, and would employ $\lceil 2\frac{w+1}{w} \rceil$ calls to the permutation for w message blocks. In their proposal of AES-PRF, Mennink and Neves increased the performance of their construction [180] by instantiating it with five-round AES. However, its security margin is thin [181] and improved cryptanalysis could break it in the near future.

Algorithm 7 Specification of CENCPP*

```
101: function CENCPP*[P, w, A].E_K(N, M)
102:     M_1 ··· M_m ←_n M
103:     ℓ ← ⌈m/w⌉
104:     for i ← 0..ℓ − 1 do
105:         j ← i · w
106:         (S_{j+1} ‖ ··· ‖ S_{j+w})
107:             ← XORPP*[P, w, A]_K(N ‖ ⟨i⟩_μ)
108:         for k ← j + 1..j + w do
109:             C_k ← msb_{|M_k|}(S_k) ⊕ M_k
110:     return (C_1 ‖ ··· ‖ C_m)

201: function CENCPP*[P, w, A].D_K(N, C)
202:     return CENCPP*[P, w, A].E_K(N, C)

301: function XORPP*[P, w, A]_K(I)
302:     (K_0, K_1) ← K
303:     (P_0, ..., P_w) ← P
304:     L_0 ← (a_{0,0} · K_0) ⊕ (a_{0,1} · K_1)
305:     Û_0 ← I ⊕ L_0
306:     X̂_0 ← P_0(Û_0) ⊕ L_0
307:     for α ← 1..w do
308:         L_α ← (a_{α,0} · K_0) ⊕ (a_{α,1} · K_1)
309:         Û_α ← I ⊕ L_α
310:         X̂_α ← P_α(Û_α) ⊕ L_α
311:         O_α ← X̂_α ⊕ X̂_0
312:     return O ← (O_1 ‖ ··· ‖ O_w)
```

More related works exist in the secret-permutation setting. Cogliati and Seurin [182] showed that a variant of EDM with a single keyed permutation – that is $E_K(E_K(M) \oplus M)$ – possesses roughly $O(2^{2n/3})$ security. The work by Guo et al. [183] followed this direction, showing $O(2^{2n/3}/n)$ security for the single-permutation variants of EDM and its dual EDMD– $E_K(E_K(M)) \oplus E_K(M)$. Moreover, they proved a similar security result also for the sum from a single permutation and its inverse, SUMPIP: $E_K(M) \oplus E_K^{-1}(M)$. The Decrypted Wegman-Carter Davies-Meyer construction [184] would also possess a security bound of $O(2^{2n/3})$ but limited the input space to $2n/3$ bits. SUMPIP could retain beyond-birthday-bound security with public permutations, i.e.

$$P(M \oplus K_1) \oplus K_1 \oplus P^{-1}(M \oplus K_2) \oplus K_2$$

could be secure beyond $O(2^{n/2})$ queries when using a public primitive P.
MACs from public permutations obtained a high level of attention recently. In [174], Chakraborti et al. proposed a PDM-MAC

$$P^{-1}(P(K \oplus M) \oplus K \oplus 2K \oplus M) \oplus 2K\,,$$

which eliminated the need for a second key from SUMPIP. They also considered a nonce-based variable-input-length PRF, PDM*MAC, and a single-key version 1K-PDM*MAC. All of their constructions maintained a security bound of $O(2^{2n/3})$. However, the instantiations needed both forward and inverse of the permutation, which is less practical for a permutation-based design compared to the construction that invokes the permutation only in forward direction.

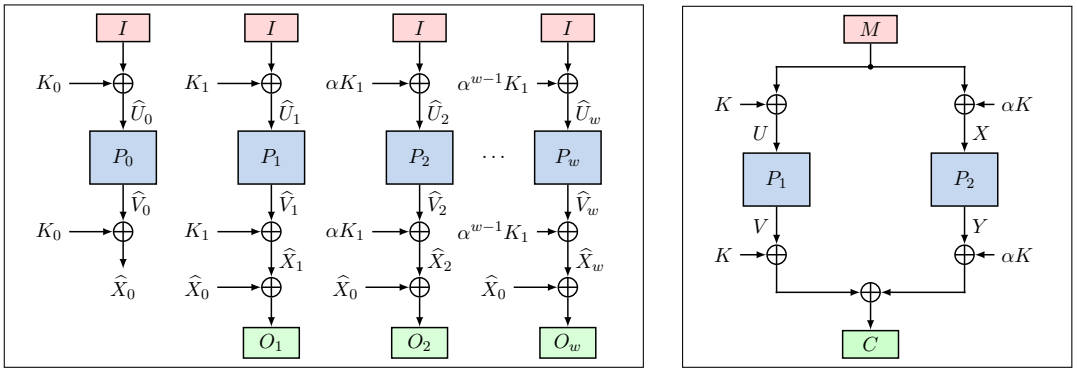


FIGURE 6.3: Example of using a weak key schedule for XORPP* (left) and SoEM′ (right)

In [177], Dutta et al. studied the security pEDM, a strongly related variant of SoKAC12, which uses the single permutation only in forward direction:

$$P(P(M \oplus K_1) \oplus (M \oplus K_1) \oplus K_2) \oplus K_1\,,$$

also with $O(2^{2n/3})$ security. These constructions consider related aspects, but are fixed-output-length PRFs, whereas CENCPP* can encrypt messages of variable lengths. Comparing CENCPP* with $w = 1$, pEDM has the advantage of using only a single primitive. Though, the latter can evaluate the primitive calls in parallel and allows a better rate for greater w, whereas for an encryption with the latter, similar arguments as for a counter mode with a two-round Even-Mansour construction would hold.

6.4 Birthday-bound Distinguisher on CENCPP* with Weak Key Scheduling

To derive the i-th round key L_i of CENCPP*, we have $L_i = (a_{i,0} \cdot K_0) \oplus (a_{i,1} \cdot K_1)$ for all $i \in [0..w]$, where $\mathbf{A} = (a_{i,j}) \in \{0,1\}^n$ is the key-scheduling matrix of dimension $(w+1) \times 2$ and K_0, K_1 are two independent n-bit keys. Using SoEM as a base, it is tempting to use a key scheduling of K_0, K_1, αK_1, $\alpha^2 K_1$, ..., which omits the addition of K_0 for all subsequent permutation calls. In matrix form, this key

scheduling would produce

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & \alpha & \cdots & \alpha^{w-1} \end{bmatrix}^{\top}}_{\mathbf{A}^{\top}} \cdot \begin{bmatrix} K_0 \\ K_1 \end{bmatrix} .$$

While the latter appears much simpler, after transposing its matrix form to $w+1$ rows, it contains dependent rows. Let two dependent rows be denoted as $\mathbf{A}_i$ and $\mathbf{A}_j$ in the key-scheduling matrix $\mathbf{A}$ such that they are linearly dependent, i.e., $\mathbf{A}_i = \alpha \mathbf{A}_j$ for some non-zero $\alpha \in \{0,1\}^n$. Then, we have $L_i = \alpha L_j$ for some $\alpha \in \{0,1\}^n \setminus \{0^n\}$. We use the idea of cancelling dependent outputs and thus reduce the distinguishing problem to that for single-key SoEM. Since the steps are not intuitive, we illustrate the birthday-bound distinguisher of CENCPP* in the following. First, we show that we can reduce the security of CENCPP* to the security of SoEM with the key usage of $(L_i, \alpha L_i)$ for some non-zero $\alpha \in \{0,1\}^n$ when $\mathbf{A}_i$ and $\mathbf{A}_j$ rows of $\mathbf{A}$ are linearly dependent. We denote this variant of SoEM as $\mathsf{SoEM}' \stackrel{\text{def}}{=} \mathsf{SoEM}[P_i, P_j]_{L_i, \alpha L_i}$.

6.4.1 Reduction to SoEM′

Suppose, $\mathbf{D}$ is an information-theoretic distinguisher on SoEM′ and $\tau = \{K\} \cup \tau_p \cup \tau_c$ is a transcript, consisting of the key, the primitive-query transcript τ_p with q_p primitive queries and their corresponding responses (U^i, V^i) to P_1 and (X^k, Y^k) to P_2 each, as well as the construction-query transcript τ_c with q_c construction queries and their corresponding responses (M^j, C^j). After the interaction, $\mathbf{D}$ is given τ, including the key $K \xleftarrow{\$} \mathbb{F}_2^n$, and sees $C = W \oplus Z$ where

$$W \stackrel{\text{def}}{=} P_1(M \oplus K) \oplus K \quad \text{and} \quad Z \stackrel{\text{def}}{=} P_2(M \oplus (\alpha \cdot K)) \oplus (\alpha \cdot K) .$$

In comparison, a distinguisher $\mathbf{D}'$ on CENCPP* $[P_0, P_i, P_j]_{K_0, K_1}$ with key schedule as above can compute $C_i \oplus C_j = (X_i \oplus X_0) \oplus (X_j \oplus X_0) = W \oplus Z = C$. Thus,

$$\mathbf{Adv}^{\mathsf{CENCPP}^*}_{\mathsf{PRF}}(\mathbf{D}') \geq \mathbf{Adv}^{\mathsf{SoEM}'}_{\mathsf{PRF}}(\mathbf{D}) ,$$

where $\mathbf{D}$ and $\mathbf{D}'$ ask the same number of construction queries q_c and primitive queries q_p to each of the primitives. Note that the distinguisher $\mathbf{D}'$ knows the values of i and j from the knowledge of the key-scheduling algorithm.

6.4.2 Birthday-bound Attack on SoEM′

Let $\mathcal{U}$ and $\mathcal{V}$ be two subspaces of $\mathbb{F}_2^n$. Then, for every $\alpha \in \mathbb{F}_2^n$, $\mathcal{U}+\mathcal{V} \stackrel{\text{def}}{=} \{u+v | u \in \mathcal{U}, v \in \mathcal{V}\}$ and $\alpha \cdot \mathcal{V} \stackrel{\text{def}}{=} \{\alpha \cdot v | v \in \mathcal{V}\}$ are also subspaces. We write $\mathbf{0}$ and $\mathbf{1}$ for the neutral elements of addition and multiplication, respectively. If $\{x_1, x_2, \cdots, x_{n/2}\}$ is a basis of $\mathcal{V}$, then $\{\alpha \cdot x_1, \alpha \cdot x_2, \cdots, \alpha \cdot x_{n/2}\}$ is also a basis of $\alpha \cdot \mathcal{V}$, where $\alpha \neq \mathbf{0}$.

Fact 1. Let $\mathcal{U}$ and $\mathcal{V}$ be two subspaces of $\mathbb{F}_{2^n}$. If their intersection contains only the zero element $\mathcal{U} \cap \mathcal{V} = \{\mathbf{0}\}$, we say that $\mathcal{U}$ and $\mathcal{V}$ have zero intersection. If both have zero intersection, it holds that $\dim(\mathcal{U}+\mathcal{V}) = \dim(\mathcal{U}) + \dim(\mathcal{V})$. Equivalently, one can say that the basis elements of $\mathcal{U}$ and $\mathcal{V}$ are linearly independent.

Theorem 6.4. Let $\alpha \notin \{\mathbf{0}, \mathbf{1}\}$. For every $1 \leq i \leq n/2$, there exists a subspace $\mathcal{V} \subseteq \mathbb{F}_2^n$ with $\dim(\mathcal{V}) = i$ such that $\mathcal{V}$ and $\alpha \cdot \mathcal{V}$ have zero intersection. In particular, there is a subspace $\mathcal{V}$ of dimension $n/2$ such that $\mathcal{V} + \alpha \cdot \mathcal{V} = \mathbb{F}_2^n$.

Proof. We prove Theorem 6.4 by induction on i. For $i = 1$, the statement is obvious by choosing non-zero x_1. For $1 \leq i < n/2$, suppose, we have picked $x_1, x_2, \cdots, x_i$ such that all elements from $\{x_1, x_2, \cdots, x_i, \alpha \cdot x_1, \alpha \cdot x_2, \cdots \alpha \cdot x_i\}$ are linearly independent. Let

$$\mathcal{S}_i \stackrel{\text{def}}{=} \mathbf{span}(\{x_1, x_2, \cdots, x_i, \alpha \cdot x_1, \alpha \cdot x_2, \cdots, \alpha \cdot x_i\}),$$

i.e., its span. Moreover, we define $\mathcal{T}_i$ as short form of

$$\mathcal{T}_i \stackrel{\text{def}}{=} \mathcal{S}_i \cup \left(\alpha^{-1} \cdot \mathcal{S}_i\right) \cup \left((1+\alpha)^{-1} \cdot \mathcal{S}_i\right).$$

It holds that $|\mathcal{T}_i| \leq 3 \cdot 2^{n-2} < 2^n$. When we choose a new element $x_{i+1} \notin \mathcal{T}_i$, it follows from the definition of $\mathcal{T}_i$ that x_{i+1}, $\alpha \cdot x_{i+1}$ and $(1+\alpha) \cdot x_{i+1}$ are not in $\mathcal{S}_i$. Hence, the elements

$$\{x_1, x_2, \cdots, x_{i+1}, \alpha \cdot x_1, \alpha \cdot x_2, \cdots, \alpha \cdot x_{i+1}\}$$

are linearly independent, which concludes the proof. Note that such a basis can be constructed efficiently, element by element. □

Distinguisher on SoEM′: Next, we demonstrate a distinguisher on SoEM′. Given the observation above, we can first construct a vector space $\mathcal{X}$ of dimension $n/2$ such that $\mathcal{X} + (1+\alpha) \cdot \mathcal{X} = \mathbb{F}_2^n$. Let $\mathcal{M} = (1+\alpha)^{-1} \cdot \mathcal{X}$. So, $\mathcal{M} + \mathcal{X} = \mathbb{F}_2^n$ and

hence there exists $X \in \mathcal{X}$ and $M \in \mathcal{M}$ with $M + X = \alpha \cdot K$. Let $\mathcal{U} = \alpha^{-1} \cdot \mathcal{X}$. Then

$$\mathcal{U} = \alpha^{-1} \cdot (1 + \alpha) \cdot \mathcal{M} = (1 + \alpha^{-1}) \cdot M\,.$$

Thus, $M + K = \alpha^{-1} \cdot X + (1 + \alpha^{-1}) \cdot M \in \mathcal{U}$ and there exists $M \in \mathcal{M}, U \in \mathcal{U}$, and $X \in \mathcal{X}$ such that $M \oplus U = K$ and $M \oplus X = \alpha K$.
Let $P_1(U) = V$ and $P_2(X) = Y$. Then, $C = \mathsf{SoEM}'(M) = (1 \oplus \alpha) \cdot K \oplus V \oplus Y$. We use shorthand notations $V^{\oplus c}$, $Y^{\oplus c}$ and $C^{\oplus c}$ to denote $P_1(U \oplus c)$, $P_2(X \oplus c)$ and $\mathsf{SoEM}'(M \oplus c)$ respectively for some non-zero $c \in \{0,1\}^n$. It is easy to see that for any c, it holds that

$$C^{\oplus c} = (1 \oplus \alpha) \cdot K \oplus V^{\oplus c} \oplus Y^{\oplus c}$$

and hence $C \oplus C^{\oplus c} = (V \oplus V^{\oplus c}) \oplus (Y \oplus Y^{\oplus c})$. We use this observation to complete our attack. Suppose that c and d are two distinct constants outside of $\mathcal{U}$, $\mathcal{X}$, and $\mathcal{M}$. Then, the distinguisher can proceed as follows:

1. It queries all values $U_i \in \mathcal{U}$, $U_i \oplus c$ and $U_i \oplus d$ to its primitive oracle P_1, and stores them together with the corresponding responses V_i, $V_i^{\oplus c}$ and $V_i^{\oplus d}$.

2. Similarly, it queries all values $X_i \in \mathcal{X}$, $X_i \oplus c$ and $X_i \oplus d$ to its primitive oracle P_2, and stores them together with the corresponding responses Y_i, $Y_i^{\oplus c}$ and $Y_i^{\oplus d}$.

3. Moreover, it queries all values $M_i \in \mathcal{M}$, $M_i \oplus c$ and $M_i \oplus d$ to its construction oracle, and stores them together with the corresponding responses C_i, $C_i^{\oplus c}$ and $C_i^{\oplus d}$.

4. After making all queries as described above, it looks for triple (i, j, k) such that the following two equalities hold:

   4.1 $C_i \oplus C_i^{\oplus c} = (V_j \oplus V_j^{\oplus c}) \oplus (Y_k \oplus Y_k^{\oplus c})$.

   4.2 $C_i \oplus C_i^{\oplus d} = (V_j \oplus V_j^{\oplus d}) \oplus (Y_k \oplus Y_k^{\oplus d})$.

5. If there exists such triple (i, j, k), it outputs real and random otherwise.

6.5 Security Analysis of CENCPP*

This section studies the NE security of CENCPP*. Prior, we briefly revisit that of CENC.

6.5.1 Recalling the Security of CENC

The security of XORP: In [16], Iwata showed that CENC[w] is secure for up to $2^{2n/3}/w$ message blocks as long as E_K is a secure block-cipher. At Dagstuhl'07 [185], he added an attack that needed $2^n/w$ queries, and showed $O(2^n/w)$ security if the total number of primitive calls remained below $\sigma < 2^{n/2}$. He conjectured that CENC may be secure for up to $2^n/w$ blocks. In [179], Iwata et al. confirmed that conjecture by a simple corollary from Patarin. We briefly recall their conclusion. In [186, Theorem 6], Patarin showed the indistinguishability for the sum of multiple independent secret permutations under assumptions on the validity of the Mirror Theory. [179] adapted this bound to upper bound the PRF security of XORP:

$$\mathbf{Adv}^{\mathsf{XORP}}_{\mathsf{PRF}}(q_c, t) \leq \frac{w^2 q}{2^n} + \mathbf{Adv}^{E}_{\mathsf{PRP}}((w+1)q_c, t). \tag{6.1}$$

Theorem 3 in [179] conjectured for m being a multiple of w, where m is the maximum number of message blocks queried:

$$\mathbf{Adv}^{\mathsf{CENC}}_{\mathsf{NE}}(q_c, m, t) \leq \frac{mwq_c}{2^n} + \mathbf{Adv}^{E}_{\mathsf{PRP}}\left(\frac{w+1}{w}mq_c, t\right). \tag{6.2}$$

Note that in Equation (6.1) and Equation (6.2), the authors considered computationally bounded distinguishers for which we included the time parameter t. Thus, CENC provided a convenient trade-off of $w+1$ calls per w message blocks with security for up to $2^n/w$ calls to E_K. The proof sketch by [179] reduced the security of CENC to the proof of the sum of two permutations. At that time, the latter analysis relied on recursive arguments of Patarin's Mirror Theory that were subject to controversies. The work by Bhattacharya and Nandi [187] proved similar security for the generalised sum of permutations and CENC using the χ^2 method [188].

6.5.2 The Security of CENCPP*

In the following, let n, w be positive integers, $P_0, \ldots, P_w \xleftarrow{\$} \mathsf{Perm}(\mathbb{F}_2^n)$ be independent public permutations, $K_0, K_1 \xleftarrow{\$} \mathbb{F}_2^n$ be a pair of n-bit independent secret keys

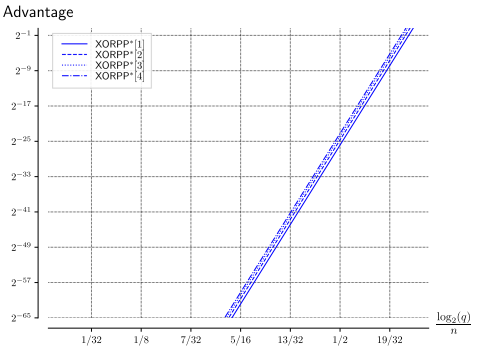


FIGURE 6.4: Security of $\mathsf{XORPP}^*[w]$ with varying w and $n = 64$

which are sampled uniformly at random from $\mathbb{F}_2^n$. Let $\mathbf{A}$ be the key-scheduling matrix of dimension $(w+1) \times 2$ such that each entry is an n-bit binary string. We write $\mathbf{K} = (K_0, K_1)$ and $\mathbf{P} = (P_0, \ldots, P_w)$ for brevity. Again, we conduct a two-step analysis, where we consider (1) the PRF security of XORPP^* $[\mathbf{P}, w, \mathbf{A}]_{\mathbf{K}}$ and (2) the NE security of $\mathsf{CENCPP}^*[\mathbf{P}, w, \mathbf{A}]_{\mathbf{K}}$. Since the matrix $\mathbf{A}$ is public, we omit it from the notation $\mathsf{XORPP}^*[\mathbf{P}, w, \mathbf{A}]_{\mathbf{K}}$ and $\mathsf{CENCPP}^*[\mathbf{P}, w, \mathbf{A}]_{\mathbf{K}}$ and simply write $\mathsf{XORPP}^*[\mathbf{P}, w]_{\mathbf{K}}$ and $\mathsf{CENCPP}^*[\mathbf{P}, w]_{\mathbf{K}}$, respectively. For simplicity of notation, we write XORPP^* $[\mathbf{P}, w]_{\mathbf{K}}$ as XORPP^* and $\mathsf{CENCPP}^*[\mathbf{P}, w]_{\mathbf{K}}$ as CENCPP^*.

Theorem 6.5. It holds that $\mathbf{Adv}^{\mathsf{CENCPP}^*}_{\mathsf{NE}}(q_p, q_c, m) \leq \mathbf{Adv}^{\mathsf{XORPP}^*}_{\mathsf{PRF}}\left(q_p, \frac{m}{w} q_c\right)$.

Proof. Recall that, m is the maximum number of message blocks in all queries. Therefore, for a maximal number of message chunks $\ell = \lceil m/w \rceil$, CENCPP^* consists of the application of ℓ instances of XORPP^*. We can replace XORPP^* by a random function ρ at the cost of

$$\mathbf{Adv}^{\mathsf{XORPP}^*}_{\mathsf{PRF}}\left(q_p, \frac{m}{w} q_c\right) .$$

Since the resulting construction is indistinguishable from random bits, Theorem 6.5 follows.

Theorem 6.6. Let $\mathbf{A}$ be a matrix of $(w+1) \times 2$ entries such that each of its elements is an n-bit binary string and all its rows are pairwise linearly independent. Let $q_p + (w+1)q_c \leq 2^n/2(w+1)$. It holds that

$$\mathbf{Adv}^{\mathsf{XORPP}^*}_{\mathsf{PRF}}(q_p, q_c) \leq \frac{(w+1)^2 q_p^2 q_c}{2^{2n+1}} + \frac{4w q_p^2 q_c}{2^{2n}} + \frac{2w q_c^2 q_p}{2^{2n}} + \frac{w^2 q_c^2 q_p}{2^{2n}} + \frac{3w^2 q_p^2 q_c}{2^{2n+1}} + \frac{3w^3 q_p^2 q_c^2}{2^{3n}} + \frac{(w+1)^2 q_c (q_p + q_c)^2}{2^{2n}} .$$

The security for varying values of w is illustrated in Figure 6.4.

Corollary 6.7. CENCPP* security results by combining Theorem 6.5 and Theorem 6.6 as follows:

$$\mathbf{Adv}^{\mathsf{CENCPP}^*}_{\mathsf{NE}}(q_p, q_c, m) \leq \frac{2wmq_p^2q_c}{2^{2n}} + \frac{4mq_p^2q_c}{2^{2n}} + \frac{2q_pm^2}{w2^{2n}} + \frac{q_pm^2}{2^{2n}} + \frac{3wmq_p^2q_c}{2^{2n+1}} + \frac{3wm^2q_p^2q_c^2}{2^{3n}} + \frac{4wmq_cq_p^2 + 8m^2q_pq_c^2}{2^{2n}} + \frac{4m^3q_c^3}{w2^{2n}},$$

where m is the maximum number of message blocks among all q_c queries and we used $2w \geq w + 1$.

Proof of Theorem 6.6. We fix a non-trivial information-theoretic deterministic distinguisher $\mathbf{D}$ who is given access to $(w + 2)$ oracles in either of the real or ideal world. In the real world, $\mathbf{D}$ is given access to the construction oracle $\mathsf{XORPP}^*[\mathbf{P}, w]_{\mathbf{K}}$ where $\mathbf{K} = (K_0, K_1)$ is a pair of n-bit random keys and $\mathbf{P} = (P_0, P_1, \ldots, P_w)$ is a tuple of $w+1$ many n-bit independent random permutations, and the primitive oracles $\mathbf{P} = (P_0, \ldots, P_w)$. In the ideal world, $\mathbf{D}$ is given access to a random function, which answers each query of $\mathbf{D}$ by w blocks of n bits uniform and independent random strings $\mathbf{O} = (O_1, \ldots, O_w)$ and to the tuple of $w+1$ many independent n-bit random permutations $\mathbf{P} = (P_0, P_1, \ldots, P_w)$. Query to the construction oracle is called the construction query and to that of the primitive oracle is called the primitive query. We assume that $\mathbf{D}$ can ask exactly q_c construction queries and q_p primitive queries to each of primitive oracle P_α, $\alpha \in [0..w]$. For queries to each of the primitive oracle P_α, $\mathbf{D}$ can either make a forward query U_α to its primitive oracle P_α and receives response V_α or can make an inverse query V_α to P_α^{-1} and receives response U_α. We summarise the interaction of the distinguisher $\mathbf{D}$ with the oracles in a transcript τ which is partitioned into $\tau = \tau_c \cup \tau_0 \cup \ldots \cup \tau_w$, where each partial transcript captures the queries and responses from a particular oracle. The construction transcript contains the queries to and responses from the construction oracle: $\tau_c = \{(I^1, \mathbf{O}^1), \ldots, (I^{q_c}, \mathbf{O}^{q_c})\}$, where $\mathbf{O}^i = (O_1^i, \ldots, O_w^i)$. The primitive transcripts $\tau_\alpha = \{(U_\alpha^1, V_\alpha^1), \ldots, (U_\alpha^{q_p}, V_\alpha^{q_p})\}$ contain exactly the queries to and responses from permutation P_α for all $\alpha \in [0..w]$. Since $\mathbf{D}$ is non-trivial, we assume that τ does not contain duplicate elements. After the interaction, we release the keys K_0, K_1 to the distinguisher before it outputs its decision bit. In the real world (K_0, K_1) are the keys used in the construction, whereas in the ideal world they are sampled uniformly at random. Hence, the transcript τ becomes $\tau = \tau_c \cup \tau_0 \cup \ldots \cup \tau_w \cup \{(K_0, K_1)\}$. With the help of the transcript τ, $\mathbf{D}$ can

compute the all the inputs $\widehat{U}^i_\alpha$ to the permutations P_α for q_c construction queries using the following equation

$$\widehat{U}^i_\alpha \stackrel{\text{def}}{=} I^i \oplus a_{\alpha,0} \cdot K_0 \oplus a_{\alpha,1} \cdot K_1 \,, \tag{6.3}$$

where $\alpha \in [0..w]$ and $i \in [q_c]$. We partition the set of all attainable transcripts Att into two disjoint sets of GoodT and BadT that represent good and bad transcripts. We denote by Θ_{real} and Θ_{ideal} random variables that represent the distribution of transcripts in the real and the ideal world, respectively.

Bad Events: Let $\tau = \tau_c \cup \tau_0 \cup \ldots \cup \tau_w \cup \{(K_0, K_1)\}$ be an attainable transcript. Since, the distinguisher is given the keys $\mathbf{K}$, it can compute all the permutation inputs $(\widehat{U}^i_\alpha)_{i\in[q_c],\alpha\in[0..w]}$ using Equation (6.3). Before defining the bad events for XORPP^*, we give a brief rationale for them.

Rationale. For $w+1$ n-bit permutations $(P_0, \ldots, P_w)$, we denote $P_\alpha(\widehat{U}^i_\alpha)$ as $\widehat{V}^i_\alpha$ for $\alpha \in [0..w]$. Then the construction for i-th query leads to the following system of equations:

$$\mathbb{E}_i = \begin{cases} \widehat{V}^i_0 \oplus \widehat{V}^i_1 = O^i_1 \oplus (a_{0,0} \oplus a_{1,0}) \cdot K_0 \oplus (a_{0,1} \oplus a_{1,1}) \cdot K_1 \\ \widehat{V}^i_0 \oplus \widehat{V}^i_2 = O^i_2 \oplus (a_{0,0} \oplus a_{2,0}) \cdot K_0 \oplus (a_{0,1} \oplus a_{2,1}) \cdot K_1 \\ \vdots \quad \vdots \quad \vdots \quad \vdots \\ \widehat{V}^i_0 \oplus \widehat{V}^i_w = O^i_w \oplus (a_{0,0} \oplus a_{w,0}) \cdot K_0 \oplus (a_{0,1} \oplus a_{w,1}) \cdot K_1, \end{cases}$$

A trivial bad event is, if for i-th construction query, both inputs to the permutation simultaneously collide with two primitive inputs, i.e., $\widehat{U}^i_\alpha = U^j_\alpha$ and $\widehat{U}^i_\beta = U^k_\beta$ for $\alpha \neq \beta \in [0..w]$. If $\widehat{U}^i_\alpha$ collides with U^j_α for some $\alpha \in [0..w]$, then this event uniquely determines the value of the permutation output for the remaining variables in $\mathbb{E}_i$. In the real world, such a collision uniquely determines the rest of the variables, whereas this property does not hold in the ideal world. A bad event occurs if any of such determined variables collides with any primitive query output. Assume that the i-th and j-th construction query, respectively, $\widehat{U}^i_0$ and $\widehat{U}^j_0$, collide with some primitive input each. In turn, this uniquely determines the value of the permutation output for the remaining variables in the respective system of equations. A bad event occurs if any two of such determined variables collide with each other. A similar situation arises when for two construction queries, let them be the i-th and j-th construction query, respectively, $\widehat{U}^i_\alpha$ and $\widehat{U}^j_\beta$ collide with some primitive

input for some $\alpha, \beta \in [0..w]$, and the determined variables collides. We say that τ is bad if any of the following bad events hold.

1. *Two inputs to the permutations for a construction query simultaneously collide with the input of corresponding two primitive queries.*
   - bad_1: $\exists i \in [q_c], j, k \in [q_p]$, and distinct permutation indices $\alpha, \beta \in [0..w]$ such that $(\widehat{U}^i_\alpha = U^j_\alpha) \wedge (\widehat{U}^i_\beta = U^k_\beta)$.
2. *For a construction query, one of the inputs collides with the input of a primitive query, which lets the output of another permutation call of the same construction query collide with the output of another primitive query.*
   - bad_2: $\exists i \in [q_c], j, k \in [q_p]$, and permutation index $\alpha \in [w]$ such that $(\widehat{U}^i_0 = U^j_0) \wedge (V^j_0 \oplus O^i_\alpha \oplus (a_{0,0} \oplus a_{\alpha,0}) \cdot K_0 \oplus (a_{0,1} \oplus a_{\alpha,1}) \cdot K_1 = V^k_\alpha)$.
   - bad_3: $\exists i \in [q_c], j, k \in [q_p]$, and permutation index $\alpha \in [w]$ such that $(\widehat{U}^i_\alpha = U^j_\alpha) \wedge (V^j_\alpha \oplus O^i_\alpha \oplus (a_{0,0} \oplus a_{\alpha,0}) \cdot K_0 \oplus (a_{0,1} \oplus a_{\alpha,1}) \cdot K_1 = V^k_0)$.
   - bad_4: $\exists i \in [q_c], j, k \in [q_p]$, and distinct permutation indices $\alpha, \beta \in [w]$ such that $(\widehat{U}^i_\alpha = U^j_\alpha) \wedge (V^j_\alpha \oplus O^i_\alpha \oplus O^i_\beta \oplus (a_{\alpha,0} \oplus a_{\beta,0}) \cdot K_0 \oplus (a_{\alpha,1} \oplus a_{\beta,1}) \cdot K_1 = V^k_\beta)$.
3. *For two construction queries i and j, one of the inputs of i-th construction query collides with the input of a primitive query, and one of the inputs of j-th construction query collides with the input of a primitive query, and the output of any two permutation calls collide.*
   - bad_5: $\exists i, j \in [q_c], k, l \in [q_p]$, and permutation index $\alpha \in [w]$ such that $(\widehat{U}^i_0 = U^k_0) \wedge (\widehat{U}^j_0 = U^l_0) \wedge (V^k_0 \oplus O^i_\alpha = V^l_0 \oplus O^j_\alpha)$.
   - bad_6: $\exists i, j \in [q_c], k, l \in [q_p]$, and permutation index $\alpha \in [w]$ such that $(\widehat{U}^i_\alpha = U^k_\alpha) \wedge (\widehat{U}^j_\alpha = U^l_\alpha) \wedge (V^k_\alpha \oplus O^i_\alpha = V^l_\alpha \oplus O^j_\alpha)$.
   - bad_7: $\exists i, j \in [q_c], k, l \in [q_p]$, and distinct permutation indices $\alpha, \beta \in [w]$ such that $(\widehat{U}^i_\alpha = U^k_\alpha) \wedge (\widehat{U}^j_\alpha = U^l_\alpha) \wedge (V^k_\alpha \oplus O^i_\alpha \oplus O^i_\beta = V^l_\alpha \oplus O^j_\alpha \oplus O^j_\beta)$.
   - bad_8: $\exists i, j \in [q_c], k, l \in [q_p]$, and distinct permutation indices $\gamma, \beta \in [w]$ such that $(\widehat{U}^i_0 = U^k_0) \wedge (\widehat{U}^j_\gamma = U^l_\gamma) \wedge (V^k_0 \oplus V^l_\gamma \oplus O^i_\beta \oplus O^j_\gamma \oplus O^j_\beta = (a_{0,0} \oplus a_{\gamma,0}) \cdot K_0 \oplus (a_{0,1} \oplus a_{\gamma,1}) \cdot K_1)$.
   - bad_9: $\exists i, j \in [q_c], k, l \in [q_p]$, and distinct permutation indices $\alpha, \gamma \in [w]$ such that $(\widehat{U}^i_\alpha = U^k_\alpha) \wedge (\widehat{U}^j_\gamma = U^l_\gamma) \wedge (V^k_\alpha \oplus V^l_\gamma \oplus O^i_\alpha \oplus O^j_\gamma = (a_{\alpha,0} \oplus a_{\gamma,0}) \cdot K_0 \oplus (a_{\alpha,1} \oplus a_{\gamma,1}) \cdot K_1)$.

- bad_{10}: $\exists i, j \in [q_c], k, l \in [q_p]$, and distinct permutation indices $\alpha, \beta, \gamma \in [w]$ such that $(\widehat{U}^i_\alpha = U^k_\alpha) \wedge (\widehat{U}^j_\gamma = U^l_\gamma) \wedge (V^k_\alpha \oplus V^l_\gamma \oplus O^i_\alpha \oplus O^i_\beta \oplus O^j_\gamma \oplus O^j_\beta = (a_{\alpha,0} \oplus a_{\gamma,0}) \cdot K_0 \oplus (a_{\alpha,1} \oplus a_{\gamma,1}) \cdot K_1)$.

Using the union bound, the probability that a transcript in the ideal world is bad is at most

$$\Pr\left[\Theta_{\text{ideal}} \in \text{BADT}\right] \leq \sum_{i=1}^{10} \Pr[\mathsf{bad}_i]\,. \tag{6.4}$$

Lemma 6.8. It holds that

$$\Pr\left[\Theta_{\text{ideal}} \in \text{BADT}\right] \leq \frac{(w+1)^2 q_p^2 q_c}{2^{2n+1}} + \frac{4w q_p^2 q_c}{2^{2n}} + \frac{2w q_c^2 q_p}{2^{2n}} + \frac{w^2 q_c^2 q_p}{2^{2n}} + \frac{3w^2 q_p^2 q_c}{2^{2n+1}} + \frac{3w^3 q_p^2 q_c^2}{2^{3n}}\,.$$

Proof. In the following, we study the probabilities of the individual bad events. Before, we recall the key-scheduling matrix $\mathbf{A}$ as follows:

$$\mathbf{A} = \begin{bmatrix} a_{0,0} & a_{1,0} & a_{2,0} & \dots & a_{w,0} \\ a_{0,1} & a_{1,1} & a_{2,1} & \dots & a_{w,1} \end{bmatrix}^\top.$$

bad_1: This event considers the collisions between two construction-query inputs and two primitive-query inputs. For this event, it must hold that

$$I^i \oplus (a_{\alpha,0} \cdot K_0 \oplus a_{\alpha,1} \cdot K_1) = U^j_\alpha \quad \text{and} \quad I^i \oplus (a_{\beta,0} \cdot K_0 \oplus a_{\beta,1} \cdot K_1) = U^k_\beta\,,$$

with $[a_{i,0}\ a_{i,1}]$ as the i-th row of the key-scheduling matrix. The two equations can be seen as

$$\mathbf{A}' \cdot \mathbf{K} = \begin{bmatrix} a_{\alpha,0} & a_{\alpha,1} \\ a_{\beta,0} & a_{\beta,1} \end{bmatrix} \cdot \begin{bmatrix} K_0 \\ K_1 \end{bmatrix} = \begin{bmatrix} I^i \oplus U^j_\alpha \\ I^i \oplus U^k_\beta \end{bmatrix}$$

Since all rows of $\mathbf{A}$ are pairwise linearly independent, $\mathbf{A}'$ is non-singular. Moreover, K_0 and K_1 are uniform random variables over $\{0,1\}^n$. Thus, we can apply Lemma 6.2 and the probability of this event for a fixed choice of indices is 2^{-2n}. Since one can choose α and β in $\binom{w+1}{2}$ ways, we obtain from the union bound over all

indices

$$\Pr[\mathsf{bad}_1] = \sum_{i\in[q_c]} \sum_{j\in[q_p]} \sum_{k\in[q_p]} \sum_{0\le\alpha<\beta\le w} \Pr\left[\widehat{U}^i_\alpha = U^j_\alpha \wedge \widehat{U}^i_\beta = U^k_\beta\right] \le \frac{\binom{w+1}{2} q_p^2 q_c}{2^{2n}}\,. \quad (6.5)$$

bad_2: This event considers the collision between the input of P_0 corresponding to a construction query and the input to P_0 corresponding to a primitive query, and the collision between the output of P_α corresponding to the same construction query and the output of P_α corresponding to a primitive query. For this event, it must hold that

$$I^i \oplus (a_{0,0} \cdot K_0 \oplus a_{0,1} \cdot K_1) = U^j_0 \quad \text{and}$$
$$(V^j_0 \oplus O^i_\alpha \oplus (a_{0,0} \oplus a_{\alpha,0}) \cdot K_0 \oplus (a_{0,1} \oplus a_{\alpha,1}) \cdot K_1 = V^k_\alpha)\,,$$

The two equations can be seen as

$$\mathbf{A}' \cdot \mathbf{K} = \begin{bmatrix} a_{0,0} & a_{0,1} \\ (a_{0,0} \oplus a_{\alpha,0}) & (a_{0,1} \oplus a_{\alpha,1}) \end{bmatrix} \cdot \begin{bmatrix} K_0 \\ K_1 \end{bmatrix} = \begin{bmatrix} I^i \oplus U^j_0 \\ V^k_\alpha \oplus V^j_0 \oplus O^i_\alpha \end{bmatrix}$$

Since all rows of $\mathbf{A}$ are pairwise linearly independent, $\mathbf{A}'$ is non-singular, because $\det(A') = (a_{0,0}a_{\alpha,1} \oplus a_{0,1}a_{\alpha,0})$ which is the determinant of the following matrix

$$\mathbf{A}'' = \begin{bmatrix} a_{0,0} & a_{0,1} \\ a_{\alpha,0} & a_{\alpha,1} \end{bmatrix}$$

and $\mathbf{A}''$ is non-singular. Moreover, K_0 and K_1 are uniform random variables over $\{0,1\}^n$. Thus, we can apply Lemma 6.2 and the probability of this event for a fixed choice of indices is 2^{-2n}. Since one can choose i in q_c ways, j and k in q_p ways and α in w ways, we obtain from the union bound over all indices

$$\Pr[\mathsf{bad}_2] \le \frac{w q_p^2 q_c}{2^{2n}}\,. \quad (6.6)$$

bad_3: This event considers the collision between the input of P_α corresponding to a construction query and the input to P_α corresponding to a primitive query for $\alpha \in [w]$, and the collision between the output of P_0 corresponding to the same construction query and the output of P_0 corresponding to a primitive query. For

this event, it must hold that

$$I^i \oplus (a_{\alpha,0} \cdot K_0 \oplus a_{\alpha,1} \cdot K_1) = U_\alpha^j \quad \text{and}$$
$$(V_\alpha^j \oplus O_\alpha^i \oplus (a_{0,0} \oplus a_{\alpha,0}) \cdot K_0 \oplus (a_{0,1} \oplus a_{\alpha,1}) \cdot K_1 = V_0^k)\,.$$

The two equations can be seen as

$$\mathbf{A}' \cdot \mathbf{K} = \begin{bmatrix} a_{\alpha,0} & a_{\alpha,1} \\ (a_{0,0} \oplus a_{\alpha,0}) & (a_{0,1} \oplus a_{\alpha,1}) \end{bmatrix} \cdot \begin{bmatrix} K_0 \\ K_1 \end{bmatrix} = \begin{bmatrix} I^i \oplus U_\alpha^j \\ V_0^k \oplus V_\alpha^j \oplus O_\alpha^i \end{bmatrix}$$

Since all rows of $\mathbf{A}$ are pairwise linearly independent, $\mathbf{A}'$ is non-singular, because $\det(A') = (a_{0,1}a_{\alpha,0} \oplus a_{0,0}a_{\alpha,1})$ which is the determinant of the matrix $\mathbf{A}''$ as defined in bad_2. Moreover, K_0 and K_1 are uniform random variables over $\{0,1\}^n$. Thus, we can apply Lemma 6.2 and the probability of this event for a fixed choice of indices is 2^{-2n}. Since one can choose i in q_c ways, j and k in q_p ways and α in w ways, we obtain from the union bound over all indices

$$\Pr[\mathsf{bad}_3] \leq \frac{wq_p^2 q_c}{2^{2n}}\,. \tag{6.7}$$

bad_4: This event considers the collision between the input of P_α corresponding to a construction query and the input to P_α corresponding to a primitive query for $\alpha \in [w]$, and the collision between the output of P_β corresponding to the same construction query and the output of P_β corresponding to a primitive query for some $\beta \neq \alpha$. For this event, it must hold that

$$\begin{cases} I^i \oplus (a_{\alpha,0} \cdot K_0 \oplus a_{\alpha,1} \cdot K_1) = U_\alpha^j \\ (V_\alpha^j \oplus O_\alpha^i \oplus O_\beta^i \oplus (a_{\alpha,0} \oplus a_{\beta,0}) \cdot K_0 \oplus (a_{\alpha,1} \oplus a_{\beta,1}) \cdot K_1 = V_\beta^k). \end{cases}$$

The two equations can be seen as

$$\mathbf{A}' \cdot \mathbf{K} = \begin{bmatrix} a_{\alpha,0} & a_{\alpha,1} \\ (a_{\alpha,0} \oplus a_{\beta,0}) & (a_{\alpha,1} \oplus a_{\beta,1}) \end{bmatrix} \cdot \begin{bmatrix} K_0 \\ K_1 \end{bmatrix} = \begin{bmatrix} I^i \oplus U_\alpha^j \\ V_\beta^k \oplus V_\alpha^j \oplus O_\alpha^i \oplus O_\beta^i \end{bmatrix}$$

Since all rows of $\mathbf{A}$ are pairwise linearly independent, $\mathbf{A}'$ is non-singular, because $\det(A') = (a_{\beta,1}a_{\alpha,0} \oplus a_{\beta,0}a_{\alpha,1})$ which is the determinant of the following matrix

$$\mathbf{A}'' = \begin{bmatrix} a_{\alpha,0} & a_{\alpha,1} \\ a_{\beta,0} & a_{\beta,1} \end{bmatrix}$$

and $\mathbf{A}''$ is non-singular. Moreover, K_0 and K_1 are uniform random variables over $\{0,1\}^n$. Thus, we can apply Lemma 6.2 and the probability of this event for a fixed choice of indices is 2^{-2n}. Since one can choose i in q_c ways, j and k in q_p ways and α and β in $\binom{w}{2}$ ways, we obtain from the union bound over all indices

$$\Pr[\mathsf{bad}_4] \leq \frac{\binom{w}{2} q_p^2 q_c}{2^{2n}}\,. \tag{6.8}$$

bad_5: This event considers the collision between the input of P_0 for two construction queries and the corresponding primitive input to P_0 and the collision between the output of P_α for some $\alpha \in [w]$ corresponding to the same two construction queries. For this event, it must hold that

$$\begin{cases} (a_{0,0} \cdot K_0 \oplus a_{0,1} \cdot K_1) = I^i \oplus U_0^k = I^j \oplus U_0^l & (\mathsf{E.1}) \\ (O_\alpha^i \oplus O_\alpha^j = V_0^k \oplus V_0^l)\,. \end{cases}$$

We can easily observe that

$$\Pr[(\mathsf{E.1})] = \Pr[I^i \oplus U_0^k = I^j \oplus U_0^l] \cdot \Pr[(\mathsf{E.1}) \mid I^i \oplus U_0^k = I^j \oplus U_0^l] \tag{6.9}$$

Let's first fix a value for α and the choice of indices of the two construction queries and the two primitive queries. We'll break down the event into two following cases. Firstly, if the last among four queries is a backward primitive query (w.l.o.g., suppose it's V_0^k to obtain U_0^k), then the probability of Equation (6.9) comes out to be $\frac{1}{2^n} \cdot \frac{1}{2^n}$. The first $\frac{1}{2^n}$ comes from the randomness over U_0^k and the second $\frac{1}{2^n}$ comes from the randomness over $a_{0,0} \cdot K_0 \oplus a_{0,1} \cdot K_1$. But in this case, $\Pr[O_\alpha^i \oplus O_\alpha^j = V_0^k \oplus V_0^l] = 1$.
Secondly, if the last among four queries is a forward positive query (w.l.o.g., suppose it's U_0^k to obtain V_0^k) or a construction query (w.l.o.g., suppose it's I^i to obtain O^i), then the probability of Equation (6.9) comes out to be $1.\frac{1}{2^n}$. The $\frac{1}{2^n}$

comes from randomness over $a_{0,0} \cdot K_0 \oplus a_{0,1} \cdot K_1$. But in this case $\Pr[O^i_\alpha \oplus O^j_\alpha = V^k_0 \oplus V^l_0] = \frac{1}{2^n}$. The $\frac{1}{2^n}$ comes from randomness over V^k_0 or O^i_α respectively.
Now, in case when the last query is a primitive query, then i and j can be chosen in $2\binom{q_c}{2}$ ways. But the value of the index corresponding to the last primitive query gets fixed once one fixes the value of the index of the other primitive query (This can be done in q_p ways). Similarly, in case when the last query is a construction query, then k and l can be chosen in q_p^2 ways. But the value of the index corresponding to the last construction query gets fixed once one fixes the value of the index of the other construction query (This can be done in q_c ways). As one can choose α in w ways, we obtain from the union bound over all indices

$$\Pr[\mathsf{bad}_5] \leq \max\left(\frac{2w\binom{q_c}{2}q_p}{2^{2n}}, \frac{wq_cq_p^2}{2^{2n}}\right) \leq \frac{2w\binom{q_c}{2}q_p}{2^{2n}} + \frac{wq_cq_p^2}{2^{2n}}\,. \tag{6.10}$$

bad_6: This event considers the collision between the input of P_α for two construction queries and the corresponding primitive input to P_α for some $\alpha \in [w]$ and the collision between the output of P_0 corresponding to the same two construction queries. For this event, it must hold that

$$\begin{cases} (a_{\alpha,0} \cdot K_0 \oplus a_{\alpha,1} \cdot K_1) = I^i \oplus U^k_\alpha = I^j \oplus U^l_\alpha & (\mathsf{E}.1) \\ (O^i_\alpha \oplus O^j_\alpha = V^k_\alpha \oplus V^l_\alpha)\,. \end{cases}$$

We'll bound the probability of this event in a way similar to that of bad_5. We can easily observe that

$$\Pr[(\mathsf{E}.1)] = \Pr[I^i \oplus U^k_\alpha = I^j \oplus U^l_\alpha] \cdot \Pr[(\mathsf{E}.1) \mid I^i \oplus U^k_\alpha = I^j \oplus U^l_\alpha] \tag{6.11}$$

Let's first fix a value for α and the choice of indices of the two construction queries and the two primitive queries. We'll break down the event into two following cases. Firstly, if the last among four queries is a backward primitive query (w.l.o.g., suppose it's V^k_α to obtain U^k_α), then the probability of Equation (6.11) comes out to be $\frac{1}{2^n} \cdot \frac{1}{2^n}$. The first $\frac{1}{2^n}$ comes from the randomness over U^k_α and the second $\frac{1}{2^n}$ comes from the randomness over $a_{0,0} \cdot K_0 \oplus a_{0,1} \cdot K_1$. But in this case, $\Pr[O^i_\alpha \oplus O^j_\alpha = V^k_\alpha \oplus V^l_\alpha] = 1$.

Secondly, if the last among four queries is a forward positive query (w.l.o.g., suppose it's U_α^k to obtain V_α^k) or a construction query (w.l.o.g., suppose it's I^i to obtain O^i), then the probability of Equation (6.11) comes out to be $1.\frac{1}{2^n}$. The $\frac{1}{2^n}$ comes from randomness over $a_{0,0} \cdot K_0 \oplus a_{0,1} \cdot K_1$. But in this case $\Pr[O_\alpha^i \oplus O_\alpha^j = V_\alpha^k \oplus V_\alpha^l] = \frac{1}{2^n}$. The $\frac{1}{2^n}$ comes from randomness over V_α^k or O_α^i respectively.
Now, in case when the last query is a primitive query, then i and j can be chosen in $2\binom{q_c}{2}$ ways. But the value of the index corresponding to the last primitive query gets fixed once one fixes the value of the index of the other primitive query (This can be done in q_p ways). Similarly, in case when the last query is a construction query, then k and l can be chosen in q_p^2 ways. But the value of the index corresponding to the last construction query gets fixed once one fixes the value of the index of the other construction query (This can be done in q_c ways). As one can choose α in w ways, we obtain from the union bound over all indices

$$\Pr[\mathsf{bad}_6] \leq \max\left(\frac{2w\binom{q_c}{2}q_p}{2^{2n}}, \frac{wq_cq_p^2}{2^{2n}}\right) \leq \frac{2w\binom{q_c}{2}q_p}{2^{2n}} + \frac{wq_cq_p^2}{2^{2n}}\,. \tag{6.12}$$

bad_7: This event considers the collision between the input of P_α for two construction queries and the corresponding primitive input to P_α for some $\alpha \in [w]$ and the collision between the output of P_β corresponding to the same two construction queries for some $\beta \in [w]$ with $\beta \neq \alpha$. For this event, it must hold that

$$\begin{cases} (a_{\alpha,0} \cdot K_0 \oplus a_{\alpha,1} \cdot K_1) = I^i \oplus U_\alpha^k = I^j \oplus U_\alpha^l & (\mathsf{E.1}) \\ (O_\alpha^i \oplus O_\beta^i \oplus O_\alpha^j \oplus O_\beta^j = V_\alpha^k \oplus V_\alpha^l)\,. \end{cases}$$

Again we'll bound the probability of this event in a way similar to that of the previous bad event. We can easily observe that

$$\Pr[(\mathsf{E.1})] = \Pr[I^i \oplus U_\alpha^k = I^j \oplus U_\alpha^l] \cdot \Pr[(\mathsf{E.1}) \mid I^i \oplus U_\alpha^k = I^j \oplus U_\alpha^l] \tag{6.13}$$

Let's first fix the values for α and β and the choice of indices of the two construction queries and the two primitive queries. We'll break down the event into two following cases.
Firstly, if the last among four queries is a backward primitive query (w.l.o.g., suppose it's V_α^k to obtain U_α^k), then the probability of Equation (6.13) comes out

to be $\frac{1}{2^n}.\frac{1}{2^n}$. The first $\frac{1}{2^n}$ comes from the randomness over U^k_α and the second $\frac{1}{2^n}$ comes from the randomness over $a_{0,0} \cdot K_0 \oplus a_{0,1} \cdot K_1$. But in this case, $\Pr[O^i_\alpha \oplus O^i_\beta \oplus O^j_\alpha \oplus O^j_\beta = V^k_\alpha \oplus V^l_\alpha] = 1$.

Secondly, if the last among four queries is a forward positive query (w.l.o.g., suppose it's U^k_α to obtain V^k_α) or a construction query (w.l.o.g., suppose it's I^i to obtain O^i), then the probability of Equation (6.13) comes out to be $1.\frac{1}{2^n}$. The $\frac{1}{2^n}$ comes from randomness over $a_{0,0} \cdot K_0 \oplus a_{0,1} \cdot K_1$. But in this case $\Pr[O^i_\alpha \oplus O^i_\beta \oplus O^j_\alpha \oplus O^j_\beta = V^k_\alpha \oplus V^l_\alpha] = \frac{1}{2^n}$. The $\frac{1}{2^n}$ comes from randomness over V^k_α or $O^i_\alpha \oplus O^i_\beta$ respectively. Now, in case when the last query is a primitive query, then i and j can be chosen in $2\binom{q_c}{2}$ ways. But the value of the index corresponding to the last primitive query gets fixed once one fixes the value of the index of the other primitive query (This can be done in q_p ways). Similarly, in case when the last query is a construction query, then k and l can be chosen in q_p^2 ways. But the value of the index corresponding to the last construction query gets fixed once one fixes the value of the index of the other construction query (This can be done in q_c ways). As one can choose α and β in $2\binom{w}{2}$ ways, we obtain from the union bound over all indices

$$\Pr[\mathsf{bad}_7] \leq \max\left(\frac{4\binom{w}{2}\binom{q_c}{2}q_p}{2^{2n}}, \frac{2\binom{w}{2}q_c q_p^2}{2^{2n}}\right) \leq \frac{4\binom{w}{2}\binom{q_c}{2}q_p}{2^{2n}} + \frac{2\binom{w}{2}q_c q_p^2}{2^{2n}}. \tag{6.14}$$

bad_8: This event considers the collision between the input of P_0 for i-th construction query and a primitive input to P_0, the collision between the input of P_γ for j-th construction query and a primitive input to P_γ for some $\gamma \in [w]$ and the collision between the output of P_β for i-th and j-th construction queries for some $\beta \in [w]$ with $\beta \neq \gamma$. For this event, it must hold that

$$\begin{cases} (a_{0,0} \cdot K_0 \oplus a_{0,1} \cdot K_1) = I^i \oplus U^k_0 \\ (a_{\gamma,0} \cdot K_0 \oplus a_{\gamma,1} \cdot K_1) = I^j \oplus U^l_\gamma \\ (a_{0,0} \oplus a_{\gamma,0}) \cdot K_0 \oplus (a_{0,1} \oplus a_{\gamma,1}) \cdot K_1) = (V^k_0 \oplus V^l_\gamma \oplus O^i_\beta \oplus O^j_\gamma \oplus O^j_\beta). \end{cases}$$

Note that the system of equations above can be written equivalently as

$$\begin{cases} (a_{0,0} \cdot K_0 \oplus a_{0,1} \cdot K_1) = I^i \oplus U^k_0 \\ (a_{\gamma,0} \cdot K_0 \oplus a_{\gamma,1} \cdot K_1) = I^j \oplus U^l_\gamma \\ I^i \oplus I^j \oplus U^k_0 \oplus U^l_\gamma = (V^k_0 \oplus V^l_\gamma \oplus O^i_\beta \oplus O^j_\gamma \oplus O^j_\beta). \end{cases}$$

Let's first fix the values for γ and β and the choice of indices of the two construction queries and the two primitive queries. The probability of each of the first two equations comes out to be $\frac{1}{2^n}$, which comes from the randomness over $a_{0,0} \cdot K_0 \oplus a_{0,1} \cdot K_1$ and $a_{\gamma,0} \cdot K_0 \oplus a_{\gamma,1} \cdot K_1$ respectively. Since the matrix

$$\begin{bmatrix} a_{0,0} & a_{0,1} \\ a_{\gamma,0} & a_{\gamma,1} \end{bmatrix}$$

is full-rank, the joint probability of the first two equations comes out to be $\frac{1}{2^{2n}}$. The probability of the third equation comes out to be $\frac{1}{2^n}$, but the randomness comes from different variables depending on the last query. The different possible cases are as follows.

1. If the last among four queries is the construction query to obtain O^i from I^i, then the randomness comes from O^i_β.
2. If the last among four queries is the construction query to obtain O^j from I^j, then the randomness comes from $O^j_\gamma \oplus O^j_\beta$.
3. If the last among four queries is the forward primitive query to obtain V^k_0 from U^k_0, then the randomness comes from V^k_0.
4. If the last among four queries is the forward primitive query to obtain V^l_γ from U^l_γ, then the randomness comes from V^l_γ.
5. If the last among four queries is the backward primitive query to obtain U^k_0 from V^k_0, then the randomness comes from U^k_0.
6. If the last among four queries is the backward primitive query to obtain U^l_γ from V^l_γ, then the randomness comes from U^l_γ.

Now, one can choose i and j together in $2\binom{q_c}{2}$ ways and k and l in q_p ways each. Moreover, γ and β together can be chosen in $2\binom{w}{2}$ ways. Thus, we obtain from the union bound over all indices

$$\Pr[\mathsf{bad}_8] \leq \frac{4\binom{w}{2}\binom{q_c}{2}q_p^2}{2^{3n}}\,. \tag{6.15}$$

bad_9: This event considers the collision between the input of P_α for i-th construction queries and a primitive input to P_α, the collision between the input of P_γ for j-th construction queries and a primitive input to P_γ for some $\alpha \neq \gamma \in [w]$ and the collision between the output of P_0 for i-th and j-th construction queries. For this event, it must hold that

$$\begin{cases} (a_{\alpha,0} \cdot K_0 \oplus a_{\alpha,1} \cdot K_1) = I^i \oplus U^k_\alpha \\ (a_{\gamma,0} \cdot K_0 \oplus a_{\gamma,1} \cdot K_1) = I^j \oplus U^l_\gamma \\ (a_{\alpha,0} \oplus a_{\gamma,0}) \cdot K_0 \oplus (a_{\alpha,1} \oplus a_{\gamma,1}) \cdot K_1) = (V^k_\alpha \oplus V^l_\gamma \oplus O^i_\alpha \oplus O^j_\gamma)\,. \end{cases}$$

Note that the above system of equations can be equivalently written as

$$\begin{cases} (a_{\alpha,0} \cdot K_0 \oplus a_{\alpha,1} \cdot K_1) = I^i \oplus U^k_\alpha \\ (a_{\gamma,0} \cdot K_0 \oplus a_{\gamma,1} \cdot K_1) = I^j \oplus U^l_\gamma \\ I^i \oplus I^j \oplus U^k_\alpha \oplus U^l_\gamma = (V^k_\alpha \oplus V^l_\gamma \oplus O^i_\alpha \oplus O^j_\gamma)\,. \end{cases}$$

Using the similar reasoning while bounding bad_8, we have

$$\Pr[\mathsf{bad}_9] \leq \frac{4\binom{w}{2}\binom{q_c}{2}q_p^2}{2^{3n}}\,. \tag{6.16}$$

bad_{10}: This event considers the collision between the input of P_α for i-th construction queries and a primitive input to P_α, the collision between the input of P_γ for j-th construction queries and a primitive input to P_γ for some $\alpha \neq \gamma \in [w]$ and the collision between the output of P_β for i-th and j-th construction queries for some $\beta \in [w]$ such that $\beta \neq \alpha$, $\beta \neq \gamma$. For this event, it must hold that

$$\begin{cases} (a_{\alpha,0} \cdot K_0 \oplus a_{\alpha,1} \cdot K_1) = I^i \oplus U^k_\alpha \\ (a_{\gamma,0} \cdot K_0 \oplus a_{\gamma,1} \cdot K_1) = I^j \oplus U^l_\gamma \\ (a_{\alpha,0} \oplus a_{\gamma,0}) \cdot K_0 \oplus (a_{\alpha,1} \oplus a_{\gamma,1}) \cdot K_1) = (V^k_\alpha \oplus V^l_\gamma \oplus O^i_\alpha \oplus O^i_\beta \oplus O^j_\gamma \oplus O^j_\beta). \end{cases}$$

Note that the above system of equations can be equivalently written as

$$\begin{cases} (a_{\alpha,0} \cdot K_0 \oplus a_{\alpha,1} \cdot K_1) = I^i \oplus U^k_\alpha \\ (a_{\gamma,0} \cdot K_0 \oplus a_{\gamma,1} \cdot K_1) = I^j \oplus U^l_\gamma \\ I^i \oplus I^j \oplus U^k_\alpha \oplus U^l_\gamma = (V^k_\alpha \oplus V^l_\gamma \oplus O^i_\alpha \oplus O^i_\beta \oplus O^j_\gamma \oplus O^j_\beta). \end{cases}$$

Using the similar reasoning while bounding bad_8, we have

$$\Pr[\mathsf{bad}_{10}] \leq \frac{2w(w-1)(w-2)\binom{q_c}{2}q_p^2}{2^{3n}}\,. \tag{6.17}$$

The bound in Lemma 6.8 follows from Equation (6.4)-Equation (6.17).

Good Transcripts: It remains to study the interpolation probabilities of good transcripts.

Lemma 6.9. Let $q_p + (w+1)q_c \leq 2^n/2(w+1)$. For any good transcript $\tau = \tau_c \cup \tau_0 \cup \ldots \tau_w \cup \{K_0, K_1\}$, it holds that

$$\frac{\Pr[\Theta_{\text{real}} = \tau]}{\Pr[\Theta_{\text{ideal}} = \tau]} \geq 1 - \frac{(w+1)^2 q_c (q_p + q_c)^2}{2^{2n}}\,.$$

Proof. Let $\mathsf{All}_{\text{real}}(\tau)$ denote the set of all oracles in the real world and $\mathsf{All}_{\text{ideal}}(\tau)$ the set of all oracles in the ideal world that produce $\tau \in \textsc{GoodT}$. Moreover, let $\mathsf{Comp}_{\text{real}}(\tau)$ denote the fraction of oracles in the real world that are compatible with τ and $\mathsf{Comp}_{\text{ideal}}(\tau)$ the corresponding fraction in the ideal world. It holds that

$$\frac{\Pr\left[\Theta_{\text{real}} = \tau\right]}{\Pr\left[\Theta_{\text{ideal}} = \tau\right]} = \frac{|\mathsf{Comp}_{\text{real}}(\tau)| \cdot |\mathsf{All}_{\text{ideal}}(\tau)|}{|\mathsf{Comp}_{\text{ideal}}(\tau)| \cdot |\mathsf{All}_{\text{real}}(\tau)|}.$$

We can easily bound the number for three out of four terms: $|\mathsf{All}_{\text{real}}(\tau)| = (2^n)^2 \cdot (2^n!\,)^{w+1}$ since there exist $(2^n)^2$ keys and $2^n!$ possible ways for each of the $w+1$ independent permutations P_α for $\alpha \in [0..w]$. The same argument holds in the ideal world $|\mathsf{All}_{\text{ideal}}(\tau)| = (2^n)^2 \cdot (2^n!\,)^{w+1} \cdot (2^{wn})^{2^n}$, combined with $(2^{wn})^{2^n}$ random functions for construction queries' answers. Moreover, $|\mathsf{Comp}_{\text{ideal}}(\tau)| = (2^{wn})^{2^n - q_c} \cdot \prod_{i=0}^{w}(2^n - q_p)!$ compatible oracles exist in the ideal world, where $(2^{wn})^{2^n-q_c}$ are the oracles that produce the correct construction-query outputs for the $2^n - q_c$ remaining non-queried inputs, and for all permutations, there exist $(2^n - q_p)!$ compatible primitives each. It remains to find $|\mathsf{Comp}_{\text{real}}(\tau)|$. Note that

$$|\mathsf{Comp}_{\text{real}}(\tau)| = \left|\left\{\mathbf{P} = (P_0, \ldots, P_w) : \mathsf{XORPP}^*[\mathbf{P}, w]_{\mathbf{K}} \mapsto \tau_c \wedge \bigwedge_{\alpha=0}^{w} P_\alpha \mapsto \tau_\alpha\right\}\right|,$$

where $\mathsf{XORPP}^*[\mathbf{P}, w]_{\mathbf{K}} \mapsto \tau_c$ denotes that $\mathsf{XORPP}^*[\mathbf{P}, w]_{\mathbf{K}}$ produces the construction query transcript τ_c. Similarly, for all $\alpha \in [0..w]$, $P_\alpha \mapsto \tau_\alpha$ denotes that the permutation P_α produces the primitive query transcript τ_α. In other words, if Dom_α

denotes the set $\{U_\alpha^i : (U_\alpha^i, V_\alpha^i) \in \tau_\alpha\}$ and Ran_α denotes the set $\{V_\alpha^i : (U_\alpha^i, V_\alpha^i) \in \tau_\alpha\}$, then $P_\alpha \mapsto \tau_\alpha$ equivalently means P_α maps elements from Dom_α to Ran_α. Now, in order to compute $|\mathsf{Comp}_{\text{real}}(\tau)|$, we regroup the queries from τ_c, τ_0, ..., τ_w to τ_c^{new}, τ_0^{new}, ..., τ_w^{new}. The new transcript sets are initialised by their corresponding old parts, and reordered as follows:

1. if $\exists i \in [q_c], j \in [q_p]$ such that $\widehat{U}_0^i = U_0^j$, then
   - $\tau_c^{\mathsf{new}} \leftarrow \tau_c^{\mathsf{new}} \setminus \{(I^i, \mathbf{O}^i)\}$ and
   - for all $\alpha \in [w]$, $\tau_\alpha^{\mathsf{new}} \leftarrow \tau_\alpha^{\mathsf{new}} \cup \{(\widehat{U}_\alpha^i, V_0^j \oplus O_\alpha^i \oplus (a_{0,0} \oplus a_{\alpha,0}) \cdot K_0 \oplus (a_{0,1} \oplus a_{\alpha,1}) \cdot K_1)\}$.
2. if $\exists i \in [q_c], j \in [q_p]$, and $\alpha \in [w]$ such that $\widehat{U}_\alpha^i = U_\alpha^j$, then
   - $\tau_c^{\mathsf{new}} \leftarrow \tau_c^{\mathsf{new}} \setminus \{(I^i, \mathbf{O}^i)\}$ and
   - $\tau_0^{\mathsf{new}} \leftarrow \tau_0^{\mathsf{new}} \cup \{(\widehat{U}_0^i, V_\alpha^j \oplus O_\alpha^i \oplus (a_{0,0} \oplus a_{\alpha,0}) \cdot K_0 \oplus (a_{0,1} \oplus a_{\alpha,1}) \cdot K_1)\}$ and
   - for all $\beta \in [w]$ with $\beta \neq \alpha$, $\tau_\beta^{\mathsf{new}} \leftarrow \tau_\beta^{\mathsf{new}} \cup \{(\widehat{U}_\beta^i, V_\alpha^j \oplus O_\alpha^i \oplus O_\beta^i \oplus (a_{\alpha,0} \oplus a_{\beta,0}) \cdot K_0 \oplus (a_{\alpha,1} \oplus a_{\beta,1}) \cdot K_1)\}$.

Note that such an addition of elements in Step (1) and Step (2) is sound. For Step (1),

- since $\widehat{U}_0^i$ collides with U_0^j, $\widehat{U}_\alpha^i$ cannot collide with any U_α^k for $\alpha \in [w]$ due to $\overline{\mathsf{bad}_1}$.
- Similarly, $(V_0^j \oplus O_\alpha^i \oplus (a_{0,0} \oplus a_{\alpha,0}) \cdot K_0 \oplus (a_{0,1} \oplus a_{\alpha,1}) \cdot K_1)$ cannot collide with any V_α^k for $\alpha \in [w]$ due to $\overline{\mathsf{bad}_2}$.
- Moroever, $(V_0^j \oplus O_\alpha^i \oplus (a_{0,0} \oplus a_{\alpha,0}) \cdot K_0 \oplus (a_{0,1} \oplus a_{\alpha,1}) \cdot K_1)$ is distinct due to $\overline{\mathsf{bad}_5}$ and $\overline{\mathsf{bad}_8}$.

For Step (2),

- since $\widehat{U}_\alpha^i$ collides with U_α^j for $\alpha \in [w]$, neither $\widehat{U}_0^i$ can collide with any U_0^k nor $\widehat{U}_\beta^i$ can collide with any U_β^k for $\beta \in [w]$ with $\beta \neq \alpha$ due to $\overline{\mathsf{bad}_1}$.
- Similarly, $(V_\alpha^j \oplus O_\alpha^i \oplus (a_{0,0} \oplus a_{\alpha,0}) \cdot K_0 \oplus (a_{0,1} \oplus a_{\alpha,1}) \cdot K_1)$ cannot collide with any V_0^k due to $\overline{\mathsf{bad}_3}$.
- $(V_\alpha^j \oplus O_\alpha^i \oplus O_\beta^i \oplus (a_{\alpha,0} \oplus a_{\beta,0}) \cdot K_0 \oplus (a_{\alpha,1} \oplus a_{\beta,1}) \cdot K_1)$ cannot collide with V_β^k for $\beta \in [w]$ with $\beta \neq \alpha$ due to $\overline{\mathsf{bad}_4}$.

- $(V_\alpha^j \oplus O_\alpha^i \oplus (a_{0,0} \oplus a_{\alpha,0}) \cdot K_0 \oplus (a_{0,1} \oplus a_{\alpha,1}) \cdot K_1)$ is distinct due to $\overline{\mathsf{bad}_6}$ and $\overline{\mathsf{bad}_9}$.
- $(V_\alpha^j \oplus O_\alpha^i \oplus O_\beta^i \oplus (a_{\alpha,0} \oplus a_{\beta,0}) \cdot K_0 \oplus (a_{\alpha,1} \oplus a_{\beta,1}) \cdot K_1)$ is distinct due to $\overline{\mathsf{bad}_7}$ and $\overline{\mathsf{bad}_{10}}$.

Also note that such an addition of elements (x, y) in the transcript $\tau_\alpha^{\mathsf{new}}$ for $\alpha \in [0..w]$ also updates the set $\mathsf{Dom}_\alpha \leftarrow \mathsf{Dom}_\alpha \cup \{x\}$ and $\mathsf{Ran}_\alpha \leftarrow \mathsf{Ran}_\alpha \cup \{y\}$. Now, given q_c constructions queries and q_p queries to each of the permutations in the original transcript, let the numbers of queries moved from τ_c be r which includes total s_α many elements into the primitive partial transcripts τ_α for $\alpha \in [0..w]$. Thus, the number of queries in the new construction transcript is denoted by $q' = q_c - r$ and the $w+1$ sets of transcripts, $(\tau_0^{\mathsf{new}}, \tau_1^{\mathsf{new}}, \ldots, \tau_w^{\mathsf{new}})$ define exactly $(q_p + s_0, q_p + s_1, \ldots, q_p + s_w)$ input-output tuples for $(P_0, \ldots, P_w)$ respectively. Therefore, it is easy to see that $(s_0 + \ldots + s_w) = rw$. Moreover, for each $\alpha \in [0..w]$, $s_\alpha \leq r \leq q_c$. Now, what remains is the counting of the number of permutations $(P_0, \ldots, P_w)$ that satisfy these $(q_p + s_0, q_p + s_1, \ldots, q_p + s_w)$ tuples respectively. That could give the remaining transcript τ_c^{new}, i.e., we are interested to count the number of permutations $(P_0, \ldots, P_w)$ that satisfies the following system of equations:

$$\mathbb{E}_i = \begin{cases} P_0(\widehat{U}_0^i) \oplus P_1(\widehat{U}_1^i) = O_1^i \oplus (a_{0,0} \oplus a_{1,0}) \cdot K_0 \oplus (a_{0,1} \oplus a_{1,1}) \cdot K_1 \\ P_0(\widehat{U}_0^i) \oplus P_2(\widehat{U}_2^i) = O_2^i \oplus (a_{0,0} \oplus a_{2,0}) \cdot K_0 \oplus (a_{0,1} \oplus a_{2,1}) \cdot K_1 \\ \vdots \quad \vdots \quad \vdots \quad \vdots \\ P_0(\widehat{U}_0^i) \oplus P_w(\widehat{U}_w^i) = O_w^i \oplus (a_{0,0} \oplus a_{w,0}) \cdot K_0 \oplus (a_{0,1} \oplus a_{w,1}) \cdot K_1 \,, \end{cases}$$

where $i \in [q']$, $U_\alpha^i = I^i \oplus (a_{\alpha,0} \cdot K_0 \oplus a_{\alpha,1} \cdot K_1)$ for all $\{(I^i, \mathbf{O}^i)\} \in \tau_c^{\mathsf{new}}$, along with the fact that for each $\alpha \in [0..w]$, P_α maps $\mathcal{D}_\alpha$ to $\mathcal{R}_\alpha$, where $\mathcal{D}_\alpha = \{0,1\}^n \setminus \mathsf{Dom}_\alpha$ and $\mathcal{R}_\alpha = \{0,1\}^n \setminus \mathsf{Ran}_\alpha$. Note that

$$\mathsf{Dom}_\alpha \stackrel{\text{def}}{=} \{U_\alpha^i : (U_\alpha^i, V_\alpha^i) \in \tau_\alpha^{\mathsf{new}}\}$$
$$\mathsf{Ran}_\alpha \stackrel{\text{def}}{=} \{V_\alpha^i : (U_\alpha^i, V_\alpha^i) \in \tau_\alpha^{\mathsf{new}}\}.$$

It is easy to see that $|\mathcal{D}_\alpha| = |\mathcal{R}_\alpha| = (2^n - q_p - s_\alpha)$. Note that $\mathsf{Ran}_\alpha = \{0,1\}^n \setminus \mathcal{R}_\alpha$, for $\alpha \in [0..w]$, as the set of range values of P_α that are prohibited (basically these are the V values in τ_α). Now, for $j = [0..q' - 1]$, let

$$\lambda_{j+1} \stackrel{\text{def}}{=} \left|\left\{(\mathsf{P}_0^1, \ldots, \mathsf{P}_0^{j+1}, \ldots, \mathsf{P}_w^1, \ldots, \mathsf{P}_w^{j+1})\right\}\right| \tag{6.18}$$

be the number of solutions that satisfy

(1) the system of equations $\mathbb{E}_1 \cup \mathbb{E}_2 \cup \ldots \cup \mathbb{E}_{j+1}$

(2) $\forall \alpha \in [0..w]$, it holds that $\mathsf{P}_\alpha^{j+1} \notin \{\mathsf{P}_\alpha^1, \ldots, \mathsf{P}_\alpha^j\} \cup \mathsf{Ran}_\alpha$.

Then, the goal is to define a recursive expression for λ_{j+1} from λ_j such that a lower bound can be found for the expression λ_{j+1}/λ_j. It holds that

$$|\mathsf{Comp}_{\text{real}}(\tau)| = \lambda_{q'} \cdot (2^n - (q_p + s_0 + q'))! \cdot \ldots \cdot (2^n - (q_p + s_w + q'))! \ ,$$

where the second term represents the number of permutations compatible with P_0 and the rightmost term contains the number of permutations compatible with P_w. We obtain

$$\frac{\Pr[\Theta_{\text{real}} = \tau]}{\Pr[\Theta_{\text{ideal}} = \tau]} = \frac{\lambda_{q'} \cdot \prod_{i=0}^{w} (2^n - (q_p + s_i + q'))! \cdot (2^n)^{w \cdot q_c}}{((2^n - q_p)!\,)^{w+1}}. \tag{6.19}$$

Let $\mathcal{B}_{(1)}$ denote the set of solutions that comply with only Condition (1) without considering Conditions (2.0) through $(2.w)$. Moreover, let $\mathcal{B}_{(2.\iota:i)}$ denote the set of solutions compatible with Condition (1), but not with $(2.\iota : i)$, for $i = 1, \ldots, j + |\mathsf{Ran}_\iota|$. From the inclusion-exclusion principle, it follows that

$$\begin{aligned}
\lambda_{j+1} &= \left|\mathcal{B}_{(1)}\right| - \left| \left(\bigcup_{i=1}^{j+|\mathsf{Ran}_0|} \mathcal{B}_{(2.0:i)} \right) \cup \cdots \cup \left(\bigcup_{i=1}^{j+|\mathsf{Ran}_w|} |\mathcal{B}_{(2.w:i)}| \right) \right| \\
&\geq \left|\mathcal{B}_{(1)}\right| - \left| \sum_{i=1}^{j+|\mathsf{Ran}_0|} |\mathcal{B}_{(2.0:i)}| \right| - \cdots - \left| \sum_{i=1}^{j+|\mathsf{Ran}_w|} |\mathcal{B}_{(2.w:i)}| \right| \\
&\quad + \sum_{i=1}^{j+|\mathsf{Ran}_0|} \sum_{i'=1}^{j+|\mathsf{Ran}_1|} \underbrace{\left|\mathcal{B}_{(2.0:i)} \cap \mathcal{B}_{(2.1:i')}\right|}_{\geq 0} + \cdots \\
&\quad + \sum_{i=1}^{j+|\mathsf{Ran}_{w-1}|} \sum_{i'=1}^{j+|\mathsf{Ran}_w|} \underbrace{\left|\mathcal{B}_{(2.(w-1):i)} \cap \mathcal{B}_{(2.w:i')}\right|}_{\geq 0} \\
&\geq 2^n \cdot \lambda_j - \sum_{i=1}^{j+|\mathsf{Ran}_0|} \lambda_j - \cdots - \sum_{i=1}^{j+|\mathsf{Ran}_w|} \lambda_j.
\end{aligned}$$

It follows that $\lambda_{j+1} \geq 2^n \cdot \lambda_j - (j + q_p + s_0) \cdot \lambda_j - \ldots - (j + q_p + s_w) \cdot \lambda_j$. Therefore,

$$\frac{\lambda_{j+1}}{\lambda_j} \geq 2^n - (w+1)j - (w+1)q_p - \sum_{\alpha=0}^{w} s_\alpha$$

with $\lambda_0 = 1$. It follows that Equation (6.19) can be written as

$$\prod_{t=0}^{s_0-1} \frac{2^n}{2^n - q_p - t} \cdot \ldots \cdot \prod_{t=0}^{s_w-1} \frac{2^n}{2^n - q_p - t} \cdot \prod_{i=0}^{q'-1} \frac{\lambda_{i+1}}{\lambda_i} \cdot \frac{(2^n)^w}{\prod_{\alpha=0}^{w}(2^n - q_p - i - s_\alpha)}$$
$$\geq \prod_{i=0}^{q'-1} \left(\frac{(2^n - (w+1)i - (w+1)q_p - \sum_{\alpha=0}^{w} s_\alpha)}{\prod_{\alpha=0}^{w}(2^n - q_p - i - s_\alpha)} \cdot 2^{nw} \right) . \tag{6.20}$$

By substituting $z_i \stackrel{\text{def}}{=} q_p + i$ and setting $p_{i,\alpha} \stackrel{\text{def}}{=} (z_i + s_\alpha)/2^n$, we get

$$(6.20) = \prod_{i=0}^{q'-1} \left(\frac{(2^n)^{w+1} - 2^{nw} \cdot ((w+1)z_i + \sum_{\alpha=0}^{w} s_\alpha)}{\prod_{\alpha=0}^{w}(2^n - (z_i + s_\alpha))} \right) \tag{6.21}$$
$$\geq \prod_{i=0}^{q'-1} \left(\frac{(2^n)^{w+1} - 2^{nw} \cdot ((w+1)z_i + \sum_{\alpha=0}^{w} s_\alpha)}{2^{n(w+1)}(1 - \sum_{\alpha=0}^{w} p_{i,\alpha} + \sum_{0 \leq \alpha < \beta \leq w} p_{i,\alpha} p_{i,\beta})} \right) \tag{6.22}$$

Note that $0 \leq p_{i,\alpha} \leq 1$ and $s_{\max} \stackrel{\text{def}}{=} \max\{s_\alpha : 0 \leq \alpha \leq w\}$ and by applying Lemma 6.3, we derived Equation (6.22) from Equation (6.21). Therefore, we have

$$(6.22) \geq \prod_{i=0}^{q'-1} \left(1 - \frac{\sum_{\alpha<\beta}(z_i + s_\alpha)(z_i + s_\beta)}{2^{2n}(1 - \sum_{\alpha=0}^{w} p_{i,\alpha} + \sum_{0 \leq \alpha < \beta \leq w} p_{i,\alpha} p_{i,\beta})} \right)$$
$$\geq \prod_{i=0}^{q'-1} \left(1 - \frac{\binom{w+1}{2}(z_i + s_{\max})^2}{2^{2n} - 2^n \sum_{\alpha=0}^{w}(z_i + s_\alpha) + \sum_{\alpha<\beta}(z_i + s_\alpha)(z_i + s_\beta)} \right)$$
$$\stackrel{(1)}{\geq} \prod_{i=0}^{q'-1} \left(1 - \frac{2 \cdot \binom{w+1}{2} \cdot (q_p + q_c)^2}{2^{2n}} \right) \stackrel{(2)}{\geq} \left(1 - \frac{(w+1)^2 q'(q_p + q_c)^2}{2^{2n}} \right) \tag{6.23}$$

where (1) follows from the fact that $z_i + s_{\max} = q_p + i + s_{\max} \leq q_p + q' + r = q_p + q_c$. Moreover, $2^n \sum_{\alpha=0}^{w}(z_i + s_\alpha) - \sum_{\alpha<\beta}(z_i + s_\alpha)(z_i + s_\beta) \leq 2^{2n}/2$, that follows due to the fact that $(q_p + q_c) \leq (q_p + (w+1)q_c) \leq 2^n/2(w+1)$. (2) holds due to Bernoulli's inequality. Finally, we used $q_c \geq q'$ to derive the final bound.

Our claim in Theorem 6.6 follows from Lemma 2.1, 6.8, and 6.9. □

Algorithm 8 Specification of CENCPP

```
101: function CENCPP[P, w].E_K(N, M)
102:     M_1 ··· M_m ←_n M
103:     ℓ ← ⌈m/w⌉
104:     for i ← 0..ℓ − 1 do
105:         j ← i · w
106:         (S_{j+1} ‖ ··· ‖ S_{j+w})
107:             ← XORPP* [P, w]_K(N ‖ ⟨i⟩_μ)
108:         for k ← j + 1..j + w do
109:             C_k ← msb_{|M_k|}(S_k) ⊕ M_k
110:     return (C_1 ‖ ··· ‖ C_m)

201: function CENCPP[P, w].D_K(N, C)
202:     return CENCPP[P, w].E_K(N, C)

301: function XORPP[P, w]_K(I)
302:     (K_0, K_1) ← K
303:     (P_0, ..., P_w) ← P
304:     L_0 ← K_0 ⊕ K_1
305:     Û_0 ← I ⊕ L_0
306:     X̂_0 ← P_0(Û_0) ⊕ L_0
307:     for α ← 1..w do
308:         L_α ← (2^α · K_0) ⊕ (2^{2α} · K_1)
309:         Û_α ← I ⊕ L_α
310:         X̂_α ← P_α(Û_α) ⊕ L_α
311:         O_α ← X̂_α ⊕ X̂_0
312:     return (O_1 ‖ ··· ‖ O_w)
```

6.5.3 CENCPP: An Instantiation of CENCPP*

A natural instantiation of CENCPP* can be realised by instantiating the key-scheduling matrix $\mathbf{A}$ of dimension $(w+1) \times 2$ of XORPP* as follows:

$$\mathbf{L} \cdot \mathbf{K} = \begin{bmatrix} 1 & \alpha^1 & \alpha^2 & \cdots & \alpha^w \\ 1 & \alpha^2 & \alpha^4 & \cdots & \alpha^{2w} \end{bmatrix}^\top \cdot \begin{bmatrix} K_0 \\ K_1 \end{bmatrix},$$

where the elements are in $\mathbb{F}_2^n$, and $\alpha \in \mathbb{F}_2^n$ is a primitive element, which is often $\alpha = 2$, that is the polynomial $\mathtt{x}^1$ for practical values of $\mathbb{F}_2^n \cdot p(\mathtt{x})$ is an irreducible modulus polynomial in $\mathbb{F}_2^n$. Note that any two rows of the matrix $\mathbf{L}$ above are linearly independent. We refer to the instantiation of XORPP* with matrix $\mathbf{L}$ as XORPP. We define the concrete nonce- and public-permutation-based encryption scheme CENCPP in Algorithm 8. Since any two rows in the key-scheduling matrix of CENCPP are linearly independent, the security of CENCPP follows from Theorems 6.5 and 6.6.

6.6 Domain-separated Variants

DS-SoEM is a sum of Even-Mansour constructions with $d = 1$ bit of domain separation, i.e., it uses $(n-1)$-bit message inputs and fixes the last bit to encode domains that are distinct for each permutation. Let $P \xleftarrow{} \mathsf{Perm}(\mathbb{F}_2^n)$ and $\mathbf{K} \xleftarrow{\text{def}} (K_0, K_1) \xleftarrow{} field_2^n)^2$. We define $\mathsf{DS\text{-}SoEM}[P]_{K_0,K_1} : (\mathbb{F}_2^n)^2 \times \mathbb{F}_2^{n-1} \to \mathbb{F}_2^n$ to compute $\mathsf{DS\text{-}SoEM}[P]_{K_0,K_1}(M)$, as listed in Algorithm 9. Note that we use $(n-1)$ bits of the key in forward direction only, i.e., the domain is **not** masked. For this

Algorithm 9 Specification of DS-CENCPP, DS-XORPP, and DS-SoEM

101: **function** DS-CENCPP$[P,w].\mathcal{E}_{\mathbf{K}}(N,M)$
102: $M_1 \cdots M_m \leftarrow_n M$
103: $\ell \leftarrow \lceil m/w \rceil$
104: **for** $i \leftarrow 0..\ell-1$ **do**
105: $j \leftarrow i \cdot w$
106: $(S_{j+1} \| \cdots \| S_{j+w})$
107: $\leftarrow$ DS-XORPP$[P,w]_{\mathbf{K}}(N \| \langle i \rangle_\mu)$
108: **for** $k \leftarrow j+1..j+w$ **do**
109: $C_k \leftarrow S_k \oplus M_k$
110: **return** $\mathsf{msb}_{|M|}(C_1 \| \cdots \| C_m)$

201: **function** DS-CENCPP$[P,w].\mathcal{D}_{\mathbf{K}}(N,C)$
202: **return** DS-CENCPP$[P,w].\mathcal{E}_{\mathbf{K}}(N,C)$

301: **function** DS-XORPP$[P,w]_{\mathbf{K}}(I)$
302: $(K_0,K_1) \leftarrow \mathbf{K}$
303: $L_0 \leftarrow K_0 \oplus K_1$
304: $\widehat{U}_0 \leftarrow (I \oplus \mathsf{msb}_{n-d}(L_0)) \| \langle 0 \rangle_d$
305: $\widehat{X}_0 \leftarrow P(\widehat{U}_0) \oplus L_0$
306: **for** $\alpha \leftarrow 1..w$ **do**
307: $L_\alpha \leftarrow (2^\alpha \cdot K_0) \oplus (2^{2\alpha} \cdot K_1)$
308: $\widehat{U}_\alpha \leftarrow (I \oplus \mathsf{msb}_{n-d}(L_\alpha)) \| \langle \alpha \rangle_d$
309: $\widehat{X}_\alpha \leftarrow P(\widehat{U}_\alpha) \oplus L_\alpha$
310: $O_\alpha \leftarrow \widehat{X}_\alpha \oplus \widehat{X}_0$
311: **return** $(O_1 \| \cdots \| O_w)$

401: **function** DS-SoEM$[P,w]_{\mathbf{K}}(M)$
402: $(K_0,K_1) \leftarrow \mathbf{K}$
403: $\widehat{U}_0 \leftarrow (\mathsf{msb}_{n-1}(K_0) \oplus M) \| \langle 0 \rangle_1$
404: $\widehat{U}_1 \leftarrow (\mathsf{msb}_{n-1}(K_1) \oplus M) \| \langle 1 \rangle_1$
405: $\widehat{V}_0 \leftarrow P(\widehat{U}_0)$
406: $\widehat{V}_1 \leftarrow P(\widehat{U}_1)$
407: **return** $\widehat{V}_0 \oplus \widehat{V}_1 \oplus K_0 \oplus K_1$

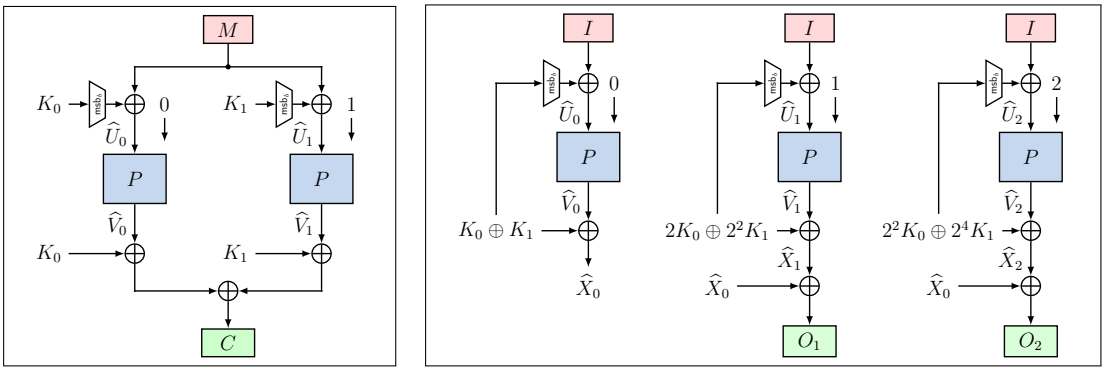
construction, we set a zero bit for the call to the left and a one bit for the domain input to the right permutation. An illustration is given in Figure 6.5a.

DS-XORPP: We can define DS-XORPP$[P,w]$ similarly. Here, $d \geq \lceil \log_2(w+1) \rceil$ bits are necessary to separate the domains. Let again $\mathbf{K} \stackrel{\text{def}}{=} (K_0,K_1) \in (\mathbb{F}_2^n)^2$. We define DS-XORPP$[P,w] : (\mathbb{F}_2^n)^2 \times \mathbb{F}_2^{n-d} \to (\mathbb{F}_2^n)^w$ as given in Algorithm 9 and a pictorial depiction is given in Figure 6.5b. The input domain is $\mathbb{F}_2^{n-d}$. Again, we use $(n-d)$ bits of the key in forward direction only, i.e., the domain is **not** masked.

DS-CENCPP is then defined naturally. Let $\mathcal{N} \stackrel{\text{def}}{=} \mathbb{F}_2^{\nu+\mu}$ be a nonce space such that $\nu+\mu = n-d$. Let $N \in \mathcal{N}$ be a nonce and $M \in \mathbb{F}_2^*$ be a message. Let again $\mathbf{K} \stackrel{\text{def}}{=} (K_0,K_1) \in (\mathbb{F}_2^n)^2$ and $P \in \mathsf{Perm}(\mathbb{F}_2^n)$. Then, the encryption and decryption algorithms $\mathcal{E}$ and $\mathcal{D}$ of DS-CENCPP$[P,w]_{\mathbf{K}}(N,M)$ are provided in Algorithm 9.

6.7 Distinguishers on DS-SoEM and DS-XORPP

This section provides a distinguisher on DS-SoEM that matches our security bound and distinguishers on variants that mask also the domain and use only a single key. Thus, they show that our bound is tight (up to a logarithmic factor) and explain our designs.



(A) DS-SoEM$[P]_{K_0,K_1}$. (B) DS-XORPP$[P,w]_{K_0,K_1}$.

FIGURE 6.5: The domain-separated constructions, here with DS-XORPP$[P,2]$. The trapezoids represent truncation of the key masks at the input to their $b = n - d$ most significant bits. For DS-SoEM, the trapezoid truncates the key masks at the input to their $n-1$ most significant bits, whereas for DS-XORPP, it truncates $n-2$ most significant bits.

The existing distinguisher from [17, Proposition 2] on SoEM12 (one permutation, two independent keys) needed $3 \cdot 2^{n/2}$ queries:

1. For $i \leftarrow 1..2^{n/2}$, query $M^i = (\langle i \rangle_{n/2} \,\|\, 0^{n/2})$ to get C^i, and $M^{*i} = M^i \oplus 1$ to get C^{*i}.

2. For $j \leftarrow 1..2^{n/2}$, query $M'^j = (0^{n/2} \| \langle j \rangle_{n/2})$ to get C'^j, and $M'^{*j} = M'^j \oplus 1$ for C'^{*j}.

After $3 \cdot 2^{n/2}$ queries, there exists one tuple $(M^i, M^{*i}, M'^j, M'^{*j})$ such that $M^i \oplus M'^j = M^{*i} \oplus M'^{*j} = K_0 \oplus K_1$, which can be seen if $C^i = C'^j$ and $C^{*i} = C'^{*j}$. Note that the fourth set of queries M'^{*j} is not new, but can be taken from the other sets. For SoEM, the distinguisher exploited that one can find two queries M and M' such that their inputs to the left and right permutation are **swapped**. For DS-SoEM, this distinguisher does **not** apply since the domain separation prevents that the permutation inputs can be swapped.
A working distinguisher can be constructed with significant advantage and $6c \cdot 2^{2n/3}$ queries, for small constant $c \geq 1$. Let $q = c \cdot 2^{2n/3}$.

1. For $j \leftarrow 1..q$, query a random M^j without replacement, get C^j. Moreover, query $M^{*j} = M^j \oplus \langle 1 \rangle_{n-1}$ to get C^{*j} and store (C^j, C^{*j}).

2. For $i \leftarrow 1..q$, sample $u_0^i \in \mathbb{F}_2^{n-1}$ without replacement, query $U_0^i = (u_0^i \;\|\; \langle 0 \rangle_1)$ to P, and obtain V_0^i. Query $U_0^{*i} = U_0^i \oplus 10^{n-1}$ to P to obtain V_0^{*i} and store (V_0^i, V_0^{*i}).

3. For $k \leftarrow 1..q$, sample $u_1^k \in \mathbb{F}_2^{n-1}$ without replacement, query $U_1^k = (u_1^k \parallel \langle 1 \rangle_1)$ to P, and get V_1^k. Query $U_1^{*k} = U_1^k \oplus 10^{n-1}$ to P to get V_1^{*k} and store (V_1^k, V_1^{*k}).

With high probability, there exists a tuple (M^j, U_0^i, U_1^k) such that

$$((M^j \oplus \mathsf{msb}_{n-1}(K_0)) \parallel \langle 0 \rangle_1) = U_0^i \text{ and } ((M^j \oplus \mathsf{msb}_{n-1}(K_1)) \parallel \langle 1 \rangle_1) = U_1^k .$$

If this is the case, check if

$$((M^{*j} \oplus \mathsf{msb}_{n-1}(K_0) \parallel \langle 0 \rangle_1) = U_0^{*i} \text{ and } ((M^{*j} \oplus \mathsf{msb}_{n-1}(K_1)) \parallel \langle 1 \rangle_1) = U_1^{*k}$$

also holds. If yes, return real; return random otherwise.

Why not also mask the domain? If the keys K_0 and K_1 would be XORed also to the domains, it could hold for DS-SoEM that $\mathsf{lsb}_1(K_0) \oplus \langle 0 \rangle_1 = \mathsf{lsb}_1(K_1) \oplus \langle 1 \rangle_1$. Similarly, it could hold for DS-XORPP for any distinct pair $i, j \in [0..w]$ that

$$\mathsf{lsb}_d(2^i K_0 \oplus 2^{2i} K_1) \oplus \langle i \rangle_d = \mathsf{lsb}_d(2^j K_0 \oplus 2^{2j} K_1)) \oplus \langle j \rangle_d$$

This would counter the distinct domains. While the distinguisher from [17, Proposition 2] would still be inapplicable, a slide attack (cf. [189, 190]) could become a threat. In the following, we consider a variant of DS-SoEM$[P]$ with the permutation inputs

$$U_0^i \leftarrow (M^i \parallel \langle 0 \rangle_1) \oplus K_0 \quad \text{and} \quad U_1^i \leftarrow (M^i \parallel \langle 1 \rangle_1) \oplus K_1.$$

Let $K_0, K_1 \xleftarrow{\$} \mathbb{F}_2^n$, and $\mathsf{lsb}_1(K_0) \oplus \mathsf{lsb}_1(K_1) = 1$, i.e., their least significant bit differs, which holds with probability 0.5. Let $c \in \mathbb{F}_2^{n-1}$ be a non-zero constant. Then:

1. For $i \leftarrow 1..2^{n/2}$, sample $M^i = (\langle i \rangle_{n/2} \parallel 0^{n/2-1})$, obtain C^i and store it.
2. Derive $M^{*i} = M^i \oplus c$, and obtain its corresponding ciphertext C^{*i}.
3. Similarly, for $j \leftarrow 1..2^{n/2-1}$, sample $M^j = (0^{n/2} \parallel \langle j \rangle_{n/2-1})$, obtain C^j and store it.
4. Derive $M^{*j} = M^j \oplus c$, and obtain its corresponding ciphertext C^{*j}.

5. If $\exists i \neq j$ such that $C^i = C^j$ and $C^{*i} = C^{*j}$, return real; return random otherwise.

Then, there exists a pair s. t. $M^i \oplus M^j = \mathsf{msb}_{n-1}(K^0 \oplus K^1)$. It follows that $U_0^i = U_1^j$ and $U_0^j = U_1^i$, from which $C^i = C^j$ follows. A similar argument holds for $C^{*i} = C^{*j}$.

A distinguisher on a single-key variant shows that the tempting approach of using a single-key domain-separated variant of DS-SoEM does not offer sufficient security in practice. Since the domain differs in both permutation calls, this would ensure distinct inputs on both sides of each query. However, this construction would possess only $n/2$-bit PRF security. In the following, we sketch a distinguisher, where we assume that both keys K_0 and K_1 are replaced by a single key K.

1. For $i \leftarrow 1..2^{n/2}$, sample $M^i = (\langle i \rangle_{n/2} \parallel 0^{n/2-1})$ to obtain C^i and store them. To each M^i, associate a plaintext $M'^i = M^i \oplus (10^{n-2})$ and its output C'^i.

2. For $j \leftarrow 1..2^{n/2-1}$, ask for the primitive encryption of $U_0^j = (\langle 0 \rangle_{n/2} \parallel \langle i \rangle_{n/2-1} \parallel \langle 0 \rangle_1)$ to obtain V_0^j. Query $U_0'^j = U_0^j \oplus (10^{n-1})$ to obtain $V_0'^j$.

3. Similarly, for $j \leftarrow 1..2^{n/2-1}$, ask for the primitive encryption of $U_1^j = (\langle 0 \rangle n/2 \parallel \langle i \rangle_{n/2-1} \parallel \langle 1 \rangle_1)$ to obtain V_1^j. Query $U_1'^j = U_1^j \oplus (10^{n-1})$ to obtain $V_1'^j$.

4. If there exists one tuple i, j s. t. $C^i = V_0^j \oplus V_1^j$ and $C'^i = V_0'^j \oplus V_1'^j$, output real and output random otherwise.

With probability one, there will be one collision for the real construction, whereas the probability of the event is negligible in the ideal world.

6.8 Security Analysis of DS-CENCPP and DS-SoEM

Here, we study the NE security of DS-CENCPP.

6.8.1 Security Result of DS-CENCPP

As before, let $P \xleftarrow{\$} \mathsf{Perm}(\mathbb{F}_2^n)$ and $K_0, K_1 \xleftarrow{\$} \mathcal{K}$ be independent secret keys; we write $\mathbf{K} = (K_0, K_1)$ for brevity. Again, we conducted a two-step analysis, where we consider (1) the PRF security of DS-XORPP$[P, w]$ and (2) the NE security of DS-CENCPP$[P, w]$. For simplicity of notation, we write DS-XORPP$[P, w]$ as DS-XORPP. Moreover, we write DS-CENCPP$[P, w]$ as DS-CENCPP.

Theorem 6.10. It holds that

$$\mathbf{Adv}^{\mathsf{DS\text{-}CENCPP}}_{\mathsf{NE}}(q_p, q_c, m) \leq \mathbf{Adv}^{\mathsf{DS\text{-}XORPP}}_{\mathsf{PRF}}\left(q_p, \frac{m}{w} q_c\right) .$$

The proof follows a similar argumentation as that of CENCPP^*.

Theorem 6.11. Let $v \stackrel{\text{def}}{=} w + 1$ and $q_c + v(q_p + q_c) \leq 2^n/2(w+1)$. It holds that

$$\begin{aligned}\mathbf{Adv}^{\mathsf{DS\text{-}XORPP}}_{\mathsf{PRF}}(q_p, q_c) \leq\ & \frac{2^{2d} v^2 q_c q_p^2}{2^{2n+1}} + \frac{2^{d+1} w v q_c q_p^2}{2^{2n}} + \frac{2^d v^3 q_c q_p^2}{2^{2n+1}} + \frac{2^{2d} w^3 q_c^2 q_p}{2^{2n}} \\ &+ \frac{2^{2d+1} w^3 q_c q_p^2}{2^{2n}} + \frac{2^{2d} w^4 q_c^2 q_p}{2^{2n}} + \frac{2^{2d+1} w^4 q_c^2 q_p^2}{2^{3n}} \\ &+ \frac{w q_c}{2^n} + \frac{\binom{w}{2} q_c}{2^n} + \frac{4v^4 q_c^3 + 4v^4 q_c^2 q_p + v^4 q_c q_p^2}{2^{2n}} .\end{aligned}$$

Corollary 6.12. The Security of $\mathsf{DS\text{-}CENCPP}$ results from combining Theorem 6.10 and Theorem 6.11 as follows:

$$\begin{aligned}\mathbf{Adv}^{\mathsf{DS\text{-}CENCPP}}_{\mathsf{NE}}(q_p, q_c, m) \ \leq\ & \frac{2^{2d} v m q_c q_p^2}{2^{2n}} + \frac{2^{d+1} v m q_c q_p^2}{2^{2n}} + \frac{2^d v^2 m q_c q_p^2}{2^{2n}} \\ &+ \frac{2^{2d} w m^2 q_c^2 q_p}{2^{2n}} + \frac{2^{2d+1} w^2 m q_c q_p^2}{2^{2n}} + \frac{2^{2d} w^2 m^2 q_c^2 q_p}{2^{2n}} \\ &+ \frac{2^{2d+1} m^2 w^2 q_c^2 q_p^2}{2^{3n}} + \frac{m q_c}{2^n} + \frac{m w q_c}{2^{n+1}} \\ &+ \frac{4(w+7) m^3 q_c^3 + 16 v^2 m^2 q_c^2 q_p + 2 v^3 m q_c q_p^2}{2^{2n}} ,\end{aligned}$$

where m is the maximum number of message blocks among all q_c queries.

For the bound in Corollary 6.12, we used that $v^2/w \leq 2v$ and $v^4/w^3 \leq (w+7)$ and $v^4/w \leq 2v^3$.

Proof of Theorem 6.11. We fix a non-trivial information-theoretic deterministic distinguisher $\mathbf{D}$ that is given access to the oracle $\mathsf{DS\text{-}XORPP}[P, w]_{\mathbf{K}}$ and the primitive oracle P, for a pair of n-bit random keys $\mathbf{K} = (K_0, K_1)$ and an n-bit random permutation P, in the real world. In the ideal world, it is given access to a random function, which answers each query of $\mathbf{D}$ by w blocks of n bits uniform and independent random strings $\mathbf{O} = (O_1, \ldots, O_w)$ and to an n-bit random permutation P. We assume that $\mathbf{D}$ can ask exactly q_c construction queries and $q'_p = (w+1)q_p$ forward and backward primitive queries altogether to the primitive P, where U^i_α is the i-th forward primitive query to the primitive P whose last d bits is equal to $\langle \alpha \rangle_d$ and V^i_α is the corresponding response and vice versa. We summarise the interaction

of the distinguisher $\mathbf{D}$ with the oracles in a transcript τ which is partitioned into $\tau = \tau_c \cup \tau_0 \cup \ldots \cup \tau_w$, where the construction transcript τ_c contains the queries to and responses from the construction oracle: $\tau_c = \{(I^1, \mathbf{O}^1), \ldots, (I^{q_c}, \mathbf{O}^{q_c})\}$ and the primitive transcripts $\tau_\alpha = \{(U_\alpha^1, V_\alpha^1), \ldots, (U_\alpha^{q_p}, V_\alpha^{q_p})\}$ contain exactly the queries to and responses from permutation P such that the last d bits of U_α^i for all $i \in [q_p]$ and for all $\alpha \in [0..w]$ is equal to $\langle \alpha \rangle_d$. Since $\mathbf{D}$ is non-trivial, τ does not contain duplicate elements. After the interaction is over, we release the keys K_0, K_1, which are the keys used in the construction in the real world, and sampled uniformly at random in the ideal world, to the distinguisher before it outputs its decision bit. Hence, the transcript τ becomes $\tau = \tau_c \cup \tau_0 \cup \ldots \cup \tau_w \cup \{(K_0, K_1)\}$. With the help of the transcript τ, $\mathbf{D}$ can compute the all the inputs $\widehat{U}_\alpha^i$ to the permutations P for q_c construction queries using the following equation

$$\widehat{U}_\alpha^i \stackrel{\text{def}}{=} I^i \oplus \mathsf{msb}_{n-d}(2^\alpha \cdot K_0 \oplus 2^{2\alpha} \cdot K_1) \,\|\, \langle \alpha \rangle_d \,, \tag{6.24}$$

where $\alpha \in [0..w]$ and $i \in [q_c]$. We partition the set of all attainable transcripts Att into two disjoint sets of GoodT and BadT that represent good and bad transcripts. We denote by Θ_{real} and Θ_{ideal} random variables that represent the distribution of transcripts in the real and the ideal world, respectively.

Bad Transcripts: Let $\tau = \tau_c \cup \tau_0 \cup \ldots \cup \tau_w \cup \{(K_0, K_1)\}$ be an attainable transcript. Since, the distinguisher is given the keys $\mathbf{K}$, it can compute all the permutation inputs $(\widehat{U}_\alpha^i)_{i \in [q_c], \alpha \in [0..w]}$ using Equation (6.24). We say that τ is bad if any one of the following bad events holds.

1. *Two inputs to the permutations for a construction query simultaneously collides with the input of corresponding two primitive queries.*

   - bad_1: $\exists i \in [q_c], j, k \in [q_p]$ and distinct permutation indices $\alpha, \beta \in [0..w]$ such that. $(\widehat{U}_\alpha^i = U_\alpha^j) \wedge (\widehat{U}_\beta^i = U_\beta^j)$.

2. *For a construction query, one of the inputs collides with the input of a primitive query, which makes the output of another permutation call of the same construction query collide with the output of another primitive query.*

   - bad_2: $\exists i \in [q_c], j, k \in [q_p]$ and permutation indices $\alpha \in [w]$ and $\beta \in [0..w]$ such that $(\widehat{U}_0^i = U_0^j) \wedge (V_0^j \oplus O_\alpha^i \oplus (2^\alpha \oplus 1) \cdot K_0 \oplus (2^{2\alpha} \oplus 1) \cdot K_1 = V_\beta^k)$.
   - bad_3: $\exists i \in [q_c], j, k \in [q_p]$ and permutation indices $\alpha \in [w]$ and $\beta \in [0..w]$ such that $(\widehat{U}_\alpha^i = U_\alpha^j) \wedge (V_\alpha^j \oplus O_\alpha^i \oplus (2^\alpha \oplus 1) \cdot K_0 \oplus (2^{2\alpha} \oplus 1) \cdot K_1 = V_\beta^k)$.

- bad_4: $\exists i \in [q_c], j, k \in [q_p]$ and disinct permutation indices $\alpha, \beta \in [w]$ and $\gamma \in [0..w]$ such that $(\widehat{U}^i_\alpha = U^j_\alpha) \wedge (V^j_\alpha \oplus O^i_\alpha \oplus O^i_\beta \oplus (2^\alpha \oplus 2^\beta) \cdot K_0 \oplus (2^{2\alpha} \oplus 2^{2\beta}) \cdot K_1 = V^k_\gamma)$.

3. *For two construction queries i and j, one of the inputs of i-th construction query collides with the input of a primitive query, and one of the inputs of j-th construction query collides with the input of a primitive query, and the output of any two permutation calls collide.*

   - bad_5: $\exists i, j \in [q_c], k, l \in [q_p]$ and permutation indices $\beta, \gamma \in [w]$ such that $(\widehat{U}^i_0 = U^k_0) \wedge (\widehat{U}^j_0 = U^l_0) \wedge (V^k_0 \oplus O^i_\beta \oplus O^j_\gamma \oplus V^l_0 = (2^\beta \oplus 2^\gamma) \cdot K_0 \oplus (2^{2\beta} \oplus 2^{2\gamma}) \cdot K_1)$.
   - bad_6: $\exists i, j \in [q_c], k, l \in [q_p]$ and permutation index $\alpha \in [w]$ and permutation indices $\beta, \gamma \in [0..w]$ such that $\alpha \neq \beta, \alpha \neq \gamma$ and $(\widehat{U}^i_\alpha = U^k_\alpha) \wedge (\widehat{U}^j_\alpha = U^l_\alpha) \wedge (V^k_\alpha \oplus V^l_\alpha \oplus O^i_\alpha \oplus O^i_\beta \oplus O^j_\alpha \oplus O^j_\gamma = (2^\beta \oplus 2^\gamma) \cdot K_0 \oplus (2^{2\beta} \oplus 2^{2\gamma}) \cdot K_1)$.
   - bad_7: $\exists i, j \in [q_c], k, l \in [q_p]$ and permutation index $\alpha \in [w]$ and permutation indices $\beta \in [w]$ and $\gamma \in [0..w]$ such that $\alpha \neq \gamma$ and $(\widehat{U}^i_0 = U^k_0) \wedge (\widehat{U}^j_\alpha = U^l_\alpha) \wedge (V^k_0 \oplus V^l_\alpha \oplus O^i_\beta \oplus O^j_\alpha \oplus O^j_\gamma = (1 \oplus 2^\alpha \oplus 2^\beta \oplus 2^\gamma) \cdot K_0 \oplus (1 \oplus 2^{2\alpha} \oplus 2^{2\beta} \oplus 2^{2\gamma}) \cdot K_1)$.
   - bad_8: $\exists i, j \in [q_c], k, l \in [q_p]$ and distinct permutation indices $\alpha, \beta \in [w]$ and permutation indices $\gamma, \rho \in [0..w]$ such that $\alpha \neq \gamma, \rho \neq \beta$ and $(\widehat{U}^i_\alpha = U^k_\alpha) \wedge (\widehat{U}^j_\beta = U^l_\beta) \wedge (V^k_\alpha \oplus V^l_\beta \oplus O^i_\alpha \oplus O^i_\gamma \oplus O^j_\beta \oplus O^j_\rho = (2^\rho \oplus 2^\alpha \oplus 2^\beta \oplus 2^\gamma) \cdot K_0 \oplus (2^{2\rho} \oplus 2^{2\alpha} \oplus 2^{2\beta} \oplus 2^{2\gamma}) \cdot K_1)$.
   - bad_9: $\exists i \in [q_c]$ and a permutation index $\alpha \in [w]$ such that $O^i_\alpha = (K_0 \oplus K_1) \oplus (2^\alpha K_0 \oplus 2^{2\alpha} K_1)$.
   - bad_{10}: $\exists i \in [q_c]$ and distinct permutation indices $\alpha, \beta \in [w]$ such that $O^i_\alpha \oplus O^i_\beta = (2^\alpha \oplus 2^\beta) \cdot K_0 (2^{2\alpha} \oplus 2^{2\beta}) \cdot K_1$.

Lemma 6.13. It holds that

$$\Pr\left[\Theta_{\text{ideal}} \in \textsc{BadT}\right] \leq \frac{2^{2d}\binom{w+1}{2} q_c q_p^2}{2^{2n}} + \frac{2^{d+1} w(w+1) q_c q_p^2}{2^{2n}} + \frac{2^d (w+1)\binom{w+1}{2} q_c q_p^2}{2^{2n}} + \frac{2^{2d} w^3 q_c^2 q_p}{2^{2n}} + \frac{2^{2d+1} w^3 q_c q_p^2}{2^{2n}} + \frac{2^{2d} w^4 q_c^2 q_p}{2^{2n}} + \frac{2^{2d+1} w^4 q_c^2 q_p^2}{2^{3n}} + \frac{w q_c}{2^n} + \frac{\binom{w}{2} q_c}{2^n}.$$

Proof. In the following, we study the probabilities of the individual bad events. Before that, we recall the key-scheduling matrix $\mathbf{A}$ as follows:

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 2^2 & \dots & 2^w \\ 1 & 2^2 & 2^4 & \dots & 2^2 w \end{bmatrix}^\top .$$

bad_1 This event considers the collision between the input of P corresponding to a construction query whose last d bits is $\langle \alpha \rangle_d$ and the input to P corresponding to a primitive query whose last d bits is $\langle \alpha \rangle_d$, and a similar collision corresponding to construction and primitive query whose last d bits is $\langle \beta \rangle_d$. To bound the event, it must hold that

$$\begin{aligned} &\mathsf{msb}_{n-d}(2^\alpha \cdot K_0 \oplus 2^{2\alpha} \cdot K_1) = I^i \oplus \mathsf{msb}_{n-d}(U^j_\alpha) \quad \text{and} \\ &\mathsf{msb}_{n-d}(2^\beta \cdot K_0 \oplus 2^{2\beta} \cdot K_1) = I^i \oplus \mathsf{msb}_{n-d}(U^k_\beta)\,. \end{aligned}$$

with $[2^\alpha \;\; 2^{2\alpha}]$ and $[2^\beta \;\; 2^{2\beta}]$ as the $(\alpha{+}1)$-th and the $(\beta{+}1)$-th row of $\mathbf{A}$ respectively. The two equations can be seen as

$$\mathsf{msb}_{n-d}\left(\mathbf{A}' \cdot \mathbf{K}\right) = \mathsf{msb}_{n-d}\left(\begin{bmatrix} 2^\alpha & 2^{2\alpha} \\ 2^\beta & 2^{2\beta} \end{bmatrix} \cdot \begin{bmatrix} K_0 \\ K_1 \end{bmatrix}\right) = \begin{bmatrix} I^i \oplus \mathsf{msb}_{n-d}(U^j_\alpha) \\ I^i \oplus \mathsf{msb}_{n-d}(U^k_\beta) \end{bmatrix}$$

Since all rows of $\mathbf{A}$ are pairwise linearly independent, $\mathbf{A}'$ is non-singular. Moreover, K_0 and K_1 are uniform random variables over $\{0,1\}^n$. Thus, we can apply Lemma 6.2 and the probability of this event for a fixed choice of indices is $2^{-2(n-d)}$. Since one can choose α and β in $\binom{w+1}{2}$ ways, we obtain from the union bound over all indices

$$\Pr[\mathsf{bad}_1] \leq \frac{2^{2d}\binom{w+1}{2} q_c q_p^2}{2^{2n}}\,.$$

bad_2 This event considers the collision between the input of P corresponding to a construction query whose last d bits is $\langle 0 \rangle_d$ and the input to P corresponding to a primitive query whose last d bits is $\langle 0 \rangle_d$, and the collision between the output of P corresponding to the same construction query whose last d bits is $\langle \alpha \rangle_d$ and the output of P corresponding to a primitive query whose last d bits is $\langle \beta \rangle_d$ for $\alpha \in [w]$ and $\beta \in [0..w]$. For this event, it must hold that

$$\mathsf{msb}_{n-d}(K_0 \oplus K_1) = I^i \oplus \mathsf{msb}_{n-d}(U^j_0) \quad \text{and}$$

$$(2^{\alpha} \oplus 1) \cdot K_0 \oplus (2^{2\alpha} \oplus 1) \cdot K_1 = O^i_{\alpha} \oplus V^j_0 \oplus V^k_{\beta} \,.$$

Note that the matrix

$$\mathbf{A}' = \begin{bmatrix} 1 & 1 \\ 2^{\alpha} \oplus 1 & 2^{2\alpha} \oplus 1 \end{bmatrix}$$

is non-singular. Since K_0 and K_1 are uniform random variables over $\{0,1\}^n$, the probability of this event for a fixed choice of indices is $2^d/2^{2n}$ as follows from Lemma 6.1. Since one can choose α in w ways and β in $w+1$ ways, we obtain from the union bound over all indices

$$\Pr[\mathsf{bad}_2] \leq \frac{2^d w(w+1) q_c q_p^2}{2^{2n}} \,.$$

bad_3 This event considers the collision between the input of P corresponding to a construction query whose last d bits is $\langle \alpha \rangle_d$ and the input to P corresponding to a primitive query whose last d bits is $\langle \alpha \rangle_d$, and the collision between the output of P corresponding to the same construction query whose last d bits is $\langle 0 \rangle_d$ and the output of P corresponding to a primitive query whose last d bits is $\langle \beta \rangle_d$ for $\alpha \in [w]$ and $\beta \in [0..w]$. For this event, it must hold that

$$\begin{aligned} \mathsf{msb}_{n-d}(2^{\alpha} \cdot K_0 \oplus 2^{2\alpha} \cdot K_1) &= I^i \oplus \mathsf{msb}_{n-d}(U^j_{\alpha}) \quad \text{and} \\ (2^{\alpha} \oplus 1) \cdot K_0 \oplus (2^{2\alpha} \oplus 1) \cdot K_1 &= O^i_{\alpha} \oplus V^j_{\alpha} \oplus V^k_{\beta} \,. \end{aligned}$$

Note that the matrix

$$\mathbf{A}' = \begin{bmatrix} 2^{\alpha} & 2^{2\alpha} \\ 2^{\alpha} \oplus 1 & 2^{2\alpha} \oplus 1 \end{bmatrix}$$

is non-singular. Since K_0 and K_1 are uniform random variables over $\{0,1\}^n$, the probability of this event for a fixed choice of indices is $2^d/2^{2n}$ as follows from Lemma 6.1. Since one can choose α in w ways and β in $w+1$ ways, we obtain from the union bound over all indices

$$\Pr[\mathsf{bad}_3] \leq \frac{2^d w(w+1) q_c q_p^2}{2^{2n}} \,.$$

bad$_4$ This event considers the collision between the input of P corresponding to a construction query whose last d bits is $\langle\alpha\rangle_d$ and the input to P corresponding to a primitive query whose last d bits is $\langle\alpha\rangle_d$, and the collision between the output of P corresponding to the same construction query whose last d bits is $\langle\beta\rangle_d$ and the output of P corresponding to a primitive query whose last d bits is $\langle\gamma\rangle_d$ for α and $\beta \in [w]$ and $\gamma \in [0..w]$. For this event, it must hold that

$$\mathsf{msb}_{n-d}(2^\alpha \cdot K_0 \oplus 2^{2\alpha} \cdot K_1) = I^i \oplus \mathsf{msb}_{n-d}(U^j_\alpha) \quad \text{and}$$
$$(2^\alpha \oplus 2^\beta) \cdot K_0 \oplus (2^{2\alpha} \oplus 2^{2\beta}) \cdot K_1 = O^i_\alpha \oplus O^i_\beta \oplus V^j_\alpha \oplus V^k_\gamma \,.$$

Note that the matrix

$$\mathbf{A}' = \begin{bmatrix} 2^\alpha & 2^{2\alpha} \\ 2^\alpha \oplus 2^\beta & 2^{2\alpha} \oplus 2^{2\beta} \end{bmatrix}$$

is non-singular. Since K_0 and K_1 are uniform random variables over $\{0,1\}^n$, the probability of this event for a fixed choice of indices is $2^d/2^{2n}$ as follows from Lemma 6.1. Since one can choose α and β in $\binom{w+1}{2}$ ways and γ in $w+1$ ways, we obtain from the union bound over all indices

$$\Pr[\mathsf{bad}_4] \le \frac{2^d(w+1)\binom{w+1}{2}q_c q_p^2}{2^{2n}} \,.$$

bad$_5$ This event considers the collision between the input of P corresponding to the i-th construction query whose last d bits is $\langle 0\rangle_d$ and to that of the input of corresponding primitive query and the collision between the input of P corresponding to the j-th construction query whose last d bits is $\langle 0\rangle_d$ and to that of the input of corresponding primitive query and the collision between the output of P corresponding to the i-th construction query whose last d bits is $\langle\beta\rangle_d$ and the output of P corresponding to the j-th construction query whose last d bits is $\langle\gamma\rangle_d$ for $\beta, \gamma \in [w]$. For this event, it must hold that

$$\begin{cases} \mathsf{msb}_{n-d}(K_0 \oplus K_1) = I^i \oplus \mathsf{msb}_{n-d}(U^k_0), \\ \mathsf{msb}_{n-d}(K_0 \oplus K_1) = I^j \oplus \mathsf{msb}_{n-d}(U^l_0), \\ (2^\beta \oplus 2^\gamma) \cdot K_0 \oplus (2^{2\beta} \oplus 2^{2\gamma}) \cdot K_1 = V^k_0 \oplus O^i_\beta \oplus O^j_\gamma \oplus V^l_0 \,. \end{cases}$$

Note that the system of equations above can be rewritten as

$$\begin{cases} K_0 \oplus K_1 = U_0^k \oplus (I^i \| \langle x \rangle_d) = U_0^l \oplus (I^j \| \langle y \rangle_d), & \text{(E.1)} \\ (2^\beta \oplus 2^\gamma) \cdot K_0 \oplus (2^{2\beta} \oplus 2^{2\gamma}) \cdot K_1 = V_0^k \oplus O_\beta^i \oplus O_\gamma^j \oplus V_0^l \,, \end{cases}$$

where $x, y \in \{0,1\}^d$. We can easily observe that

$$\begin{aligned} \Pr[\text{(E.1)}] = \Pr[U_0^k \oplus (I^i \| \langle x \rangle_d) = U_0^l \oplus (I^j \| \langle y \rangle_d)] \\ \cdot \Pr[\text{(E.1)} \mid U_0^k \oplus (I^i \| \langle x \rangle_d) = U_0^l \oplus (I^j \| \langle y \rangle_d)]. \end{aligned} \tag{6.25}$$

Let's first fix the choice of indices of the two construction queries and the two primitive queries, and the values of β, γ, x and y. Now in the first case, if the last among four queries is a backward primitive query (w.l.o.g., suppose it's V_0^k to obtain U_0^k), then the probability of Equation (6.25) comes out to be $\frac{1}{2^n}.\frac{1}{2^n}$. The first $\frac{1}{2^n}$ comes from the randomness over U_0^k and the second $\frac{1}{2^n}$ comes from the randomness over $K_0 \oplus K_1$. And in the second case, if the last among four queries is a forward positive query (w.l.o.g., suppose it's U_0^k to obtain V_0^k) or a construction query (w.l.o.g., suppose it's I^i to obtain O^i), then the probability of Equation (6.25) comes out to be $1.\frac{1}{2^n}$. The $\frac{1}{2^n}$ comes from randomness over $K_0 \oplus K_1$. In both the cases, $\Pr[(2^\beta \oplus 2^\gamma) \cdot K_0 \oplus (2^{2\beta} \oplus 2^{2\gamma}) \cdot K_1 = V_0^k \oplus O_\beta^i \oplus O_\gamma^j \oplus V_0^l] = \frac{1}{2^n}$. The $\frac{1}{2^n}$ comes from randomness over $(2^\beta \oplus 2^\gamma) \cdot K_0 \oplus (2^{2\beta} \oplus 2^{2\gamma}) \cdot K_1$. Since the matrix

$$\begin{bmatrix} 1 & 1 \\ (2^\beta \oplus 2^\gamma) & (2^{2\beta} \oplus 2^{2\gamma}) \end{bmatrix}$$

is full-rank, the probability of the third equation, conditioned on the first two equations, comes out to be $\frac{1}{2^n}$, and as a result, the joint probability of all the three equations corresponding to bad_5 comes out to be $\frac{1}{2^{3n}}$ (in the first case) or $\frac{1}{2^{2n}}$ (in the second case). In the first case, one can choose i and j together in $\binom{q_c}{2}$ ways, and k and l in q_p ways each. In the second case, if the last query is a forward primitive query, then i and j can be chosen in $2\binom{q_c}{2}$ ways. But the value of the index corresponding to the last primitive query gets fixed once one fixes the value of the index of the other primitive query (This can be done in q_p ways). Similarly, if the last query is a construction query, then k and l can be chosen in q_p^2 ways.

But the value of the index corresponding to the last construction query gets fixed once one fixes the value of the index of the other construction query (This can be done in q_c ways). Moreover, β and γ together can be chosen in w^2 ways. Thus, we obtain from the union bound over all indices and all possible values of x and y,

$$\begin{aligned}\Pr[\mathsf{bad}_5] &\leq \max\left(\frac{2^{2d}w^2\binom{q_c}{2}q_p^2}{2^{3n}}, \frac{2^{2d}w^2\binom{q_c}{2}q_p}{2^{2n}}, \frac{2^{2d}w^2q_cq_p^2}{2^{2n}}\right)\\ &\leq \frac{2^{2d}w^2\binom{q_c}{2}q_p^2}{2^{3n}} + \frac{2^{2d}w^2\binom{q_c}{2}q_p}{2^{2n}} + \frac{2^{2d}w^2q_cq_p^2}{2^{2n}}\,.\end{aligned}$$

bad_6 This event considers the collision between the input of P corresponding to the i-th construction query whose last d bits is $\langle\alpha\rangle_d$ and to that of the input of corresponding primitive query and collision between the input of P corresponding to the j-th construction query whose last d bits is $\langle\alpha\rangle_d$ and to that of the input of corresponding primitive query for some $\alpha \in [w]$ and the collision between the output of P corresponding to the i-th construction query whose last d bits is $\langle\beta\rangle_d$ and the output of P corresponding to the j-th construction query whose last d bits is $\langle\gamma\rangle_d$ for $\beta, \gamma \in [0..w]$ such that $\alpha \neq \beta, \alpha \neq \gamma$. For this event, it must hold that

$$\begin{cases}\mathsf{msb}_{n-d}(2^\alpha \cdot K_0 \oplus 2^{2\alpha} \cdot K_1) = I^i \oplus \mathsf{msb}_{n-d}(U^k_\alpha),\\ \mathsf{msb}_{n-d}(2^\alpha \cdot K_0 \oplus 2^{2\alpha} \cdot K_1) = I^j \oplus \mathsf{msb}_{n-d}(U^l_\alpha),\\ (2^\beta \oplus 2^\gamma) \cdot K_0 \oplus (2^{2\beta} \oplus 2^{2\gamma}) \cdot K_1 = V^k_\alpha \oplus V^l_\alpha \oplus O^i_\alpha \oplus O^i_\beta \oplus O^j_\alpha \oplus O^j_\gamma\,.\end{cases}$$

Note that the above system of equations can be rewritten as

$$\begin{cases}2^\alpha \cdot K_0 \oplus 2^{2\alpha} \cdot K_1 = U^k_\alpha \oplus (I^i \| \langle x\rangle_d),\\ 2^\alpha \cdot K_0 \oplus 2^{2\alpha} \cdot K_1 = U^l_\alpha \oplus (I^j \| \langle y\rangle_d),\\ (2^\beta \oplus 2^\gamma) \cdot K_0 \oplus (2^{2\beta} \oplus 2^{2\gamma}) \cdot K_1 = V^k_\alpha \oplus V^l_\alpha \oplus O^i_\alpha \oplus O^i_\beta \oplus O^j_\alpha \oplus O^j_\gamma\,,\end{cases}$$

where $x, y \in \{0,1\}^d$. Using similar reasoning while bounding bad_5, we have

$$\Pr[\mathsf{bad}_6] \leq \frac{2^{2d}w^3\binom{q_c}{2}q_p^2}{2^{3n}} + \frac{2^{2d}w^3\binom{q_c}{2}q_p}{2^{2n}} + \frac{2^{2d}w^3q_cq_p^2}{2^{2n}}\,.$$

bad$_7$ This event considers the collision between the input of P corresponding to the i-th construction query whose last d bits is $\langle 0 \rangle_d$ and to that of the input of corresponding primitive query and collision between the input of P corresponding to the j-th construction query whose last d bits is $\langle \alpha \rangle_d$ and to that of the input of corresponding primitive query for some $\alpha \in [w]$ and the collision between the output of P corresponding to the i-th construction query whose last d bits is $\langle \beta \rangle_d$ and the output of P corresponding to the j-th construction query whose last d bits is $\langle \gamma \rangle_d$ for $\beta \in [w]$ and $\gamma \in [0..w]$ such that $\alpha \neq \gamma$. For this event, it must hold that

$$\begin{cases} \mathsf{msb}_{n-d}(K_0 \oplus K_1) = I^i \oplus \mathsf{msb}_{n-d}(U_0^k)\,, \\ \mathsf{msb}_{n-d}(2^\alpha \cdot K_0 \oplus 2^{2\alpha} \cdot K_1) = I^j \oplus \mathsf{msb}_{n-d}(U_\alpha^l)\,, \\ (1 \oplus 2^\alpha \oplus 2^\beta \oplus 2^\gamma) \cdot K_0 \oplus (1 \oplus 2^{2\alpha} \oplus 2^{2\beta} \oplus 2^{2\gamma}) \cdot K_1 \\ \quad = V_0^k \oplus V_\alpha^l \oplus O_\beta^i \oplus O_\alpha^j \oplus O_\gamma^j\,. \end{cases}$$

Note that the above system of equations can be rewritten as

$$\begin{cases} K_0 \oplus K_1 = U_0^k \oplus (I^i \| \langle x \rangle_d)\,, \\ 2^\alpha \cdot K_0 \oplus 2^{2\alpha} \cdot K_1 = U_\alpha^l \oplus (I^j \| \langle y \rangle_d)\,, \\ (1 \oplus 2^\alpha \oplus 2^\beta \oplus 2^\gamma) \cdot K_0 \oplus (1 \oplus 2^{2\alpha} \oplus 2^{2\beta} \oplus 2^{2\gamma}) \cdot K_1 \\ \quad = V_0^k \oplus V_\alpha^l \oplus O_\beta^i \oplus O_\alpha^j \oplus O_\gamma^j\,, \end{cases}$$

where $x, y \in \{0,1\}^d$. Let's first fix the choice of indices of the two construction queries and the two primitive queries, and the values of α, β, γ, x and y. The rank of the first two equations over K_0 and K_1 is 2 and hence the joint probability of the first two equations comes out to be $\frac{1}{2^{2n}}$. Once we bound the probability of the first two equations, K_0 and K_1 get fixed. The probability of the third equation depends on the randomness of different variables depending on the last query.

1. If the last among four queries is the construction query to obtain O^i from I^i, then the randomness comes from O_β^i, and the probability of the third equation comes out to be $\frac{1}{2^n}$.

2. If the last among four queries is the construction query to obtain O^j from I^j, then the randomness comes from $O_\alpha^j \oplus O_\gamma^j$, and the probability of the

third equation comes out to be $\frac{1}{2^n}$.

3. If the last among four queries is the forward primitive query to obtain V_0^k from U_0^k, then the randomness comes from V_0^k, and the probability of the third equation comes out to be $\frac{1}{2^n}$.

4. If the last among four queries is the forward primitive query to obtain V_α^l from U_α^l, then the randomness comes from V_α^l, and the probability of the third equation comes out to be $\frac{1}{2^n}$.

5. If the last among four queries is the backward primitive query to obtain U_0^k from V_0^k, then the probability of the third equation comes out to be 1.

6. If the last among four queries is the backward primitive query to obtain U_α^l from V_α^l, then the probability of the third equation comes out to be 1.

Now, one can choose i and j together in $2\binom{q_c}{2}$ ways. If the last query is a construction query or a forward primitive query, then one can choose k and l in q_p ways each. but if the last query is a backward primitive query, then the value of the index of the last primitive query gets fixed once one fixes the value of the index of the other primitive query (This can be done in q_p ways). Moreover, β, γ can be chosen in w^2 ways and α can be chosen in w ways. Thus, we obtain from the union bound over all indices and all possible values of x and y,

$$\Pr[\mathsf{bad}_7] \leq \max\left(\frac{2^{2d}w^3\binom{q_c}{2}q_p^2}{2^{3n}}, \frac{2^{2d}w^3\binom{q_c}{2}q_p}{2^{2n}}\right) \leq \frac{2^{2d}w^3\binom{q_c}{2}q_p^2}{2^{3n}} + \frac{2^{2d}w^3\binom{q_c}{2}q_p}{2^{2n}}.$$

bad$_8$ This event considers the collision between the input of P corresponding to the i-th construction query whose last d bits is $\langle\alpha\rangle_d$ and to that of the input of corresponding primitive query for some $\alpha \in [w]$ and collision between the input of P corresponding to the j-th construction query whose last d bits is $\langle\beta\rangle_d$ and to that of the input of corresponding primitive query for some $\beta \in [w]$ and the collision between the output of P corresponding to the i-th construction query whose last d bits is $\langle\gamma\rangle_d$ and the output of P corresponding to the j-th construction query whose last d bits is $\langle\rho\rangle_d$ for $\beta \in [w]$ and $\gamma, \rho \in [0..w]$ such that $\alpha \neq \gamma$ and $\rho \neq \beta$. For this event, it must hold that

$$\begin{cases} \mathsf{msb}_{n-d}(2^{\alpha} \cdot K_0 \oplus 2^{2\alpha} \cdot K_1) = I^i \oplus \mathsf{msb}_{n-d}(U^k_{\alpha})\,, \\ \mathsf{msb}_{n-d}(2^{\beta} \cdot K_0 \oplus 2^{2\beta} \cdot K_1) = I^j \oplus \mathsf{msb}_{n-d}(U^l_{\beta})\,, \\ (2^{\rho} \oplus 2^{\alpha} \oplus 2^{\beta} \oplus 2^{\gamma}) \cdot K_0 \oplus (2^{2\rho} \oplus 2^{2\alpha} \oplus 2^{2\beta} \oplus 2^{2\gamma}) \cdot K_1 \\ \quad = V^k_{\alpha} \oplus V^l_{\beta} \oplus O^i_{\alpha} \oplus O^i_{\gamma} \oplus O^j_{\beta} \oplus O^j_{\rho}\,. \end{cases}$$

Note that the above system of equations can be rewritten as

$$\begin{cases} 2^{\alpha} \cdot K_0 \oplus 2^{2\alpha} \cdot K_1 = U^k_{\alpha} \oplus (I^i \| \langle x \rangle_d)\,, \\ 2^{\beta} \cdot K_0 \oplus 2^{2\beta} \cdot K_1 = U^l_{\beta} \oplus (I^j \| \langle y \rangle_d)\,, \\ (2^{\rho} \oplus 2^{\alpha} \oplus 2^{\beta} \oplus 2^{\gamma}) \cdot K_0 \oplus (2^{2\rho} \oplus 2^{2\alpha} \oplus 2^{2\beta} \oplus 2^{2\gamma}) \cdot K_1 \\ \quad = V^k_{\alpha} \oplus V^l_{\beta} \oplus O^i_{\alpha} \oplus O^i_{\gamma} \oplus O^j_{\beta} \oplus O^j_{\rho}\,, \end{cases}$$

where $x, y \in \{0,1\}^d$. Using similar reasoning while bounding bad_7, we have

$$\Pr[\mathsf{bad}_8] \leq \frac{2^{2d} w^4 \binom{q_c}{2} q_p^2}{2^{3n}} + \frac{2^{2d} w^4 \binom{q_c}{2} q_p}{2^{2n}}\,.$$

bad_9 To bound the event, it must hold that

$$(2^{\alpha} + 1) \cdot K_0 \oplus (2^{2\alpha} + 1) \cdot K_1 = O^i_{\alpha}\,.$$

Since K_0 and K_1 are uniform random variables over $\{0,1\}^n$, the probability of this event for a fixed choice of indices is 2^n. Since one can choose α in at most w ways and i in at most q_c ways, we obtain from the union bound over all indices

$$\Pr[\mathsf{bad}_9] \leq \frac{w q_c}{2^n}\,.$$

bad_{10} To bound the event, it must hold that

$$(2^{\alpha} + 2^{\beta}) \cdot K_0 \oplus (2^{2\alpha} + 2^{2\beta}) \cdot K_1 = O^i_{\alpha} + O^i_{\beta}\,.$$

Since K_0 and K_1 are uniform random variables over $\{0,1\}^n$, the probability of this event for a fixed choice of indices is 2^n. Since one can choose α and β in at most

$\binom{w}{2}$ ways and i in at most q_c ways, we obtain from the union bound over all indices

$$\Pr[\mathsf{bad}_{10}] \leq \frac{\binom{w}{2} q_c}{2^n}\,.$$

The bound in Lemma 6.13 follows from the sum of probabilities of the individual bad events. □

Good Transcripts: It remains to study the interpolation probabilities of good transcripts.

Lemma 6.14. Let $v \stackrel{\text{def}}{=} w+1$ and $q_c + v(q_p + q_c) \leq 2^n/2(w+1)$. For any good transcript $\tau = \tau_c \cup \tau_0 \cup \ldots \tau_w \cup \{K_0, K_1\}$, it holds that

$$\frac{\Pr[\Theta_{\text{real}} = \tau]}{\Pr[\Theta_{\text{ideal}} = \tau]} \geq 1 - \frac{4v^4 q_c^3 + 4v^4 q_c^2 q_p + v^4 q_c q_p^2}{2^{2n}}.$$

Proof. Let $\mathsf{All}_{\text{real}}(\tau)$ denote the set of all oracles in the real world, and $\mathsf{All}_{\text{ideal}}(\tau)$ the set of all oracles in the ideal world. Let $\mathsf{Comp}_{\text{real}}(\tau)$ denote the fraction of oracles in the real world that are compatible with τ and $\mathsf{Comp}_{\text{ideal}}(\tau)$ the corresponding fraction in the ideal world. It holds that

$$\frac{\Pr[\Theta_{\text{real}} = \tau]}{\Pr[\Theta_{\text{ideal}} = \tau]} = \frac{|\mathsf{Comp}_{\text{real}}(\tau)| \cdot |\mathsf{All}_{\text{ideal}}(\tau)|}{|\mathsf{Comp}_{\text{ideal}}(\tau)| \cdot |\mathsf{All}_{\text{real}}(\tau)|}.$$

We can easily bound three out of four terms:

$$|\mathsf{All}_{\text{real}}(\tau)| = (2^n)^2 \cdot (2^n)!$$

since there exist $(2^n)^2$ keys and $2^n!$ possible permutations. The same argument holds in the ideal world, i.e.,

$$|\mathsf{All}_{\text{ideal}}(\tau)| = (2^n)^2 \cdot (2^n)! \cdot (2^{wn})^{2^n},$$

combined with $(2^{wn})^{2^n}$ random functions for the answers to the construction queries. Moreover,

$$|\mathsf{Comp}_{\text{ideal}}(\tau)| = (2^{wn})^{2^n - q_c} \cdot (2^n - (w+1) \cdot q_p)!$$

compatible oracles exist in the ideal world, where $(2^{wn})^{2^n - q_c}$ are the random function oracles that are compatible with the construction query transcripts and

$(2^n - (w+1)q_p)!$ permutation oracles that are compatible with primitive query transcripts. Now, it remains to determine $|\mathsf{Comp}_{\text{real}}(\tau)|$. Note that

$$|\mathsf{Comp}_{\text{real}}(\tau)| = \left| \left\{ P : \mathsf{DS\text{-}XORPP}[P,w]_{\mathbf{K}} \mapsto \tau_c \wedge \bigwedge_{\alpha=0}^{w} P \mapsto \tau_\alpha \right\} \right| .$$

For $\alpha \in [0..w]$, let Dom_α denotes the set $\{U_\alpha^i : (U_\alpha^i, V_\alpha^i) \in \tau_\alpha\}$ and Ran_α denotes the set $\{V_\alpha^i : (U_\alpha^i, V_\alpha^i) \in \tau_\alpha\}$, then $\bigwedge_{\alpha=0}^{w} P \mapsto \tau_\alpha$ equivalently means that for each $\alpha \in [0..w]$, P maps elements from Dom_α to Ran_α. Now, in order to compute $|\mathsf{Comp}_{\text{real}}(\tau)|$, we regroup the queries from $\tau_c, \tau_0, \ldots, \tau_w$ to $\tau_c^{\mathsf{new}}, \tau_0^{\mathsf{new}}, \ldots, \tau_w^{\mathsf{new}}$. Using a similar regrouping technique, the new transcript sets are initialised by their corresponding old parts, and reordered as follows:

1. if $\exists i \in [q_c], j \in [q_p]$, such that $\widehat{U}_0^i = U_0^j$, then
   - $\tau_c^{\mathsf{new}} \leftarrow \tau_c^{\mathsf{new}} \setminus \{(I^i, \mathbf{O}^i)\}$ and
   - for all $\alpha \in [w]$, $\tau_\alpha^{\mathsf{new}} \leftarrow \tau_\alpha^{\mathsf{new}} \cup \{(\widehat{U}_\alpha^i, V_0^j \oplus O_\alpha^i \oplus (2^\alpha \oplus 1) \cdot K_0 \oplus (2^{2\alpha} \oplus 1) \cdot K_1)\}$.
2. if $\exists i \in [q_c], j \in [q_p]$, and $\alpha \in [w]$ such that $\widehat{U}_\alpha^i = U_\alpha^j$, then
   - $\tau_c^{\mathsf{new}} \leftarrow \tau_c^{\mathsf{new}} \setminus \{(I^i, \mathbf{O}^i)\}$ and
   - $\tau_0^{\mathsf{new}} \leftarrow \tau_0^{\mathsf{new}} \cup \{(\widehat{U}_0^i, V_\alpha^j \oplus O_\alpha^i \oplus (2^\alpha \oplus 1) \cdot K_0 \oplus (2^{2\alpha} \oplus 1) \cdot K_1)\}$ and
   - for all $\beta \in [w]$ with $\beta \neq \alpha$, $\tau_\beta^{\mathsf{new}} \leftarrow \tau_\beta^{\mathsf{new}} \cup \{(\widehat{U}_\beta^i, V_\alpha^j \oplus O_\alpha^i \oplus O_\beta^i \oplus (2^\alpha \oplus 2^\beta) \cdot K_0 \oplus (2^{2\alpha} \oplus 2^{2\beta}) \cdot K_1)\}$.

Note that the addition of elements in Steps (1) and (2) is sound. For Step (1),

- since $\widehat{U}_0^i$ collides with U_0^j, $\widehat{U}_\alpha^i$ cannot collide with any U_α^k for $\alpha \in [w]$ due to $\overline{\mathsf{bad}_1}$.
- Similarly, $(V_0^j \oplus O_\alpha^i \oplus (2^\alpha \oplus 1) \cdot K_0 \oplus (2^{2\alpha} \oplus 1) \cdot K_1)$ cannot collide with any V_β^k for $\beta \in [0..w]$ due to $\overline{\mathsf{bad}_2}$.
- Moroever, $(V_0^j \oplus O_\alpha^i \oplus (2^\alpha \oplus 1) \cdot K_0 \oplus (2^{2\alpha} \oplus 1) \cdot K_1)$ is distinct due to $\overline{\mathsf{bad}_5}$ and $\overline{\mathsf{bad}_7}$.

For Step (2),

- since $\widehat{U}_\alpha^i$ collides with U_α^j for $\alpha \in [w]$, neither $\widehat{U}_0^i$ can collide with any U_0^k nor $\widehat{U}_\beta^i$ can collide with any U_β^k for $\beta \in [w]$ with $\beta \neq \alpha$ due to $\overline{\mathsf{bad}_1}$.

- Similarly, $(V_\alpha^j \oplus O_\alpha^i \oplus (2^\alpha \oplus 1) \cdot K_0 \oplus (2^{2\alpha} \oplus 1) \cdot K_1)$ cannot collide with any V_β^k for any $\beta \in [0..w]$ due to $\overline{\mathsf{bad}_3}$ and
- $(V_\alpha^j \oplus O_\alpha^i \oplus O_\beta^i \oplus (2^\alpha \oplus 2^\beta) \cdot K_0 \oplus (2^{2\alpha} \oplus 2^{2\beta}) \cdot K_1)$ cannot collide with V_γ^k for any $\gamma \in [0..w]$ due to $\overline{\mathsf{bad}_4}$.
- Moroever, $(V_\alpha^j \oplus O_\alpha^i \oplus (2^\alpha \oplus 1) \cdot K_0 \oplus (2^{2\alpha} \oplus 1) \cdot K_1)$ is distinct due to $\overline{\mathsf{bad}_6}$ and $\overline{\mathsf{bad}_8}$.

Further note that such an addition of elements (x, y) in the transcript $\tau_\alpha^{\mathsf{new}}$ for $\alpha \in [0..w]$ also updates the set $\mathsf{Dom}_\alpha \leftarrow \mathsf{Dom}_\alpha \cup \{x\}$ and $\mathsf{Ran}_\alpha \leftarrow \mathsf{Ran}_\alpha \cup \{y\}$. Now, given q_c constructions queries and $q_p' = (w+1)q_p$ primitive queries to the permutation P in the original transcript, let the numbers of queries moved from τ_c be r which includes total s_α elements into the primitive partial transcripts τ_α for $\alpha \in [0..w]$. Thus, the number of queries in the new construction transcript is denoted by $q' = q_c - r$. Moreover, we define $q_\alpha'' = q_p + s_\alpha$, for all $0 \le \alpha \le w$ and for each $\alpha \in [0..w]$, $s_\alpha \le q_c$. The $w+1$ sets of transcripts, $(\tau_0^{\mathsf{new}}, \tau_1^{\mathsf{new}}, \ldots, \tau_w^{\mathsf{new}})$ define exactly $(q_0'', q_1'', \ldots, q_w'')$ input-output tuples for P. What remains is the counting of the number of permutations P that satisfy these $q_0'' + \ldots + q_w''$ tuples, and that could give the remaining transcript τ_c^{new}, i.e., we are interested to count the number of permutation P that satisfies the following system of equations:

$$\mathbb{E}_i = \begin{cases} P(\widehat{U}_0^i) \oplus P(\widehat{U}_1^i) = O_1^i \oplus 2 \cdot K_0 \oplus 2^2 \cdot K_1 \oplus K_0 \oplus K_1 \\ P(\widehat{U}_0^i) \oplus P(\widehat{U}_2^i) = O_2^i \oplus 2^2 \cdot K_0 \oplus 2^4 \cdot K_1 \oplus K_0 \oplus K_1 \\ \vdots \quad \vdots \quad \vdots \quad \vdots \\ P(\widehat{U}_0^i) \oplus P(\widehat{U}_w^i) = O_w^i \oplus 2^w \cdot K_0 \oplus 2^{2w} \cdot K_1 \oplus K_0 \oplus K_1, \end{cases}$$

where $i \in [q']$, $U_\alpha^i = I^i \oplus \mathsf{msb}_{n-d}(2^\alpha \cdot K_0 \oplus 2^{2\alpha} \cdot K_1) \,\|\, \langle \alpha \rangle_d$ for all $\{(I^i, \mathbf{O}^i)\} \in \tau_c^{\mathsf{new}}$, along with the fact that for each $\alpha \in [0..w], P$ maps $\mathcal{D}_\alpha$ to $\mathcal{R}_\alpha$, where $\mathcal{D}_\alpha = \{0,1\}^n \setminus \mathsf{Dom}_\alpha$ and $\mathcal{R}_\alpha = \{0,1\}^n \setminus \mathsf{Ran}_\alpha$. Since τ is a good transcript, it follows that the constants in the right hand side of each equation of $\mathbb{E}_i$, i.e., $O_\alpha^i \oplus 2^\alpha \cdot K_0 \oplus 2^{2\alpha} \cdot K_1 \oplus K_0 \oplus K_1$, is non-zero, for $\alpha \in [w]$ (due to $\overline{\mathsf{bad}_9}$). Similarly, due to $\overline{\mathsf{bad}_{10}}$, we have all the constants in the right hand side of equations $\mathbb{E}_i$ distinct from each other. Note that,

$$\mathsf{Dom}_\alpha \stackrel{\text{def}}{=} \{U_\alpha^i : (U_\alpha^i, V_\alpha^i) \in \tau_\alpha^{\mathsf{new}}\}$$
$$\mathsf{Ran}_\alpha \stackrel{\text{def}}{=} \{V_\alpha^i : (U_\alpha^i, V_\alpha^i) \in \tau_\alpha^{\mathsf{new}}\}.$$

It is easy to see that $|\mathcal{D}_\alpha| = |\mathcal{R}_\alpha| = (2^n - q_p - s_\alpha)$. Note that, for each $\alpha \in [0..w]$, $V_\alpha^{\mathsf{out}} = \{0,1\}^n \setminus \mathcal{R}_\alpha$ is the set of range values of P that are prohibited (basically these are the V values in τ_α). Now, for $j = [0..q'-1]$, let

$$\lambda_{j+1} \stackrel{\text{def}}{=} \left|\left\{(\mathsf{P}_0^1, \ldots, \mathsf{P}_0^{j+1}, \ldots, \mathsf{P}_w^1, \ldots, \mathsf{P}_w^{j+1})\right\}\right| \tag{6.26}$$

be the number of solutions that satisfy

(1) the system of equations $\mathbb{E}_1 \cup \mathbb{E}_2 \cup \ldots \cup \mathbb{E}_{j+1}$.

(2) For all $\alpha \in [0..w]$, it holds that $\mathsf{P}_\alpha^{j+1} \notin \{\mathsf{P}_\alpha^1, \ldots, \mathsf{P}_\alpha^j\} \cup \mathsf{Ran}_0 \cup \mathsf{Ran}_1 \cup \ldots \cup \mathsf{Ran}_w$.

Then, the goal is to define a recursive expression for λ_{j+1} from λ_j such that a lower bound can be found for the expression λ_{j+1}/λ_j. It holds that

$$|\mathsf{Comp}_{\text{real}}(\tau)| = \lambda_{q'} \cdot \left(2^n - \left(\sum_{\alpha=0}^{w} q''_\alpha + (w+1)q'\right)\right)!$$

We obtain

$$\frac{\Pr[\Theta_{\text{real}} = \tau]}{\Pr[\Theta_{\text{ideal}} = \tau]} = \frac{\lambda_{q'} \cdot (2^n - (\sum_{\alpha=0}^{w} q''_\alpha + (w+1)q'))!}{(2^n - (w+1)q_p)!} \cdot (2^{wn})^{q_c}\,. \tag{6.27}$$

Let $\mathcal{B}_{(1)}$ denote the set of solutions that comply with only Condition (1) without considering Conditions (2.0) through $(2.w)$. Moreover, let $\mathcal{B}_{(2.\iota:i)}$ denote the set of solutions compatible with Condition (1), but not with $(2.\iota : i)$, for $i = 1, \ldots, j + \sum_{\alpha=0}^{w} |\mathsf{Ran}_\alpha|$. From the inclusion-exclusion principle, it follows that λ_{j+1} can be written as

$$\begin{aligned}
&\left|\mathcal{B}_{(1)}\right| - \left|\left(\bigcup_{i=1}^{j+|\mathsf{Ran}_0|+\ldots+|\mathsf{Ran}_w|} \mathcal{B}_{(2.0:i)}\right) \cup \cdots \cup \left(\bigcup_{i=1}^{j+|\mathsf{Ran}_0|+\ldots+|\mathsf{Ran}_w|} \mathcal{B}_{(2.w:i)}\right)\right| \\
&\geq \left|\mathcal{B}_{(1)}\right| - \left|\sum_{i=1}^{j+|\mathsf{Ran}_0|+\ldots+|\mathsf{Ran}_w|} |\mathcal{B}_{(2.0:i)}|\right| - \cdots - \left|\sum_{i=1}^{j+|\mathsf{Ran}_w|+\ldots+|\mathsf{Ran}_w|} |\mathcal{B}_{(2.w:i)}|\right| \\
&\geq 2^n \cdot \lambda_j - \sum_{i=1}^{j+|\mathsf{Ran}_0|+\ldots+|\mathsf{Ran}_w|} \lambda_j - \cdots - \sum_{i=1}^{j+|\mathsf{Ran}_0|+\ldots+|\mathsf{Ran}_w|} \lambda_j\,.
\end{aligned}$$

So, it follows that

$$\lambda_{j+1} \geq 2^n \cdot \lambda_j - \left(j + \sum_{\alpha=0}^{w} q''_\alpha\right) \cdot \lambda_j - \ldots - \left(j + \sum_{\alpha=0}^{w} q''_\alpha\right) \cdot \lambda_j$$

where recall that $q''_\alpha = q_p + s_\alpha$ for $\alpha \in [0..w]$. Therefore,

$$\frac{\lambda_{j+1}}{\lambda_j} \geq 2^n - (w+1)j - (w+1)\left(\sum_{\alpha=0}^{w} q''_\alpha\right).$$

with $\lambda_0 = 1$. Let $s = s_0 + \ldots + s_w$. It follows from Equation (6.27) that

$$\begin{aligned}(6.27) &= \prod_{t=0}^{s-1} \frac{2^n}{2^n - (w+1)q_p - t} \cdot \prod_{j=0}^{q'-1} \frac{\lambda_{j+1}}{\lambda_j} \cdot \frac{(2^n)^w}{\prod_{\alpha=0}^{w}(2^n - \sum_{k=0}^{w} q''_k - jq' - j)} \\ &\geq \prod_{j=0}^{q'-1} \left(\frac{(2^n - (w+1)j - (w+1)\sum_{\alpha=0}^{w} q''_\alpha)}{\prod_{\alpha=0}^{w}(2^n - \sum_{k=0}^{w} q''_k - jq' - j)}\right) 2^{nw}. \qquad (6.28)\end{aligned}$$

We use $q_{\mathsf{sum}} \stackrel{\text{def}}{=} \sum_{k=0}^{w} q''_k$ and define $p = (q' + q_{\mathsf{sum}})/2^n$. Note that, $0 \leq p \leq 1$ and therefore by applying Lemma 6.3 on Equation (6.28), we have

$$\begin{aligned}(6.28) &\geq \prod_{j=0}^{q'-1} \left(\frac{(2^n)^{w+1} - (w+1)\cdot p \cdot 2^{n(w+1)}}{(2^n - p\cdot 2^n))^{w+1}}\right) && (6.29)\\ &\geq \prod_{j=0}^{q'-1} \left(\frac{1-(w+1)p}{1-(w+1)p + \binom{w+1}{2}p^2}\right) && (6.30)\\ &= \prod_{j=0}^{q'-1} \left(1 - \frac{\binom{w+1}{2}p^2}{1-(w+1)p + \binom{w+1}{2}p^2}\right) \\ &= \prod_{j=0}^{q'-1} \left(1 - \frac{\binom{w+1}{2}(q'+q_{\mathsf{sum}})^2}{2^{2n} - (w+1)2^n(q'+q_{\mathsf{sum}}) + \binom{w+1}{2}(q'+q_{\mathsf{sum}})^2}\right) \\ &\geq \prod_{j=0}^{q'-1} \left(1 - \frac{2\binom{w+1}{2}(q'+q_{\mathsf{sum}})^2}{2^{2n}}\right) && (6.31)\\ &\geq \left(1 - \frac{(w+1)^2 q'(q'+q_{\mathsf{sum}})^2}{2^{2n}}\right) && (6.32)\\ &\stackrel{(1)}{\geq} \left(1 - \frac{v^2(q_c^3 + 2q_c^2 v(q_c+q_p) + q_c v^2(q_c+q_p)^2)}{2^{2n}}\right) && (6.33)\\ &\geq \left(1 - \frac{4v^4 q_c^3 + 4v^4 q_c^2 q_p + v^4 q_c q_p^2}{2^{2n}}\right). && (6.34)\end{aligned}$$

Note that, $(w+1)2^n(q'+q_{\mathsf{sum}}) - \binom{w+1}{2}(q'+q_{\mathsf{sum}})^2 \leq 2^{2n}/2$ as $q'+q_{\mathsf{sum}} = q' + \sum_{\alpha=0}^{w} q''_\alpha = q' + (w+1)q_p + \sum_{\alpha=0}^{w} s_\alpha$ and for each $\alpha \in [0..w]$, $s_\alpha \leq q_c$ and $q' \leq q_c$, it follows that $q_{\mathsf{sum}} \leq (w+1)(q_p+q_c)$, and thereby $q' + q_{\mathsf{sum}} \leq q_c + (w+1)(q_p+q_c) \leq 2^n/2(w+1)$. (1) follows due to $v \stackrel{\text{def}}{=} w+1$ and $q' \leq q_c$ and $q_{\mathsf{sum}} \leq v(q_p+q_c)$. which concludes our proof. □

Our claim in Theorem 6.11 follows from Lemmas 2.1, 6.13, and 6.14. □

6.8.2 Security Result of DS-SoEM

We consider $\mathsf{DS\text{-}SoEM}[P]_{\mathbf{K}}$ with $d = 1$, $P \xleftarrow{\$} \mathsf{Perm}(\mathbb{F}_2^n)$, $K_0, K_1 \xleftarrow{\$} \mathbb{F}_2^n$, and $\mathbf{K} = (K_0, K_1)$. For the simplicity of the notation, we write $\mathsf{DS\text{-}SoEM}[P]_{\mathbf{K}}$ as DS-SoEM. Note that the security result of DS-SoEM can be trivially deduced from the security result of DS-XORPP by putting the value of $w = 1, d = 1$ and letting $K_0' = K_0 \oplus K_1$ and $K_1' = 2K_0 \oplus 2^2 K_1$ in Theorem 6.11, where K_0' and K_1' are the keys of DS-SoEM. It remains to argue that K_0' and K_1' are statistically independent random variables. This is easy to see as the two equations $K_0 \oplus K_1$ and $2K_0 \oplus 2^2 K_1$ are linearly independent, where K_0, K_1 are uniformly sampled two independent n-bit keys, i.e, for any pair of n bit string k_0', k_1',

$$\Pr[K_0' = k_0', K_1' = k_1'] = 2^{-2n}.$$

This holds true as

$$\Pr[K_0' = k_0', K_1' = k_1'] = \Pr[K_0 \oplus K_1 = k_0', 2K_0 \oplus 2^2 K_1 = k_1']$$

which can be equivalently written as

$$\Pr\left[\underbrace{\begin{bmatrix} 1 & 1 \\ 2 & 2^2 \end{bmatrix}}_{\mathbf{A}'} \cdot \begin{bmatrix} K_0 \\ K_1 \end{bmatrix} = \begin{bmatrix} k_0' \\ k_1' \end{bmatrix}\right]. \tag{6.35}$$

Since the matrix $\mathbf{A}'$ is non-singular, using Lemma 6.2 we can deduce the probability in Equation (6.35) is exactly 2^{-2n}. Moreover, it is also easy to see that

$$\Pr[K_0' = k_0'] = \Pr[K_1' = k_1'] = 2^{-n},$$

which establishes the independence the round keys used in DS-SoEM. Thus, the security result of DS-SoEM is stated as follows:

Corollary 6.15. Let $\mathbf{D}$ be a distinguisher with exactly q_c construction queries and q_p primitive queries. Let $2q_p + 3q_c \leq 2^{n-2}$. Then

$$\mathbf{Adv}^{\mathsf{DS\text{-}SoEM}}_{\mathsf{PRF}}(\mathbf{D}) \leq \frac{56q_cq_p^2}{2^{2n}} + \frac{100q_c^2q_p}{2^{2n}} + \frac{64q_c^3}{2^{2n}} + \frac{q_c}{2^n} + \frac{8q_c^2q_p^2}{2^{3n}}\,.$$

6.9 Conclusion

This chapter has proposed a variant of CENC from public permutations, CENCPP*. From that base, it is straightforward to obtain a nonce-based encryption scheme or a fixed-input-length variable-output-length PRF with a security bound of up to $O(2^{2n/3}/w^2)$ queries. Our result can be combined with a beyond-birthday-secure MAC from public permutations to obtain an authenticated encryption scheme. The doubling-based key schedule ensures pairwise independent keys for all pairs of permutation inputs in XORPP* and DS-XORPP. Although the key masks can be cached, for values of $w \leq 2$, the choice of keys can be improved in terms of computations. For $w = 1$, XORPP* degenerates to the SoEM construction and can simply use (K_0, K_1) for the permutation calls. For $w = 2$, XORPP* can use $(K_0, K_0 \oplus K_1, K_1)$ for the calls to the permutations to ensure independent keys without the need for doubling. A natural follow-up to this work is to implement CENC and CENCPP to compare their speed. Also, we see the recent summation-truncation-hybrid by Gunsing and Mennink [191] to be similar to the sum of permutation, although it is based on secret permutations. Adapting it to beyond-birthday-bound security with public permutations seems an interesting future work.

References

[1] Charanjit S. Jutla. Encryption modes with almost free message integrity. In Birgit Pfitzmann, editor, *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding*, volume 2045 of *Lecture Notes in Computer Science*, pages 529–544. Springer, 2001. doi: 10.1007/3-540-44987-6_32. URL https://doi.org/10.1007/3-540-44987-6_32.

[2] Charanjit S. Jutla. Encryption modes with almost free message integrity. *J. Cryptol.*, 21(4):547–578, 2008. doi: 10.1007/S00145-008-9024-Z. URL https://doi.org/10.1007/s00145-008-9024-z.

[3] Phillip Rogaway. Authenticated-encryption with associated-data. In Vijayalakshmi Atluri, editor, *Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS 2002, Washington, DC, USA, November 18-22, 2002*, pages 98–107. ACM, 2002. doi: 10.1145/586110.586125. URL https://doi.org/10.1145/586110.586125.

[4] CAESAR Committee. CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness. http://competitions.cr.yp.to/caesar.html/.

[5] NIST. Lightweight cryptography. https://csrc.nist.gov/Projects/Lightweight-Cryptography.

[6] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2: Lightweight authenticated encryption and hashing. *J. Cryptol.*, 34(3):33, 2021. doi: 10.1007/s00145-021-09398-9. URL https://doi.org/10.1007/s00145-021-09398-9.

[7] Arghya Bhattacharjee, Cuauhtemoc Mancillas López, Eik List, and Mridul Nandi. The oribatida v1.3 family of lightweight authenticated encryption schemes. *J. Math. Cryptol.*, 15(1):305–344, 2021. doi: 10.1515/jmc-2020-0018. URL https://doi.org/10.1515/jmc-2020-0018.

[8] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, and Thomas Unterluggauer. ISAP - towards side-channel secure authenticated encryption. *IACR Trans. Symmetric Cryptol.*, 2017(1):80–105, 2017. doi: 10.13154/tosc.v2017.i1.80-105. URL https://doi.org/10.13154/tosc.v2017.i1.80-105.

[9] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, Bart Mennink, Robert Primas, and Thomas Unterluggauer. Isap v2.0. *IACR Trans. Symmetric Cryptol.*, 2020(S1):390–416, 2020. doi: 10.13154/tosc.v2020.iS1.390-416. URL https://doi.org/10.13154/tosc.v2020.iS1.390-416.

[10] Arghya Bhattacharjee, Avik Chakraborti, Nilanjan Datta, Cuauhtemoc Mancillas-López, and Mridul Nandi. sf ISAP+: sf ISAP with fast authentication. In Takanori Isobe and Santanu Sarkar, editors, *Progress in Cryptology - INDOCRYPT 2022 - 23rd International Conference on Cryptology in India, Kolkata, India, December 11-14, 2022, Proceedings*, volume 13774 of *Lecture Notes in Computer Science*, pages 195–219. Springer, 2022. doi: 10.1007/978-3-031-22912-1_9. URL https://doi.org/10.1007/978-3-031-22912-1_9.

[11] Arghya Bhattacharjee, Avik Chakraborti, Nilanjan Datta, Cuauhtemoc Mancillas-López, and Mridul Nandi. ISAP+: ISAP with fast authentication. *IACR Cryptol. ePrint Arch.*, page 1591, 2022. URL https://eprint.iacr.org/2022/1591.

[12] Ted Krovetz and Phillip Rogaway. The Software Performance of Authenticated-Encryption Modes. In Antoine Joux, editor, *Fast Software Encryption*, pages 306–327, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-21702-9.

[13] Ritam Bhaumik and Mridul Nandi. Improved security for OCB3. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017*, pages 638–666, Cham, 2017. Springer International Publishing. ISBN 978-3-319-70697-9.

[14] Arghya Bhattacharjee, Ritam Bhaumik, and Mridul Nandi. Offset-based bbb-secure tweakable block-ciphers with updatable caches. In Takanori Isobe and Santanu Sarkar, editors, *Progress in Cryptology - INDOCRYPT 2022 - 23rd International Conference on Cryptology in India, Kolkata, India, December 11-14, 2022, Proceedings*, volume 13774 of *Lecture Notes in Computer Science*, pages 171–194. Springer, 2022. doi: 10.1007/978-3-031-22912-1_8. URL https://doi.org/10.1007/978-3-031-22912-1_8.

[15] Arghya Bhattacharjee, Ritam Bhaumik, and Mridul Nandi. Offset-based bbb-secure tweakable block-ciphers with updatable caches. *IACR Cryptol. ePrint Arch.*, page 1776, 2022. URL https://eprint.iacr.org/2022/1776.

[16] Tetsu Iwata. New Blockcipher Modes of Operation with Beyond the Birthday Bound Security. In Matthew J. B. Robshaw, editor, *FSE*, volume 4047 of *LNCS*, pages 310–327. Springer, 2006. doi: 10.1007/11799313_20.

[17] Yu Long Chen, Eran Lambooij, and Bart Mennink. How to Build Pseudorandom Functions from Public Random Permutations. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO I*, volume 11692 of *LNCS*, pages 266–293. Springer, 2019. doi: 10.1007/978-3-030-26948-7_10.

[18] Arghya Bhattacharjee, Avijit Dutta, Eik List, and Mridul Nandi. Cencpp*: beyond-birthday-secure encryption from public permutations. *Des. Codes Cryptogr.*, 90(6):1381–1425, 2022. doi: 10.1007/s10623-022-01045-z. URL https://doi.org/10.1007/s10623-022-01045-z.

[19] Arghya Bhattacharjee, Avijit Dutta, Eik List, and Mridul Nandi. CENCPP - beyond-birthday-secure encryption from public permutations. *IACR Cryptol. ePrint Arch.*, page 602, 2020. URL https://eprint.iacr.org/2020/602.

[20] 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, *ASIACRYPT I*, volume 8873 of *LNCS*, pages 105–125. Springer, 2014. doi: 10.1007/978-3-662-45611-8_6.

[21] Jacques Patarin. The "coefficients h" technique. In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *Selected Areas in Cryptography*,

pages 328–345, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-04159-4.

[22] Shan Chen and John Steinberger. Tight security bounds for key-alternating ciphers. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, pages 327–350, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. ISBN 978-3-642-55220-5.

[23] Shan Chen and John Steinberger. Tight security bounds for key-alternating ciphers. Cryptology ePrint Archive, Report 2013/222, 2013. https://ia.cr/2013/222.

[24] Viet Tung Hoang and Stefano Tessaro. Key-alternating ciphers and key-length extension: Exact bounds and multi-user security. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016*, pages 3–32, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg. ISBN 978-3-662-53018-4.

[25] Morris Dworkin. FIPS 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. Technical report, 2015.

[26] Nicky Mouha, Bart Mennink, Anthony Van Herrewege, Dai Watanabe, Bart Preneel, and Ingrid Verbauwhede. Chaskey: An efficient MAC algorithm for 32-bit microcontrollers. In Antoine Joux and Amr M. Youssef, editors, *Selected Areas in Cryptography - SAC 2014 - 21st International Conference, Montreal, QC, Canada, August 14-15, 2014, Revised Selected Papers*, volume 8781 of *Lecture Notes in Computer Science*, pages 306–323. Springer, 2014. doi: 10.1007/978-3-319-13051-4_19. URL https://doi.org/10.1007/978-3-319-13051-4_19.

[27] Guido Bertoni, Joan Daemen, Seth Hoffert, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer. Farfalle: parallel permutation-based cryptography. *IACR Trans. Symmetric Cryptol.*, 2017(4):1–38, 2017. URL https://tosc.iacr.org/index.php/ToSC/article/view/801.

[28] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2 Submission to the CAESAR Competition. September 15 2016. Submission to the CAESAR competition. http://competitions.cr.yp.to/caesar-submissions.html.

[29] Elena Andreeva, Begül Bilgin, Andrey Bogdanov, Atul Luykx, Bart Mennink, Nicky Mouha, and Kan Yasuda. APE: Authenticated Permutation-Based Encryption for Lightweight Cryptography. In Carlos Cid and Christian Rechberger, editors, *FSE*, volume 8540 of *LNCS*, pages 168–186. Springer, 2014. doi: 10.1007/978-3-662-46706-0_9.

[30] Jean-Philippe Aumasson, Philipp Jovanovic, and Samuel Neves. NORX: Parallel and Scalable AEAD. In Miroslaw Kutylowski and Jaideep Vaidya, editors, *ESORICS II*, volume 8713 of *LNCS*, pages 19–36. Springer, 2014. doi: 10.1007/978-3-319-11212-1_2.

[31] Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles van Assche, and Ronny van Keer. Ketje v2. 2016. Submission to the CAESAR competition http://competitions.cr.yp.to/caesar-submissions.html.

[32] Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles van Assche, and Ronny van Keer. Keyak v2. 2016. Submission to the CAESAR competition http://competitions.cr.yp.to/caesar-submissions.html.

[33] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Sponge functions. In *ECRYPT hash workshop*, volume 2007. Citeseer, 2007.

[34] 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. doi: 10.1007/978-3-642-28496-0_19. URL https://doi.org/10.1007/978-3-642-28496-0_19.

[35] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. On the security of the keyed sponge construction. In *SHA-3 competition (round 3)*, volume 2011, 2011.

[36] Bart Mennink. Key Prediction Security of Keyed Sponges. *IACR Trans. Symmetric Cryptol.*, 2018(4):128–149, 2018. doi: 10.13154/tosc.v2018.i4.128-149.

[37] Christoph Dobraunig and Bart Mennink. Security of the Suffix Keyed Sponge. *IACR Trans. Symmetric Cryptol.*, 2019(4):223–248, 2019. doi: 10.13154/tosc.v2019.i4.223-248.

[38] Donghoon Chang, Morris Dworkin, Seokhie Hong, John Kelsey, and Mridul Nandi. A Keyed Sponge Construction with Pseudorandomness in the Standard Model. In *The Third SHA-3 Candidate Conference*, volume 3, page 7, March 2012.

[39] 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. doi: 10.1007/978-3-662-48116-5_18. URL https://doi.org/10.1007/978-3-662-48116-5_18.

[40] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Permutation-based encryption, authentication and authenticated encryption. *Directions in Authenticated Ciphers*, 2012.

[41] Joan Daemen, Bart Mennink, and Gilles Van Assche. Full-state keyed duplex with built-in multi-user support. 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 II*, volume 10625 of *Lecture Notes in Computer Science*, pages 606–637. Springer, 2017. doi: 10.1007/978-3-319-70697-9_21. URL https://doi.org/10.1007/978-3-319-70697-9_21.

[42] Peter Gaži, Krzysztof Pietrzak, and Stefano Tessaro. The Exact PRF Security of Truncation: Tight Bounds for Keyed Sponges and Truncated CBC. In Rosario Gennaro and Matthew Robshaw, editors, *CRYPTO I*, volume 9215 of *LNCS*, pages 368–387. Springer, 2015. doi: 10.1007/978-3-662-47989-6_18.

[43] Bart Mennink, Reza Reyhanitabar, and Damian Vizár. Security of full-state keyed sponge and duplex: Applications to authenticated encryption. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT II*, volume 9453 of *LNCS*, pages 465–489. Springer, 2015. doi: 10.1007/978-3-662-48800-3_19.

[44] Yusuke Naito and Kan Yasuda. New Bounds for Keyed Sponges with Extendable Output: Independence Between Capacity and Message Length. In Thomas Peyrin, editor, *FSE*, volume 9783 of *LNCS*, pages 3–22. Springer, 2016. doi: 10.1007/978-3-662-52993-5_1.

[45] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. On the indifferentiability of the sponge construction. In Nigel P. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, volume 4965 of *Lecture Notes in Computer Science*, pages 181–197. Springer, 2008. doi: 10.1007/978-3-540-78967-3_11. URL https://doi.org/10.1007/978-3-540-78967-3_11.

[46] Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology. In Moni Naor, editor, *TCC*, volume 2951 of *LNCS*, pages 21–39. Springer, 2004. doi: 10.1007/978-3-540-24638-1_2.

[47] Philipp Jovanovic, Atul Luykx, and Bart Mennink. Beyond $2^{c/2}$ Security in Sponge-Based Authenticated Encryption Modes. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT I*, volume 8873 of *LNCS*, pages 85–104. Springer, 2014. doi: 10.1007/978-3-662-45611-8_5.

[48] 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. doi: 10.13154/tches.v2018.i2.218-241. URL https://doi.org/10.13154/tches.v2018.i2.218-241.

[49] Raghvendra Rohit and Sumanta Sarkar. [lwc-forum] ROUND 2 OFFICIAL COMMENT: Oribatida. NIST lwc forum mailing list, 17 September 17:09 2019.

[50] Arghya Bhattacharjee, Eik List, Cuauhtemoc Mancillas López, and Mridul Nandi. The Oribatida Family of Lightweight Authenticated Encryption Schemes Version v1.2. Technical report, Sep 27 2019. Second-round submission to the NIST Lightweight Cryptography Competition. https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/oribatida-spec-round2.pdf.

[51] Riham AlTawy, Guang Gong, Morgan He, Ashwin Jha, Kalikinkar Mandal, Mridul Nandi, and Raghvendra Rohit. SpoC: An Authenticated Cipher. Technical report, Feb 24 2019. First-round submission to the NIST Lightweight Cryptography Competition. https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/round-1/spec-doc/spoc-spec.pdf.

[52] Avik Chakraborti, Nilanjan Datta, Mridul Nandi, and Kan Yasuda. Beetle Family of Lightweight and Secure Authenticated Encryption Ciphers. http://eprint.iacr.org/2018/805, 2018.

[53] Mihir Bellare, Alexandra Boldyreva, Lars R. Knudsen, and Chanathip Namprempre. Online Ciphers and the Hash-CBC Construction. In Joe Kilian, editor, *CRYPTO*, volume 2139 of *LNCS*, pages 292–309. Springer, 2001. doi: 10.1007/3-540-44647-8_18.

[54] Jacques Patarin. The "coefficients h" technique. In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *Selected Areas in Cryptography, 15th International Workshop, SAC 2008, Sackville, New Brunswick, Canada, August 14-15, Revised Selected Papers*, volume 5381 of *Lecture Notes in Computer Science*, pages 328–345. Springer, 2008. doi: 10.1007/978-3-642-04159-4_21. URL https://doi.org/10.1007/978-3-642-04159-4_21.

[55] Avik Chakraborti, Ashwin Jha, Cuauhtemoc Mancillas Lopez, Mridul Nandi, and Yu Sasaki. ESTATE. Technical report, Mar 29 2019. First-round submission to the NIST Lightweight Cryptography Competition. https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/round-1/spec-doc/spoc-spec.pdf.

[56] Han Sui, Wenling Wu, Lei Zhang, and Danxia Zhang. LAEM (Lightweight Authentication Encryption Mode). Technical report, Mar 25 2019. First-round submission to the NIST Lightweight Cryptography Competition. https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/round-1/spec-doc/LAEM-spec.pdf.

[57] Avik Chakraborti, Nilanjan Datta, Ashwin Jha, Cuauhtemoc Mancillas Lopez, Mridul Nandi, and Yu Sasaki. LOTUS-AEAD and

LOCUS-AEAD. Technical report, Feb 26 2019. First-round submission to the NIST Lightweight Cryptography Competition. https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/round-1/spec-doc/lotus-aead-and-locus-aead-spec.pdf.

[58] 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. doi: 10.1007/11761679_23. URL https://doi.org/10.1007/11761679_23.

[59] John Black and Phillip Rogaway. A Block-Cipher Mode of Operation for Parallelizable Message Authentication. In Lars R. Knudsen, editor, *EUROCRYPT*, volume 2332 of *LNCS*, pages 384–397. Springer, 2002.

[60] Kazuhiko Minematsu. Parallelizable rate-1 authenticated encryption from pseudorandom functions. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 275–292. Springer, 2014. doi: 10.1007/978-3-642-55220-5_16. URL https://doi.org/10.1007/978-3-642-55220-5_16.

[61] Phillip Rogaway, Mihir Bellare, John Black, and Ted Krovetz. OCB: a block-cipher mode of operation for efficient authenticated encryption. In Michael K. Reiter and Pierangela Samarati, editors, *ACM-CCS*, pages 196–205. ACM, 2001. doi: 10.1145/501983.502011.

[62] Shai Halevi. EME*: Extending EME to Handle Arbitrary-Length Messages with Associated Data. In Anne Canteaut and Kapalee Viswanathan, editors, *INDOCRYPT*, volume 3348 of *LNCS*, pages 315–327. Springer, 2004. doi: 10.1007/978-3-540-30556-9_25.

[63] Jean-Sébastien Coron, Yevgeniy Dodis, Avradip Mandal, and Yannick Seurin. A Domain Extender for the Ideal Cipher. In Daniele Micciancio, editor, *TCC*, volume 5978 of *LNCS*, pages 273–289. Springer, 2010. Full version at https://eprint.iacr.org/2009/356.

[64] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK Families of Lightweight Block Ciphers. *IACR Cryptology ePrint Archive*, 2013:404, 2013.

[65] Farzaneh Abed, Eik List, Stefan Lucks, and Jakob Wenzel. Differential Cryptanalysis of Round-Reduced Simon and Speck. In Carlos Cid and Christian Rechberger, editors, *FSE*, volume 8540 of *LNCS*, pages 525–545. Springer, 2014. doi: 10.1007/978-3-662-46706-0_27.

[66] Huaifeng Chen and Xiaoyun Wang. Improved Linear Hull Attack on Round-Reduced Simon with Dynamic Key-Guessing Techniques. In Thomas Peyrin, editor, *FSE*, volume 9783 of *LNCS*, pages 428–449. Springer, 2016. doi: 10.1007/978-3-662-52993-5_22.

[67] Zhengbin Liu, Yongqiang Li, and Mingsheng Wang. Optimal Differential Trails in SIMON-like Ciphers. *IACR Trans. Symmetric Cryptol.*, 2017(1): 358–379, 2017. doi: 10.13154/tosc.v2017.i1.358-379.

[68] Håvard Raddum. Algebraic Analysis of the Simon Block Cipher Family. In Kristin E. Lauter and Francisco Rodríguez-Henríquez, editors, *LATINCRYPT*, volume 9230 of *LNCS*, pages 157–169. Springer, 2015. doi: 10.1007/978-3-319-22174-8_9.

[69] Zejun Xiang, Wentao Zhang, Zhenzhen Bao, and Dongdai Lin. Applying MILP Method to Searching Integral Distinguishers Based on Division Property for 6 Lightweight Block Ciphers. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT I*, volume 10031 of *LNCS*, pages 648–678, 2016. doi: 10.1007/978-3-662-53887-6_24.

[70] ISO/IEC. Information technology – Automatic identification and data capture techniques – Part 21: Crypto Suite SIMON Security Services for Air Interface Communications. https://www.iso.org/standard/70388.html, Oct 2018.

[71] 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, *EUROCRYPT (1)*, volume 9056 of *LNCS*, pages 15–44. Springer, 2015. Full version at https://eprint.iacr.org/2014/793.

[72] Alex Biryukov, Arnab Roy, and Vesselin Velichkov. Differential Analysis of Block Ciphers SIMON and SPECK. In Carlos Cid and Christian Rechberger,

editors, *FSE*, volume 8540 of *LNCS*, pages 546–570. Springer, 2014. doi: 10.1007/978-3-662-46706-0_28.

[73] Itai Dinur, Orr Dunkelman, Masha Gutman, and Adi Shamir. Improved Top-Down Techniques in Differential Cryptanalysis. In Kristin E. Lauter and Francisco Rodríguez-Henríquez, editors, *LATINCRYPT*, volume 9230 of *LNCS*, pages 139–156. Springer, 2015. doi: 10.1007/978-3-319-22174-8_8.

[74] Stefan Kölbl, Gregor Leander, and Tyge Tiessen. Observations on the SIMON Block Cipher Family. In Rosario Gennaro and Matthew Robshaw, editors, *CRYPTO*, volume 9215 of *LNCS*, pages 161–185. Springer, 2015.

[75] Mitsuru Matsui. On Correlation Between the Order of S-boxes and the Strength of DES. In Alfredo De Santis, editor, *EUROCRYPT*, volume 950 of *LNCS*, pages 366–375. Springer, 1994. doi: 10.1007/BFb0053451.

[76] Javad Alizadeh, Nasour Bagheri, Praveen Gauravaram, Abhishek Kumar, and Somitra Kumar Sanadhya. Linear Cryptanalysis of Round Reduced SIMON. *IACR Cryptology ePrint Archive*, 2013:663, 2013.

[77] Mohamed Ahmed Abdelraheem, Javad Alizadeh, Hoda AlKhzaimi, Mohammad Reza Aref, Nasour Bagheri, Praveen Gauravaram, and Martin M. Lauridsen. Improved Linear Cryptanalysis of Round Reduced SIMON. *IACR Cryptology ePrint Archive*, 2014:681, 2014.

[78] Zhengbin Liu, Yongqiang Li, and Mingsheng Wang. The Security of SIMON-like Ciphers Against Linear Cryptanalysis. *IACR Cryptology ePrint Archive*, 2017:576, 2017.

[79] Mohamed Ahmed Abdelraheem, Javad Alizadeh, Hoda A. AlKhzaimi, Mohammad Reza Aref, Nasour Bagheri, and Praveen Gauravaram. Improved Linear Cryptanalysis of Reduced-Round SIMON-32 and SIMON-48. In Alex Biryukov and Vipul Goyal, editors, *INDOCRYPT*, volume 9462 of *LNCS*, pages 153–179. Springer, 2015. doi: 10.1007/978-3-319-26617-6_9.

[80] Huiling Zhang, Wenling Wu, and Yanfeng Wang. Integral Attack Against Bit-Oriented Block Ciphers. In Soonhak Kwon and Aaram Yun, editors, *ICISC*, volume 9558 of *LNCS*, pages 102–118. Springer, 2015. doi: 10.1007/978-3-319-30840-1_7.

[81] Yosuke Todo and Masakatu Morii. Bit-Based Division Property and Application to Simon Family. In Thomas Peyrin, editor, *FSE*, volume 9783 of *LNCS*, pages 357–377. Springer, 2016.

[82] Kota Kondo, Yu Sasaki, Yosuke Todo, and Tetsu Iwata. On the Design Rationale of SIMON Block Cipher: Integral Attacks and Impossible Differential Attacks against SIMON Variants. *IEICE Transactions*, 101-A(1): 88–98, 2018. doi: 10.1587/transfun.E101.A.88.

[83] Raghvendra Rohit and Guang Gong. Correlated Sequence Attack on Reduced-Round Simon-32/64 and Simeck-32/64. *IACR Cryptology ePrint Archive*, 2018:699, 2018.

[84] Xuzi Wang, Baofeng Wu, Lin Hou, and Dongdai Lin. Automatic Search for Related-Key Differential Trails in SIMON-like Block Ciphers Based on MILP. In Liqun Chen, Mark Manulis, and Steve Schneider, editors, *ISC*, volume 11060 of *LNCS*, pages 116–131. Springer, 2018. doi: 10.1007/978-3-319-99136-8_7.

[85] Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In Neal Koblitz, editor, *CRYPTO 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996. doi: 10.1007/3-540-68697-5_9. URL https://doi.org/10.1007/3-540-68697-5_9.

[86] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *CRYPTO 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999. doi: 10.1007/3-540-48405-1_25. URL https://doi.org/10.1007/3-540-48405-1_25.

[87] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults (extended abstract). In Walter Fumy, editor, *EUROCRYPT 1997*, volume 1233 of *Lecture Notes in Computer Science*, pages 37–51. Springer, 1997. doi: 10.1007/3-540-69053-0_4. URL https://doi.org/10.1007/3-540-69053-0_4.

[88] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In Burton S. Kaliski Jr., editor, *CRYPTO '97, Proceedings*, volume

1294 of *Lecture Notes in Computer Science*, pages 513–525. Springer, 1997. doi: 10.1007/BFb0052259. URL https://doi.org/10.1007/BFb0052259.

[89] Louis Goubin and Jacques Patarin. DES and differential power analysis (the "duplication" method). In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems, CHES 1999, Proceedings*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172. Springer, 1999. doi: 10.1007/3-540-48059-5_15. URL https://doi.org/10.1007/3-540-48059-5_15.

[90] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Michael J. Wiener, editor, *CRYPTO 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 398–412. Springer, 1999. doi: 10.1007/3-540-48405-1_26. URL https://doi.org/10.1007/3-540-48405-1_26.

[91] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002. ISBN 3-540-42580-2. doi: 10.1007/978-3-662-04722-4. URL https://doi.org/10.1007/978-3-662-04722-4.

[92] Daniel J. Bernstein. ChaCha, a variant of Salsa20. 2008.

[93] Joan Daemen, Michaël Peeters, Gilles Van Assche, and Vincent Rijmen. The NOEKEON block cipher, 2000. Nessie Proposal. 2020. https://competitions.cr.yp.to/round3/acornv3.pdf.

[94] Gilles Piret, Thomas Roche, and Claude Carlet. PICARO - A block cipher allowing efficient higher-order side-channel resistance. In Feng Bao, Pierangela Samarati, and Jianying Zhou, editors, *ACNS 2012. Proceedings*, volume 7341 of *Lecture Notes in Computer Science*, pages 311–328. Springer, 2012. doi: 10.1007/978-3-642-31284-7_19. URL https://doi.org/10.1007/978-3-642-31284-7_19.

[95] Benoît Gérard, Vincent Grosso, María Naya-Plasencia, and François-Xavier Standaert. Block ciphers that are easier to mask: How far can we go? In Guido Bertoni and Jean-Sébastien Coron, editors, *Cryptographic Hardware and Embedded Systems - CHES 2013. Proceedings*, volume 8086 of *Lecture Notes in Computer Science*, pages 383–399. Springer, 2013.

doi: 10.1007/978-3-642-40349-1_22. URL https://doi.org/10.1007/978-3-642-40349-1_22.

[96] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2. Submission to CAESAR. 2016. https://competitions.cr.yp.to/round3/asconv12.pdf.

[97] Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles Van Assche. The Keccak reference (version 3.0). 2011. https://keccak.team/files/Keccak-reference-3.0.pdf.

[98] National Institute of Standards and Technology. FIPS PUB 202: SHA-3Standard: Permutation-based hash and extendable-output functions. . Federal Information Processing Standards Publication 202, U.S. Department ofCommerce, 8 2015.

[99] Christoph Dobraunig and Maria Eichlseder and Florian Mendel and Martin Schläffer. Ascon v1.2. Submission to NIST Lightweight Cryptography, 2019. 2019. https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/finalist-round/updated-spec-doc/ascon-spec-final.pdf.

[100] Elena Andreeva, Begül Bilgin, Andrey Bogdanov, Atul Luykx, Florian Mendel, Bart Mennink, Nicky Mouha, Qingju Wang, and Kan Yasuda. PRIMATEs v1.02. Submission to CAESAR. 2016. https://competitions.cr.yp.to/round2/primatesv102.pdf.

[101] Vincent Grosso, Gaëtan Leurent, Francois-Xavier Standaert, Kerem Varici, Anthony Journault, Francois Durvaux, Lubos Gaspar, and Stéphanie Kerckhof. SCREAM Side-Channel Resistant Authenticated Encryption with Masking. Submission to CAESAR. 2015. https://competitions.cr.yp.to/round2/screamv3.pdf.

[102] Guido Bertoni, Michaël Peeters Joan Daemen, Gilles Van Assche, and Ronny Van Keer. Ketje v2. Submission to CAESAR. 2016. https://competitions.cr.yp.to/round3/ketjev2.pdf.

[103] Marcel Medwed, François-Xavier Standaert, Johann Großschädl, and Francesco Regazzoni. Fresh re-keying: Security against side-channel and fault attacks for low-cost devices. In Daniel J. Bernstein and

Tanja Lange, editors, *AFRICACRYPT 2010, Proceedings*, volume 6055 of *Lecture Notes in Computer Science*, pages 279–296. Springer, 2010. doi: 10.1007/978-3-642-12678-9_17. URL https://doi.org/10.1007/978-3-642-12678-9_17.

[104] Marcel Medwed, Christophe Petit, Francesco Regazzoni, Mathieu Renauld, and François-Xavier Standaert. Fresh re-keying II: securing multiple parties against side-channel and fault attacks. In Emmanuel Prouff, editor, *CARDIS 2011*, volume 7079 of *Lecture Notes in Computer Science*, pages 115–132. Springer, 2011. doi: 10.1007/978-3-642-27257-8_8. URL https://doi.org/10.1007/978-3-642-27257-8_8.

[105] Sonia Belaïd, Fabrizio De Santis, Johann Heyszl, Stefan Mangard, Marcel Medwed, Jörn-Marc Schmidt, François-Xavier Standaert, and Stefan Tillich. Towards fresh re-keying with leakage-resilient prfs: cipher design principles and analysis. *J. Cryptogr. Eng.*, 4(3):157–171, 2014. doi: 10.1007/s13389-014-0079-5. URL https://doi.org/10.1007/s13389-014-0079-5.

[106] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Sponge functions, 2007. Ecrypt Hash Workshop 2007.

[107] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. 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 531–545. Springer, 2000. doi: 10.1007/3-540-44448-3_41. URL https://doi.org/10.1007/3-540-44448-3_41.

[108] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. *J. Cryptol.*, 21(4):469–491, 2008. doi: 10.1007/S00145-008-9026-X. URL https://doi.org/10.1007/s00145-008-9026-x.

[109] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986. doi: 10.1145/6490.6503. URL https://doi.org/10.1145/6490.6503.

[110] Christoph Dobraunig and Maria Eichlseder and Stefan Mangard and Florian Mendel and Bart Mennink and Robert Primas and Thomas Unterluggauer. ISAP v2.0. Submission to NIST. 2019. https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/round-1/spec-doc/ISAP-spec.pdf.

[111] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, Bart Mennink, Robert Primas, and Thomas Unterluggauer. Isap v2.0. https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/finalist-round/updated-spec-doc/isap-spec-final.pdf.

[112] https://keccak.team/hardware.html.

[113] Zhenzhen Bao, Seongha Hwang, Akiko Inoue, ByeongHak Lee, Jooyoung Lee, and Kazuhiko Minematsu. XOCB: beyond-birthday-bound secure authenticated encryption mode with rate-one computation. 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 532–561. Springer, 2023. doi: 10.1007/978-3-031-30634-1_18. URL https://doi.org/10.1007/978-3-031-30634-1_18.

[114] Jonathan Katz and Moti Yung. Unforgeable encryption and chosen ciphertext secure modes of operation. In Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, and Bruce Schneier, editors, *Fast Software Encryption*, pages 284–299, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. ISBN 978-3-540-44706-1.

[115] Moses Liskov, Ronald L. Rivest, and David Wagner. Tweakable Block Ciphers. J Cryptol 24, 588–613 (2011). https://doi.org/10.1007/s00145-010-9073-y.

[116] Moses Liskov, Ronald L. Rivest, and David Wagner. Tweakable Block Ciphers. In Moti Yung, editor, *Advances in Cryptology — CRYPTO 2002*, pages 31–46, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. ISBN 978-3-540-45708-4.

[117] Richard Schroeppel. The Hasty Pudding Cipher. AES submission to NIST, 1998.

[118] Niels Ferguson, Stefan Lucks, Bruce Schneier, Doug Whiting, Mihir Bellare, Tadayoshi Kohno, Jon Callas, and Jesse Walker. The Skein Hash Function Family. SHA3 submission to NIST (Round 3), 2010.

[119] Jérémy Jean, Ivica Nikolic, Thomas Peyrin, and Yannick Seurin. The Deoxys AEAD family. *J. Cryptol.*, 34(3):31, 2021. doi: 10.1007/s00145-021-09397-w. URL https://doi.org/10.1007/s00145-021-09397-w.

[120] Jérémy Jean, Ivica Nikolic, and Thomas Peyrin. Joltik v1.3. CAESAR Round, 2, 2015.

[121] Jérémy Jean, Ivica Nikolić, and Thomas Peyrin. Tweaks and keys for block ciphers: The tweakey framework. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014*, pages 274–288, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. ISBN 978-3-662-45608-8.

[122] Vincent Grosso, Gaëtan Leurent, François-Xavier Standaert, Kerem Varici, Anthony Journault, François Durvaux, Lubos Gaspar, and Stéphanie Kerckhof. SCREAM v3. Submission to CAESAR competition, 2015.

[123] CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness. https://competitions.cr.yp.to/caesar-submissions.html.

[124] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The skinny family of block ciphers and its low-latency variant mantis. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016*, pages 123–153, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg. ISBN 978-3-662-53008-5.

[125] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. SKINNY-AEAD and SKINNY-Hash. *IACR Transactions on Symmetric Cryptology*, 2020(S1):88–131, Jun. 2020. doi: 10.13154/tosc.v2020.iS1.88-131. URL https://tosc.iacr.org/index.php/ToSC/article/view/8619.

[126] Roberto Avanzi. The QARMA block cipher family. almost mds matrices over rings with zero divisors, nearly symmetric even-mansour constructions

with non-involutory central rounds, and search heuristics for low-latency s-boxes. *IACR Transactions on Symmetric Cryptology*, 2017(1):4–44, Mar. 2017. doi: 10.13154/tosc.v2017.i1.4-44. URL https://tosc.iacr.org/index.php/ToSC/article/view/583.

[127] Christof Beierle, Gregor Leander, Amir Moradi, and Shahram Rasoolzadeh. Craft: Lightweight tweakable block cipher with efficient protection against dfa attacks. *IACR Transactions on Symmetric Cryptology*, 2019(1):5–45, Mar. 2019. doi: 10.13154/tosc.v2019.i1.5-45. URL https://tosc.iacr.org/index.php/ToSC/article/view/7396.

[128] 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*, pages 263–293, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg. ISBN 978-3-662-49890-3.

[129] Yusuke Naito. Tweakable blockciphers for efficient authenticated encryptions with beyond the birthday-bound security. *IACR Transactions on Symmetric Cryptology*, 2017(2):1–26, Jun. 2017. doi: 10.13154/tosc.v2017.i2.1-26. URL https://tosc.iacr.org/index.php/ToSC/article/view/636.

[130] Phillip Rogaway, Mihir Bellare, John Black, and Ted Krovetz. OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption. In *Proceedings of the 8th ACM Conference on Computer and Communications Security*, CCS '01, page 196–205, New York, NY, USA, 2001. Association for Computing Machinery. ISBN 1581133855. doi: 10.1145/501983.502011. URL https://doi.org/10.1145/501983.502011.

[131] Information technology – Security techniques – Authenticated encryption. ISO/IEC 19772:2009, 2009.

[132] Phillip Rogaway. Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC. In Pil Joong Lee, editor, *Advances in Cryptology - ASIACRYPT 2004*, pages 16–31, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. ISBN 978-3-540-30539-2.

[133] Akiko Inoue, Tetsu Iwata, Kazuhiko Minematsu, and Bertram Poettering. Cryptanalysis of OCB2: Attacks on Authenticity and Confidentiality. J Cryptol 33, 1871–1913 (2020). https://doi.org/10.1007/s00145-020-09359-8.

[134] Akiko Inoue, Tetsu Iwata, Kazuhiko Minematsu, and Bertram Poettering. Cryptanalysis of OCB2: Attacks on Authenticity and Confidentiality. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019*, pages 3–31, Cham, 2019. Springer International Publishing. ISBN 978-3-030-26948-7.

[135] Kazumaro Aoki and Kan Yasuda. The security of the OCB mode of operation without the sprp assumption. In Willy Susilo and Reza Reyhanitabar, editors, *Provable Security*, pages 202–220, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-41227-1.

[136] Tomer Ashur, Orr Dunkelman, and Atul Luykx. Boosting authenticated encryption robustness with minimal modifications. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017*, pages 3–33, Cham, 2017. Springer International Publishing. ISBN 978-3-319-63697-9.

[137] Phillip Rogaway, Mihir Bellare, and John Black. OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption. *ACM Trans. Inf. Syst. Secur.*, 6(3):365–403, aug 2003. ISSN 1094-9224. doi: 10.1145/937527.937529. URL https://doi.org/10.1145/937527.937529.

[138] T. Krovetz and P. Rogaway. The OCB Authenticated-Encryption Algorithm. RFC 7253, DOI 10.17487/RFC7253, May 2014, https://www.rfc-editor.org/info/rfc7253.

[139] Kazuhiko Minematsu. Parallelizable Rate-1 Authenticated Encryption from Pseudorandom Functions. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, pages 275–292, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. ISBN 978-3-642-55220-5.

[140] Ping Zhang, Peng Wang, and Honggang Hu. The int-rup security of ocb with intermediate (parity) checksum. Cryptology ePrint Archive, Report 2016/1059, 2016. https://ia.cr/2016/1059.

[141] Ping Zhang, Peng Wang, Honggang Hu, Changsong Cheng, and Wenke Kuai. Int-rup security of checksum-based authenticated encryption. In Tatsuaki Okamoto, Yong Yu, Man Ho Au, and Yannan Li, editors, *Provable Security*, pages 147–166, Cham, 2017. Springer International Publishing. ISBN 978-3-319-68637-0.

[142] Zhenzhen Bao, Jian Guo, Tetsu Iwata, and Kazuhiko Minematsu. ZOCB and ZOTR: Tweakable blockcipher modes for authenticated encryption with full absorption. *IACR Transactions on Symmetric Cryptology*, 2019(2):1–54, Jun. 2019. doi: 10.13154/tosc.v2019.i2.1-54. URL https://tosc.iacr.org/index.php/ToSC/article/view/8313.

[143] Jean Liénardy and Frédéric Lafitte. A weakness in OCB3 used with short nonces allowing for a break of authenticity and confidentiality. *Inf. Process. Lett.*, 183:106404, 2024. doi: 10.1016/J.IPL.2023.106404. URL https://doi.org/10.1016/j.ipl.2023.106404.

[144] Avik Chakraborti, Nilanjan Datta, Ashwin Jha, Cuauhtemoc Mancillas-López, and Mridul Nandi. Light-OCB: Parallel lightweight authenticated cipher with full security. In Lejla Batina, Stjepan Picek, and Mainack Mondal, editors, *Security, Privacy, and Applied Cryptography Engineering*, pages 22–41, Cham, 2022. Springer International Publishing. ISBN 978-3-030-95085-9.

[145] Avik Chakraborti, Nilanjan Datta, Ashwin Jha, Cuauhtemoc Mancillas-López, Mridul Nandi, and Yu Sasaki. Int-rup secure lightweight parallel ae modes. *IACR Transactions on Symmetric Cryptology*, 2019(4):81–118, Jan. 2020. doi: 10.13154/tosc.v2019.i4.81-118. URL https://tosc.iacr.org/index.php/ToSC/article/view/8454.

[146] Ritam Bhaumik, Xavier Bonnetain, André Chailloux, Gaëtan Leurent, María Naya-Plasencia, André Schrottenloher, and Yannick Seurin. QCB: Efficient quantum-secure authenticated encryption. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2021*, pages 668–698, Cham, 2021. Springer International Publishing. ISBN 978-3-030-92062-3.

[147] Phillip Rogaway and Thomas Shrimpton. A provable-security treatment of the key-wrap problem. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006*, pages 373–390, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-34547-3.

[148] Shay Gueron, Adam Langley, and Yehuda Lindell. AES-GCM-SIV: Nonce Misuse-Resistant Authenticated Encryption. RFC 8452, DOI 10.17487/RFC8452, April 2019, https://www.rfc-editor.org/info/rfc8452.

[149] Shay Gueron and Yehuda Lindell. GCM-SIV: Full Nonce Misuse-Resistant Authenticated Encryption at Under One Cycle per Byte. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, page 109–119, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450338325. doi: 10.1145/2810103.2813613. URL https://doi.org/10.1145/2810103.2813613.

[150] Tetsu Iwata. New Blockcipher Modes of Operation with Beyond the Birthday Bound Security. In Matthew Robshaw, editor, *Fast Software Encryption*, pages 310–327, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-36598-3.

[151] Tetsu Iwata. Authenticated Encryption Mode for Beyond the Birthday Bound Security. In Serge Vaudenay, editor, *Progress in Cryptology – AFRICACRYPT 2008*, pages 125–142, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-68164-9.

[152] 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*, pages 33–63, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg. ISBN 978-3-662-53018-4.

[153] Avijit Dutta, Mridul Nandi, and Abishanka Saha. Proof of mirror theory for $\xi_{\max} = 2$. *IEEE Trans. Inf. Theory*, 68(9):6218–6232, 2022. doi: 10.1109/TIT.2022.3171178. URL https://doi.org/10.1109/TIT.2022.3171178.

[154] Benoît Cogliati, Avijit Dutta, Mridul Nandi, Jacques Patarin, and Abishanka Saha. Proof of mirror theory for a wide range of $\xi _{\max }$. 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 470–501. Springer, 2023. doi: 10.1007/978-3-031-30634-1_16. URL https://doi.org/10.1007/978-3-031-30634-1_16.

[155] Srimanta Bhattacharya and Mridul Nandi. Revisiting variable output length xor pseudorandom function. *IACR Transactions on Symmetric Cryptology*, 2018(1):314–335, Mar. 2018. doi: 10.13154/tosc.v2018.i1.314-335. URL https://tosc.iacr.org/index.php/ToSC/article/view/853.

[156] Wei Dai, Viet Tung Hoang, and Stefano Tessaro. Information-theoretic indistinguishability via the chi-squared method. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017*, pages 497–523, Cham, 2017. Springer International Publishing. ISBN 978-3-319-63697-9.

[157] Srimanta Bhattacharya and Mridul Nandi. Luby-rackoff backwards with more users and more security. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2021, Proceedings, Part III*, volume 13092 of *LNCS*, pages 345–375. Springer, 2021. doi: 10.1007/978-3-030-92078-4_12. URL https://doi.org/10.1007/978-3-030-92078-4_12.

[158] Daniel J. Bernstein. Salsa20 specification. *eSTREAM Project algorithm description*, 2005.

[159] NIST. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. *Federal Information Processing Standards (FIPS) Publication*, 202, 2015. URL http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf.

[160] David A. McGrew and John Viega. The Security and Performance of the Galois/Counter Mode (GCM) of Operation. In Anne Canteaut and Kapalee Viswanathan, editors, *INDOCRYPT*, volume 3348 of *LNCS*, pages 343–355. Springer, 2004. doi: 10.1007/978-3-540-30556-9_27.

[161] Ted Krovetz and Phillip Rogaway. The Software Performance of Authenticated-Encryption Modes. In Antoine Joux, editor, *FSE*, volume 6733 of *LNCS*, pages 306–327. Springer, 2011. doi: 10.1007/978-3-642-21702-9_18.

[162] Kan Yasuda. A New Variant of PMAC: Beyond the Birthday Bound. In Phillip Rogaway, editor, *CRYPTO*, volume 6841 of *LNCS*, pages 596–609. Springer, 2011.

[163] Kan Yasuda. The Sum of CBC MACs Is a Secure PRF. In Josef Pieprzyk, editor, *CT-RSA*, volume 5985 of *LNCS*, pages 366–381. Springer, 2010. doi: 10.1007/978-3-642-11925-5_25.

[164] Liting Zhang, Wenling Wu, Han Sui, and Peng Wang. 3kf9: Enhancing 3GPP-MAC beyond the Birthday Bound. In Xiaoyun Wang and Kazue

Sako, editors, *ASIACRYPT*, volume 7658 of *LNCS*, pages 296–312. Springer, 2012. doi: 10.1007/978-3-642-34961-4_19.

[165] Yusuke Naito. Blockcipher-Based MACs: Beyond the Birthday Bound Without Message Length. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT III*, volume 10626 of *LNCS*, pages 446–470. Springer, 2017. doi: 10.1007/978-3-319-70700-6_16.

[166] Tetsu Iwata and Kazuhiko Minematsu. Stronger Security Variants of GCM-SIV. *IACR Trans. Symmetric Cryptol.*, 2016(1):134–157, 2016. doi: 10.13154/tosc.v2016.i1.134-157.

[167] Nilanjan Datta, Avijit Dutta, Mridul Nandi, and Goutam Paul. Double-block Hash-then-Sum: A Paradigm for Constructing BBB Secure PRF. *IACR Trans. Symmetric Cryptol.*, 2018(3):36–92, 2018. doi: 10.13154/tosc.v2018.i3.36-92.

[168] Moses D. Liskov, Ronald L. Rivest, and David A. Wagner. Tweakable Block Ciphers. In Moti Yung, editor, *CRYPTO*, volume 2442 of *LNCS*, pages 31–46. Springer, 2002. doi: 10.1007/3-540-45708-9_3.

[169] Tim Beyne, Yu Long Chen, Christoph Dobraunig, and Bart Mennink. Dumbo, jumbo, and delirium: Parallel authenticated encryption for the lightweight circus. *IACR Trans. Symmetric Cryptol.*, 2020(S1):5–30, 2020. doi: 10.13154/tosc.v2020.iS1.5-30. URL https://doi.org/10.13154/tosc.v2020.iS1.5-30.

[170] 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, *EUROCRYPT I*, volume 9665 of *LNCS*, pages 263–293. Springer, 2016. doi: 10.1007/978-3-662-49890-3_11.

[171] Jian Guo, Thomas Peyrin, and Axel Poschmann. The PHOTON family of lightweight hash functions. In *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, pages 222–239, 2011. doi: 10.1007/978-3-642-22792-9_13. URL https://doi.org/10.1007/978-3-642-22792-9_13.

[172] Andrey Bogdanov, Miroslav Knezevic, Gregor Leander, Deniz Toz, Kerem Varici, and Ingrid Verbauwhede. SPONGENT: A Lightweight Hash Function. In Bart Preneel and Tsuyoshi Takagi, editors, *CHES*, volume 6917 of *LNCS*, pages 312–325. Springer, 2011. doi: 10.1007/978-3-642-23951-9_21.

[173] Mridul Nandi. Mind the Composition: Birthday Bound Attacks on EWCDMD and SoKAC21. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT I*, volume 12105 of *LNCS*, pages 203–220. Springer, 2020. doi: 10.1007/978-3-030-45721-1_8.

[174] Avik Chakraborti, Mridul Nandi, Suprita Talnikar, and Kan Yasuda. On the Composition of Single-Keyed Tweakable Even-Mansour for Achieving BBB Security. *IACR Trans. Symmetric Cryptol.*, 2020(2):1–39, 2020. doi: 10.13154/tosc.v2020.i2.1-39. URL https://doi.org/10.13154/tosc.v2020.i2.1-39.

[175] Avijit Dutta and Mridul Nandi. BBB Secure Nonce Based MAC Using Public Permutations. In Abderrahmane Nitaj and Amr M. Youssef, editors, *AFRICACRYPT*, volume 12174 of *LNCS*, pages 172–191. Springer, 2020. doi: 10.1007/978-3-030-51938-4_9.

[176] Yu Sasaki, Yosuke Todo, Kazumaro Aoki, Yusuke Naito, Takeshi Sugawara, Yumiko Murakami, Mitsuru Matsui, and Shoichi Hirose. Minalpher v1.1. 29 August 2015. Second-round submission to the CAESAR competition.

[177] Avijit Dutta, Mridul Nandi, and Suprita Talnikar. Permutation Based EDM: An Inverse Free BBB Secure PRF. *IACR Trans. Symmetric Cryptol.*, 2021 (2):39, 2021. To appear.

[178] Bart Mennink and Samuel Neves. Encrypted Davies-Meyer and Its Dual: Towards Optimal Security Using Mirror Theory. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO, Part III*, volume 10403 of *LNCS*, pages 556–583. Springer, 2017. doi: 10.1007/978-3-319-63697-9_19. Full version at https://eprint.iacr.org/2017/473.

[179] Tetsu Iwata, Bart Mennink, and Damian Vizár. CENC is Optimally Secure. *IACR Cryptology ePrint Archive*, 2016:1087, 2016. URL http://eprint.iacr.org/2016/1087.

[180] Bart Mennink and Samuel Neves. Optimal PRFs from Blockcipher Designs. *IACR Trans. Symmetric Cryptol.*, 2017(3):228–252, 2017. doi: 10.13154/tosc.v2017.i3.228-252.

[181] Patrick Derbez, Tetsu Iwata, Ling Sun, Siwei Sun, Yosuke Todo, Haoyang Wang, and Meiqin Wang. Cryptanalysis of AES-PRF and Its Dual. *IACR Trans. Symmetric Cryptol.*, 2018(2):161–191, 2018. doi: 10.13154/tosc.v2018.i2.161-191.

[182] Benoît Cogliati and Yannick Seurin. Analysis of the single-permutation encrypted Davies-Meyer construction. *Des. Codes Cryptogr.*, 86(12):2703–2723, 2018. doi: 10.1007/s10623-018-0470-9.

[183] Chun Guo, Yaobin Shen, Lei Wang, and Dawu Gu. Beyond-birthday secure domain-preserving PRFs from a single permutation. *Des. Codes Cryptogr.*, 87(6):1297–1322, 2019. doi: 10.1007/s10623-018-0528-8.

[184] Nilanjan Datta, Avijit Dutta, Mridul Nandi, and Kan Yasuda. Encrypt or Decrypt? To Make a Single-Key Beyond Birthday Secure Nonce-Based MAC. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO I*, volume 10991 of *LNCS*, pages 631–661. Springer, 2018. doi: 10.1007/978-3-319-96884-1_21.

[185] Tetsu Iwata. Tightness of the Security Bound of CENC. In Eli Biham, Helena Handschuh, Stefan Lucks, and Vincent Rijmen, editors, *Symmetric Cryptography*, volume 07021 of *Dagstuhl Seminar Proceedings*. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2007. URL http://drops.dagstuhl.de/opus/volltexte/2007/1016.

[186] Jacques Patarin. Introduction to Mirror Theory: Analysis of Systems of Linear Equalities and Linear Non Equalities for Cryptography. *IACR Cryptology ePrint Archive*, 2010:287, 2010.

[187] Srimanta Bhattacharya and Mridul Nandi. Revisiting Variable Output Length XOR Pseudorandom Function. *IACR Trans. Symmetric Cryptol.*, 2018(1):314–335, 2018. doi: 10.13154/tosc.v2018.i1.314-335.

[188] Wei Dai, Viet Tung Hoang, and Stefano Tessaro. Information-Theoretic Indistinguishability via the Chi-Squared Method. In Jonathan Katz and

Hovav Shacham, editors, *CRYPTO Part III*, volume 10403 of *LNCS*, pages 497–523. Springer, 2017. doi: 10.1007/978-3-319-63697-9_17. Full version at http://eprint.iacr.org/2017/537, latest version 20170616:190106.

[189] Itai Dinur, Orr Dunkelman, Nathan Keller, and Adi Shamir. Key Recovery Attacks on 3-round Even-Mansour, 8-step LED-128, and Full AES^2. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT I*, volume 8269 of *LNCS*, pages 337–356. Springer, 2013. doi: 10.1007/978-3-642-42033-7_18.

[190] Orr Dunkelman, Nathan Keller, and Adi Shamir. Minimalism in Cryptography: The Even-Mansour Scheme Revisited. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT*, volume 7237 of *LNCS*, pages 336–354. Springer, 2012. doi: 10.1007/978-3-642-29011-4_21.

[191] Aldo Gunsing and Bart Mennink. The Summation-Truncation Hybrid: Reusing Discarded Bits for Free. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO I*, volume 12170 of *LNCS*, pages 187–217. Springer, 2020. doi: 10.1007/978-3-030-56784-2_7.