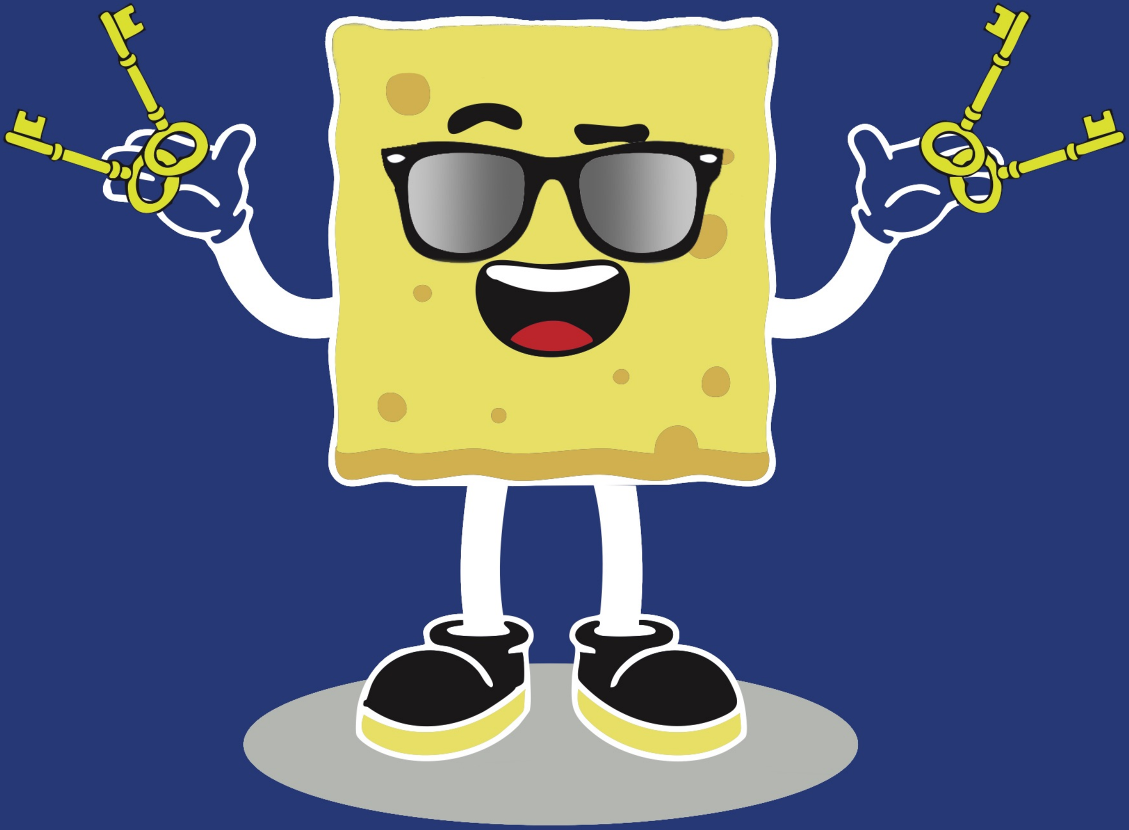
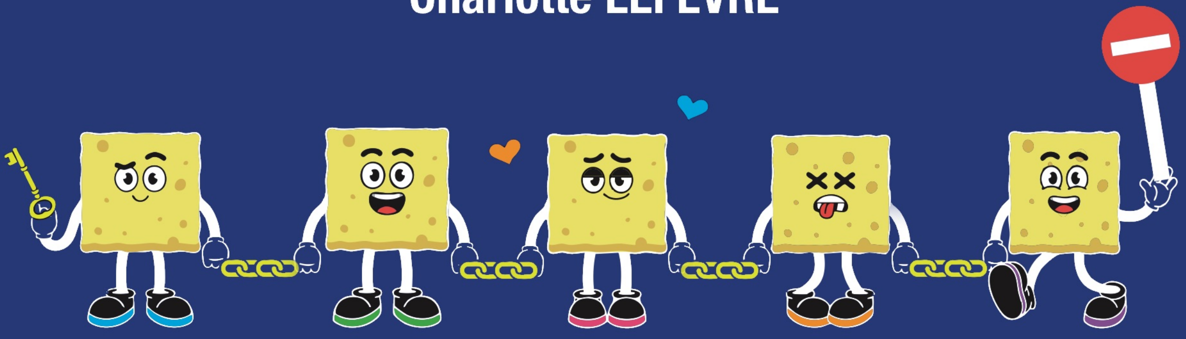
PROVABLE SECURITY OF PERMUTATION-BASED CRYPTOGRAPHY

Never throw in the sponge!



Charlotte LEFEVRE



Provable Security of Permutation-Based Cryptography

Charlotte Simone Thérèse Lefevre

This work has been supported by the Netherlands Organisation for Scientific Research (NWO) under grant OCENW.KLEIN.435

Printed by Proefschriftspecialist, Zaandam

Provable Security of Permutation-Based Cryptography

Proefschrift

ter verkrijging van de graad van doctor
aan de Radboud Universiteit Nijmegen
op gezag van de rector magnificus prof. dr. J.M. Sanders,
volgens besluit van het college voor promoties
in het openbaar te verdedigen op

dinsdag 10 februari 2026
om 12.30 uur precies

door

Charlotte Simone Thérèse Lefevre

geboren op 11 september 1997
te Clermont-Ferrand (Frankrijk)

Acknowledgments

Before my PhD even began, I tried to imagine what qualities an *ideal supervisor* would have. Then I met Bart Mennink, and I quickly realized that no security model could even capture half of his brilliance.

Bart, your expertise, intellectual rigor, boundless enthusiasm, and ability to challenge have been a constant source of inspiration. You always made time to discuss research and give constructive feedback; meticulously proofreading every text and every single slide, so much so that not even a 0.1cm accidental shift in a TikZ picture escaped your notice.

You taught me to think independently. You encouraged me to try things I never would have believed possible, to take calculated leaps, and to do so with the comfort of a safety net you always provided. When I was on the verge of throwing my laptop out the window, one meeting with you was enough to convince me that everything would be okay and that I needed to rest. Turns out, you were (again) right.

You have been a true role model to me, and everything I've learned from you will stay with me far beyond this thesis. Thank you for being such a supportive and brilliant mentor at every single step of the way.

While I'm excited to move forward, part of me is sad that this PhD is over, because it means I no longer have the privilege of being supervised by a random oracle^{++} anymore. I will dearly miss your daily jokes and witnessing your LaTeX wizardry in action!

I would also like to thank my promotor, Joan Daemen, for giving me great academic freedom while always offering support and valuable guidance throughout these years. I am deeply grateful for the discussions, the opportunities you opened up for me, and the insightful advice you shared along the way.

I would like to thank my manuscript committee, consisting of Herman Geuvers, Maria Eichlseder, Tetsu Iwata, Gaëtan Leurent, and Thomas Ristenpart for taking the time and effort to carefully read my thesis and for providing valuable feedback that helped improve this work.

I had the opportunity to work with and learn from many brilliant people. Thanks to Damian Vizár for his supervision during the internship. Your sharp mind and endless curiosity consistently challenged me to think differently. In addition, many thanks to Yanis Belkheyar, Christoph Dobraunig, Shun Li, Mario Marhuenda Beltrán, Siwei Sun, Zhiyu Zhang, Zhen Qin and Dengguo Feng. I have learned so much from you, and it was a privilege to collaborate and exchange ideas together.

My deepest thanks go to Pierre Karpman, who taught cryptography courses and supervised my master's thesis. Pierre, it was your dedication, enthusiasm, and your outstanding sense of pedagogy that led me to choose cryptography. Throughout the master's thesis you were an excellent supervisor; discussing ideas with you was always enriching. I remember clumsily telling you how much I really liked the Even-Mansour construction, distinguishing games, and random oracles. Just a few days later you sent a PhD proposal from Bart and Joan, which proved a perfect match. Thank you so much for everything.

Mario, you are my academic brother. I loved our late-night idea sessions, whiteboard scribbles, building a tree of thoughts of size $\mathcal{O}(\infty)$ and the way we'd dig into a paper until we reached an existential crisis, only to cycle back to the beginning (just like those infamous counterexamples...). I'll never forget to be careful about reviewer B, and that constant factors are rarely to be worried about in real life. Thank you for the laughs and for being a constantly cheerful collaborator full of ideas.

Yanis, your support has been remarkable, from that infamous sleepless night to symbolic bungee jumps through hard times. You were always attentive, generous, and ready to help while bringing delicious home-made cakes. Co-organizing GelreCrypt with you and Bart was truly enjoyable. Thank you.

It was a lot of fun being part of the legendary organization of the Three Musketeers, a name found by Bart, with co-organizers Mario and Yanis. Together, we managed to pull off countless jokes, especially at Bart's expense. And somehow, thanks to the three of you (and Clémence!), I ended up training for and running a marathon, which was definitely *not* on my bucket list!

Solane, thank you so much for your unconditional support and constant company throughout these years. From long talks to all our trips, you were there for the big moments and the small ones. No matter what, life is lifing!

To Louise, who I met during my licence and who always positively influenced me to be better. Even from afar during the PhD, your influence has stayed with me.

Cas, the spritzs, the bathing grazers, and all the fun will be fondly remembered.

Cédric, thanks to you I was able to survive through the cold winter in Neuchâtel. I will continue to cherish the memories of our weekly fonclette evenings and all the fun we had as you introduced me to so many wonderful places.

Thanks to my office mates for creating such a supportive, calm, and joyful environment. I will dearly miss you, Zahra, as a dedicated and constant companion in the office. Working alongside you made even the longest days enjoyable, whether it was sharing hardcore deadlines, complaining about the new coffee machine, or having so many interesting discussions. Alex, I enjoyed our shared experiences from long office days to our conference adventures (remember that room upgrade?). I helped you settle in, and now we swap roles!

Many thanks to the secretaries Janet and Kira for their constant support and kindness. Thanks to the ESCADA team members for the interesting conversations and weekly meetings. Thank you in general to all digital security members.

Thank you to everyone I met at conferences and events for all the enriching discussions. It would be impossible to cite everyone individually, but I would like to thank, in no particular order: Rachelle, Clémence, Ashwin, Augustin, Christina, André, Yann, Morten, Patrick, Baptiste, Gaëtan, Marcel, César, Aurel, Maria, María, Mustafa, Maxime, Anne.

Many thanks go to Yliès Falcone, Bruno Laurent, Florent Bouchez-Tichadou, Yan Gerard, Christophe Massard. You were exemplary teachers/researchers who played a significant positive role in my career path. You encouraged and helped me to take decisions I have never regretted.

Cathy Bouhard, sans toi la couverture de cette thèse aurait ressemblé au travail d'un enfant de six ans. Même si tu n'es pas familière avec le domaine, tu as patiemment écouté toutes mes idées absurdes et les as transformées en cette couverture. Je suis vraiment fière du résultat. Merci infiniment.

Romaric, j'ai littéralement soumis cette thèse à tes côtés, sur une aire d'autoroute complètement miteuse. C'était un moment à la fois absurde et chargé d'émotion. Tu es incroyable. Merci de faire partie de ma vie.

Papa, Maman, il faudrait un manuscript dédié pour vous remercier proprement. Une manière simple d'illustrer votre soutien inconditionnel est votre charmante habitude d'organiser des vacances en fonction de l'endroit où j'atterris. J'espère que vous n'avez pas fait une overdose de gouda. Merci pour tout. Merci, Tonton, pour ton soutien.

À Anne-Marie, Simone, et Thérèse.

Contents

Part I

Foundations

Chapter 1

Introduction

Modern digital systems fundamentally rely on the secure transmission and storage of information. To meet this critical need, cryptology, the science of secrets, provides the core techniques for protecting information in the presence of active adversaries. Its history is as old as humanity's need to safeguard sensitive messages, but its modern form took shape more recently, with the works of Shannon [Sha48, Sha49] and Kerckhoffs [Ker83] that laid the theoretical foundations for modern cryptography.[1]

Early cryptology focused mostly on *confidentiality*, which takes care of transforming the messages into unintelligible ciphertexts. Although the ability to tamper with communication is not new, the widespread deployment of modern communication technologies makes it easier than ever for attackers to eavesdrop, inject, modify, or replay information. Therefore, ensuring that the data comes from a legitimate source and remains unaltered is a crucial requirement. Those two properties are named respectively *authenticity* and *integrity*.

Until the mid-1970s, cryptology considered a setting where the sender and receiver share the same secret key, which is now known as *symmetric-key cryptology*. The work of Diffie and Hellman [DH76] broke this barrier: two parties can now establish a shared secret over an insecure channel. This paved the way for *asymmetric cryptography* or *public-key cryptography*, in which each user publishes a public key to encrypt or verify signatures while keeping a separate private key to decrypt or sign messages. Asymmetric cryptography

[1]Refer to Kahn [Kah96] for a complete history of cryptography.

can be used on its own for encryption, but at a far greater computational cost than the symmetric counterparts. In practice, modern cryptographic protocols often use a hybrid approach by combining public-key cryptography to establish shared secrets with symmetric-key cryptography for bulk encryption. Well-known examples of this approach include TLS [SCM08], SSH [IS09], and Signal [PM16].

1.1 Symmetric-Key Cryptography

Symmetric-key schemes are typically required to guarantee either confidentiality, authenticity, or both. These schemes are often built in a modular way, where a small core primitive is combined with a construction. One of the most widely used core primitives is the *block cipher*, which is an invertible keyed function that encrypts fixed-length messages. A tweakable block cipher extends this notion by incorporating an additional non-secret input named the tweak, which enables more flexible constructions [LRW02]. Nowadays, the most used block cipher is AES [DR02], which replaced the previous standard DES [Nat77].

In the following, we will review several types of symmetric-key algorithms, and briefly explain when and how they are used.

1.1.1 Encryption Schemes

Encryption-only schemes aim to guarantee confidentiality, so that ciphertexts leak no useful information about the plaintexts. In the strongest security model, ciphertexts are required to be *indistinguishable from random strings*. It has been shown [Sha49] that perfect indistinguishability can only be reached by the one-time pad (OTP) [Mil82], which encrypts a message by XORing it with a one-time key of the same length.

However, OTPs are impractical for most applications. *Stream ciphers* approximate the OTP by expanding a short seed key with the help of a diversifier into a long pseudorandom keystream. With keys as small as 128 bits, they may produce streams that are computationally indistinguishable from random. For instance, one popular method to build a stream cipher is the counter mode of operation [Dwo01].

The aforementioned stream ciphers produce ciphertexts such that one bit flip in the ciphertext results in one bit flip on the plaintext. In contrast, some encryption schemes ensure that every bit of the ciphertext depends on every bit of the plaintext. This is realized by *wide block ciphers* that encrypt an entire plaintext via an invertible transformation. Examples are the wide block cipher Adiantum [CB18], or the modes HCTR [WFW05] and (docked-)double-decker [GDM19]. At the time of writing this thesis, the US National Institute of Standards and Technology (NIST) has expressed interest in standardizing an AES-based *cryptographic accordion* [CDD+25], which is defined as a tweakable wide block cipher.

1.1.2 Message Authentication Schemes

Message Authentication Codes (MACs) ensure that a message comes from a legitimate sender and has not been tampered with. To illustrate the concept, suppose that Alice and Bob share a secret key. When Bob wants to send a message to Alice, he computes a MAC tag using their shared key and appends it to the message. When Alice receives a message that appears to come from Bob, she recomputes the tag on the message using the same key and verifies if it matches the one received. A successful verification convinces Alice that the message was produced by someone who knows the key (presumably Bob) and that it has not been tampered with. In other words, without the secret key, it is computationally infeasible for an adversary to create a fresh message that will pass the verification. Widely used types of MAC include the HMAC construction [BCK96], and the CMAC mode [Dwo05].

1.1.3 Authenticated Encryption Schemes

Authenticated encryption schemes aim to combine both confidentiality and authenticity into one cryptographic suite. These primitives are particularly important because many applications require both security properties simultaneously. By design, this unified approach avoids the pitfalls of ad hoc composition of an encryption scheme with a MAC, and often improves performance by sharing internal components. The OCB family [RBBK01, RBB03, KR11], CCM [Dwo04, WHF03], and GCM [Dwo07, MV04] are examples of prominent AEAD modes.

1.1.4 Hash Functions

A cryptographic hash function maps an input of arbitrary length to a fixed-length digest. Variants that allow variable-length output are called *extendable-output functions* (XOFs). Hash functions do not require a secret key, yet their internal design often uses components and techniques similar to those found in symmetric-key primitives. A secure hash function must be resistant to preimage, second-preimage, and collision attacks. However, as we discuss in Section 2.4.1, sometimes stronger security notions are required.

Hash functions are highly versatile tools and appear in a wide range of use cases. To cite a few examples, they can be used for digital signatures, password protection, proof-of-works, integrity checks, and zero-knowledge proof systems. They can also be used as a component of an encryption/authentication scheme, as in for instance the aforementioned HMAC. The SHA-2 [Nat15a] and SHA-3 [Nat15b] families, standardized by the NIST, are examples of widely used hash functions.

1.2 Modern Challenges in Symmetric-Key Cryptography

The rapid proliferation of the Internet of Things (IoT) has made resource-constrained environments ubiquitous. IoT devices, such as sensors, smart-cards, RFID tags, and embedded controllers, may transfer sensitive data that needs cryptographic protection, yet they must do so with minimal resources. Indeed, these devices often operate on limited area, RAM, ROM, computational throughput, and/or may run on battery power for months or even years. Therefore, improving the performance and efficiency of cryptographic algorithms is essential not only to satisfy these technical constraints, but also to reduce the overall costs and enable their widespread adoption across a diverse range of applications. Standard AES-based cryptographic schemes can exceed these tight memory and energy budgets, making them impractical for those aforementioned applications (see, e.g., [BKL+07, BCG+12])

This limitation highlighted a need to design algorithms tailored to meet those harsh constraints without compromising security, which is the focus of *lightweight cryptography*. Perhaps one of the earliest proposals that fits this description is the NOEKEON block cipher [DPVR00], dating back to 2000. Starting from the mid-2000s, a new generation of explicitly lightweight branded (tweakable) block ciphers appeared, for example HIGHT [HSH+06], PRESENT [BKL+07], KATAN [CDK09], PRINCE [BCG+12], PRINCEv2 [BEK+20], SIMON [BSS+13], SKINNY [BJK+16], GIFT [BPP+17], QARMA [Ava17], and QARMAv2 [ABD+23].

Concurrently to these publications of lightweight block ciphers, the EU ECRYPT network launched the eSTREAM project in 2004 [ECR04] to identify new stream ciphers suited for both high-throughput and constrained devices. With the project completed in 2008, eSTREAM's portfolio includes

Grain [HJMM08], MICKEY [BD08], and Trivium [CP08] in the constrained environments category.

A similar evolution appeared in authenticated encryption over the past decade, in the form of two competitions. First, the CAESAR competition [CAE14] (2014-2019) was launched to encourage the design of authenticated encryption schemes. Its final portfolio includes, in the category lightweight, the schemes Ascon [DEMS14] and ACORN [Wu14] as first and second choices, respectively. Right after CAESAR, the lightweight cryptography competition organized by the NIST [Nat19] began in 2019. Its goal was to standardize an authenticated encryption algorithm and optionally a hash function that offer robust security while meeting the tight resource budgets of embedded systems. This competition was concluded in 2023, with the Ascon suite selected as an overall winner [DEMS21b, DEMS21a, SMC$^+$25].

Ascon, which won both the CAESAR competition (in the category lightweight) and the NIST competition, is based on the sponge construction [BDPV07], which we discuss in the next section.

1.3 The Sponge Construction

The introduction of cryptographic sponge functions by Bertoni et al. [BDPV07] in 2007 marked a major advance in the field of permutation-based cryptography. The idea of sponge functions originated during the design of RadioGatún [BDPV06], initially motivated by the need for a clear reference to express security claims [BDPV11a, Section 1.1]. Building on this foundation, the authors proposed the sponge construction and instantiated it with the Keccak-f[1600] permutation [BDPV11d]. The resulting family of sponge functions was submitted to the NIST SHA-3 competition [Nat12], where it was ultimately selected as the winner [Nat15b].

In more detail, the sponge is a construction for building an extendable output function out of a fixed-size function. Since most applications use a permutation as the building block, we will focus on that case. Let $\mathcal{P}$ be the permutation, and $b \geq 1$ its bit size. The sponge operates on a state of size b bits, which is split into an inner part of size c bits (the *capacity*) and an outer part of size r bits (the *rate*), where $b = r + c$. The sponge first absorbs the input data by injecting r bits at a time into the state, applying $\mathcal{P}$ to update the state after each block. After the last plaintext block is absorbed, the sponge squeezes the digest by extracting it from the state r bits at a time, each extraction being interleaved with an evaluation of $\mathcal{P}$. See Figure 1.1 for an illustration, and Section 3.1.1 for a detailed description.

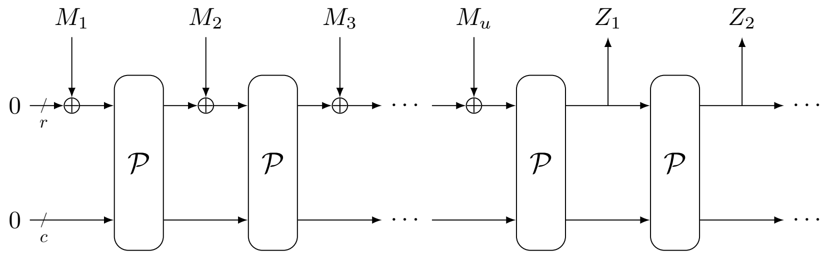


Figure 1.1: Illustration of the sponge construction.

The rate determines at which pace the messages are absorbed and the digest blocks extracted, while the capacity is a security parameter. Therefore, the choice of r/c is an efficiency/security tradeoff. For example, the members of the SHA-3 family operate on a state of $b = 1600$ bits, with capacities ranging from $c = 448$ (with rate $r = 1152$) to $c = 1024$ (with rate $r = 576$).

Extensive research has shown that the sponge paradigm can be extended beyond hashing to support a wider range of cryptographic functionalities. For instance, the sponge can be keyed to build pseudorandom functions, which are useful for building MACs and for keystream generation. Likewise, the duplex construction [BDPV11b, DMV17] is a stateful variant of the sponge and is used to build authenticated encryption schemes. For more details, refer to Sections 3.2 and 3.3.

It was quickly acknowledged that the sponge was particularly well-suited for lightweight hashing, see, e.g., QUARK [AHMN10], Spongent [BKL$^+$11], and PHOTON [GPP11]. Similarly, the designers of the sponge and duplex constructions recognized the duplex's suitability for lightweight authenticated encryption [BDPV11b]. They instantiated their MonkeyDuplex construction [BDPV12] with Keccak-p-based permutations [BDPV11d],[2] which led to the Ketje family of authenticated encryption functions [BDP$^+$16a]. Fundamental research as well as the development of designs in this direction has been significantly boosted by the aforementioned CAESAR [CAE14] and NIST lightweight cryptography [Nat19] competitions. In the CAESAR competition, there were 10 out of 57 submissions based on or inspired by the duplex (including Ketje). In the NIST lightweight cryptography competition, 22 out of 57 submissions (including 5 finalists) were duplex-inspired. As mentioned earlier, Ascon was

[2]More precisely, they defined the MonkeyWrap mode [BDP$^+$16a] on top of MonkeyDuplex, which is then instantiated with these permutations.

selected as a winner for the CAESAR competition (in the category lightweight) and the NIST competition.

1.4 Provable Security

In Section 1.1 we used the term *computationally infeasible* to describe the absence of any algorithm running with practical computational resources that can break the scheme with non-negligible probability. In principle, we would like to prove such statements mathematically, but in most cases cryptographers do not know how to formally prove an entire cryptosystem. For instance, there exists no mathematical theorem stating that SHA-2 or SHA-3 is secure. Instead, the analysis relies on two complementary methodologies that give strong evidence of security.

The first approach is *cryptanalysis*, which consists of applying various attack techniques (e.g., differential cryptanalysis [BS90, BS91], linear cryptanalysis [Mat93]) to test whether the scheme is resistant to those attacks. The absence of any attack better than brute-force, despite extensive public scrutiny, gives a certain level of confidence in the security of the scheme. For instance, AES is considered as highly secure due to the vast amount of research attempting, yet failing, to break it.

The second one, and the focus of this thesis, is *provable security*, which refers to mathematical techniques used to claim the security of constructions or protocols. The foundations of this framework were laid by Goldwasser and Micali [GM82]. For keyed block cipher-based constructions, proofs typically work in the *standard model*: security of the construction is reduced to that of the underlying block cipher. For permutation-based constructions, analysts often work in the *ideal model*, where the permutation is modeled as uniformly random. The ideal model is a stronger assumption than the aforementioned standard model, but it is often the best available framework given the lack of alternative assumptions for analyzing permutations-based schemes.

Even if those models are fundamentally different, they still share one common point: they serve to argue that a construction or protocol is sound. Thus, if a concrete instance built on top of a secure construction is broken, then the underlying weakness arises from the underlying primitive. In our context, provable security consists of proving upper bounds on the probability that the adversary breaks a given construction based on an ideal permutation. It shows that, up to a certain number of queries/evaluations, the scheme is *generically* secure.

A crucial aspect in provable security is the *tightness* of the bounds. We call a bound *tight* if there exists a generic attack that succeeds with a probability close to the bounds. If a bound is not tight, this leaves a gap that could be interpreted in two ways: either the best known attack is not optimal, or the proof techniques used are not sharp enough to capture the true security of the construction. Security bounds determine the maximum level of security that can be confidently claimed by designers, and a loose bound due to the proof technique may lead to a false sense of insecurity. Having tight bounds is thus important, especially in lightweight cryptography, where the security margins are pushed at their limits. For example, consider the sponge construction. It has benefited from an extensive provable security treatment, and notably it has been proven to behave like a random oracle in the indifferentiability framework [MRH04, CDMP05][3] as long as the total number of permutation evaluations is at most $2^{c/2}$, and the obtained bound is tight [BDPV08]. Here, we clearly see the link between tight security bounds and efficiency: consider for instance a permutation of size 320 bits (the size of the Ascon permutation). Then, in order to have 128 bits of security, c needs to be of size at least 256, which limits the rate (hence the efficiency parameter) to 64. If the bound were not tight, then r would need to be decreased to ensure the same level of provable security, hence slowing absorption and squeezing.

1.5 Research Direction

The goal of this thesis is to support the design of permutation-based cryptographic schemes from the angle of provable security. To that end, we explore the following directions:

- Prove tight security bounds and perform an extensive analysis of existing permutation-based constructions;
- Design permutation-based constructions;
- Analyze permutation-based constructions in specific use cases, where they may offer stronger security guarantees than those derived from general-purpose security models.

[3]The indifferentiability framework is discussed in detail in Section 2.4.1.2.

1.6 Outline

The remainder of the thesis is divided into five parts, with each chapter detailed in the following.

Part I: Foundations

This part introduces the foundational concepts on which this thesis is based.

Chapter 2: Preliminaries. This chapter introduces useful notation and important notions used in the remainder of the thesis, including several security models, relevant cryptographic constructions, and proof techniques. Contribution-wise, it also contains a slight improvement of a multicollision bound.

Chapter 3: the Sponge Construction and Its Variants. In this chapter, we provide a comprehensive overview of the sponge construction, the duplex construction, and several of their variants, which are used to build hash functions, pseudorandom functions, and authenticated encryption schemes. Additionally, the chapter discusses the specifics of adversarial resources in these contexts, and reviews existing security bounds for each construction introduced.

Part II: Security Analysis of Sponge-Based Hash Function Constructions

This part focuses on core results around sponge-based constructions for hashing purposes. It consists of three chapters.

Chapter 4: Tight Preimage Resistance of the Sponge Construction. In this chapter, we show that the security bound for the sponge construction's preimage resistance [AMP10b, BDPV08] can be tightened to match the best-known attack [BDPV07]. To fill this gap, we introduce a new proof technique and demonstrate that several lightweight hash functions have higher generic preimage security than previously believed. Notably, the hash function of the Ascon suite achieves 192-bit generic security, an improvement from the previously believed 128-bit bound.

Chapter 5: Indifferentiability of the Sponge Construction with a Restricted Number of Message Blocks. In this chapter, we revisit the indifferentiability bound of the sponge construction [BDPV08] in the context of restricted-length input messages. We establish a tight upper bound on the indifferentiability of the sponge when the number of message blocks is restricted. This result indicates that, depending on the maximum input size, the absorption rate may be increased without compromising security. Moreover, we also prove a bound on the public indifferentiability [YMO09, DRS09, MPS12] of the (restricted) sponge, a weaker but still relevant security notion. We then extend this result to the unrestricted setting, and prove tight second preimage and collision resistance for a generalized version of the sponge. In particular, this result shows that, for such applications, the squeezing rate of the sponge can be increased up to the full width of the underlying permutation.

Chapter 6: Permutation-Based Hashing Beyond the Birthday Bound. In chapter, we introduce a construction named the double sponge, which positively answers the fundamental question as to whether it is possible to have a hash function based on a permutation of size b bits, with more than $b/2$ bits of security. The double sponge can be seen as the sponge embedded within the double block length hashing paradigm, making two permutation calls in parallel interleaved with an efficient mixing function. Similarly to the sponge, the permutation size is split as $b = r + c$, and the underlying compression function absorbs r bits at a time. We prove that the double sponge is indifferentiable from a random oracle up to approximately $2^{2c/3}$ queries. In particular, when $c > \frac{3b}{4}$, the construction exceeds the birthday bound in the primitive size, making it, to our knowledge, the first hash function construction based on a permutation to achieve this feature.

Part III: Application-Specific Permutation-Based Hashing

This part looks into two specific use cases of permutation-based hash functions, each discussed in one chapter.

Chapter 7: Permutation-Based Hash Chains with Applications to Password Hashing. In this chapter, we focus on the provable security of hash chains used for authentication with one-time passwords, notably with the second factor authentication scheme T/Key [KMB17]. To that end, we introduce a new security model that distinguishes offline and online complexities, enabling a refined analysis. We argue that when considering specific hash function constructions, using the general security notion of indifferentiability results in lossy bounds. We then conduct a dedicated analysis with the sponge construction, which results in refined security guarantees, as well as an enhanced understanding of the preimage resistance of cascaded sponge evaluations. The technical core involves solving a complexified variant of preimage resistance, building upon the results from Chapter 4. For most use cases, however, a truncated permutation is sufficient to instantiate the hash function. In this setting, we prove strong security guarantees, showing that T/Key can be instantiated with permutations of sizes as small as 200 bits.

Chapter 8: To Pad or not to Pad? Padding-Free Arithmetization-Oriented Sponges and Duplexes. This chapter focuses on sponges in the context of arithmetization-oriented applications, that operate on elements on a finite field of large size. In these cases, the padding required by the sponge construction can lead to a significant efficiency loss. We explore an alternative approach that replaces the padding overhead with a non-cryptographic permutation applied to the inner state. We demonstrate how this approach can be used in arithmetization-oriented element-wise sponge-based hashing (similar to the SAFE framework [AKMQ23]) and with the duplex, with a focus on authenticated encryption. Our security bounds are comparable to those of the sponge and duplex, with a constant-factor security loss that is field-agnostic.

Part IV: Design and Analysis of Sponge-Based Pseudorandom Functions and Authenticated Encryption Constructions

This part focuses on sponge-based authenticated encryption and pseudorandom functions.

Chapter 9: Security of the Ascon Modes. In this chapter, we dive into the provable security of the Ascon modes. The mode underlying Ascon authenticated encryption (Ascon-AE) is based on the duplex, which has itself been extensively analyzed with a large body of duplex-based modes. However, Ascon-AE has a unique feature, namely *key blindings*, added by the designers to achieve strong security guarantees, notably in scenarios where internal states may be leaked. In this chapter, we systematize the existing knowledge on the security of Ascon-AE mode, identify and fill gaps when needed, with a particular focus on how the aforementioned key blindings contribute to enhanced security. Moreover, we also systematize the knowledge around the hash function and PRF modes of Ascon.

Chapter 10: Kirby and MacaKey: Sponge-Based Pseudorandom Functions with Enhanced Squeezing Rate. In this chapter, we explore the design of efficient sponge-based PRF constructions, with a focus on optimizing the squeezing phase without sacrificing generic security. We introduce two constructions, Kirby and MacaKey. The rationale of those constructions are different, but they share the similarity that they *squeeze the entire state size*. We show that Kirby achieves $b/2$ bits of security, while MacaKey has generic security comparable to that of the full-state keyed sponge [MRV15], but squeezes the data over the entire state size.

Part V: Conclusions and Perspectives

Chapter 11: Conclusions and Perspectives. In this chapter, we summarize the key findings of this thesis and elaborate on open questions.

1.7 Contributions

This section details the source works for each chapter, explicitly notes any content difference that goes beyond minor changes, and clarifies my specific contributions. Unless stated otherwise, all contributions mentioned are significant.

Chapter 3: the Sponge Construction and Its Variants. Although this chapter mainly reviews a collection of existing works, a small portion is based on the following note:

[Lef24] *Charlotte Lefevre.* **A Note on Adversarial Online Complexity in Security Proofs of Duplex-Based Authenticated Encryption Modes.** Cryptology ePrint Archive, Paper 2024/213. url: `https://eprint.iacr.org/2024/213`.

This note highlights nuances in how security proofs of sponge-based constructions count adversarial resources, which may not always reflect realistic attack scenarios.

Chapter 4: Tight Preimage Resistance of the Sponge Construction. This chapter is based on the following work:

[LM22] *Charlotte Lefevre and Bart Mennink.* **Tight Preimage Resistance of the Sponge Construction.** CRYPTO 2022 (IV). LNCS, vol. 13510, pp. 185-204 Springer (2022).

The majority of the results presented in this chapter are derived directly from the paper, except Section 4.4.2, which refers to follow-up works and extends the applicability of our findings to scenarios where a random preimage is known to exist (i.e., *domain-oriented preimage* resistance [AS11]). This will be used as the technical core to derive the results in Section 7.4.

Contribution: I am the main author of the paper. The proof technique was developed in collaboration with Bart Mennink.

Chapter 5: Indifferentiability of the Sponge Construction with a Restricted Number of Message Blocks. This chapter is based on the following work:

[Lef23] *Charlotte Lefevre.* **Indifferentiability of the Sponge Construction with a Restricted Number of Message Blocks.** IACR Transactions on Symmetric Cryptology 2023, pp. 224-243.

This chapter contains all results presented in this paper, but the part on public indifferentiability of the unrestricted sponge and its application to collision and second preimage resistance (Section 5.5) is new.

Chapter 6: Permutation-Based Hashing Beyond the Birthday Bound. This chapter is based on the following work:

[LM24c] *Charlotte Lefevre and Bart Mennink.* **Permutation-Based Hashing Beyond the Birthday Bound.** IACR Transactions on Symmetric Cryptology 2024, pp. 71-113.

Most of the content has not been changed.

Contribution: I am the main author of the paper; the core ideas of the construction were developed in collaboration with Bart Mennink.

Chapter 7: Permutation-Based Hash Chains with Applications to Password Hashing. This chapter is based on the following work:

[LM24b] *Charlotte Lefevre and Bart Mennink.* **Permutation-Based Hash Chains with Application to Password Hashing.** IACR Transactions on Symmetric Cryptology 2024, pp. 249-286.

Most of the content has not been changed.

Contribution: I contributed to refining the security model, carrying out the security proofs, and writing the paper.

Chapter 8: To Pad or not to Pad? Padding-Free Arithmetization-Oriented Sponges and Duplexes. This chapter is based on the following work:

[LBM25] *Charlotte Lefevre and Mario Marhuenda Beltrán and Bart Mennink.* **To Pad or Not to Pad? Padding-Free Arithmetization-Oriented Sponges.** IACR Transactions on Symmetric Cryptology 2025, pp. 97-137.

All results presented in Chapter 8 can be found in that paper. This paper additionally presents and analyzes a sponge-based PRF that is not included in this thesis.

Contribution: The core idea for these constructions came from Mario Marhuenda Beltrán. I contributed to the development and refinement of the constructions, to writing the paper, and carried out the two security proofs included in this thesis.

Chapter 9: Security of the Ascon Modes. The content of this chapter is based on two publications:

[LM24a] *Charlotte Lefevre and Bart Mennink.* **Generic Security of the Ascon Mode: On the Power of Key Blinding.** Selected Areas in Cryptography 2024 (II). LNCS, vol. 15517, pp. 3-32 Springer (2024).

[LM25a] *Charlotte Lefevre and Bart Mennink.* **SoK: Security of the Ascon Modes.** IACR Transactions on Symmetric Cryptology 2025, pp. 138-210.

The results presented in this chapter are mostly following the second paper, as it includes all the results from the first paper, with a slight adjustment to the security model.

Contribution: I contributed to identifying gaps in existing research, defining the scope of focus, carrying out the security proofs and generic attacks, and writing the papers.

Chapter 10: Kirby and MacaKey: Sponge-Based Pseudorandom Functions with Enhanced Squeezing Rate. The content of this chapter is based on two works in submission. This chapter contains major changes, beyond the necessary changes to merge the two papers together: some results have been omitted, and a comparison between the two constructions has been added.
The first work is:

[LBD23] *Charlotte Lefevre and Yanis Belkheyar and Joan Daemen.* **Kirby: A Robust Permutation-Based PRF Construction.** Cryptology ePrint Archive, Paper 2023/1520. url: `https://eprint.iacr.org/2023/1520`.

This work introduces the construction Kirby and proves its security.

Contribution: I was slightly involved in the design of the construction, with my main responsibility being to carry out the security proof.
The second work is:

[LB25] *Charlotte Lefevre and Mario Marhuenda Beltrán.* **MacaKey: Full-State Keyed Sponge Meets the Summation-Truncation Hybrid.** Cryptology ePrint Archive, Paper 2025/893. url: `https://eprint.iacr.org/2025/893`.

This work introduces the construction MacaKey and proves its security.

Contribution: The core idea behind MacaKey came from Mario Marhuenda Beltrán. I contributed to refining the construction, addressing technical challenges in the security proofs, and writing the paper.

Chapter 2

Preliminaries

2.1 Notation

Let $\mathbb{N}$ denote the set of natural numbers including 0, and $\mathbb{N}^*$ the set excluding 0. We use $x := y$ to define x as being equal to y. For $k, n \in \mathbb{N}$ such that $k \leq n$, $[\![k, n]\!]$ denotes the set $\{k, \ldots, n\}$. We use $[n]_k$ to denote the falling factorial of n of depth k, i.e., the product $\prod_{i=0}^{k-1}(n - i)$. If $x \in \mathbb{N}$ with $x \leq 2^n - 1$, $\langle x \rangle_n$ denotes the n-bit binary representation of x.

Let ϵ denote the empty string. For $a \in \mathbb{N}$, we write $\{0,1\}^a$ to denote the set of binary strings of size a ($\{0,1\}^0$ abuses notation for the set only containing the empty string ϵ), and let $\{0,1\}^{\leq a} := \bigcup_{i=0}^{a}\{0,1\}^i$. We define

$$\{0,1\}^* := \bigcup_{i\in\mathbb{N}}\{0,1\}^i\,, \qquad\qquad (\{0,1\}^a)^* := \bigcup_{i\in\mathbb{N}}\{0,1\}^{i\cdot a}\,.$$

In words, $\{0,1\}^*$ is the set of all finite binary strings, and $(\{0,1\}^a)^*$ those whose length is a multiple of a (including ϵ). We also denote by $\{0,1\}^\infty$ the set of infinite binary strings.

Let $X, Y \in \{0,1\}^*$. We write $X\|Y$ for the concatenation of X and Y. Moreover, if $|X| = |Y|$, $X \oplus Y$ denotes the bitwise addition of X and Y. If L is an ordered list and $1 \leq i < j \leq |L|$, then $L[i : j]$ is the sub-list of L containing the i^{th} to the j^{th} element (latter included). The same notation applies to finite or infinite strings: for $Z \in \{0,1\}^* \cup \{0,1\}^\infty$, $Z[i : j]$ is the substring of Z from position i to j, and we write $Z[i]$ for $Z[i : i]$. Moreover, if

$Z \in \{0,1\}^*$ and $1 \leq s \leq |Z|$,

$$\mathrm{outer}_s(Z) := Z[1:s]\,, \qquad \mathrm{inner}_s(Z) := Z\left[|Z|-s+1:|Z|\right]\,.$$

Additionally, if $s \leq \min\{|X|,|Y|\}$, we write $X \overset{s}{=} Y$ whenever $\mathrm{inner}_s(X) = \mathrm{inner}_s(Y)$. If $|X| \leq |Y|$, we say that X is a prefix of Y, denoted by $X \prec Y$ if Y truncated to its first $|X|$ bits is equal to X. Moreover, a set $\mathcal{E} \subseteq \{0,1\}^*$ is prefix-free if for any $X, Y \in \mathcal{E}$ distinct, X is not a prefix of Y.

Let $\mathcal{S}, \mathcal{S}'$ be two sets. If $\mathcal{S}$ is finite, $x \xleftarrow{\$} \mathcal{S}$ means that x is sampled uniformly at random from $\mathcal{S}$. We denote $(\mathcal{S} \cup \mathcal{S}') \setminus (\mathcal{S} \cap \mathcal{S}')$ by $\mathcal{S} \sqcup \mathcal{S}'$. $\mathbf{1}_{x\in\mathcal{S}}$ denotes the indicator function of $\mathcal{S}$, i.e.,

$$\mathbf{1}_{x\in\mathcal{S}} = \begin{cases} 1 \text{ if } x \in \mathcal{S}\,, \\ 0 \text{ otherwise}\,. \end{cases}$$

Finally, $\exists^{\neq} x, y \in \mathcal{S}$ indicates the existence of x and y as two distinct elements in $\mathcal{S}$.

Throughout this section and in all chapters except for Chapter 8, we will work in the setting of a binary alphabet. However, the definitions and objects presented can be easily generalized to other alphabets by replacing $\{0,1\}$ accordingly.

2.2 Cryptographic Primitives

In this thesis, we use the term *primitive* to refer to a function that serves as a building block for constructing cryptographic schemes. Examples of such primitives are block ciphers and cryptographic permutations.

Let $k, n, b \in \mathbb{N}^*$. Traditionally, most cryptographic schemes are based on block ciphers. A block cipher is a function:

$$\begin{aligned} E : \{0,1\}^k \times \{0,1\}^n &\longrightarrow \{0,1\}^n \\ (K, P) \qquad &\longrightarrow \ C\,. \end{aligned}$$

It has the property that, for every $K \in \{0,1\}^k$, $E(K,\cdot)$ is invertible. Its inverse is denoted by E^{-1} and given $K \in \{0,1\}^k$, one must have $E\left(K, E^{-1}(K,P)\right) = E^{-1}(K, E(K,P)) = P$.

In this thesis, we will focus on permutations. A permutation $\mathcal{P}$ is an invertible function:

$$\begin{aligned} \mathcal{P} : \{0,1\}^b &\longrightarrow \{0,1\}^b \\ X \quad &\longrightarrow \ Y\,. \end{aligned}$$

We denote the set of all b-bit permutations $\mathcal{P} : \{0,1\}^b \to \{0,1\}^b$ by $\texttt{Perm}(b)$. Strictly seen, a block cipher with a fixed key is a permutation. In the other direction, the Even-Mansour construction [EM91, EM97] turns generically a permutation into a block cipher. We will not analyze concrete permutations, but rather consider them as *ideal primitives*. This means that in the security proofs, the underlying permutation will be sampled as $\mathcal{P} \xleftarrow{\$} \texttt{Perm}(b)$ at the beginning of the security experiments.

In Section 2.4, we will discuss cryptographic constructions, among them hash functions. A hash function may also be considered as a primitive, but in this thesis we mostly treat them at a higher level. This is because we model their underlying building blocks as ideal primitives, and focus on analyzing the properties of the underlying hash function constructions.

2.3 Security Games, Adversaries, and Indistinguishability

In cryptography, the security of a scheme is typically argued via a *security game*, an interactive experiment between an adversary $\mathcal{A}$ and a collection of oracles $\mathcal{O}$. The goal of the adversary is to violate a claimed property, and we write $\mathcal{A}^{\mathcal{O}} \to o$ to indicate that, after its interaction with $\mathcal{O}$, the adversary outputs o. In this thesis, we consider permutation-based schemes where the underlying permutations are modeled as uniformly random. As a result, we assume adversaries to be *information-theoretic*, meaning that their computational power is unbounded, and their complexity is measured solely by the number of oracle queries they make. A *distinguisher* is a special type of adversary that outputs a decision bit. Without loss of generality, we will assume that adversaries are deterministic and do not query the same oracle on the same input.

Indistinguishability. Indistinguishability is a fundamental concept in cryptography that forms the basis of many security definitions. It captures the idea that an adversary should not be able to distinguish a real object from an ideal one.

To formalize this, we consider a cryptosystem $\mathcal{C}$ that relies on a randomized primitive $\mathcal{F}$, denoted by $\mathcal{C}^{\mathcal{F}}$. The actual behavior of the scheme, including how it uses $\mathcal{F}$, is modeled by a collection of oracles denoted by W_R, representing the *real world*. For instance, in the permutation-based constructions studied in this thesis, $\mathcal{F}$ is a random permutation that the adversary can directly query (in both directions), and $\mathcal{C}$ a sponge-like construction built from it. The *ideal world*, denoted by W_I, is defined by the security model and represents the idealized target functionality that $\mathcal{C}$ aims to fulfill (see Section 2.4 for concrete examples of ideal worlds).

In a distinguishing experiment, the adversary is a *distinguisher* and is given access to either W_R or W_I, chosen at random. Its task is to determine which world it is interacting with. The advantage of a distinguisher in telling the two worlds apart is formalized in the following definition.

Definition 2.3.1. *Let W_R, W_I be two collections of oracles, and $\mathcal{A}$ a distinguisher. The distinguishing advantage of $\mathcal{A}$ is defined as follows:*

$$\Delta_{\mathcal{A}}\left(W_R\ ;\ W_I\right) = \left|\mathbf{Pr}\left(\mathcal{A}^{W_R} \to 1\right) - \mathbf{Pr}\left(\mathcal{A}^{W_I} \to 1\right)\right| .$$

Random Oracles. A random oracle is an idealized primitive and is fundamental in many security models. The notion of a random oracle was formalized by Bellare and Rogaway [BR93] to facilitate the analysis of cryptographic schemes in the so-called random oracle model. A random oracle assigns to each input a uniformly random output, while ensuring consistency: if an input is queried more than once, the same output is returned each time. In its most general form, let

$$\mathcal{RO}^\infty : \{0,1\}^* \longrightarrow \{0,1\}^\infty ,$$

which assigns to every input string an infinite random bitstring. For convenience, we often work with the truncated version

$$\mathcal{RO} : \{0,1\}^* \times \mathbb{N}^* \longrightarrow \{0,1\}^* ,$$

which, on input (M, ν), returns $\mathcal{RO}^\infty(M)[1 : \nu]$. We refer to both $\mathcal{RO}^\infty$ and $\mathcal{RO}$ as random oracles. Several models work with restricted versions that accept fixed-length inputs and/or return outputs of fixed length; we also refer to these as random oracles.

In the indistinguishability frameworks considered in this thesis, random oracles will appear in the ideal worlds, and sometimes also in the real worlds as idealized primitives. They may appear under different names (such as random functions or online random oracle) depending on the domain, range, or input/output behavior required by the construction. However, these are all variants of the same concept: an oracle that provides random outputs, while ensuring consistency on repeated inputs.

Security Bounds, Tightness. Security bounds provide an upper limit on the probability that the adversary succeeds in breaking a cryptographic scheme. These bounds are typically expressed as a function of the adversary's resources (such as query complexity) and specific parameters of the construction (e.g., permutation size or key length). A security bound against property X is called *tight* if there exists a generic attack of property X with success probability close to the one of the security bound. Here, "close" means that constant or logarithmic factors are ignored. We will use the big-O notation $\tilde{\mathcal{O}}(\cdot)$ when interpreting security bounds.

2.4 Relevant Cryptographic Constructions

In this section, we introduce three fundamental types of constructions for symmetric-key cryptography, as well as their security goals. Here, we focus on security in the *ideal primitive* model: at the beginning of the game, the primitive underlying the construction is sampled uniformly at random from the set of primitives `Prim` as $\mathcal{F} \xleftarrow{\$} \texttt{Prim}$.

2.4.1 Hash Functions and Extendable Output Functions

A cryptographic hash function h takes as input an arbitrarily long message and produces a fixed-length digest:

$$\begin{aligned} h : \{0,1\}^* &\longrightarrow \{0,1\}^n \\ M \quad &\longrightarrow \quad Z\,. \end{aligned}$$

An extendable output function (XOF) h_X is more general than a hash function, as it supports arbitrarily long outputs:

$$\begin{aligned} h_X : \{0,1\}^* \times \mathbb{N}^* &\longrightarrow \{0,1\}^* \\ (M,\nu) \quad &\longrightarrow Z \in \{0,1\}^\nu\,. \end{aligned}$$

The security model that applies to a hash function/XOF varies depending on its intended use. We discuss in Section 2.4.1.1 several classical security requirements for hash functions, and in Section 2.4.1.2 we discuss indifferentiability, which is one of the most powerful security notions for arguing security of constructions. As a hash function is a special case of a XOF, in the following we will focus solely on XOFs, and the security models generalize directly to hash functions. We denote by $\mathcal{H}$ a XOF construction, and given $\mathcal{F} \in \texttt{Prim}$, $\mathcal{H}^{\mathcal{F}}$ denotes the obtained XOF.

2.4.1.1 Classical Security Goals

Traditionally, the security of XOFs is characterized by the following three fundamental properties, assuming a minimal output length $\nu \in \mathbb{N}^*$:

- Preimage resistance: given Z, it should be hard to find M such that $h_X(M, \nu) = Z$;
- Second preimage resistance: given M, it should be hard to find $M' \neq M$ such that $h_X(M, \nu) = h_X(M', \nu)$;
- Collision resistance: it should be hard to find $M \neq M'$ such that $h_X(M, \nu) = h_X(M', \nu)$.

These aforementioned definitions are rather informal and can yield ambiguity. Instead, we want formal definitions following the game-based framework from Section 2.3, where each security property is defined by an adversarial experiment and a precise success probability. Rogaway and Shrimpton [RS04] formalized 7 different notions for preimage, second preimage, and collision resistance. Below, we take their collision resistance and their notions of *everywhere* preimage and second preimage resistance, and adapt the terminology to fit our context.

Definition 2.4.1. *Let $\mathcal{H}$ be a XOF construction, with primitive set* `Prim`*. Let $\mathcal{F} \xleftarrow{\$}$* `Prim`*, and $\kappa, \nu \in \mathbb{N}^*$.*

- *The collision resistance of $\mathcal{H}$ against an adversary $\mathcal{A}$ is defined as*

$$\mathbf{Adv}_{\mathcal{H}}^{\mathrm{col}[\nu]}(\mathcal{A}) = \mathbf{Pr}\left(\mathcal{A}^{\mathcal{F}} \to (M, M') \quad \begin{array}{c} \textit{such that } M \neq M' \textit{ and} \\ \mathcal{H}^{\mathcal{F}}(M, \nu) = \mathcal{H}^{\mathcal{F}}(M', \nu) \end{array}\right) ;$$

- *The (everywhere) preimage resistance of $\mathcal{H}$ against an adversary $\mathcal{A}$ is defined as*

$$\mathbf{Adv}_{\mathcal{H}}^{\mathrm{ePre}[\nu]}(\mathcal{A}) = \max_{Z \in \{0,1\}^{\nu}} \mathbf{Pr}\left(\mathcal{A}^{\mathcal{F}}(Z) \to M \quad \textit{such that } \mathcal{H}^{\mathcal{F}}(M, \nu) = Z\right) ;$$

- *The (everywhere) second preimage resistance of $\mathcal{H}$ against an adversary $\mathcal{A}$ is defined as*

$$\mathbf{Adv}_{\mathcal{H}}^{\mathrm{eSec}[\kappa, \nu]}(\mathcal{A}) = \max_{M \in \{0,1\}^{\leq \kappa}} \mathbf{Pr}\left(\mathcal{A}^{\mathcal{F}}(M) \to M' \quad \begin{array}{c} \textit{such that } M \neq M' \textit{ and} \\ \mathcal{H}^{\mathcal{F}}(M, \nu) = \mathcal{H}^{\mathcal{F}}(M', \nu) \end{array}\right) .$$

Moreover, we define $\mathbf{Adv}_{\mathcal{H}}^{\mathrm{col}[\nu]}(q)$, $\mathbf{Adv}_{\mathcal{H}}^{\mathrm{ePre}[\nu]}(q)$, *and* $\mathbf{Adv}_{\mathcal{H}}^{\mathrm{eSec}[\kappa,\nu]}(q)$ *to be respectively the maximum of* $\mathbf{Adv}_{\mathcal{H}}^{\mathrm{col}[\nu]}(\mathcal{A})$, $\mathbf{Adv}_{\mathcal{H}}^{\mathrm{ePre}[\nu]}(\mathcal{A})$, *and* $\mathbf{Adv}_{\mathcal{H}}^{\mathrm{eSec}[\kappa,\nu]}(\mathcal{A})$, *over all adversaries allowed to make at most q primitive queries.*

Everywhere (second) preimage resistance captures security for any fixed challenge. Rogaway and Shrimpton additionally defined the *always* variants of (second) preimage resistance, where the challenge value itself is randomized rather than the underlying primitive. Those notions fall outside the ideal primitive model, hence we do not consider them.

The notion of (everywhere) preimage resistance on its own might have limited practical applications, as in most cases the challenge to invert comes from an evaluation on a secret random preimage. This corresponds to the notion "pre" in [RS04], and was later named *domain-oriented* preimage resistance by Andreeva and Stam [AS11]. We will adopt this terminology, and re-state the definition below.

Definition 2.4.2. *Let $\mathcal{H}$ be a XOF construction, with primitive set* `Prim`. *Let* $\mathcal{F} \xleftarrow{\$} \texttt{Prim}$, $\kappa, \nu \in \mathbb{N}^*$. *The domain-oriented preimage resistance of $\mathcal{H}$ against an adversary $\mathcal{A}$ is defined as*

$$\mathbf{Adv}_{\mathcal{H}}^{\mathrm{dPre}[\kappa,\nu]}(\mathcal{A}) = \mathbf{Pr}\begin{pmatrix} M \xleftarrow{\$} \{0,1\}^{\kappa}, \mathcal{A}^{\mathcal{F}}\left(\mathcal{H}^{\mathcal{F}}(M,\nu)\right) \to M' \\ \textit{such that } \mathcal{H}^{\mathcal{F}}(M,\nu) = \mathcal{H}^{\mathcal{F}}(M',\nu) \end{pmatrix}.$$

Moreover, we define $\mathbf{Adv}_{\mathcal{H}}^{\mathrm{dPre}[\kappa,\nu]}(q)$ *to be the maximum of* $\mathbf{Adv}_{\mathcal{H}}^{\mathrm{dPre}[\kappa,\nu]}(\mathcal{A})$, *over all adversaries allowed to make at most q primitive queries.*

As pointed by Andreeva and Stam [AS11], in general, everywhere preimage resistance does not imply domain-oriented preimage resistance, and vice versa.

2.4.1.2 Indifferentiability

Due to the wide range of applications in which XOFs are used, the aforementioned security properties are not enough for all of their applications. A well-known example is the Merkle-Damgård (MD) construction [Mer89, Dam89], which is one of the earliest generic constructions for building hash functions and has been used, for example, by the SHA-2 family [Nat15a]. Assuming that the underlying primitive (i.e., a compression function) is ideal, then the resulting hash function is collision and (second) preimage resistant.[1] However,

[1] In fact, existing results are stronger as they reduce the resistance of the hash function to that of the underlying compression function, sometimes under certain conditions [Dam89, Mer89, ANPS07]. However, this subtlety is not relevant to the current discussion.

it remains vulnerable to the *length extension attack* [Tsu92, CDMP05], which is problematic particularly in scenarios where secret keys are prepended to the message (see for instance Section 3.2.1). For further discussion of these issues, see [Men25].

A stronger security requirement for a XOF construction is to behave like a random oracle. Unfortunately, it is provably impossible to realize a genuine random oracle by any concrete hash function [CGH98, CGH04]. However, we want to "get as close as possible" to this ideal behavior; this can be done by assuming that the underlying primitive $\mathcal{F}$ is ideal, using the indistinguishability framework (cf., Definition 2.3.1). Since $\mathcal{F}$ is *public* in the real world, the adversary must be able to query it in W_R. This leads to the setup $W_R := \left(\mathcal{H}^{\mathcal{F}}, \mathcal{F}\right)$. On the other hand, the ideal world would replace the XOF $\mathcal{H}^{\mathcal{F}}$ with a random oracle $\mathcal{RO}$. A random oracle, by definition, does not rely on primitives, which raises the challenge of how to implement the primitive oracle in the ideal world.

To solve that, Maurer et al. [MRH04] introduced the notion of *indifferentiability*, which was refined in the context of hash functions by Coron et al. [CDMP05]. In this framework, the ideal-world counterpart for the public primitive $\mathcal{F}$ is a *simulator* $\mathbf{S}$. $\mathbf{S}$ is an algorithm that has oracle access to $\mathcal{RO}$ and with the same query interface as a primitive $\mathcal{F} \in \texttt{Prim}$. The goal of $\mathbf{S}$ is to simulate the behavior of an ideal primitive in order to make $W_I := \left(\mathcal{RO}, \mathbf{S}^{\mathcal{RO}}\right)$ indistinguishable from $W_R := \left(\mathcal{H}^{\mathcal{F}}, \mathcal{F}\right)$. Intuitively, $\mathbf{S}$ needs to (i) make sure that its answers are consistent with the ones of a random oracle (formalized in Definition 2.4.4) and (ii) ensure that its outputs have a distribution similar to the one of a random primitive. The setup is illustrated in Figure 2.1, and the formal definition is given below.

Definition 2.4.3. *Let $\mathcal{H}$ be a XOF construction, with primitive set* $\texttt{Prim}$. *Let $\mathcal{F} \xleftarrow{\$} \texttt{Prim}$. Let $\mathbf{S}$ be a simulator with the same interface as $\mathcal{F}$. The indifferentiability advantage of an adversary $\mathcal{A}$ against the simulator $\mathbf{S}$ is defined as follows:*

$$\mathbf{Adv}^{\text{indif}}_{\mathcal{H},\mathbf{S}}(\mathcal{A}) = \Delta_{\mathcal{A}}\left(\mathcal{H}^{\mathcal{F}}, \mathcal{F}\ ;\ \mathcal{RO}, \mathbf{S}^{\mathcal{RO}}\right) .$$

In the following we define $\mathcal{RO}$-consistency, which captures the fact that the simulator answers match the $\mathcal{RO}$ outputs. This is an essential property to guarantee indifferentiability of the construction.

Definition 2.4.4. *A simulator $\mathbf{S}$ for an iterated XOF construction is $\mathcal{RO}$-consistent if $\forall \nu \in \mathbb{N}^*, \forall M \in \{0,1\}^*$, whenever one can compute $\mathcal{H}^{\mathbf{S}}(M,\nu)$ from the queries received by $\mathbf{S}$, then it equals $\mathcal{RO}(M,\nu)$.*

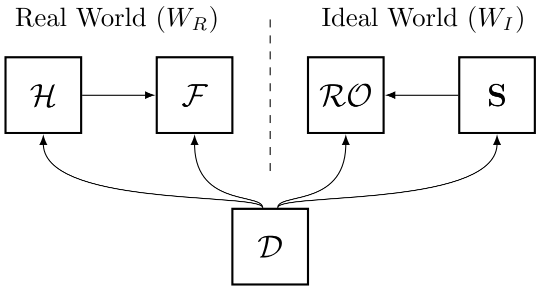


Figure 2.1: Illustration of the indifferentiability framework tailored for XOFs. Here $\mathcal{H}$ denotes a XOF construction, $\mathcal{F} \xleftarrow{\$} \texttt{Prim}$, $\mathcal{RO}$ is a random oracle, and $\mathbf{S}$ a simulator.

Note that the $\mathcal{RO}$-consistency for an indifferentiable construction can be guaranteed under certain conditions, e.g., as long as a bad event does not occur.

Indifferentiability and Composition. When proving the security of a hash-based cryptosystem, it is common to model the XOF as a random oracle. Indifferentiability then allows these proofs to be composed. This means that, under some conditions, instead of proving security of a cryptosystem $\mathcal{C}^{\mathcal{H}^{\mathcal{F}}}$, one may prove security of $\mathcal{C}^{\mathcal{RO}}$, and then compose the bound with the indifferentiability advantage of $\mathcal{H}$ to derive the security of $\mathcal{C}^{\mathcal{H}^{\mathcal{F}}}$.

However, there is a caveat: the adversary must be *single-stage*, meaning that the security game must be equivalent to another one with a single stateful adversary. This limitation was identified by Ristenpart et al. [RSS11], who gave a counter-example of a hash-based storage auditing scheme that, although secure in the random oracle model, is completely broken when instantiated with typical indifferentiable hash function constructions. To resolve this issue, they proposed the notion of *reset indifferentiability*, which allows composition with multi-stage adversaries. Yet, this more robust notion faces a fundamental limitation: Luykx et al. [LAMP12] showed that no domain extender could satisfy this notion.

Therefore, plain indifferentiability is still one of the most widely used frameworks, and the strongest achievable notion in practice. Looking ahead, in this thesis we will always consider adversaries that are single-stage.

Among others, indifferentiability guarantees the classical security properties from Section 2.4.1.1. Andreeva et al. [AMP10b, Appendix A] made this implication explicit, and we state their result for our use case below.

Lemma 2.4.1 ([AMP10b]). *Let $\mathcal{H}$ be a XOF construction with primitive set* $\texttt{Prim}$. *Let $\mathcal{RO}$ be a random oracle, $\mathcal{F} \xleftarrow{\$} \texttt{Prim}$, and $\mathbf{S}$ be a simulator with oracle access to $\mathcal{RO}$. Let $\kappa, \nu \in \mathbb{N}^*$. Let $\texttt{x} \in \{\mathrm{col}[\nu], \mathrm{ePre}[\nu], \mathrm{eSec}[\kappa, \nu], \mathrm{dPre}[\kappa, \nu]\}$, and let $\mathcal{A}$ be an $\texttt{x}$ adversary. There exists an indifferentiability adversary $\mathcal{A}'$ with respect to the simulator, and an $\texttt{x}$ adversary $\mathcal{A}''$ such that*

$$\mathbf{Adv}^{\texttt{x}}_{\mathcal{H}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{indif}}_{\mathcal{H},\mathbf{S}}(\mathcal{A}') + \mathbf{Adv}^{\texttt{x}}_{\mathcal{RO}}(\mathcal{A}'')\,.$$

Here, $\mathbf{Adv}^{\texttt{x}}_{\mathcal{RO}}(\mathcal{A}'')$ slightly abuses notation as $\mathcal{A}''$ gets direct access to $\mathcal{RO}$ and aims to break $\texttt{x}$ security. Moreover, $\mathcal{A}'$ has at most the same query complexity as $\mathcal{A}$, and $\mathcal{A}''$ has at most the same query complexity as $\mathcal{A}$ and $\mathbf{S}$ combined.

Note that, for an adversary $\mathcal{A}$ making q $\mathcal{RO}$ queries,

$$\mathbf{Adv}^{\mathrm{col}[\nu]}_{\mathcal{RO}}(\mathcal{A}) \leq \frac{\binom{q}{2}}{2^\nu}\,, \qquad \mathbf{Adv}^{\mathrm{eSec}[\kappa,\nu]}_{\mathcal{RO}}(\mathcal{A}) \leq \frac{q}{2^\nu}\,,$$
$$\mathbf{Adv}^{\mathrm{ePre}[\nu]}_{\mathcal{RO}}(\mathcal{A}) \leq \frac{q}{2^\nu}\,, \qquad \mathbf{Adv}^{\mathrm{dPre}[\kappa,\nu]}_{\mathcal{RO}}(\mathcal{A}) \leq \frac{q}{2^\nu} + \frac{q}{2^\kappa}\,.$$

2.4.2 Pseudorandom Functions

In this section we focus on a special type of keyed functions, so-called *pseudorandom functions* (PRFs). Note that this is a slight abuse of terminology, as the term "pseudorandom function" refers to the security goal.

Let us define a PRF construction F, parametrized further by a primitive $\mathcal{F} \in \texttt{Prim}$. $F^{\mathcal{F}}$ takes as input a key $K \in \{0,1\}^k$, a message $M \in \{0,1\}^*$, the output length $\nu \in \mathbb{N}^*$, and returns a binary string of length ν:

$$\begin{aligned} F^{\mathcal{F}} : \{0,1\}^k \times \{0,1\}^* \times \mathbb{N}^* &\longrightarrow \quad \{0,1\}^* \\ (K, M, \nu) \qquad\quad &\longrightarrow Z \in \{0,1\}^\nu\,. \end{aligned}$$

2.4.2.1 PRF security

PRF security is argued in the indistinguishability framework. A "good" PRF with a secret key K should be indistinguishable from a random oracle. We consider security in a multi-user setting, where $\mu \in \mathbb{N}^*$ instances of the PRF are used with different keys each.

Definition 2.4.5. *Let $\mu \in \mathbb{N}^*$. Let F be a keyed function construction, with primitive set* $\texttt{Prim}$. *Let $\mathcal{F} \xleftarrow{\$} \texttt{Prim}$. For $m \in [\![1, \mu]\!]$, denote by $F^{\mathcal{F}}_{K_m} : \{0,1\}^* \times$*

$\mathbb{N}^* \to \{0,1\}^*$ the specific instance of $F^{\mathcal{F}}$ of user m. The multi-user security of F against an adversary $\mathcal{A}$ is defined as

$$\mathbf{Adv}_F^{\mu\text{-prf}}(\mathcal{A}) = \Delta_{\mathcal{A}}\left((F_{K_m}^{\mathcal{F}})_{m=1}^{\mu}, \mathcal{F} \,;\, (\mathcal{RO}_m)_{m=1}^{\mu}, \mathcal{F}\right) ,$$

where $\mathcal{RO}_1, \ldots, \mathcal{RO}_\mu$ are random oracles. Without loss of generality, we can assume that the adversary makes at least one construction query for every user, since otherwise we can consider the multi-user setting with a smaller number of users.

Note that in this definition, the instance of one specific user is left vague on purpose. Looking ahead, we will assume that users' keys are sampled uniformly at random, and that user instances differ only in their keys and, sometimes, in their initial values. We will also consider PRF constructions where the output length is fixed, akin to hash functions that are restrictions of XOFs.

2.4.2.2 Message Authentication Codes

One prominent use case of PRFs is *message authentication codes* (MACs). Let $t \in \mathbb{N}^*$. A MAC, potentially based on a primitive $\mathcal{F}$, comprises two algorithms, indexed by a key $K \in \{0,1\}^k$: one for generating tags

$$\begin{aligned} \mathcal{M}_K^{\mathcal{F}} : \{0,1\}^* &\longrightarrow \{0,1\}^t \\ M &\longrightarrow T , \end{aligned}$$

and one verification algorithm

$$\begin{aligned} \mathcal{V}_K^{\mathcal{F}} : \{0,1\}^* \times \{0,1\}^t &\longrightarrow \{\top, \bot\} \\ (M,T) &\longrightarrow d . \end{aligned}$$

It has the property that for any $M \in \{0,1\}^*$, $\mathcal{V}_K^{\mathcal{F}}(M, \mathcal{M}_K^{\mathcal{F}}(M)) = \top$. In the associated security game, the adversary has oracle access to $\mathcal{M}_K^{\mathcal{F}}$ and $\mathcal{V}_K^{\mathcal{F}}$, where $\mathcal{F} \xleftarrow{\$} \texttt{Prim}$. The adversary is never allowed to query $\mathcal{V}_K^{\mathcal{F}}$ with an input (M,T) that corresponds to a past query $\mathcal{M}_K(M) = T$. The adversary is said to produce a *forgery* if it makes a query to $\mathcal{V}_K$ that returns $\top$.

Definition 2.4.6. *Let $\mathcal{MAC} = (\mathcal{M}, \mathcal{V})$ be a MAC construction, with primitive set* `Prim`. *Let $\mathcal{F} \xleftarrow{\$}$* `Prim`. *The advantage of the adversary $\mathcal{A}$ is defined as follows:*

$$\mathbf{Adv}_{\mathcal{MAC}}^{\mathrm{MAC}}(\mathcal{A}) = \mathbf{Pr}\left(\mathcal{A}^{\mathcal{M}_K^{\mathcal{F}}, \mathcal{V}_K^{\mathcal{F}}, \mathcal{F}} \textit{ forges}\right) .$$

A well-known result is that PRF security implies MAC security [GGM84, BKR94]. More formally, let $\mathcal{A}$ be an adversary that makes q_v queries to $\mathcal{V}_K^{\mathcal{F}}$. There exists a distinguisher $\mathcal{A}'$ against the PRF security of $\mathcal{M}$ in the single-user setting with a most the same query complexity as $\mathcal{A}$ such that

$$\mathbf{Adv}_{\mathcal{MAC}}^{\mathrm{MAC}}(\mathcal{A}) \leq \mathbf{Adv}_{\mathcal{M}}^{\text{1-prf}}(\mathcal{A}') + \frac{q_v}{2^t}\,.$$

2.4.3 Authenticated Encryption

Authenticated encryption schemes allow to encrypt and authenticate the plaintext at the same time. They often also offer the possibility to authenticate some associated data that is not meant to be secret, but only authentic. Those schemes are referred to as *Authenticated Encryption with Associated Data* (AEAD). We focus on *nonce-based* schemes, meaning that security relies on the fact that a *nonce*, a public input, never repeats.[2] Furthermore, all schemes we consider produce a ciphertext whose length matches the plaintext and output the authentication tag separately.

Let $k, n, t \in \mathbb{N}^*$. Consider an AEAD construction $\mathbf{AE}$, parametrized further by a primitive $\mathcal{F} \in \mathtt{Prim}$. It consists of two algorithms: the encryption algorithm $\mathbf{Enc}^{\mathcal{F}}$ and decryption algorithm $\mathbf{Dec}^{\mathcal{F}}$. Encryption $\mathbf{Enc}^{\mathcal{F}}$ takes as input a key $K \in \{0,1\}^k$, a nonce $N \in \{0,1\}^n$, associated data $A \in \{0,1\}^*$, plaintext $P \in \{0,1\}^*$, and it outputs a ciphertext $C \in \{0,1\}^*$ with $|C| = |P|$ and a tag $T \in \{0,1\}^t$, so that

$$\begin{aligned} \mathbf{Enc}^{\mathcal{F}} : \{0,1\}^k \times \{0,1\}^n \times \{0,1\}^* \times \{0,1\}^* &\longrightarrow \{0,1\}^* \times \{0,1\}^t \\ (K, N, A, P) &\longrightarrow (C, T)\,. \end{aligned}$$

The corresponding decryption function $\mathbf{Dec}^{\mathcal{F}}$ takes as input a key $K \in \{0,1\}^k$, a nonce $N \in \{0,1\}^n$, associated data $A \in \{0,1\}^*$, a ciphertext $C \in \{0,1\}^*$, and a tag $T \in \{0,1\}^t$, and it outputs either $P \in \{0,1\}^*$ with $|P| = |C|$ if verification with the tag is correct or a failure symbol $\perp$, so that

$$\begin{aligned} \mathbf{Dec}^{\mathcal{F}} : \{0,1\}^k \times \{0,1\}^n \times \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^t &\longrightarrow \{0,1\}^* \cup \{\perp\} \\ (K, N, A, C, T) &\longrightarrow P \text{ or } \perp\,. \end{aligned}$$

It has the property that, for any $K \in \{0,1\}^k$, $N \in \{0,1\}^n$, $A, P \in \{0,1\}^*$, if $\mathbf{Enc}^{\mathcal{F}}(K, N, A, P) = (C, T)$, then $\mathbf{Dec}^{\mathcal{F}}(K, N, A, C, T) = P$.

[2] As we will see later, some schemes may still offer limited guarantees (typically only authenticity) even if the nonce is reused.

2.4.3.1 AEAD Security

An AEAD scheme aims to ensure both confidentiality and authenticity in the nonce-respecting setting. Informally, we require that: (i) ciphertexts reveal no information about the plaintexts, except for their length (confidentiality), and (ii) it is infeasible to forge a ciphertext that decrypts to a valid message (authenticity). The first to formally study this notion were Bellare and Namprempre [BN00, BN08], though in a left-or-right setting where the adversary receives the encryption of either M_0 or M_1. Shrimpton [Shr04] introduced the notion of IND-CCA3 security, which at a high level gives the adversary access to either the encryption and decryption functionality, or to a random oracle that always outputs random responses of expected length and a $\perp$-function that always returns the $\perp$-sign.

In this thesis, we will adopt Shrimpton's IND-CCA3 security notion, referring to it simply as AEAD security, and adapt it to our terminology.

Definition 2.4.7. *Let $\mu \in \mathbb{N}^*$. Let $\mathbf{AE} = (\mathbf{Enc}, \mathbf{Dec})$ be an AEAD construction, with primitive set* `Prim`. *Let $\mathcal{F} \xleftarrow{\$}$* `Prim`. *For $m \in [\![1, \mu]\!]$, denote by $\left(\mathbf{Enc}_{K_m}^{\mathcal{F}}, \mathbf{Dec}_{K_m}^{\mathcal{F}}\right)$ the specific instance of user m.*

The multi-user security of $\mathbf{AE}$ against an adversary $\mathcal{A}$ is defined as

$$\mathbf{Adv}_{\mathbf{AE}}^{\mu\text{-ae}}(\mathcal{A}) = \Delta_{\mathcal{A}}\left(\left(\mathbf{Enc}_{K_m}^{\mathcal{F}}, \mathbf{Dec}_{K_m}^{\mathcal{F}}\right)_{m=1}^{\mu}, \mathcal{F} \,;\, \left(\$_m, \perp\right)_{m=1}^{\mu}, \mathcal{F}\right), \tag{2.1}$$

where $\$_1, \ldots, \$_\mu$ are random functions that for each new input (N, A, P) outputs a tuple of two random strings of size $|P|$ and t bits, respectively. The function $\perp$ returns the failure symbol $\perp$ for each query.

Moreover, $\mathcal{A}$ is not allowed to make a decryption query on input of the result of an earlier encryption query.

An adversary $\mathcal{A}$ is called *nonce-respecting* if it is never allowed to make two distinct encryption queries that contain the same nonce.

2.4.3.2 Separation Into Confidentiality and Authenticity

Sometimes, it will be more convenient to separate the analysis of the distance (2.1) into confidentiality and authenticity:

$$\mathbf{Adv}_{\mathbf{AE}}^{\mu\text{-conf}}(\mathcal{A}) = \Delta_{\mathcal{A}}\left(\left(\mathbf{Enc}_{K_m}^{\mathcal{F}}\right)_{m=1}^{\mu}, \mathcal{F} \,;\, \left(\$_m\right)_{m=1}^{\mu}, \mathcal{F}\right), \tag{2.2}$$

$$\mathbf{Adv}_{\mathbf{AE}}^{\mu\text{-auth}}(\mathcal{A}) = \mathbf{Pr}\left(\mathcal{A}^{\left(\mathbf{Enc}_{K_m}^{\mathcal{F}}, \mathbf{Dec}_{K_m}^{\mathcal{F}}\right)_{m=1}^{\mu}, \mathcal{F}} \text{ forges}\right), \tag{2.3}$$

where for authenticity, we say that $\mathcal{A}$ "forges" if it ever makes a query to one of its decryption oracles that is successful and that is not the result of an earlier encryption query. The same remarks on the randomness and the behavior of $\mathcal{A}$ as in (2.1) apply.

Note that authenticity can be equivalently stated as a distance, assuming the same restrictions on relaying queries:

$$\Delta_{\mathcal{A}}\left(\left(\mathbf{Enc}^{\mathcal{F}}_{K_m}, \mathbf{Dec}^{\mathcal{F}}_{K_m}\right)^{\mu}_{m=1}, \mathcal{F} \,;\, \left(\mathbf{Enc}^{\mathcal{F}}_{K_m}, \perp\right)^{\mu}_{m=1}, \mathcal{F}\right) .$$

From this, we can easily conclude that AEAD security implies confidentiality and authenticity. We repeat the reduction as given by Shrimpton [Shr04]. Let $\mathcal{A}$ be any adversary against the AEAD security of $\mathbf{AE}$. Then, by the triangle inequality,

$$\begin{aligned}
\mathbf{Adv}^{\mu\text{-ae}}_{\mathbf{AE}}(\mathcal{A}) &= \Delta_{\mathcal{A}}\left(\left(\mathbf{Enc}^{\mathcal{F}}_{K_m}, \mathbf{Dec}^{\mathcal{F}}_{K_m}\right)^{\mu}_{m=1}, \mathcal{F} \,;\, (\$_m, \perp)^{\mu}_{m=1}, \mathcal{F}\right) \\
&\leq \Delta_{\mathcal{A}}\left(\left(\mathbf{Enc}^{\mathcal{F}}_{K_m}, \mathbf{Dec}^{\mathcal{F}}_{K_m}\right)^{\mu}_{m=1}, \mathcal{F} \,;\, \left(\mathbf{Enc}^{\mathcal{F}}_{K_m}, \perp\right)^{\mu}_{m=1}, \mathcal{F}\right) \\
&+ \Delta_{\mathcal{A}}\left(\left(\mathbf{Enc}^{\mathcal{F}}_{K_m}, \perp\right)^{\mu}_{m=1}, \mathcal{F} \,;\, (\$_m, \perp)^{\mu}_{m=1}, \mathcal{F}\right) \\
&\leq \mathbf{Adv}^{\mu\text{-auth}}_{\mathbf{AE}}(\mathcal{A}') + \mathbf{Adv}^{\mu\text{-conf}}_{\mathbf{AE}}(\mathcal{A}'') , \qquad (2.4)
\end{aligned}$$

for some adversaries $\mathcal{A}'$ and $\mathcal{A}''$ with the same query complexities as $\mathcal{A}$. This separation allows for a modular proof, but it comes at the price of some terms in the bounds being counted twice.

Looking ahead, this separation will be useful when looking at refined security guarantees. More precisely, the schemes considered in this thesis may achieve authenticity in adversarial models stronger than nonce-respecting ones (e.g., nonce-misuse, release of unverified plaintext, or state recovery, refer to Chapter 9).

2.5 Technical Tools

In this section, we describe two tools that will be useful for the security proofs in this thesis.

2.5.1 H-Coefficient Technique

The H-Coefficient technique by Patarin [Pat91, Pat08b], and modernized by Chen and Steinberger [CS14] is a powerful and widely used technique to upper bound a distinguishing advantage.

In indistinguishability-based games, the interaction between the distinguisher $\mathcal{A}$ and the world W_R/W_I can be summarized in a transcript τ, which contains tuples of query-responses. In order to reduce the analysis to a simpler combinatorial problem, one may extend the transcript *at the end of the interaction.* Typically, in the real world of an iterated construction, this is done by adding evaluations of the building blocks necessary to compute the outputs (including the keys), while in the ideal world, the evaluations would be replaced by dummy random values.

A transcript τ is called *attainable* if $\mathbf{Pr}\left(\mathcal{A}^{W_I} \text{ generates } \tau\right) > 0$, meaning that it can occur during the interaction with W_I. We denote by $\mathcal{T}$ the set of attainable transcripts.

The H-coefficient technique allows to split the analysis into two, and is described in the following lemma.

Lemma 2.5.1 (H-Coefficient Technique [Pat91, Pat08b, CS14])**.** *Let $\mathcal{A}$ be a deterministic distinguisher. Let $\mathcal{T} = \mathcal{T}_{good} \sqcup \mathcal{T}_{bad}$ be a partition of $\mathcal{T}$ into good and bad transcripts. If there exist $\epsilon_1, \epsilon_2 \geq 0$ such that*

$$\begin{aligned} \forall \tau \in \mathcal{T}_{good},\ \frac{\mathbf{Pr}\left(\mathcal{A}^{W_R} \textit{ generates } \tau\right)}{\mathbf{Pr}\left(\mathcal{A}^{W_I} \textit{ generates } \tau\right)} &\geq 1 - \epsilon_1 \,, \\ \textit{and} \quad \mathbf{Pr}\left(\mathcal{A}^{W_I} \textit{ generates } \tau \in \mathcal{T}_{bad}\right) &\leq \epsilon_2 \,, \end{aligned}$$

then

$$\Delta_{\mathcal{A}}\left(W_R \,;\, W_I\right) \leq \epsilon_1 + \epsilon_2 \,.$$

A proof of this lemma can be found in [CS14]. The roles of the worlds W_R and W_I can be reverted, but analyzing the probabilities of $\tau \in \mathcal{T}_{bad}$ in the ideal world is typically simpler.

The set $\mathcal{T}_{good}$ is "good" in the sense that we expect the ratio to be close to 1. In this thesis, this will mean that as long as no bad event occurs, the statistical distance between W_R and W_I is negligible for transcripts from $\mathcal{T}_{good}$. The set $\mathcal{T}_{bad}$ can be characterized by *bad events*, defined over the (extended) transcript. Those bad events capture degenerate behavior from the side of the construction (e.g., adversary guesses an internal secret state, or collisions).

The H-coefficient technique can be seen as a generalization of the fundamental lemma of game playing [BR06]. Conversely, there exists further generalizations of this technique, such as the expectation method [HT16], where ϵ_1 may be dependent on the transcript, but we will not use those in the thesis.

2.5.2 A Bins-and-Balls Result

Let $b, r, c \in \mathbb{N}^*$ with $b = r + c$. Looking ahead, a tool central to several security proofs in this thesis is the concept of multicollisions: given a set $\mathcal{S}$ of q elements sampled (with or without replacement) from $\{0,1\}^b$, it should be hard to find too many elements in $\mathcal{S}$ that all collide on their outer r bits. The problem reduces to a well-known bins-and-balls experiment: given q balls thrown randomly into R bins, we want to determine the expected maximum load. This problem has been extensively studied (e.g., [JK77, Gon81, RS98]), but those bounds are asymptotic. Our applications, however, require precise upper bounds, we thus turn to more recent results made in the context of provable security.

To tame multicollisions, there exist two main strategies. One of them was first used in the context of sponges by Jovanovic et al. [JLM14]: they use a simple Stirling approximation to bound the event that a multicollision exceeds a value θ, and subsequently reason on the mode's security under the assumption that the multicollision is at most θ. Omitting details, this results in the following strategy:

$$\mathbf{Pr}\left(\mathbf{success}\right) \leq \mathbf{Pr}\left(\mathbf{success} \mid \mathbf{mult} \leq \theta\right) + \mathbf{Pr}\left(\mathbf{mult} > \theta\right) . \tag{2.5}$$

In a follow-up work, Jovanovic et al. [JLM+19] improved the multicollision bounds by performing a case distinction depending on the values (r, c), thus improving the second probability of this equation. Daemen et al. [DMV17] introduced the multicollision limit function as a definition specifically tailored towards the sponge/duplex. In a bit more detail, they observed that for a sponge-/duplex-based analysis, the left probability is often of the form $\theta q/2^c$, where q is the number of permutation queries. Therefore, by defining θ such that the right probability is of the form $\theta/2^c$, it can be subsumed within the first term to get a joint term of the form $\theta(q+1)/2^c$.

An alternative approach is to only compute the *expected* size of the maximum multicollision, $\mathsf{E}\left(\mathbf{mult}\right)$, and then bound using the following strategy:

$$\mathbf{Pr}\left(\mathbf{success}\right) = \sum_{\theta} \mathbf{Pr}\left(\mathbf{success} \mid \mathbf{mult} = \theta\right) \mathbf{Pr}\left(\mathbf{mult} = \theta\right) . \tag{2.6}$$

The expectation on **mult** can then be used, observing that for a sponge-/duplex-based analysis, the left probability is often linear in θ (e.g., of the form $\theta q/2^c$ as mentioned above). Choi et al. [CLL19] provided bounds in the context of proving indifferentiability of truncated permutations. Their bound is general (it does not consider different parameter settings). Chakraborty et al. [CDN23b] used the same approach for their security analysis of the AEAD mode of Ascon, but with a fine-tuned bound that distinguishes different parameter setups. Their proof technique basically refines the one of Chakraborty et al. [CJN20]. Depending on the parameter setting, one bound may be better than the other one, but the tightest upper bound applies.

We remark that above two approaches to bound multicollisions are incompatible, but in many cases, one can replace the other. Typically, the expectation approach is tighter, and we will use this approach in the thesis. We slightly improve upon the result [CLL19, page 187] by removing a logarithmic factor and making the argument explicit when sampling is done without replacement. We present both bounds on the expected value of the maximum multicollision size in the following lemma.

Lemma 2.5.2. *Let $q, b, r \in \mathbb{N}^*$ such that $r \leq b$. Suppose we sample q distinct elements uniformly with replacement from $\{0,1\}^b$. Let $\mathcal{S}$ denote the resulting multiset of sampled values. Define*

$$\mathrm{mucol}_r(\mathcal{S}) = \max_{T \in \{0,1\}^r} |\{S \in \mathcal{S} \,:\, \mathrm{outer}_r(S) = T\}| \,,$$

and let $\mathrm{mucol}(q, 2^r) = \mathsf{E}(\mathrm{mucol}_r(\mathcal{S}))$. *We have*

$$\mathrm{mucol}(q, 2^r) \leq \begin{cases} 3 & \text{if } 4 \leq q \leq \sqrt{2^r}\,, \\ \frac{4\log_2(q)}{\log_2(\log_2(q))} & \text{if } \sqrt{2^r} < q \leq 2^r\,, \\ 5r\lceil \frac{q}{r2^r} \rceil & \text{if } 2^r < q\,, \end{cases}$$

$$\text{and } \mathrm{mucol}(q, 2^r) \leq \frac{2q}{2^r} + 3\ln(2^r) + 4\,.$$

Moreover, the result also holds when the values from $\mathcal{S}$ are sampled without replacement in $\{0,1\}^b$.

Proof. A proof of the first bound can be found in [CDN23b]. The current proof focuses on the second inequality. Let $R = 2^r$, $p = \frac{1}{R}$, $T \in \{0,1\}^r$, and denote by $X^{(T)} = |\{S \in \mathcal{S} \,:\, \mathrm{outer}_r(S) = T\}|$. It is clear that $X^{(T)}$ follows a binomial law with parameters p and q. Therefore, we can use the Chernoff bound, so that for any $j \geq 2pq$,

$$\mathbf{Pr}\left(X^{(T)} \geq j\right) \leq e^{-\frac{j-pq}{3}}\,.$$

Finally,

$$\begin{aligned}
\mathrm{mucol}(q,R) &= \mathsf{E}\left(\max_T X^{(T)}\right) \\
&= \sum_{j\geq 1} \mathbf{Pr}\left(\max_T X^{(T)} \geq j\right) \\
&\leq 2pq + 3\ln(R) + \sum_{j=2pq+3\ln(R)}^{q} \mathbf{Pr}\left(\bigvee_T X^{(T)} \geq j\right) \\
&\leq 2pq + 3\ln(R) + \sum_{t=1}^{R} \sum_{j=2pq+3\ln(R)}^{q} e^{-\frac{j-pq}{3}} \\
&\leq 2pq + 3\ln(R) + R\cdot e^{\frac{pq}{3}} \cdot \frac{e^{-\frac{2pq+3\ln(R)}{3}} - e^{\frac{-q-1}{3}}}{1-e^{-\frac{1}{3}}} \\
&\leq 2pq + 3\ln(R) + 4R\cdot e^{-\frac{pq}{3}} e^{-\ln(R)} \\
&\leq \frac{2q}{R} + 3\ln(R) + 4\,.
\end{aligned}$$

Moreover, when the sampling is performed without replacement, we can use [Hoe94, Theorem 4], which states that for any continuous and convex function,

$$\mathsf{E}\left(f\left(X^{(T)}\right)\right) \leq \mathsf{E}\left(f\left(Y^{(T)}\right)\right)\,,$$

where $Y^{(T)}{\sim}\mathrm{Binomial}(p,q)$. In particular, this holds when $f(x) = e^{t\cdot x}$ for any $t > 0$. Because the Chernoff bound is obtained by upper bounding $\mathsf{E}\left(e^{t\cdot X^{(T)}}\right)$, the proof also carries over to this case. $\square$

Those bounds will be heavily used in the proofs of sponge-based designs. Looking ahead, whenever possible, the bounds that include multicollisions will be presented in an abstract form, i.e., using the term mucol(q,R).

Chapter 3

The Sponge Construction and Its Variants

This chapter provides an overview of the sponge construction and its variants. Section 3.1 presents the sponge construction and its key security properties, Section 3.2 covers several sponge-based PRFs, and Section 3.3 discusses various sponge-based AEAD schemes.

3.1 The Sponge Construction

In this section, we describe the sponge construction of Bertoni et al. [BDPV07]. In Section 3.1.1 we provide a detailed description, in Section 3.1.2 we give its security bounds for the properties discussed in Section 2.4.1, along with generic attacks in Section 3.1.3.

3.1.1 Description

Let $b, c, r \in \mathbb{N}^*$ be such that $b = r + c$. Consider a padding function

$$\begin{aligned} pad_r : \{0,1\}^* &\longrightarrow (\{0,1\}^r)^* \\ M \quad &\longrightarrow M_1 \| \cdots \| M_u \,, \end{aligned}$$

which injectively splits the input into $u \geq 1$ r-bit blocks, such that the last message block is different from 0^r. A padding function satisfying this property is called *sponge-compliant* [BDPV11b, Definition 1]. Let $unpad_r$ denote the inverse of pad_r, which returns $\perp$ if the input does not correspond to a valid padding.

A minimal example of a sponge-compliant padding is the 10^*-padding, denoted by $pad_r^{10^*}$, which appends a 1 to the input message, and as many zeros as necessary to obtain a message of length multiple of r bits.

The sponge construction allows to build a XOF from a permutation $\mathcal{P} \in \mathtt{Perm}(b)$:[1]

$$\begin{aligned} \mathrm{Sponge}^{\mathcal{P}} : \{0,1\}^* \times \mathbb{N}^* &\longrightarrow \{0,1\}^* \\ (M, \nu) \quad &\longrightarrow Z \in \{0,1\}^{\nu} . \end{aligned} \tag{3.1}$$

Let $M \in \{0,1\}^*$ be an input message, $\nu \in \mathbb{N}^*$ be the requested output length, and $\ell := \lceil \frac{\nu}{r} \rceil$. The sponge operates as follows:

1. *Initialization.* M is first padded into message blocks using pad_r: $M_1 \| \cdots \| M_u \leftarrow pad_r(M)$. The state S is initialized as 0^b;
2. *Absorbing phase.* At the ith iteration, for $i = 1, \ldots, u$, the state is updated as $S \leftarrow \mathcal{P}(S \oplus (M_i \| 0^c))$;
3. *Squeezing phase.* At the ith iteration, for $i = 1, \ldots, \ell$, the outer r bits of S are extracted as $Z_i \leftarrow \mathrm{outer}_r(S)$ and the state is updated as $S \leftarrow \mathcal{P}(S)$;
4. The output is computed as $Z \leftarrow (Z_1 \| \cdots \| Z_\ell)[1 : \nu]$.

Generalizations. We will consider the following slight generalizations in the sponge:

- The squeezing rate r' may differ from the absorbing rate r, with $r' \geq r$. This variant was introduced by the designers of PHOTON [GPP11];
- The rate of the first absorption call r'' may differ from r with $r'' \geq r$, as suggested by Naito and Ohta [NO14]. In this case, define $pad_{r'',r}(M)$ as a sponge-compliant padding function that splits the input message into a first block of r'' bits, followed by (optional) blocks of r bits each;
- The initial state is set to an arbitrary value $IV \in \{0,1\}^b$. In some cases (e.g., Chapter 8), we analyze security in the presence of different IVs.

This generalized construction is illustrated in Figure 3.1 and formally described in Algorithm 1. For simplicity, we will continue to refer to it as the "sponge" throughout.

[1]More generally, the sponge can be built over any b-bit function, but in practice it is almost always defined using a permutation, since designing permutations is typically easier than non-invertible functions.

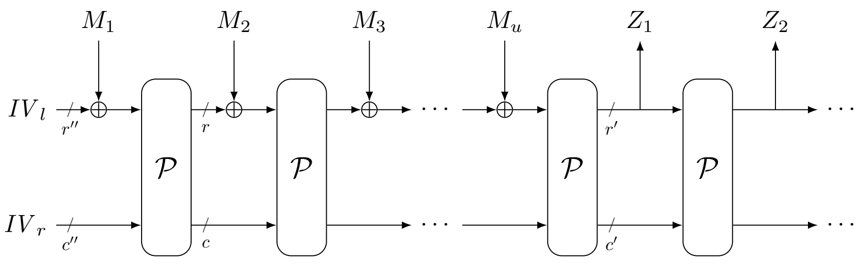


Figure 3.1: Sponge construction. The input message M is padded as $M_1 \| \cdots \| M_u \leftarrow pad_r(M)$. Here, the initial value $IV \in \{0,1\}^b$ is split as $IV_l \| IV_r$ with $IV_l \in \{0,1\}^{r''}$, $IV_r \in \{0,1\}^{c''}$.

Algorithm 1 Sponge construction.

```
1:  function Sponge$^{\mathcal{P}}(M, \nu)$
2:      // Initialization
3:      $S \leftarrow IV$; // State of the sponge, $IV$ is any fixed $b$-bit string
4:      $Z \leftarrow \epsilon$; // Output string
5:      $M_1 \| \cdots \| M_u \leftarrow pad_{r'',r}(M)$
6:      // Absorption
7:      $S \leftarrow \mathcal{P}(S \oplus (M_1 \| 0^{c''}))$
8:      for $i = 2, \ldots, u$ do
9:          $S \leftarrow \mathcal{P}(S \oplus (M_i \| 0^{c}))$
10:     // Squeezing
11:     for $i = 1, \ldots, \lceil \frac{\nu}{r'} \rceil$ do
12:         $Z \leftarrow Z \| \mathrm{outer}_{r'}(S)$
13:         $S \leftarrow \mathcal{P}(S)$
14:     return $Z[1:\nu]$
```

3.1.2 Security Bounds

In this section, we systematize the existing security bounds of the sponge in terms of indifferentiability, collision, and (second) preimage resistance.

3.1.2.1 Indifferentiability

In the indifferentiability setting, the real world is implemented by $W_R := \big(\text{Sponge}^{\mathcal{P}}, \mathcal{P}\big)$, where $\mathcal{P} \xleftarrow{\$} \texttt{Perm}(b)$. Before presenting the indifferentiability results on the sponge [BDPV08], we first focus on the metrics used for adversarial queries.

Counting Queries. The way adversarial queries are counted in the indifferentiability proof [BDPV08] is natural and works well with composition. Given an adversary $\mathcal{A}$, we measure its complexity by counting the total number of *distinct* calls to $\mathcal{P}$ that would be required in the real world, either due to direct primitive queries or through the internal evaluations induced by construction queries. This total count is denoted by $\mathcal{N}$. One might worry that $\mathcal{N}$ could be ambiguous in the following situations:

- The adversary might guess an intermediate state that the simulator/sponge construction has generated internally without making the queries in order;
- Internal collisions between sponge states might occur, which would lead to two distinct queries having the same internal permutation evaluations.

These situations might seem problematic as the adversary may not detect the occurrence of those events, hence it cannot deduce the value of $\mathcal{N}$. However, such events can only reduce the actual value of $\mathcal{N}$, so that ignoring them still yields a valid upper bound on $\mathcal{N}$. Moreover, these events are formally treated as *bad events* in the proof. When they occur, the simulator may abort and/or produce inconsistent outputs that can be detected. As a result, this method of counting is meaningful: as long as no bad event occurs, the adversary's query history precisely determines the value of $\mathcal{N}$; and even in the presence of bad events, counting from the query history still provides an upper bound on $\mathcal{N}$.

Let us now consider the following example to illustrate how this counting works in practice (excluding the bad events).

Example 3.1.1. *Assume $r = r' = r''$, $IV = 0^b$, and suppose the inputs are already padded using the function $pad_r^{10^*}$. Consider an adversary making the following queries in this order:*

1. *Forward permutation query with input $1\|0^{b-1}$, output X_1: then $\mathcal{N}$ is incremented by one;*

2. *Construction query with input $(1\|0^{r-1}, 2r)$: this induces two permutation calls. However, the first evaluation has already been counted by the previous query, so $\mathcal{N}$ is incremented by one;*

3. *Forward permutation query with input X_1, output X_2: the evaluation has already been counted by the previous query, so $\mathcal{N}$ is not incremented;*

4. *Forward permutation query with input X_2, output X_3: then $\mathcal{N}$ is incremented by one;*

5. *Construction query with input $(1\|0^{r-1}, 3r)$: this counts as three permutation calls. However, the first evaluation has been counted by query 1, the second by query 2, and the third one by query 4, hence $\mathcal{N}$ is not incremented;*

6. *Construction query with input $(1\|0^{r-1}\|0^r\|1\|0^{r-1}, r)$: this counts as three permutation calls. However, the first two have been counted by queries 1 and 2, so $\mathcal{N}$ is incremented by one.*

Note that it is possible to unify the construction and the primitive queries this way due to the structure of the simulator. When a construction or primitive query does not increment $\mathcal{N}$, it means that either the adversary or the simulator repeats a random oracle query. In the proof of Lemma 2.4.1, the adversary attacking the random oracle internally runs the simulator. We can then assume, without loss of generality, that the adversary avoids making the same random oracle query more than once.

Remark 3.1.1. *Looking ahead, in Chapters 5 and 6, we will count primitive-originated queries and construction-originated queries separately. This is slightly less tight than the unified counting method presented above, but the resulting loss is at most a factor of 4 and does not affect the conclusions.*

Indifferentiability of the Sponge. Bertoni et al. [BDPV08] proved that the sponge construction (with $r' = r'' = r$) is indifferentiable from a random oracle. In detail, they proved a bound up to

$$1 - \prod_{i=0}^{\mathcal{N}-1} \left(\frac{1 - \frac{i+1}{2^c}}{1 - \frac{i}{2^{r+c}}} \right) ,$$

which then gets approximated to $\binom{\mathcal{N}+1}{2}/2^c$. However, this approximation uses the inequality $1-x \leq e^{-x}$ in two directions (first to lower bound, then to upper bound), de facto making this approximation a true approximation instead of a strict upper bound. An alternative approach would give a proper upper bound of the form $2\binom{\mathcal{N}+1}{2}/2^c$. We adopt their result but with this simplified upper bound.

Theorem 3.1.1 ([BDPV08])**.** *Let $b, c, r, \mathcal{N} \in \mathbb{N}^*$ with $b = r + c$. Consider the sponge construction of Algorithm 1 with parameters b, c, r and $r' = r'' = r$. Let $\mathcal{A}$ be an adversary with complexity $\mathcal{N}$. There exists a simulator $\mathbf{S}$ with complexity $\tilde{\mathcal{O}}(\mathcal{N})$ queries such that*

$$\mathbf{Adv}^{\mathrm{indif}}_{\mathrm{Sponge},\mathbf{S}}(\mathcal{A}) \leq \frac{\mathcal{N}(\mathcal{N}+1)}{2^c}\,. \tag{3.2}$$

Intuitively, the simulator in the proof $\mathbf{S}$ maintains a graph construction, where each node simulates an intermediate sponge state. When it receives a fresh inverse query, it replies with a random value in $\{0,1\}^b$.[2] On a fresh forward query, $\mathbf{S}$ checks whether the input corresponds to an intermediate state right before or during a squeezing phase. If so, $\mathbf{S}$ reconstructs the corresponding input message using the graph and queries this message to the random oracle. It then ensures consistency by setting the outer part of the output to match the random oracle's response, while choosing the inner part at random.[2] The simulator is successful to provide consistent answers as long as there are no inner collisions. Later proofs in Chapter 5 and Chapter 8 use simulators that are variants of $\mathbf{S}$. The one in Chapter 6, however, is slightly more complex.

Naito and Ohta [NO14] generalized the indifferentiability result for arbitrary $r', r'' \geq r$. As a matter of fact, their proof uses a multicollision-taming strategy of the form (2.5). For consistency with the rest of this thesis, we prefer to express the result with an approach of the form (2.6). Hence, this gives

$$\mathbf{Adv}^{\mathrm{indif}}_{\mathrm{Sponge},\mathbf{S}}(\mathcal{A}) = \tilde{\mathcal{O}}\left(\frac{\mathcal{N}^2}{2^c} + \frac{\mathcal{N}\cdot \mathrm{mucol}(\mathcal{N}, 2^{r'-r})}{2^{c'}} + \frac{\mathcal{N}}{2^{c''}}\right)\,. \tag{3.3}$$

In short, this means that the same type of indifferentiability bound holds even if the initial absorption is $r'' = r + c/2$ bits, and squeezing is performed at a rate of $r' = r + c/2 - \log_2(c)$ bits at a time. This allows encoding extra information

[2]To be more precise, their simulator excludes some values from the sampling process, but this does not change the reasoning.

into the initial state of the sponge at no extra cost and may reduce the number of required permutation calls during squeezing.[3]

Interpretation of the Bound. The sponge is secure up to the birthday bound in the capacity. To achieve a target security level of λ bits, the minimum required values for c, c', and c'' are respectively 2λ, $\lambda + \log_2(2\lambda)$, and λ.

3.1.2.2 Collision, Preimage, and Second Preimage Resistance

The security of the sponge against collision, preimage, and second preimage attacks follows directly from the indifferentiability result in Theorem 3.1.1 combined with the general composition theorem of Lemma 2.4.1. We state the result below for the sponge with $r = r' = r''$.

Corollary 3.1.1.1. *Let $b, c, r, \mathcal{N} \in \mathbb{N}^*$ with $b = r + c$. Consider the sponge construction of Algorithm 1 with parameters b, c, r and $r' = r'' = r$. Let $\mathcal{A}$ be an adversary that makes $\mathcal{N}$ permutation queries. We have*

$$\mathbf{Adv}^{\mathrm{col}[\nu]}_{\mathrm{Sponge}}(\mathcal{A}) \le \frac{\mathcal{N}(\mathcal{N}+1)}{2^c} + \frac{\mathcal{N}(\mathcal{N}-1)}{2^{\nu+1}}, \tag{3.4}$$

$$\mathbf{Adv}^{\mathrm{eSec}[\kappa,\nu]}_{\mathrm{Sponge}}(\mathcal{A}) \le \frac{\mathcal{N}(\mathcal{N}+1)}{2^c} + \frac{\mathcal{N}}{2^{\nu}}, \tag{3.5}$$

$$\mathbf{Adv}^{\mathrm{ePre}[\nu]}_{\mathrm{Sponge}}(\mathcal{A}) \le \frac{\mathcal{N}(\mathcal{N}+1)}{2^c} + \frac{\mathcal{N}}{2^{\nu}}, \tag{3.6}$$

$$\mathbf{Adv}^{\mathrm{dPre}[\kappa,\nu]}_{\mathrm{Sponge}}(\mathcal{A}) \le \frac{\mathcal{N}(\mathcal{N}+1)}{2^c} + \frac{\mathcal{N}}{2^{\nu}} + \frac{\mathcal{N}}{2^{\kappa}}. \tag{3.7}$$

3.1.3 Generic Attacks

In this section, we restrict our attention to the case $r' = r'' = r$. For arbitrary values of r', r'', and r, additional intermediate results are needed to establish full tightness for second preimage and collision resistance; we postpone that discussion to Section 5.5.

It was clear from the start that the indifferentiability bound of the sponge is tight. Because the formal tightness analysis is rather technical, we defer it to Section 5.2.3. Below, we discuss generic collision and (second) preimage attacks which were already known by the designers of the sponge [BDPV07,

[3]Note that in practice, using these extended rates with a fixed permutation may require strengthening the latter to maintain the desired security guarantees.

Section 5]. Throughout, we assume the minimal padding rule $pad_r^{10^*}$.[4] As we will focus on tightness up to constant, we will sometimes ignore the fact that any sponge evaluation of a message M of length α costs α permutation calls, and simply count any such evaluation as 1 query.

3.1.3.1 Collision Attack

The best-known collision attack exploiting the structure of the sponge resembles the indifferentiability attack, and is tight. Indeed, after $\approx 2^{c/2}$ absorb calls, one can find collisions on the inner part of sponge states, and by using a *subsequent* absorb call, the inner collision can be transformed into a full-state collision. We state the proposition below.

Proposition 3.1.1 ([BDPV07]). *Let $b, c, r, \nu, \mathcal{N} \in \mathbb{N}^*$ with $b = r + c$. Consider the sponge construction of Algorithm 1 with parameters b, c, r, $r' = r'' = r$. There exists an adversary $\mathcal{A}$ with $\mathcal{N} \approx \min\left\{2^{\nu/2}, 2^{c/2}\right\}$ such that*

$$\mathbf{Adv}^{\mathrm{col}[\nu]}_{\mathrm{Sponge}}(\mathcal{A}) \approx 1 .$$

Proof. Denote by $\mathcal{H}^{\mathcal{P}}(\cdot)$ the function $\mathrm{Sponge}^{\mathcal{P}}(\cdot, \nu)$. We make a case distinction depending on the values c, ν.

- Case $\nu \le c$. The adversary takes $\mathcal{N}$ messages and queries it to the construction. With probability around $\mathcal{N}(\mathcal{N}-1)/2^{\nu+1}$, there exists two messages M, M' queried that satisfy $\mathcal{H}^{\mathcal{P}}(M) = \mathcal{H}^{\mathcal{P}}(M')$. After $\mathcal{N} \approx 2^{\nu/2}$ attempts, the adversary has with high probability found a collision;

- Case $c < \nu$. Starting from the initial state IV, the adversary computes $Y_i := \mathcal{P}(IV \oplus M_i \| 0^c)$ for $\mathcal{N}$ different values M_i. If $\mathcal{N} \approx 2^{c/2}$, there will with high probability be two indices $i \neq j$ such that

$$\mathrm{inner}_c(Y_i \oplus Y_j) = 0^c .$$

  Define the collision as the unique messages M, M' such that

$$\begin{aligned} pad_r^{10^*}(M) &= M_i \| \left(\mathrm{outer}_r(Y_i) \oplus \Delta\right) , \\ pad_r^{10^*}(M') &= M_j \| \left(\mathrm{outer}_r(Y_j) \oplus \Delta\right) , \end{aligned}$$

[4]Note that some attacks may become more subtle when, for example, the padded message includes length encoding. For a detailed example of this in a second preimage setting, see [SLZ+25, Section 3.3.3].

where $\Delta \in \{0,1\}^r$ ensures that the two above sequence of message blocks are valid padded messages. We remark that if $r \leq c/2$ one will need multiple message blocks in order to make $\mathcal{N} \approx 2^{c/2}$ evaluations, but the attack works in a comparable way. In total, in this case the attack requires $\mathcal{N} \approx 2^{c/2}$ evaluations. $\square$

3.1.3.2 Second Preimage Attack

The second preimage bound is also tight, and the best known attack also leverages inner collisions.

Proposition 3.1.2 ([BDPV07])**.** *Let $b, c, r, \nu, \kappa, \mathcal{N} \in \mathbb{N}^*$ with $b = r + c$. Consider the sponge construction of Algorithm 1 with parameters b, c, r, $r' = r'' = r$. There exists an adversary $\mathcal{A}$ with $\mathcal{N} \approx \min\left\{2^\nu, 2^{c/2}\right\}$ such that*

$$\mathbf{Adv}^{\mathrm{eSec}[\kappa,\,\nu]}_{\mathrm{Sponge}}(\mathcal{A}) \approx 1\,.$$

Proof. Denote by $\mathcal{H}^{\mathcal{P}}(\cdot)$ the function $\mathrm{Sponge}^{\mathcal{P}}(\cdot, \nu)$. Let $M \in \{0,1\}^*$ be the first preimage, and let $Z = \mathcal{H}^{\mathcal{P}}(M)$. We again make a case distinction depending on the values c, ν.

- Case $\nu \leq c/2$. The adversary fixes a message $M' \neq M$ and queries it to the construction. The query satisfies $\mathcal{H}^{\mathcal{P}}(M') = Z$ with probability around $1/2^\nu$. After $\mathcal{N} \approx 2^\nu$ attempts, the adversary has with high probability found a preimage M';
- Case $c/2 < \nu$. The attack consists of two sequential parts:
  - The adversary computes $\mathcal{H}^{\mathcal{P}}(M)$, in particular, let Y_1 be the state value during the first squeeze (following the terminology of Figure 3.1, this corresponds to the state from which Z_1 is extracted);
  - Starting from the initial state IV, it computes $Y_0^{\rightarrow} := \mathcal{P}(IV \oplus M_1 \| 0^c)$ for $\mathcal{N}$ different values M_1 (excluding the first message block of $pad_r^{10^*}(M)$). Starting from the value Y_1, it computes $Y_0^{\leftarrow} := \mathcal{P}^{-1}(\mathcal{P}^{-1}(Y_1) \oplus (M_3 \| 0^c))$ for $\mathcal{N}$ different non-zero values M_3. If $\mathcal{N} \approx 2^{c/2}$, there will with high probability be two values M_1, M_3 such that

  $$\mathrm{inner}_c(Y_0^{\rightarrow} \oplus Y_0^{\leftarrow}) = 0^c\,.$$

  Let $M_2 := \mathrm{outer}_r(Y_0^{\rightarrow} \oplus Y_0^{\leftarrow})$. Define the second preimage as the unique message M' such that $pad_r^{10^*}(M') = M_1 \| M_2 \| M_3$. We remark that if $r \leq c/2$ one will need multiple message blocks for both the forward and the inverse part in order to make $\mathcal{N} \approx 2^{c/2}$ evaluations, but the attack works in a comparable way. In total, in this case the attack requires $\mathcal{N} \approx 2^{c/2}$ evaluations. □

Remark 3.1.2. *Contrarily to the collision attack of Proposition 3.1.1, this second preimage attack fundamentally relies on the invertibility of the permutation $\mathcal{P}$. If $\mathcal{P}$ were instead a random transformation, the attack would no longer apply. In that setting, Foekens [Foe23] established a tighter second preimage bound of approximately* $\min\{2^\nu, 2^c/\alpha\}$ *queries, where α denotes the length (in blocks) of the first preimage. Moreover, we showed [SLZ+25] that a comparable bound can be achieved when adding a feed-forward during absorption.*

3.1.3.3 (Domain-Oriented) Preimage Attacks

Bertoni et al. [BDPV07] gave a first preimage attack, but it does not match the bound given in Corollary 3.1.1.1. We therefore postpone the preimage attack description to Section 4.2.2, as improving the preimage security bound is the goal of that chapter. Roughly speaking, the best-known attack is a more complex variant of the second preimage attack: the adversary must find by itself a state that squeezes to the target output Z, which may require more effort than in the second preimage scenario.

3.1.4 Duplex Construction

The duplex construction was introduced by Bertoni et al. [BDPV11b]. Its main goal is to extend the sponge paradigm to support efficient authenticated encryption. In particular, SpongeWrap [BDPV11b] (described in Section 3.3.1) is the first AEAD mode built on top of the duplex.

3.1.4.1 Description

Unlike the sponge, which separates absorption and squeezing into distinct phases, the duplex construction allows absorption and squeezing to be interleaved in a stateful way. It provides two interfaces: one to initialize the internal state, and another one to perform a duplex call that absorbs the input and then squeezes the output. The duplex construction is illustrated in Figure 3.2 and described in Algorithm 2. Notably, each duplex call applies a sponge-compliant padding function to the input before absorbing it into the state.

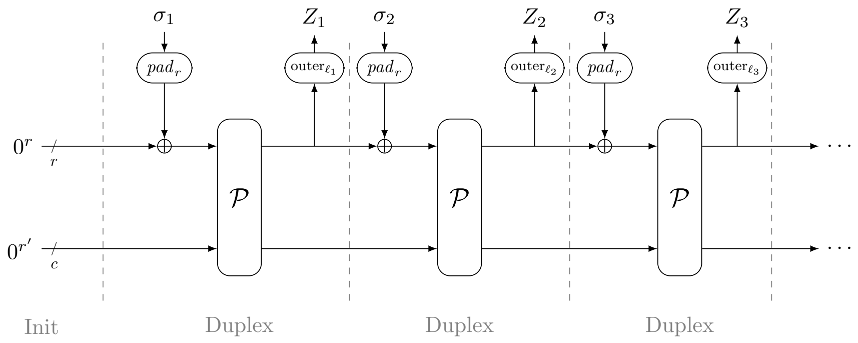


Figure 3.2: Duplex construction. The inputs σ_i and requested lengths ℓ_i must satisfy $|pad_r(\sigma_i)| = r$ and $0 \leq \ell_i \leq r$.

Algorithm 2 Duplex construction based on $\mathcal{P}$.

```
1: function D.initialize()
2:     $S \leftarrow 0^b$; // State of the duplex (kept in memory)

3: function D.duplexing($\sigma, \ell$)
       require: $\sigma \in \{0,1\}^*$ with $|pad_r(\sigma)| = r$ and $0 \leq \ell \leq r$
4:     $S \leftarrow \mathcal{P}(S \oplus (pad_r(\sigma) \| 0^c))$
5:     return $\text{outer}_\ell(S)$
```

3.1.4.2 Security

Bertoni et al. showed that any duplex call can be represented as a sponge call via the *duplexing-sponge lemma* [BDPV11b, Lemma 3]. This allows to analyze the security of duplex-based constructions through the indifferentiability of the sponge, as they did for the security proof of SpongeWrap (see Section 3.3.1). This justifies the use of padding at every duplexing call, as it is necessary for the reduction to indifferentiability.

Degabriele et al. [DFG23] took a different approach and provided an idealized version of the *padding-free* duplex named *online random oracle* (ORO). This is similar to the previously defined ideal extendable input function (IXIF) of Daemen et al. [DMV17], which was tailored for the full-state keyed duplex (see Section 3.1.4.3). The ORO can be seen as a stateful extension of a random oracle: it maintains an internal state and, when receiving an input sequence of duplexing calls, returns outputs exactly as a random oracle would, but in a way that "remembers" all previously queried sequences.[5] Under this abstraction, padding is no longer needed at the level of the duplex itself and can instead be handled by a mode built on top of the duplex. Degabriele et al. proved that the (padding-free) duplex is indifferentiable from an ORO up to $\approx 2^{c/2}$ queries [DFG23, Theorem 2], matching the bound of the sponge.

Degabriele et al. adopt a counting method that differs from that of the sponge designers: repeated duplex (or ORO) calls are counted again. For instance, if an adversary performs a sequence of l duplex/ORO calls, then re-initializes the state and repeats the exact same sequence of calls, the total query complexity is considered to be $2l$. This way of counting has the advantage to be more intuitive in stateful contexts and to better reflect how attacks are typically evaluated (see the discussion at the end of Section 3.3.1.3). However, it also leads to potentially looser bounds. This is because, with current proof techniques, the security analysis typically revolves around bounding the probability of certain bad events, themselves associated with fresh permutation or random oracle calls. Counting repeated paths as new queries increments the complexity without providing new information, which thus makes the bounds less tight.[6]

3.1.4.3 Generalization to the Full-State Keyed Duplex

The main application of the duplex construction is for AEAD purposes. In such settings, the key is injected into the internal state, making the entire state secret. This allows for different strategies to optimize the absorbing rate/throughput. In particular, it was shown that the key can be loaded into the full state (as in monkeyDuplex [BDPV12]), and absorption can be done *over the entire state* (but squeezing is still done at a rate of r bits) without sacrificing security [BDPV11c, BDPV12, CDH$^+$12, ADMV15].[7] This led to the full-state keyed duplex (FSKD) of Mennink et al. [MRV15], later further generalized by Daemen et al. [DMV17] and by Dobraunig and Mennink [DM19a] who proved leakage resilience of the duplex.

The generalization of FSKD resulted in very fine-grained security bounds, allowing tight analysis in a wide range of applications. However, the increased

[5]As a matter of fact, we describe an ORO in Algorithm 14 of Chapter 8 tailored to the context of this chapter.

[6]Looking ahead, in Chapter 8, we rather stick to the query counting made by the sponge designers, and assume that repeated paths *do not* increment the query complexity.

[7]Note that in Chapter 10, we aim, among other goals, to break the limitation of the r-bit squeezing rate for sponge-based PRF constructions.

generality came at the cost of complexity, as these bounds are difficult to interpret. Hence, we will not venture into describing in detail the bounds, and instead refer to the excellent systematization of knowledge by Mennink [Men23], which offers a clear and comprehensive interpretation in various use cases. Looking ahead, we will focus on specific instantiations of FSKD: the full-state keyed sponge and MonkeySpongeWrap, discussed in respectively Sections 3.2.2 and 3.3.1.3, for which we provide simplified (big-O) security bounds.

3.2 Sponge-Based PRF Constructions

In this section, we describe relevant sponge-based PRF constructions. We begin with the outer-keyed sponge in Section 3.2.1, then focus on an optimized variant known as the full-state keyed sponge in Section 3.2.2.

3.2.1 The Outer-Keyed Sponge

The outer-keyed sponge (OKS) is arguably the most natural way to build a PRF from a sponge. It was introduced and first analyzed by Bertoni et al. [BDPV11c], and got subsequent analysis [GPT15,ADMV15,NY16,Men18].[8]

3.2.1.1 Description

Let $\mathcal{P} \in \texttt{Perm}(b)$, $k \in \mathbb{N}^*$, and $r, c \in \mathbb{N}^*$ with $b = r + c$. The construction OKS based on the permutation $\mathcal{P}$ with key $K \in \{0,1\}^k$, is denoted as:

$$\begin{aligned} \mathrm{OKS}_K^{\mathcal{P}} : \{0,1\}^* \times \mathbb{N}^* &\longrightarrow \{0,1\}^* \\ (M, \nu) \quad &\longrightarrow \mathrm{Sponge}^{\mathcal{P}}(K \| M, \nu)\,, \end{aligned}$$

where Sponge denotes the sponge construction with $r' = r'' = r$. Alternative ways to inject the key are discussed in Remarks 3.2.2 and 3.2.3.

3.2.1.2 Security

We consider the PRF security as in Definition 2.4.5, where the primitive set is $\texttt{Perm}(b)$, and the keys of the users sampled uniformly at random, independently. More precisely, the real world is

$$W_R := \left((\mathrm{OKS}_{K_m}^{\mathcal{P}})_{m=1}^{\mu}, \mathcal{P}\right)\,,$$

[8]The name “outer-keyed sponge” comes from [ADMV15].

where $\mathcal{P} \xleftarrow{\$} \texttt{Perm}(b)$ and $K_1, \ldots, K_\mu \xleftarrow{\$} \{0,1\}^k$, and the ideal world is

$$W_I := \left((\mathcal{RO}_m)_{m=1}^{\mu}, \mathcal{P} \right) ,$$

where $\mathcal{P} \xleftarrow{\$} \texttt{Perm}(b)$, and all $\mathcal{RO}_m$'s are independent random oracles.

Counting Queries. Similarly to Section 3.1.2.1, we measure the adversary's resources in terms of permutation calls. However, one difference with indifferentiability is that the *online* and *offline* complexity are clearly separated:

- $\mathcal{N}$ denotes the total number of distinct queries to the primitive oracle;
- $\mathcal{M}$ denotes the number of *non-duplicate* queries to the construction oracles, measured as the minimal number of permutation calls that these construction queries would induce with the oracles $\left(\mathrm{OKS}_{K_m}^{\mathcal{P}}\right)_m$. As in the indifferentiability setting, a construction query is considered duplicate if it follows a path that has already been queried.

In the remainder of the section, we will assume $\mu \leq \mathcal{M} \ll \mathcal{N}$ when presenting big-O bounds.

Security Bound. The analysis of OKS is somewhat scattered. After Bertoni et al.'s initial analysis [BDPV11c], the scheme got renewed analysis by Andreeva et al. [ADMV15] and Naito and Yasuda [NY16]. Both provided bounds that were not tight in full generality, due to a specific bad event called *key prediction*. For this event, they relied on the bounds from Gaži et al. [GPT15], which were later improved by Mennink [Men18], hence closing the gap. Let $\mathcal{A}$ be an adversary with query complexity $(\mathcal{M}, \mathcal{N})$. Assuming that $\lceil \frac{k}{r} \rceil$ is a small constant, we have

$$\mathbf{Adv}_{\mathrm{OKS}}^{\mu\text{-prf}}(\mathcal{A}) = \tilde{\mathcal{O}}\left(\frac{\mu\mathcal{N}}{2^k} + \frac{\mathcal{M}\mathcal{N}}{2^c} \right) . \tag{3.8}$$

The bound is tight. In Chapter 9, we will describe attacks against Ascon-AE that use these ideas, and the attacks appear in Section 9.7.1.2 (targeting the first term) and Section 9.7.2 (targeting the second term). At a high level, the first term corresponds to the probability that the adversary guesses one of the user keys via permutation queries, which would trivially break security. The second term corresponds to the probability that the adversary guesses an internal state via permutation queries, which would likewise allow them to distinguish. The denominator is 2^c because, in this model, the adversary can always control the outer part of a state that was once used for squeezing. Let us clarify how the adversary can do this with the following example.

Example 3.2.1. *Assume for simplicity that the padding function is* $\mathrm{pad}_r^{10^*}$*, and that the key has a length divisible by* r*. The adversary can proceed as follows:*

- *Make one* OKS *call with input message* $M_1 \in \{0,1\}^{r-1}$ *and requested output length* r*, obtain an output* Z_1*. Let us assume that* Z_1 *is different from* 0^r*, so that it can be written as* $z\|1\|0^*$ *for* $z \in \{0,1\}^{\leq r-1}$*;*
- *Make another* OKS *call with input message* $M_1\|1\|z$*. The last block XORed into the state will now cancel out the outer part (which was previously* $z\|1\|0^*$*), setting it to* 0^r*.*

This shows that the adversary can deliberately manipulate the outer part of the state. In general, this strategy allows the outer part of a squeezing state to take up to $2^r - 1$ *different values.*

Remark 3.2.1. *Andreeva et al. [ADMV15] (as well as Bertoni et al. [BDPA10], and Mennink et al. [MRV15]) parametrized the adversarial resources by an additional term called* ***multiplicity****, which intuitively bounds the maximum number of times a given outer part appears among all OKS states. This parameter captures the adversary's ability to control the outer part of the permutation inputs and outputs, and thus depends heavily on the use case considered. It can yield significantly better security bounds and applies to all keyed sponge constructions. However, as pointed out by Daemen et al. [DMV17] and further discussed by Mennink [Men23], multiplicity is a proof-inherent term, thus it should have been bounded* within *the proof and should not appear in the final bounds. The multiplicity was later replaced with higher-level metrics, see Section 3.2.2.3.*

Remark 3.2.2. *An alternative version of* OKS*, namely where the input to the sponge* $K\|M$ *is roughly replaced by* $M\|K$*, appeared in the original specification of Bertoni et al. [BDPV11a, Section 5.11.2]. This approach was later generalized and analyzed by Dobraunig and Mennink [DM19b, DM20] who dubbed the construction the* ***suffix-keyed sponge*** *and proved its leakage resilience. Berendsen and Mennink [BM24] subsequently fine-tuned and improved this leakage resilience analysis. This design aims to minimize leakage: only the permutation evaluations at the end that process the key are exposed to leakage, and therefore only those need to be protected. The security bounds of* OKS *does not apply there, as the adversary can always find inner collisions before the key is processed (as in Proposition 3.1.1), so that the security of the suffix-keyed sponge has a term in* $\tilde{\mathcal{O}}\left(\frac{\mathrm{N}^2}{2^c}\right)$ *(see [DM19b, Theorem 1] for the exact bound).*

Remark 3.2.3. *Naito [Nai16] investigated an alternative construction by "sandwiching" the message between two keys (so that the input to the sponge is roughly replaced by $K\|M\|K$). This modification improves the bound: the term in $\frac{\mathcal{M}\mathcal{N}}{2^c}$ from* (3.8) *is replaced by a term of the order $\frac{\mathcal{N}}{2^c} + \frac{\mathcal{M}\mathcal{N}}{2^b}$.*

3.2.2 The Full-State-Keyed Sponge

The full-state keyed sponge (FSKS) uses the optimization strategies mentioned in Section 3.1.4.3 that are possible due to the keyed setting. The idea of absorbing over the entire state dates back to the MAC donkeySponge of Bertoni et al. [BDPV12], and was later formalized and generalized by Mennink et al. [MRV15].

3.2.2.1 Description

We follow Mennink's description [Men23] of FSKS. Let $k, b, r', c' \in \mathbb{N}^*$ such that $k \leq b$ and $r' + c' = b$. Let $\mathcal{P} \in \texttt{Perm}(b)$, and pad_b be a sponge-compliant padding function that returns a message of size multiple of b. Let $\mathcal{IV} \subset \{0,1\}^{b-k}$ be a set of initialization vectors, and $K \in \{0,1\}^k$ be a key. The FSKS based on the key K and the b-bit permutation $\mathcal{P}$ is denoted as:

$$\begin{aligned} \mathrm{FSKS}_K^{\mathcal{P}} : \mathcal{IV} \times \{0,1\}^* \times \mathbb{N}^* &\longrightarrow \{0,1\}^* \\ (IV, M, \nu) &\longrightarrow Z \in \{0,1\}^\nu . \end{aligned}$$

Let $IV \in \mathcal{IV}$ be an initialization vector, $M \in \{0,1\}^*$ the input message, $\nu \in \mathbb{N}^*$ the requested output length, and $\ell := \lceil \frac{\nu}{r'} \rceil$. $\mathrm{FSKS}_K^{\mathcal{P}}$ operates as follows:

1. *Initialization.* M is first padded into message blocks using pad_b: $M_1\| \cdots \|M_u \leftarrow pad_b(M)$. The state S is initialized as $\mathcal{P}(K\|IV)$;
2. *Absorbing phase.* At the i^{th} iteration, for $i = 1, \ldots, u$, the state is updated as $S \leftarrow \mathcal{P}(S \oplus M_i)$;
3. *Squeezing phase.* At the i^{th} iteration, for $i = 1, \ldots, \ell$, the outer r' bits of S are extracted as $Z_i \leftarrow \mathrm{outer}_{r'}(S)$ and the state is updated as $S \leftarrow \mathcal{P}(S)$;
4. The output is $Z \leftarrow (Z_1\| \cdots \|Z_\ell)[1 : \nu]$.

The FSKS construction is illustrated in Figure 3.3. Strictly seen, it can be seen as an optimized sponge with the initial rate r'' and the absorbing rate r set to b.

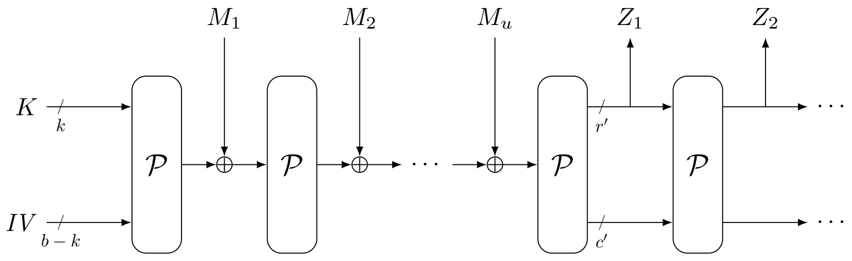


Figure 3.3: Full-state keyed sponge. The input message M is padded as $M_1 \| \cdots \| M_u \leftarrow pad_b(M)$.

3.2.2.2 Security

Again, we consider the PRF security as in Definition 2.4.5, where the real and ideal worlds are defined analogously to Section 3.2.1.2. The adversarial resource complexity $(\mathcal{M}, \mathcal{N})$ is defined as in that section. The security of FSKS was established by Mennink et al. [MRV15] but their bound parametrizes the adversarial resources according to their *multiplicity*, which should have been left implicit (see Remark 3.2.1).

Let $\mathcal{A}$ be an adversary with resources $(\mathcal{M}, \mathcal{N})$. Simplifying to the extreme Mennink's interpretation [Men23, Theorem 6] of the bound given in [DMV17],[9] we obtain

$$\mathbf{Adv}^{\mu\text{-prf}}_{\mathrm{FSKS}}(\mathcal{A}) = \tilde{\mathcal{O}}\left(\frac{\mu\mathcal{N}}{2^k} + \frac{\mathcal{M}\mathcal{N}}{2^{c'}}\right) . \tag{3.9}$$

The bound is of the same order as that of the outer-keyed sponge, and is therefore tight. In particular, one can absorb over *the entire state*[10] without loosing generic security.

[9] In [DMV17, Men23], there is a clear distinction between the number of construction queries (denoted by Q) and the online complexity $\mathcal{M}$ (itself defined in Section 3.2.1.2). Here, we use that $Q \leq \mathcal{M}$, and for the tightness analysis we assume that the adversary can freely choose Q within their online complexity budget $\mathcal{M}$.

[10] Except during the initialization, where absorption should be limited to the IV; otherwise, security degrades to at worst $\frac{\mathcal{M}\mathcal{N}}{2^k}$ [MRV15].

3.2.2.3 Optimizing Security and Efficiency

Dobraunig and Mennink [DM24] focused on the initialization of FSKD and investigated the possibility to improve the term $\frac{\mu \mathcal{N}}{2^k}$ of (3.9). In particular, when each user is assigned a distinct IV, they showed that this term can be improved to $\tilde{\mathcal{O}}\left(\frac{\mathcal{N}}{2^k}\right)$.

Moreover, the duplex's bound of [DMV17] keeps track of the maximum number of times the adversary can control the outer part of the state. This is captured by two parameters, Ω and $\mathcal{L}$, which upper bound the number of duplexing calls in which the adversary may have deliberately set the outer part of the permutation input to values of its choice.

Remark 3.2.4. *The quantities Ω and $\mathcal{L}$ are higher-level than the multiplicity (Remark 3.2.1). From Ω, $\mathcal{L}$, and a multicollision-taming technique, one can upper bound the multiplicity, as done in [DMV17]. Roughly speaking, if we denote the multiplicity by ρ, we have*

$$\mathsf{E}(\rho) \approx \Omega + \mathcal{L} + \mathrm{mucol}(\mathcal{M}, 2^{r'}) .$$

The bounds (3.8) and (3.9) implicitly assume that the adversary can always overwrite the outer part of the state. If the construction imposes a more restrictive padding rule and/or includes a unique nonce at every call, the adversary loses this ability, hence setting $\Omega = \mathcal{L} = 0$. In such cases, the terms in $\frac{\mathcal{M}\mathcal{N}}{2^{c'}}$ from (3.8) and (3.9) can be replaced by a (tight) term of the form

$$\frac{\mathcal{N} \cdot \mathrm{mucol}(\mathcal{M}, 2^{r'})}{2^{c'}} = \tilde{\mathcal{O}}\left(\frac{\mathcal{M}\mathcal{N}}{2^b} + \frac{\mathcal{N}}{2^{c'}}\right) .$$

A concrete example is the PRF Ascon-PRF [DEMS24] that keeps a small absorbing capacity c and separates the absorption and squeezing phases via domain separation (see Section 9.9 for a detailed description and analysis of the mode of Ascon-PRF).

In summary, combining those aforementioned improvements, FSKS can reach up to $\min\{k, c', b - \log_2(\mathcal{M})\}$ bits of security, even in the multi-user setting.

3.3 Sponge-Based AEAD Modes

In the previous section, we explored sponge-based PRFs and several of their variants, which form a rich design space. When moving from PRFs to AEADs, this complexity only grows, as AEAD schemes must handle multiple distinct inputs and outputs. Exhaustively describing all these variants in detail is not feasible. Instead, we focus on SpongeWrap in Section 3.3.1, which is the first sponge-based AEAD mode, and describe one of its generalizations, MonkeySpongeWrap in Section 3.3.1.3. We then briefly describe several alternative design directions in Section 3.3.2.

3.3.1 SpongeWrap

SpongeWrap was introduced together with the duplex [BDPV11b] and is the first duplex-based AEAD mode. It is a session-supporting AEAD mode, meaning that it processes sequences of messages by producing intermediate tags, ensuring that the encryption and authenticity of each message depend on all previously sent ciphertexts in the session.

3.3.1.1 Description

Let $k, b, r, c, t \in \mathbb{N}^*$ with $b = r+c$. Let $\mathcal{P} \in \texttt{Perm}(b)$, and $K \in \{0,1\}^k$ be the key. SpongeWrap based on the permutation $\mathcal{P}$ and keyed with K consists of two algorithms: $\mathbf{EncW}_K^{\mathcal{P}}$ for encryption and $\mathbf{DecW}_K^{\mathcal{P}}$ for decryption. $\mathbf{EncW}_K^{\mathcal{P}}$ takes associated data and plaintext as input, and returns a ciphertext of the same length as the plaintext, along with a tag:

$$\begin{aligned}\mathbf{EncW}_K^{\mathcal{P}} : \{0,1\}^* \times \{0,1\}^* &\longrightarrow \{0,1\}^* \times \{0,1\}^t \\ (A, P) \qquad &\longrightarrow (C \in \{0,1\}^{|P|}, T)\,.\end{aligned}$$

$\mathbf{DecW}_K^{\mathcal{P}}$ takes associated data, ciphertext, and tag as input, and returns either the plaintext or the failure symbol $\perp$ if the tag is invalid:

$$\begin{aligned}\mathbf{DecW}_K^{\mathcal{P}} : \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^t &\longrightarrow \{0,1\}^* \cup \{\perp\} \\ (A, C, T) \qquad &\longrightarrow P \in \{0,1\}^{|C|} \text{ or } \perp\,.\end{aligned}$$

It has the property that, for any $K \in \{0,1\}^k$, $A, P \in \{0,1\}^*$, if $\mathbf{EncW}_K^{\mathcal{P}}(A, P) = (C, T)$, then $\mathbf{DecW}_K^{\mathcal{P}}(A, C, T) = P$. Note that SpongeWrap does not take an explicit nonce as input. Instead, nonce-respecting security is achieved under the condition that the associated data is unique for each encryption call.

At a high level, $\mathbf{EncW}_K^{\mathcal{P}}$ begins by padding the key, associated data, and plaintext. These padded blocks are then processed in order via duplex calls:

- First, the key blocks are absorbed into the state, and each duplex call requires an output of $\ell = 0$ bits. As discussed in Section 3.1.4, each duplex call applies an additional layer of internal padding;
- Then, the associated data blocks are absorbed in the same fashion;
- Then, the plaintext is processed: each block is absorbed, and the outer parts before absorption of the blocks essentially serve as keystream.

After all plaintext blocks have been processed, the tag is generated through additional duplex calls in squeeze-only mode.

Decryption mirrors this structure. It starts by absorbing the padded key and associated data, as in the encryption algorithm. Then, the ciphertext blocks are processed: each block overwrites the outer part of the state, and the corresponding plaintext is recovered by XORing the outer part of the state with the ciphertext blocks. After all ciphertext blocks have been handled, the tag is recomputed via squeezing and compared to the one provided. If the tags do not match, the decryption returns the failure symbol $\perp$.

SpongeWrap with the 10^*-padding is illustrated in Figure 3.4. Here, cut_s denotes the function that gets as input a bit string $X \in \{0,1\}^*$, and that splits it into s-bit blocks, where the last block is of size between 0 and $s-1$ bits.

3.3.1.2 Security

The adversarial resources are refined, compared to the ones defined in Section 3.2.1.2, as follows:

- The offline complexity $\mathcal{N}$ remains unchanged;
- The online complexity $\mathcal{M}$ now spans both encryption and decryption queries. It is split as $\mathcal{M} = \mathcal{M}_E + \mathcal{M}_D$, where $\mathcal{M}_E$ denotes the online complexity of encryption queries and $\mathcal{M}_D$ the online complexity of decryption queries. Here, redundant calls between encryption and decryption queries *are* doubly counted;
- The number of construction queries is denoted by Q. It is split as $Q = Q_E + Q_D$ for encryption and decryption queries, respectively.

Throughout the remainder of this section, we focus on *nonce-respecting* security.

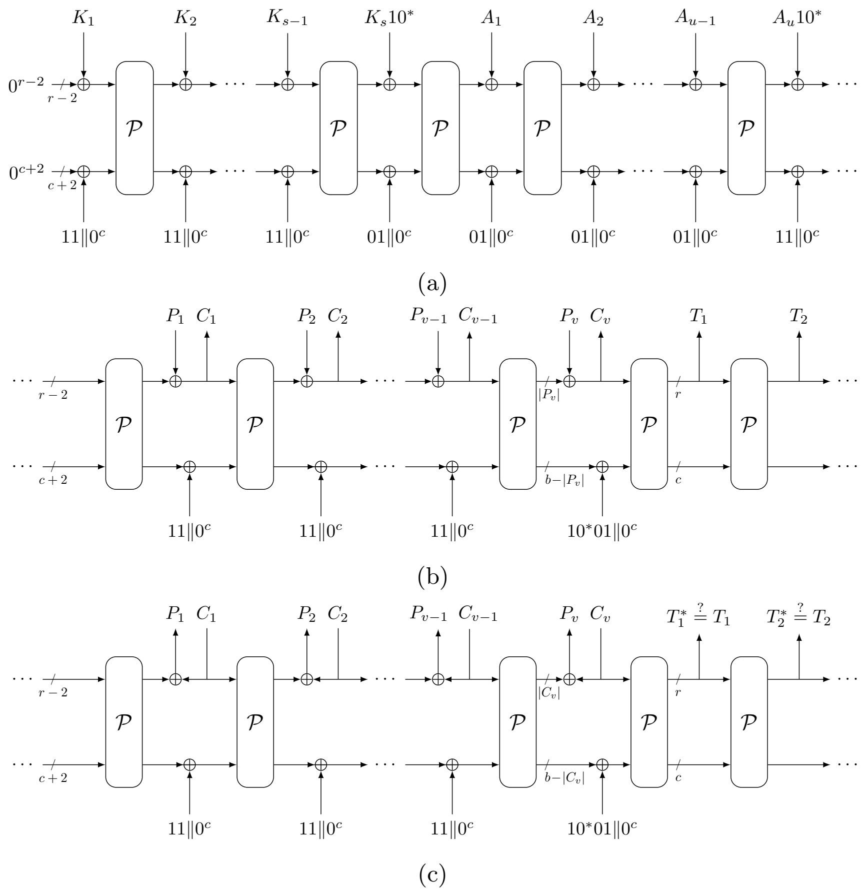


Figure 3.4: The SpongeWrap mode of operation with the 10^*-padding : (a) common processing of key and associated data by encryption and decryption; (b) additional steps for encryption **EncW**; and (c) additional steps for decryption **DecW**. Here, K and A are split as $(K_1, \ldots, K_s) \leftarrow cut_{r-2}(K)$ and $(A_1, \ldots, A_u) \leftarrow cut_{r-2}(A)$. For encryption, the plaintext $P \in \{0,1\}^*$ is split as $(P_1, \ldots, P_v) \leftarrow cut_{r-2}(P)$, and for decryption the ciphertext $C \in \{0,1\}^*$ is split as $(C_1, \ldots, C_v) \leftarrow cut_{r-2}(C)$. Both the duplex padding and the 10^*-padding are shown explicitly in the figure.

The designers of SpongeWrap proved that, in the single-user setting, one has

$$\begin{aligned}\mathbf{Adv}^{\text{1-m-conf}}_{\text{SpongeWrap}}(\mathcal{A}) &\leq \frac{Q}{2^k} + \frac{(\mathcal{N}+\mathcal{M}_E)(\mathcal{N}+\mathcal{M}_E-1)}{2^c}\,,\\ \mathbf{Adv}^{\text{1-m-auth}}_{\text{SpongeWrap}}(\mathcal{A}) &\leq \frac{Q}{2^k} + \frac{Q_D}{2^t} + \frac{(\mathcal{N}+\mathcal{M})(\mathcal{N}+\mathcal{M}-1)}{2^c}\,. \end{aligned} \tag{3.10}$$

Their proof reduces to the indifferentiability of the sponge.

Degabriele et al. [DFG23, Section 7.1, first paragraph] observed that the indifferentiability-based analysis of SpongeWrap may yield a loose security bound, suggesting that the $\tilde{\mathcal{O}}\left(\mathcal{N}^2/2^c\right)$ term in (3.10) should instead be replaced with $\tilde{\mathcal{O}}\left(\mathcal{N}^4/2^c\right)$. However, this conclusion comes from their query counting method (see Section 3.1.4.2), which differs from that of the designers. Indeed, as discussed in Section 3.1.2.1, the sponge designers intentionally avoid counting repeated queries multiple times, so that the bounds (3.10) are correct. That said, Degabriele et al.'s analysis contributes additional guarantees on SpongeWrap, such as key-dependent-message security.

On the other side, resorting to indifferentiability for solely confidentiality and authenticity security is a bit overkill, as the obtained bound is lossy. Jovanovic et al. [JLM14, JLM+19] derived a refined security bound for the NORX mode [AJN14a], which as they mention, also applies to SpongeWrap.[11] In the single-user setting, they proved a bound of the form:

$$\begin{aligned}\mathbf{Adv}^{\text{1-m-conf}}_{\text{SpongeWrap}}(\mathcal{A}) &= \tilde{\mathcal{O}}\left(\frac{\mathcal{N}}{2^{b/2}} + \frac{\theta\mathcal{N}}{2^c} + \frac{\mathcal{N}}{2^k}\right)\,,\\ \mathbf{Adv}^{\text{1-m-auth}}_{\text{SpongeWrap}}(\mathcal{A}) &= \tilde{\mathcal{O}}\left(\frac{\mathcal{N}}{2^{b/2}} + \frac{\theta\mathcal{N}}{2^c} + \frac{\mathcal{N}}{2^k} + \frac{\mathcal{M}_D\mathcal{N}}{2^c} + \frac{Q_D}{2^t}\right)\,,\end{aligned}$$

where θ tames the maximal multicollision size.[12] In particular security holds as long as $\mathcal{N} \ll \min\left\{2^{b/2}, 2^c/\theta, 2^k, 2^c/\mathcal{M}_D\right\}$ and $Q_D \ll 2^t$. A tightness analysis is postponed to the next section, as the corresponding attacks also apply to MonkeySpongeWrap.

3.3.1.3 MonkeySpongeWrap

Most modern duplex-based AEAD schemes can be viewed as slight variants of SpongeWrap. This was discussed in detail by Mennink [Men23, Section 9],

[11]Strictly speaking, their bound applies only when the key is processed by a single permutation call. Otherwise, a key prediction event (as discussed in Section 3.2.1.2) should be included in the analysis, but the resulting big-O bounds would be similar.

[12]They adopt a multicollision-taming strategy of the form (2.5), see [JLM+19, Lemma 1, Theorem 1] for the exact bounds.

who introduced a modernized variant called MonkeySpongeWrap. The main differences with SpongeWrap are:

- The state is initialized with a key and a nonce, making the scheme fall into the monkeyDuplex [BDPV12] and FSKD descriptions;
- Padding is no longer directly inherited from the sponge construction, and domain separation is handled differently.

MonkeySpongeWrap is illustrated in Figure 3.5. MonkeySpongeWrap generalizes SpongeWrap in a way similar to how FSKS generalizes OKS. Given this similarity, one might wonder why MonkeySpongeWrap does not absorb over the entire state. The reason is that the duplex does not support full-state squeezing, and because ciphertext blocks must match the size of plaintext blocks, this prevents straightforward full-state absorption of the plaintext. However, full-state absorption is possible for associated data. Since there are multiple ways to do this, we defer the discussion to Section 3.3.2.

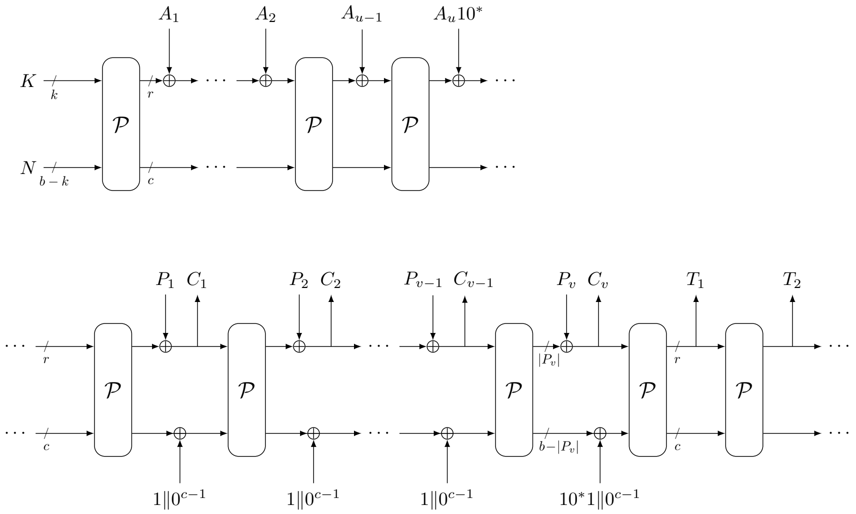


Figure 3.5: The MonkeySpongeWrap mode of operation for encryption. Here, A is split as $(A_1, \ldots, A_u) \leftarrow cut_r(A)$, and the plaintext $P \in \{0,1\}^*$ as $(P_1, \ldots, P_v) \leftarrow cut_r(P)$, noting that we put the 10^*-padding explicit in the picture.

Security. The security of MonkeySpongeWrap follows from [DMV17] and was made explicit in [Men23, Theorem 7]. Let MSW denote the MonkeySpongeWrap construction. We have

$$\mathbf{Adv}^{\mu\text{-ae}}_{\mathrm{MSW}}(\mathcal{A}) = \tilde{\mathcal{O}}\left(\frac{\mathcal{N}\cdot \mathrm{mucol}(\mathcal{M},2^r)}{2^c} + \frac{\mu\mathcal{N}}{2^k} + \frac{Q_D}{2^t} + \frac{\mathcal{M}_D\mathcal{N}}{2^c}\right). \tag{3.11}$$

The first three terms are tight. The first term corresponds to an adversary guessing an internal state of the construction (see Section 9.7.1.3 from Chapter 9, the underlying idea also applies here). The second and third terms correspond to the probability of generic key-guessing and tag forgery events. The last term $\frac{\mathcal{M}_D\mathcal{N}}{2^c}$ is more subtle, and we discuss it in the next paragraph.

Last Term: Tightness Attack and Query Counting. Gilbert et al. [GBKR23] describe an attack that covers a broad class of duplex-based AE modes where the key is incorporated to the state once, during the initialization phase (as in MonkeySpongeWrap). This attack needs only decryption and permutation queries, and targets the term $\frac{\mathcal{M}_D\mathcal{N}}{2^c}$ in the security bound. Their attack exploits the fact that for a fixed $\beta \in \{0,1\}^r$, and a random permutation $\mathcal{P}$, the function $F_\beta(x) = \mathrm{inner}_c(\mathcal{P}(\beta\|x))$ behaves like a random function, and uses its cyclic properties. A simplified overview of the attack is as follows:

1. Offline phase: find a $\beta \in \{0,1\}^r$ such that the function F_β has a large component, i.e., a set of nodes whose paths all terminate in the same cycle $\mathcal{C}$. Then, each element in $\mathcal{C}$ is seen as a candidate state value before computing the tag. For each $x \in \{0,1\}^c$ in $\mathcal{C}$, and for $i = 1,\ldots,\lceil\frac{t}{r}\rceil$, compute $\mathcal{P}^i(\beta\|x)$ to obtain a candidate tag T_x;

2. Online phase: fix a nonce N, and let $C = \beta^\gamma$. All subsequent decryption queries will have the nonce N, ciphertext C, and empty associated data. In particular, the path induced by the permutation evaluations, and hence the tag to guess, is the same for all decryption queries. If γ is sufficiently large, with high probability the $\gamma - 1$ successive permutation evaluations made to absorb C end up in the cycle $\mathcal{C}$. In that case, submitting all the decryption queries (N, ϵ, C, T_x) with all the T_x tags computed beforehand allow the adversary to forge.

In the first step, they suggest spending $\approx 2^{3c/4}$ permutation queries, which would give a cycle of length $\approx 2^{c/4}$ with high probability. Then, with $\gamma \approx 2^{c/2}$ in the second phase, the attacks succeeds with a high probability. From this information, we can already deduce that $\mathcal{N} \approx 2^{3c/4}$, $Q_D \approx 2^{c/4}$.

Regarding the online complexity, they indicate a total online complexity of $2^{3c/4}$. The rationale is as follows: there are in total $Q_D = 2^{c/4}$ decryption queries, each query requiring $2^{c/2}$ permutation calls, thus leading a total online complexity of $2^{3c/4}$. Let us call this quantity the attack-wise cost, and denote it by $\tilde{\mathcal{M}}_D$. $\tilde{\mathcal{M}}_D$ is a valid upper bound of the $\mathcal{M}_D$ used in the security proofs, but could be decreased if we take the metrics used by the security proofs.

As explained throughout this section, in the security proofs of duplex-based modes, the blocks for encryption/decryption queries that repeat are not doubly counted. Therefore, in the setting of this attack, only the first decryption query increments $\mathcal{M}_D$, as the subsequent decryption queries only change the tag. With the metrics of the security proof, $\mathcal{M}_D = 2^{c/2}$, and $Q_D = 2^{c/4}$, which gives $\mathcal{M}_D\mathcal{N} = 2^{5c/4}$.

If we parametrize this attack differently, i.e., by taking a cycle in the first phase of size $2^{c/2}$, then the attack succeeds as well with high probability when $\mathcal{N} \approx 2^{c/2}$, and has a cost of $Q_D \approx 2^{c/2}$, $\mathcal{M}_D \approx 2^{c/2}$ (but $\tilde{\mathcal{M}}_D$ is as large as $\approx 2^c$). While the attack-wise cost $\tilde{\mathcal{M}}_D$ is much higher in this parametrization, the proof-wise cost $\mathcal{M}_D$ is lower. Notably, when the tag size and key length are larger than $c/2$, this parametrization of their attack matches the security bound. This observation does not enhance the attack per se but offers alternative compromises when considering the counting method used in security proofs.

To conclude, the term in $\frac{\mathcal{M}_D\mathcal{N}}{2^c}$ is tight *for certain parameter sets*, but tightness beyond those cases is unclear.

3.3.2 Relevant Duplex-Based Variants

This section presents a selection of variants of duplex-based AEAD modes. These schemes show how some modifications to the design can enhance security and/or efficiency. We intentionally keep the discussion high-level, as the goal of this section is rather to survey the rich diversity of approaches in sponge-based AEAD designs.

Boosting Multi-User Security. The insights from Dobraunig and Mennink [DM24] (already mentioned in Section 3.2.2.3), are also applicable to boost multi-user security of duplex-based designs.

Absorbing Associated Data More Efficiently. Sasaki and Yasuda [SY15], as well as Mennink et al. [MRV15], have explored several techniques to process associated data more efficiently. These include:

- Absorbing associated data concurrently with plaintext blocks, in the inner part of the state;
- Absorbing associated data before and/or after the plaintext over the entire state;
- Absorbing associated data in parallel over the entire state, where the output is XORed to the tag (a technique known as ciphertext translation [Rog02]).

The first two methods are covered by the security treatment of [DMV17]. As an example, the CAESAR candidate Keyak [BDP+16b] combines these two methods, applying one or both depending on the relative lengths of the associated data and plaintext.

Beetle Modes. As we have seen, the security bound of SpongeWrap includes a term of the form $\frac{\mathcal{M}_D \mathcal{N}}{2^c}$, which is due to the ability of the adversary to control the outer part of the state during decryption. The Beetle mode [CDNY18] introduces a combined feedback mechanism, where both the outer part of the state and the ciphertext block are used together to form the input for the next permutation call. This reduces adversarial influence on the state and, roughly speaking, makes the term $\frac{\mathcal{M}_D \mathcal{N}}{2^c}$ disappear in the security bounds.

Full-Rate Modes. The mode of ORANGE-ZEST [CN19], a submission to the NIST Lightweight Cryptography competition [Nat19], incorporates an auxiliary state into the sponge to enable absorbing and squeezing *over the entire state*. ORANGE-ZEST has been attacked by Dobraunig et al. [DMM20] (and a modified version was attacked by Khairallah et al. [KRS19]). However, the flaws in ORANGE-ZEST have been fixed, and the concept was generalized and proven secure by Chakraborty et al. [CDN23a]. Noteworthy is that this approach requires storing and maintaining an additional state, and the security of the mode is directly dependent on the size of this state.

Parallel Modes. When SpongeWrap was introduced, the designers noted that the mode could be parallelized similarly to tree hashing. This idea has since been realized in several schemes. For example, during the CAESAR competition [CAE14], several sponge-based AEAD schemes that support parallel processing were submitted, including NORX [AJN14b], the parallel mode of ICEPOLE [MGH+15], Keyak [BDP+16b], and π-Cipher [GMS+15]. Jovanovic et al. provided security proofs covering NORX and ICEPOLE [JLM14, JLM+19].

Leakage Resilient Modes. Some use cases require AEAD schemes that prioritize resistance to implementation attacks while using a minimal amount of resources. ISAP [DEM$^+$17,DEM$^+$20], a finalist in NIST's lightweight cryptography competition [DEM$^+$19], meets these goals by combining a lightweight encrypt-then-MAC architecture with provable leakage resilience. To do that, it employs an efficient re-keying mechanism, and the authentication part is a special type of suffix keyed sponge (see in Remark 3.2.2). Degabriele et al. [DJS19] proposed an alternative sponge-based leakage-resilient design.

Ascon [DEMS21b] is another notable variant which achieves strong leakage resilience and more. However, we will not give more details here since it is the subject of Chapter 9.

Variable-Round Permutations. Some designs vary the number of rounds of the underlying permutation between phases to strike a balance between performance and security. For instance, Ascon [DEMS21b] uses 12 rounds during initialization and finalization, but only 6 or 8 rounds during the data-processing phase (with different round constants as well). As we discuss in Remark 9.2.1, it is hard to capture this in a generic security proof.

Conclusion. The analysis of Daemen et al. [DMV17] is already quite broad, and due to the generality of the result, their bound was extremely challenging to interpret. Yet, it does not cover all the variants that we discussed. Providing a single, comprehensive framework that encompasses them all would, of course, be an extremely challenging task. Combining desirable features from different schemes is rarely straightforward and typically requires a dedicated and careful security analysis.

Part II

Security Analysis of Sponge-Based Hash Function Constructions

Chapter 4

Tight Preimage Resistance of the Sponge Construction

4.1 Introduction

In Theorem 3.1.1, we saw the indifferentiability result of the sponge with $r = r' = r''$. This result implies that the sponge "behaves" like a random oracle and that it can be used in (most) applications that were proven secure in the random oracle model. In words, assuming that the query complexity is at most $2^{c/2}$, finding collisions, preimages, or second preimages for the sponge is not easier than for a random oracle. This was formally captured in Corollary 3.1.1.1, which states that for a sponge function outputting a fixed-length digest of ν bits, finding collisions requires at least

$$\mathcal{N} \approx \min\{2^{c/2}, 2^{\nu/2}\} \tag{4.1}$$

work, and finding preimages or second preimages requires at least

$$\mathcal{N} \approx \min\{2^{c/2}, 2^{\nu}\} \tag{4.2}$$

work. These bounds have directly influenced the parameter choices of many sponge-based hash designs. Most notably, the SHA-3 hash function family consists of four functions: SHA3-ν, where $\nu \in \{224, 256, 384, 512\}$ defines the output size, and each of these four functions has its capacity c equal to *twice* the digest length ν (see also Table 4.1).

As a matter of fact, in around $2^{c/2}$ work, an adversary can find inner collisions, i.e., different sponge evaluations that collide on the c-bit inner part, and it can use these inner collisions to form a full collision for the sponge and this way distinguish it from random. As shown by the designers and established in Proposition 3.1.1, the collision security bound of (4.1) is tight: a collision for a sponge with fixed ν-bit output can be obtained either by finding a c-bit inner collision or a ν-bit output collision. Similarly, for second preimage resistance, the bound in (4.2) is also tight, as established in Proposition 3.1.2. Indeed, one approach the adversary can take to find a second preimage is an exhaustive search in 2^{ν} work. Alternatively, given the first preimage, the attacker can recompute the sponge on input of this first preimage to determine the final state value *before* squeezing (denoted by X_1 in Figure 4.1). Then, it computes the sponge forward from the initial value IV and backward from the state value *before* squeezing in order to find a collision on the c-bit inner part.

For preimage security, the situation is different, and it appears that *for certain values c and ν*, the bound of (4.2) is *not tight*. This is mainly caused by the fact that, unlike for second preimage security, the final state *before* squeezing cannot always be easily found. Already in the original introduction of the sponge construction in 2007, it was claimed that a preimage attack can only be mounted in $\max\{2^{\nu-r}, 2^{c/2}\}$ work [BDPV07, Section 5.3]. In 2011, both the developers of PHOTON and Spongent made a comparable claim regarding the preimage security of their construction [BKL+11, GPP11]. We discuss this generic attack in detail in Section 4.2.2. Unfortunately, *proving* tight preimage security has remained an open problem since.

4.1.1 Tight Preimage Security

We solve this open problem and prove tight preimage security of the sponge construction. In detail, assuming that the underlying permutation $\mathcal{P}$ is random, we prove that the sponge achieves preimage security up to around

$$\mathcal{N} \approx \min\left\{\max\left\{2^{\nu-r'}, \min\left\{2^{c/2}, 2^{c''}\right\}\right\}, 2^{\nu}\right\} \tag{4.3}$$

work, where we recall that ν is the digest size, c the capacity of the sponge (during absorption), and r', r'' the rates during squeezing and initialization, respectively. A detailed bound is given in Section 4.3, and the bound tightly matches the generic attack of Section 4.2.2 (up to constant). The security relies on a careful investigation of what events are needed to happen in order for a preimage to be found, and subsequently a detailed computation of the probability of these events to occur.

At a very high level, suppose the attacker aims to obtain a preimage for a digest Z consisting of ℓ r'-bit blocks $Z_1\|\cdots\|Z_\ell$, assuming $r' \mid \nu$ for the sake of simplicity. We assume, by definition, that the attacker is required to make all permutation queries that are required for the computation of its eventual preimage, and in particular, it must definitely obtain a cascaded evaluation of $\ell-1$ permutation queries that correspond to outputs $Z_1,\ldots,Z_\ell$. In Figure 4.1, these are the first permutation evaluation *after* outputting Z_1 up to and including the last permutation evaluation *before* outputting Z_ℓ. As we demonstrate in our proof, the attacker succeeds in finding such a path only after around $\mathcal{N} \approx 2^{\nu-r'}$ queries.

However, the adversary is not done after just finding such cascade of permutation evaluations: the evaluations must also be *reached* from IV through the absorption of certain message blocks – these message blocks eventually constitute the preimage that the adversary would output. The adversary could succeed in this in two ways: either the last permutation query before squeezing is made in forward direction, or it is made in inverse direction. If it is made in forward direction, we have to go one step back in our reasoning, namely to the discussion of the squeezing cascade, and observe that in this case the cascade of ℓ permutation evaluations can only be found in $\mathcal{N} \approx 2^\nu$ queries. If it is in inverse direction, this particular permutation query can be made *for free* from the cascade of above $\ell-1$ evaluations, *but* in order to then connect the cascade to the initial value IV, the adversary must necessarily ever find a forward and an inverse permutation evaluation that collide on the inner part or an inverse permutation evaluation that connects to the initial value IV_r. This, in turn requires approximately $\min\left\{2^{c/2}, 2^{c''}\right\}$ work.

In summary, finding a preimage requires either around 2^ν work, or the maximum of $2^{\nu-r'}$ and $\min\left\{2^{c/2}, 2^{c''}\right\}$ work, exactly as expressed in (4.3). Needless to say, the actual security analysis, and in particular the derivation of an upper bound on the probability of finding a matching cascaded permutation evaluation of length $\ell-1$ or ℓ, is much more involved, among others as any permutation query of the adversary may appear at any position in this cascade.

4.1.2 Application

For hash functions with a large rate, e.g., Keccak and eventually the SHA-3 hash function family, the old bound of (4.2) accurately described the preimage security. However, with the advent of lightweight cryptography, many sponge constructions with small permutation size b, small capacity c, and small squeezing rate r' have appeared. In many of these cases, our bound has immediate implications as it confirms higher preimage security.

The ISO/IEC standardized Spongent hash function of Bogdanov et al. [BKL+11] and the PHOTON hash function of Guo et al. [GPP11] are two such cases. Spongent consists of five hash functions, all of which are sponges instantiated with a permutation of size $b \in \{88, 136, 176, 240, 272\}$ bits, a rate of $r = r' = r'' = 8$ bits for the smallest two versions and $r = r' = r'' = 16$ for the larger three, and a capacity $c = b - r$. The smallest version outputs $\nu = b = 88$ bits whereas the other versions output $\nu = c$ bits. The old bound of (4.2) implied that a preimage attack required at least $2^{c/2}$ work, whereas our new bound (4.3) implies that a preimage attack requires at least $2^{\nu - r}$ work. For the smallest version of Spongent, this is an improvement from 2^{40} to 2^{80}, and for the largest version, this is an improvement from 2^{128} to 2^{240}. PHOTON, likewise, consists of five sponge hash functions (with larger squeezing rate than absorbing rate), instantiated with a permutation of size $b \in \{100, 144, 196, 256, 288\}$, corresponding capacities $c \in \{80, 128, 160, 224, 256\}$, and with output size $\nu = c$. The squeezing rate differs for the five versions, but also here, a significant gain in the security bound is achieved: 2^{40} to 2^{64} for the smallest variant and 2^{128} to 2^{224} for the largest variant.

More recently, a notable example is Ascon-Hash, the hash function in the Ascon [DEMS21a] suite, the winner of the NIST Lightweight Cryptography competition [Nat19]. Ascon-Hash is a plain sponge construction on top of a $b = 320$-bit permutation, with a capacity $c = 256$ and a rate $r' = 64$. It outputs digests of size $\nu = 256$ bits, which are thus generated in four squeezes. In this case, the old bound of (4.2) implied generic preimage security up to 2^{128} work, whereas our new bound (4.3) implies generic preimage security up to 2^{192} work. A similar effect is achieved for the modes of other second round and final candidates in the NIST Lightweight Cryptography competition, such as ACE [AAG+19], KNOT [ZDY+19], SKINNY-HASH [BJK+19], Subterranean 2.0 [DMR19], the hash proposal of Isap [DEM+19], and PHOTON-Beetle [BCD+19]. These sponge-based functions all have their parameters (c, r', ν) satisfying $\nu - r' > c/2$.

In Table 4.1, we give a summary of these hash function constructions, and show how the new preimage security bound improves over the earlier bound. A more detailed evaluation of our new bound for SHA3-256, Spongent with $\nu = 256$, and Ascon-Hash with $\nu = 256$ is given in Section 4.4. We remark that in Table 4.1, we did not include hash functions that are sponge(-like) but squeeze digests in one round, such as Grindahl [KRT07] and Cube-Hash [Ber09], as our bound only improves over the state-of-the-art bound for sponge(-like) constructions that squeeze their digest in multiple rounds. Likewise, we did not include hash functions that squeeze digests over multi-

Table 4.1: Preimage security of the modes of SHA-3 (added for reference only, as our bound does not improve the state-of-the-art bound) and selected lightweight hash functions. Security bounds only hold under the assumption that the underlying permutations are ideal. "LWC Rx" indicates that the mode is used by a round x candidate in the NIST Lightweight Cryptography competition.

Scheme	Parameters						Security bound		Note
	b	c	r	r'	ν	ℓ	Old (4.2)	New (4.3)	
SHA3-ν	1600	448	1152	1152	224	1	2^{224}	2^{224}	SHA-3
	1600	512	1088	1088	256	1	2^{256}	2^{256}	standard [Nat15b]
	1600	768	832	832	384	1	2^{384}	2^{384}	
	1600	1024	576	576	512	1	2^{512}	2^{512}	
Spongent	88	80	8	8	88	11	2^{40}	2^{80}	ISO/IEC
	136	128	8	8	128	16	2^{64}	2^{120}	standard [BKL+11]
	176	160	16	16	160	10	2^{80}	2^{144}	
	240	224	16	16	224	14	2^{112}	2^{208}	
	272	256	16	16	256	16	2^{128}	2^{240}	
PHOTON	100	80	20	16	80	5	2^{40}	2^{64}	ISO/IEC
	144	128	16	16	128	8	2^{64}	2^{112}	standard [GPP11]
	196	160	36	36	160	5	2^{80}	2^{124}	
	256	224	32	32	224	7	2^{112}	2^{192}	
	288	256	32	32	256	8	2^{128}	2^{224}	
U-QUARK	136	128	8	8	128	16	2^{64}	2^{120}	[AHMN10]
D-QUARK	176	160	16	16	160	10	2^{80}	2^{144}	
T-QUARK	256	224	32	32	224	7	2^{112}	2^{192}	
ACE-Hash	320	256	64	64	256	4	2^{128}	2^{192}	LWC R2 [AAG+19]
KNOT Hash	256	224	32	128	256	2	2^{112}	2^{128}	LWC R2 [ZDY+19]
	384	256	128	128	256	2	2^{128}	2^{128}	
	384	336	48	192	384	2	2^{168}	2^{192}	
	512	448	64	256	512	2	2^{224}	2^{256}	
SKINNY-tk2-Hash	256	224	32	128	256	2	2^{112}	2^{128}	LWC R2 [BJK+19]
Subterranean 2.0	257	248	9	32	256	8	2^{124}	2^{224}	LWC R2 [DMR19]
Ascon-Hash	320	256	64	64	256	4	2^{128}	2^{192}	LWC winner [DEMS21a]
PHOTON-Beetle-Hash	256	224	32	128	256	2	2^{112}	2^{128}	LWC R3 [BCD+19]

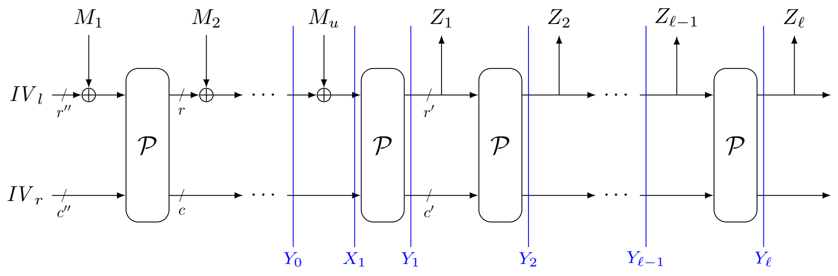


Figure 4.1: Illustration of the sponge construction with intermediate states.

ple rounds but that have a large enough c such that $\nu - r' \leq c/2$, such as Gimli [BKL+19], ESCH [BBC+19], and Xoodyak [DHP+21].

4.2 State-of-the-Art Generic Security Results

We will discuss the best known security lower bound in Section 4.2.1 and the best known generic attack in Section 4.2.2.

4.2.1 Security Lower Bound

We already discussed indifferentiability in Section 2.4.1.2. As a brief reminder, a hash function $\mathcal{H}$ based on an ideal permutation $\mathcal{P}$ is said to be indifferentiable from a random oracle $\mathcal{RO}$ if there exists a simulator $\mathbf{S}$ such that $(\mathcal{H}^{\mathcal{P}}, \mathcal{P})$ is hard to distinguish from $(\mathcal{RO}, \mathbf{S}^{\mathcal{RO}})$. Bertoni et al. [BDPV08] proved that the sponge is indifferentiable from a random oracle up to bound

$$\mathbf{Adv}^{\text{indif}}_{\text{Sponge},\mathbf{S}}(\mathcal{N}) \leq \frac{\mathcal{N}(\mathcal{N}+1)}{2^c},$$

where $\mathcal{N}$ denotes the total query complexity of the adversary. This result is formally restated in Theorem 3.1.1. Naito and Ohta [NO14] proved that for the PHOTON construction, indifferentiability holds with a bound of the form

$$\tilde{\mathcal{O}}\left(\frac{\mathcal{N}^2}{2^c} + \frac{\mathcal{N} \cdot \text{mucol}(\mathcal{N}, 2^{r'})}{2^{c'}} + \frac{\mathcal{N}}{2^{c''}}\right).$$

Indifferentiability of a hash function $\mathcal{H}$ from a random oracle $\mathcal{R}$ in words means that the hash function "behaves" like a random oracle. In the context of preimage resistance, this means that (see Lemma 2.4.1)

$$\mathbf{Adv}_{\mathcal{H}}^{\mathrm{ePre}}(\mathcal{N}) \leq \mathbf{Adv}_{\mathcal{H}}^{\mathrm{indif}}(\mathcal{N}) + \mathbf{Adv}_{\mathcal{R}}^{\mathrm{ePre}}(\mathcal{N}),$$

where we are slightly abusing notation for the latter term: to be precise, for $\mathbf{Adv}_{\mathcal{R}}^{\mathrm{ePre}}(\mathcal{N})$ we consider the adversary to have query access to the random oracle $\mathcal{R}$ and its goal is to output a message M such that $\mathcal{R}(M) = Z$ for the predetermined Z. Clearly, $\mathbf{Adv}_{\mathcal{R}}^{\mathrm{ePre}}(\mathcal{N}) = \mathcal{N}/2^{\nu}$, and we thus obtain the state-of-the-art bound for preimage resistance of the sponge, as stated in Corollary 3.1.1.1:

$$\mathbf{Adv}_{\mathrm{Sponge}}^{\mathrm{ePre}}(\mathcal{N}) \leq \frac{\mathcal{N}(\mathcal{N}+1)}{2^{c}} + \frac{\mathcal{N}}{2^{\nu}}. \tag{4.4}$$

Note that this is the bound that supports the complexity estimation given in (4.2): generically finding a preimage for $\mathrm{Sponge}^{\mathcal{P}}$ requires at least $\min\{2^{c/2}, 2^{\nu}\}$ evaluations.

4.2.2 Security Upper Bound

The best known attack, however, does not meet the first term of (4.4). In this section, we describe the best known preimage attack against the sponge construction. The attack de facto resembles the generic exhaustive preimage search attack and the attack that the sponge developers described in [BDPV07].

Proposition 4.2.1 ([BDPV07])**.** *Let $b, c, r, r', c', r'', c'', \nu, \mathcal{N} \in \mathbb{N}^*$ with $b = r + c = r' + c' = r'' + c''$ and $\nu \leq b$. Consider the sponge construction of Algorithm 1 with parameters $b, c, r, r', c', r'', c''$. There exists an adversary $\mathcal{A}$ with*

$$\mathcal{N} \approx \min\left\{\max\left\{2^{\nu-r'}, \min\left\{2^{c/2}, 2^{c''}\right\}\right\}, 2^{\nu}\right\},$$

such that

$$\mathbf{Adv}_{\mathrm{Sponge}}^{\mathrm{ePre}[\nu]}(\mathcal{A}) \approx 1.$$

Proof. Let $Z \in \{0,1\}^{\nu}$ be any given image. W.l.o.g., we assume the minimal padding $pad_{r'',r}^{10^*}$. We make a case distinction depending on the values c, c'', ν. Denote by $\mathcal{H}^{\mathcal{P}}(\cdot)$ the function $\mathrm{Sponge}^{\mathcal{P}}(\cdot, \nu)$. As we will focus on tightness up to constant, we will sometimes ignore the fact that any sponge evaluation of a message M of length α costs $\alpha + \ell$ permutation calls, and simply count any such evaluation as 1 query.

Case $\nu \leq \min\{c/2, c''\}$. The adversary fixes a message M and queries it to the construction. The query satisfies $\mathcal{H}^{\mathcal{P}}(M) = Z$ with probability around $1/2^{\nu}$. After $\mathcal{N} \approx 2^{\nu}$ attempts, the adversary has with high probability found a preimage M.

Case $\min\{c/2, c''\} < \nu$. The attack consists of two sequential parts.

- First, the adversary fixes a state value Y_1 such that $Z_1 = \text{outer}_{r'}(Y_1)$. It queries $Y_2 = \mathcal{P}(Y_1)$, $Y_3 = \mathcal{P}^2(Y_1)$, $\dots$, $Y_\ell = \mathcal{P}^{\ell-1}(Y_1)$. The queries satisfy

$$Z_i = \begin{cases} \text{outer}_{r'}(Y_i) & \text{for } i = 2, \dots, \ell-1\,, \\ \text{outer}_{\nu-(\ell-1)r'}(Y_i) & \text{for } i = \ell\,, \end{cases}$$

  with probability approximately

$$\left(\frac{1}{2^{r'}}\right)^{\ell-2} \cdot \frac{1}{2^{\nu-(\ell-1)r'}} = \frac{1}{2^{\nu-r'}}\,.$$

  After $\mathcal{N} \approx 2^{\nu-r'}$ attempts, the adversary has found a state value Y_1 such that ℓ squeezes result in Z;

- For the second part, we make another case distinction depending on the values c, c''.

  - Case $c/2 < c''$. Starting from IV, the adversary computes $Y_0^{\rightarrow} := \mathcal{P}(M_1\|0^{c''} \oplus IV)$ for $\mathcal{N}$ different values M_1. Starting from the value Y_1 found in the first part of the attack, it computes $Y_0^{\leftarrow} := \mathcal{P}^{-1}(\mathcal{P}^{-1}(Y_1) \oplus (M_3\|0^c))$ for $\mathcal{N}$ different non-zero values M_3. If $\mathcal{N} \approx 2^{c/2}$, there will with high probability be two values M_1, M_3 such that

$$\text{inner}_c(Y_0^{\rightarrow} \oplus Y_0^{\leftarrow}) = 0^c\,.$$

    Let $M_2 := \text{outer}_r(Y_0^{\rightarrow} \oplus Y_0^{\leftarrow})$. Define the preimage as the unique message M such that $pad^{10^*}_{r'',r}(M) = M_1\|M_2\|M_3$. We remark that if $r'' \leq c/2$ one may need multiple message blocks for both the forward and the inverse part in order to make $q \approx 2^{c/2}$ evaluations, but the attack works in a comparable way;

- Case $c'' \leq c/2$. The adversary computes $Y_0^{\leftarrow} := \mathcal{P}^{-1}(\mathcal{P}^{-1}(Y_1) \oplus (M_2 \| 0^c))$ for $\mathcal{N}$ different non-zero values M_2. If $\mathcal{N} \approx 2^{c''}$, there will with high probability be a value M_2 such that

$$\mathrm{inner}_{c''}(Y_0^{\leftarrow}) = \mathrm{inner}_{c''}(IV)\,.$$

Let $M_1 := \mathrm{outer}_r(Y_0^{\leftarrow} \oplus IV)$. Define the preimage as the unique message M such that $pad^{10^*}_{r'',r}(M) = M_1 \| M_2$.

In total, in this case the attack requires $\mathcal{N} \approx 2^{\nu - r'} + \min\left\{2^{c/2}, 2^{c''}\right\}$ evaluations. □

In general, the attack thus has a complexity of around

$$\mathcal{N} \approx \min\{2^{\nu - r'} + \min\{2^{c/2}, 2^{c''}\}, 2^{\nu}\}$$

evaluations. We remark that the generic second preimage attack, as described in Proposition 3.1.2, is basically a simplification of above preimage attack, where in the case of $c = c' = c''$ and $c/2 < \nu$, the attacker does not need to perform the first part of the attack but can rather compute Y_1 in a constant number of permutation evaluations from the first preimage. The generic collision attack, as described in Proposition 3.1.1, in turn differs from this second preimage attack in the sense that for $\nu/2 \leq c/2$ the attacker can perform exhaustive collision search in around $2^{\nu/2}$ evaluations (instead of 2^{ν}).

Remark 4.2.1. *In the case where $2^{c'} \leq 2^{\nu - r'}$, or equivalently, if $b \leq \nu$, it is possible that no state Y_1 exists that squeezes to the target digest Z, and in that case a preimage does not exist. However, if we look at domain-oriented preimage resistance, where we know that a preimage of size κ bits exists (Definition 2.4.2), then the situation differs. In that case, there exists an adversary that can find a preimage with high probability using approximately*

$$\mathcal{N} \approx \min\left\{2^{\kappa}, \max\left\{\min\left\{2^{c'}, 2^{\nu - r'}\right\}, \min\left\{2^{c/2}, 2^{c''}\right\}\right\}, 2^{\nu}\right\}\,,$$

queries, where $\mathbf{Adv}^{\mathrm{dPre}[\kappa,\,\nu]}_{\mathcal{RO}}(\mathcal{A}) \approx 1$, regardless of whether $\nu \leq b$ or $\nu > b$. Here, the term 2^{κ} corresponds to exhaustive search over the input domain, and $2^{c'}$ corresponds to exhausting all possible values for the capacity of the state Y_1.

4.3 Improved Preimage Resistance Lower Bound

In the following theorem, we state our main result.

Theorem 4.3.1. *Let $b, c, r, c', r', c'', r'', \nu, \mathcal{N} \in \mathbb{N}^*$ with $b = c+r = c'+r' = c''+r''$, and let $\ell := \left\lceil \frac{\nu}{r'} \right\rceil$. If $\mathcal{N} \leq 2^{c'-1}/3$ and $(\ell-1)^2 \leq 2^b$ the sponge construction of Algorithm 1 with parameters $b, c, r, r', c', r'', c''$ satisfies the following bound:*

$$\mathbf{Adv}^{\mathrm{ePre}}_{\mathrm{Sponge}}(\mathcal{N}) \leq \frac{4\mathcal{N}}{2^{\nu}} + \min\left\{ \frac{4\ell\mathcal{N}}{2^{\nu-r'}}, \frac{\mathcal{N}(\mathcal{N}-1)}{2^c} + \frac{2\mathcal{N}}{2^{c''}} \right\} . \qquad (4.5)$$

The proof is given in the remainder of this section. We note that the bound indeed matches the generic attack of Section 4.2.2 up to constant. In Section 4.4, we will discuss applications of this result, notably to domain-oriented preimage resistance, and will evaluate the bound of Theorem 4.3.1 more closely, and compare it with the generic attack of Section 4.2.2 and the state-of-the-art bound of Section 4.2.1.

4.3.1 Setup

Let $Z \in \{0,1\}^{\nu}$ be any image, and write $Z = Z_1 \| Z_2 \| \cdots \| Z_\ell$, where $|Z_i| = r'$ for $i \in \{1, \ldots, \ell-1\}$ and $|Z_\ell| = s \leq r'$. Consider any adversary $\mathcal{A}$ as in the theorem statement. We denote by $\mathcal{Q}$ its query history, which contains tuples of the form (X, Y, d), where $\mathcal{P}(X) = Y$, and the query was made in direction $\mathrm{d} \in \{\mathit{fwd}, \mathit{inv}\}$. To represent $\mathcal{A}$'s knowledge from $\mathcal{Q}$, we use a graph representation as done for example in [BDPV08, NO14]. Initially, the graph contains the nodes $\{0,1\}^b$, which represent all possible internal states of the sponge. For each query $(X, Y, \mathrm{d}) \in \mathcal{Q}$ with $d \in \{\mathrm{fwd}, \mathrm{inv}\}$ and $\mathrm{inner}_{c''}(X) = \mathrm{inner}_{c''}(IV)$, the edge $IV \xrightarrow{M} Y$ is added, where $M := \mathrm{outer}_{r''}(IV \oplus X)$. For the remainder queries $(X, Y, \mathrm{d}) \in \mathcal{Q}$ with $d \in \{\mathrm{fwd}, \mathrm{inv}\}$ and $\mathrm{inner}_{c''}(X) \neq \mathrm{inner}_{c''}(IV)$, and for any $M \in \{0,1\}^r$, the edge $Y' \xrightarrow{M} Y$ is added, where $Y' := X \oplus (M \| 0^c)$. Note that in the squeezing phase, such edge must appear for a zero-block message. In this case, the label is omitted. Let $\boldsymbol{Z}_i$ be defined as follows:

$$\boldsymbol{Z}_i := \begin{cases} \{Y_i \in \{0,1\}^b \mid \mathrm{outer}_{r'}(Y_i) = Z_i\} , & \text{for } i \in \{1, \ldots, \ell-1\} , \\ \{Y_i \in \{0,1\}^b \mid \mathrm{outer}_{s}(Y_i) = Z_i\} , & \text{for } i = \ell . \end{cases}$$

Then, the goal of $\mathcal{A}$ is to find a preimage of Z, which implies the following event $\mathsf{PRE}(\mathcal{Q})$:

$$\begin{aligned}\mathsf{PRE}(\mathcal{Q}) : \mathcal{Q} \text{ defines a path } IV \xrightarrow{M_1} \cdots \xrightarrow{M_{u-1}} Y_0 \xrightarrow{M_u} Y_1 \longrightarrow \cdots \longrightarrow Y_\ell \\ \text{such that } Y_i \in \boldsymbol{Z}_i \text{ for } i = 1, \ldots, \ell\,.\end{aligned}$$

We refer to Figure 4.1 for a depiction of these parameters. In the case of the minimal injective padding $pad^{10^*}_{r'',r}$, finding a preimage corresponds to $\mathsf{PRE}(\mathcal{Q})$ with the restriction that the last message block is not zero. In such case, the preimage found by $\mathcal{A}$ is the unique message M such that $pad^{10^*}_{r'',r}(M) = M_1\|\cdots\|M_u$.

Before proceeding to the next steps of the proof, remark that, given $k, n \in \mathbb{N}^*$ such that $k^2 \leq n$, we have

$$\begin{aligned}[n]_k &= n^k \prod_{i=0}^{k-1} \frac{n-i}{n} \\ &\geq n^k e^{\sum_{i=0}^{k-1} \frac{-i}{n-i}} \\ &\geq n^k e^{\sum_{i=0}^{k-1} \frac{-i}{n-k}} \\ &= n^k e^{\frac{-k(k-1)}{2(n-k)}} \\ &\geq n^k e^{-1/2} \\ &\geq \frac{n^k}{2}\,, \end{aligned} \tag{4.6}$$

where the first inequality uses $1 + x \leq e^x$ applied with $x = \frac{i}{n-i}$.

4.3.2 Logic

We separate the event $\mathsf{PRE}(\mathcal{Q})$ as the disjoint union of the following two events:

$$\begin{aligned}\mathsf{PREFWD}(\mathcal{Q}) :\ & \mathsf{PRE}(\mathcal{Q}) \text{ with the restriction that the query} \\ & \text{linking } Y_0 \text{ and } Y_1 \text{ must be made in forward direction}\,, \\ \mathsf{PREINV}(\mathcal{Q}) :\ & \mathsf{PRE}(\mathcal{Q}) \text{ with the restriction that the query} \\ & \text{linking } Y_0 \text{ and } Y_1 \text{ must be made in inverse direction}\,.\end{aligned}$$

Clearly,

$$\mathsf{PRE}(\mathcal{Q}) \iff \mathsf{PREFWD}(\mathcal{Q}) \vee \mathsf{PREINV}(\mathcal{Q})\,. \tag{4.7}$$

We will consider dedicated trigger points for $\mathsf{PREFWD}(\mathcal{Q})$ and $\mathsf{PREINV}(\mathcal{Q})$. Let $\mathcal{S} = \{Y_1 \mid \mathcal{P}^{i-1}(Y_1) \in \boldsymbol{Z}_i \text{ for all } i \in \{1, \dots, \ell\}\} \subseteq \boldsymbol{Z}_1$. We define the set $\mathcal{S}_{\text{fwd}}$ and the multiset $\mathcal{S}_{\text{inv}}$ as follows:

$$\begin{aligned} \mathcal{S}_{\text{fwd}} &= \{\mathcal{P}^{-1}(Y_1) \mid Y_1 \in \mathcal{S}\}\,, \\ \mathcal{S}_{\text{inv}} &= \{Y_1, \mathcal{P}(Y_1), \dots, \mathcal{P}^{\ell-1}(Y_1) \mid Y_1 \in \mathcal{S}\}\,. \end{aligned}$$

Intuitively, $\mathcal{S}_{\text{inv}}$ includes all the nodes $Y_1, \dots, Y_\ell$ appearing in paths which set $\mathsf{PRE}(\mathcal{Q})$ if discovered, while $\mathcal{S}_{\text{fwd}}$ captures all values X_1 from which such path $Y_1 \to \cdots \to Y_\ell$ starts. Looking ahead, $\mathcal{S}_{\text{fwd}}$ includes the set of trigger points for $\mathsf{PREFWD}(\mathcal{Q})$, and $\mathcal{S}_{\text{inv}}$, as a multiset, includes the set of trigger points for $\mathsf{PREINV}(\mathcal{Q})$. These trigger points can be repeated in $\mathcal{S}_{\text{inv}}$ when some values Z_i are colliding. (Based on this, we will typically simply refer to $\mathcal{S}_{\text{inv}}$ as a set.) We now introduce the following three events:

$$\begin{aligned} \mathsf{BADFWD}(\mathcal{Q}) &: \exists (X, Y, \text{fwd}) \in \mathcal{Q} \text{ such that } X \in \mathcal{S}_{\text{fwd}}\,, \\ \mathsf{BADINV}(\mathcal{Q}) &: \exists (X, Y, \text{fwd}) \in \mathcal{Q} \text{ such that } X \in \mathcal{S}_{\text{inv}} \text{ or} \\ &\quad\ \exists (X, Y, \text{inv}) \in \mathcal{Q} \text{ such that } Y \in \mathcal{S}_{\text{inv}}\,, \\ \mathsf{INNER}(\mathcal{Q}) &: \exists (X, Y, \text{fwd}), (X', Y', \text{inv}) \in \mathcal{Q} \text{ such that } \text{inner}_c(Y) = \text{inner}_c(X') \\ &\text{or } \exists (X, Y, \text{inv}) \in \mathcal{Q} \text{ such that } \text{inner}_{c''}(X) = \text{inner}_{c''}(IV)\,. \end{aligned}$$

Intuitively, for $\mathsf{PREFWD}(\mathcal{Q})$ to be set, the adversary must among others make a query $\mathcal{P}(X_1)$ with $X_1 \in \mathcal{S}_{\text{fwd}}$. Thus, $\mathsf{PREFWD}(\mathcal{Q})$ implies $\mathsf{BADFWD}(\mathcal{Q})$. Likewise, for $\mathsf{PREINV}(\mathcal{Q})$ to be set, the adversary must ever make *a* query that appears in *a* path $Y_1 \to \cdots \to Y_\ell$ in *a* query direction. In other words, it must ever query a value in $\mathcal{S}_{\text{inv}}$. Hence, $\mathsf{PREINV}(\mathcal{Q})$ implies $\mathsf{BADINV}(\mathcal{Q})$. Moreover, given that $\mathsf{PREINV}(\mathcal{Q})$ defines a path starting from IV that contains an edge between Y_0 and Y_1 corresponding to an inverse query, *somewhere* in this path from IV to Y_1 there must be a collision between an inverse query and a descendant of IV. This means that $\mathsf{PREINV}(\mathcal{Q})$ also implies $\mathsf{INNER}(\mathcal{Q})$. More formally:

$$\begin{aligned} \mathsf{PREFWD}(\mathcal{Q}) &\implies \mathsf{BADFWD}(\mathcal{Q})\,, & (4.8) \\ \mathsf{PREINV}(\mathcal{Q}) &\implies \mathsf{BADINV}(\mathcal{Q}) \wedge \mathsf{INNER}(\mathcal{Q})\,. & (4.9) \end{aligned}$$

From (4.7) to (4.9), we logically obtain

$$\mathsf{PRE}(\mathcal{Q}) \implies \mathsf{BADFWD}(\mathcal{Q}) \vee (\mathsf{BADINV}(\mathcal{Q}) \wedge \mathsf{INNER}(\mathcal{Q}))\,, \qquad (4.10)$$

and thus

$$\begin{aligned}\mathbf{Pr}\left(\mathsf{PRE}(\mathcal{Q})\right) \leq \mathbf{Pr}\left(\mathsf{BADFWD}(\mathcal{Q})\right) \\ + \min\left\{\mathbf{Pr}\left(\mathsf{BADINV}(\mathcal{Q})\right), \mathbf{Pr}\left(\mathsf{INNER}(\mathcal{Q})\right)\right\} . \quad (4.11)\end{aligned}$$

4.3.3 Probability computation

We upper bound the three probabilities of (4.11), starting with $\mathbf{Pr}\left(\mathsf{INNER}(\mathcal{Q})\right)$ in Lemma 4.3.2, then $\mathbf{Pr}\left(\mathsf{BADFWD}(\mathcal{Q})\right)$ in Lemma 4.3.3, and terminate with $\mathbf{Pr}\left(\mathsf{BADINV}(\mathcal{Q})\right)$ in Lemma 4.3.4.

Lemma 4.3.2. *We have*

$$\mathbf{Pr}\left(\mathsf{INNER}(\mathcal{Q})\right) \leq \frac{\mathcal{N}(\mathcal{N}-1)}{2^c} + \frac{2\mathcal{N}}{2^{c''}} . \quad (4.12)$$

Proof of Lemma 4.3.2. We index the queries by the query number, i.e., the i^{th} query is denoted by (X^i, Y^i, d^i). $\mathsf{INNER}(\mathcal{Q})$ translates to the fact that either there is an inner collision between the set of forward and inverse queries, or that the output of an inverse query inner collides with $\mathrm{inner}_{c''}(IV)$. More formally, this implies that there exists $i \in \{1, \dots, \mathcal{N}\}$ such that one of the following two events happens:

$$\begin{aligned}\mathsf{HIT}_i^{\mathrm{fwd}}(\mathcal{Q}) :\ & (X^i, Y^i, \mathrm{fwd}) \in \mathcal{Q} \text{ and} \\ & \mathrm{inner}_c(Y^i) \in \{\mathrm{inner}_c(X^1), \dots, \mathrm{inner}_c(X^{i-1})\} , \\ \mathsf{HIT}_i^{\mathrm{inv}}(\mathcal{Q}) :\ & (X^i, Y^i, \mathrm{inv}) \in \mathcal{Q} \text{ and} \\ & \text{either } \mathrm{inner}_c(X^i) \in \{\mathrm{inner}_c(Y^1), \dots, \mathrm{inner}_c(Y^{i-1})\} \\ & \text{or } \mathrm{inner}_{c''}(X^i) = \mathrm{inner}_{c''}(IV) .\end{aligned}$$

By basic probability theory,

$$\begin{aligned}\mathbf{Pr}\left(\mathsf{INNER}(\mathcal{Q})\right) &\leq \sum_{i=1}^{\mathcal{N}} \mathbf{Pr}\left(\mathsf{INNER}(\mathcal{Q}_i) \wedge \neg\mathsf{INNER}(\mathcal{Q}_{i-1})\right) \\ &\leq \sum_{i=1}^{\mathcal{N}} \mathbf{Pr}\left(\mathsf{HIT}_i^{\mathrm{fwd}}(\mathcal{Q}_i) \vee \mathsf{HIT}_i^{\mathrm{inv}}(\mathcal{Q}_i) \;\middle|\; \neg\mathsf{INNER}(\mathcal{Q}_{i-1})\right) .\end{aligned}$$

For any i, the query is either in forward direction or in inverse direction, so it can only set one of the two events. The response at the i^{th} query is uniformly drawn from a set of size at least $2^b - \mathcal{N}$, among which at most $(i-1)2^r + 2^{r''}$ elements set $\mathsf{HIT}_i^{\mathrm{fwd}}(\mathcal{Q}_i)$ or $\mathsf{HIT}_i^{\mathrm{inv}}(\mathcal{Q}_i)$. Thus:

$$\mathbf{Pr}\left(\mathsf{HIT}_i^{\mathrm{fwd}}(\mathcal{Q}_i) \vee \mathsf{HIT}_i^{\mathrm{inv}}(\mathcal{Q}_i) \;\middle|\; \neg\mathsf{INNER}(\mathcal{Q}_{i-1})\right) \leq \frac{(i-1)2^r}{2^b - \mathcal{N}} + \frac{2^{r''}}{2^b - \mathcal{N}} \,.$$

Then, as $\mathcal{N} \leq 2^{b-1}$,

$$\mathbf{Pr}\left(\mathsf{INNER}(\mathcal{Q})\right) \leq \sum_{i=1}^{\mathcal{N}} \left(\frac{2(i-1)2^r}{2^b} + \frac{2^{r''+1}}{2^b} \right) \leq \frac{\mathcal{N}(\mathcal{N}-1)}{2^c} + \frac{2\mathcal{N}}{2^{c''}} \,. \qquad \square$$

Lemma 4.3.3. *We have*

$$\mathbf{Pr}\left(\mathsf{BADFWD}(\mathcal{Q})\right) \leq \frac{4\mathcal{N}}{2^\nu} \,. \tag{4.13}$$

Proof of Lemma 4.3.3. By basic probability theory,

$$\begin{aligned}&\mathbf{Pr}\left(\mathsf{BADFWD}(\mathcal{Q})\right) \\ &\qquad = \sum_{y=1}^{2^{c'}} \mathbf{Pr}\left(\mathsf{BADFWD}(\mathcal{Q}) \mid |\mathcal{S}_{\mathrm{fwd}}| = y\right) \cdot \mathbf{Pr}\left(|\mathcal{S}_{\mathrm{fwd}}| = y\right) \,. \end{aligned} \tag{4.14}$$

We start by upper bounding the probability of the conditioned $\mathsf{BADFWD}(\mathcal{Q})$ event for any $y = 1, \ldots, 2^{c'}$, which is similar to a guessing game: in order to win, the adversary must guess $X_1 \in \mathcal{S}_{\mathrm{fwd}}$ with a forward query. We start by remarking that $\mathcal{S}_{\mathrm{fwd}}$ is defined via inverse $\mathcal{P}$-calls, and that the adversary has no a priori knowledge about those. Thus, one single query $\mathcal{P}(X)$ from the adversary succeeds with probability at most $\frac{y}{2^b}$. Moreover, one query eliminates at most one candidate: if the query (X, Y, d) does not set $\mathsf{BADFWD}(\mathcal{Q})$, then X can be removed from the set of candidates values to be in $\mathcal{S}_{\mathrm{inv}}$. If additionally $\mathrm{d} = \mathrm{inv}$ and $X \in S_{\mathrm{fwd}}$, the adversary cannot guess this value anymore. Thus, defining $\mathsf{BADFWD}(\mathcal{Q}_0) := \perp$,

$$\begin{aligned}&\mathbf{Pr}\left(\mathsf{BADFWD}(\mathcal{Q}) \mid |\mathcal{S}_{\mathrm{fwd}}| = y\right) \\ &\qquad \leq \sum_{i=1}^{\mathcal{N}} \mathbf{Pr}\left(\mathsf{BADFWD}(\mathcal{Q}_i) \mid |\mathcal{S}_{\mathrm{fwd}}| = y \wedge \neg\mathsf{BADFWD}(\mathcal{Q}_{i-1})\right) \\ &\qquad \leq \sum_{i=1}^{\mathcal{N}} \frac{y}{2^b - i + 1} \leq \frac{y\mathcal{N}}{2^b - \mathcal{N}} \leq 2\frac{y\mathcal{N}}{2^b} \,, \end{aligned} \tag{4.15}$$

where in the last inequality, we used $\mathcal{N} \leq 2^{b-1}$.

Plugging this bound into (4.14) gives

$$\begin{aligned}\mathbf{Pr}\left(\mathsf{BADFWD}(\mathcal{Q})\right) &\leq \frac{2\mathcal{N}}{2^b}\sum_{y=1}^{2^{c'}} y \cdot \mathbf{Pr}\left(|\mathcal{S}_{\text{fwd}}| = y\right) \\ &\leq \frac{2\mathcal{N}}{2^b}\mathsf{E}\left(|\mathcal{S}_{\text{fwd}}|\right) . \end{aligned} \tag{4.16}$$

It remains to compute $\mathsf{E}\left(|\mathcal{S}_{\text{fwd}}|\right) = \mathsf{E}\left(|\mathcal{S}|\right)$. For any $Y \in \{0,1\}^b$, define Bernoulli variable I_Y as

$$I_Y = 1 \iff Y \in \mathcal{S} .$$

Note that $I_Y = 0$ whenever $Y \notin \boldsymbol{Z}_1$. We have

$$\begin{aligned}\mathsf{E}\left(|\mathcal{S}|\right) &= \mathsf{E}\left(\sum_{Y\in\{0,1\}^b} I_Y\right) \\ &= \sum_{Y\in\boldsymbol{Z}_1} \mathsf{E}\left(I_Y\right) \\ &= \sum_{Y\in\boldsymbol{Z}_1} \mathbf{Pr}\left(Y \in \mathcal{S}\right) \\ &\leq \sum_{Y\in\boldsymbol{Z}_1} \frac{2^{c'}}{2^b}\frac{2^{c'}}{2^b-1}\cdots\frac{2^{b-s}}{2^b-(\ell-2)} \\ &= \frac{(2^{c'})^{\ell-1}\cdot 2^{b-s}}{[2^b]_{\ell-1}} \\ &\leq 2\frac{(2^{c'})^{\ell-1}\cdot 2^{b-s}}{(2^b)^{\ell-1}} ,\end{aligned}$$

where the last inequality uses (4.6). Therefore,

$$\mathsf{E}\left(|\mathcal{S}|\right) \leq 2\frac{2^b}{2^\nu} . \tag{4.17}$$

Finally, from (4.16) and (4.17), we thus obtain

$$\mathbf{Pr}\left(\mathsf{BADFWD}(\mathcal{Q})\right) \leq \frac{4\mathcal{N}}{2^\nu} , \tag{4.18}$$

which completes the proof. □

Lemma 4.3.4. *We have*

$$\mathbf{Pr}\left(\mathsf{BADINV}(\mathcal{Q})\right) \leq \frac{4\ell\mathcal{N}}{2^{\nu-r'}} \,. \tag{4.19}$$

Proof of Lemma 4.3.4. We first note that if $\ell = 1$, $|\mathcal{S}_{\mathrm{inv}}| = 2^{c'} \leq 2^{b-\nu}$, and the result will be meaningless as $\mathsf{BADINV}(\mathcal{Q})$ can be set with probability 1. We will henceforth focus on the case $\ell \geq 2$.

Similar to the proof of Lemma 4.3.3, by basic probability we obtain

$$\mathbf{Pr}\left(\mathsf{BADINV}(\mathcal{Q})\right) = \sum_{y=1}^{2^{c'}} \mathbf{Pr}\left(\mathsf{BADINV}(\mathcal{Q}) \mid |\mathcal{S}_{\mathrm{inv}}| = \ell y\right) \cdot \mathbf{Pr}\left(|\mathcal{S}_{\mathrm{inv}}| = \ell y\right) \,, \tag{4.20}$$

where we used that $\mathcal{S}_{\mathrm{inv}}$ is a multiset with a size multiple of ℓ. We start by investigating the conditioned $\mathsf{BADINV}(\mathcal{Q})$ event for any $y = 1, \ldots, 2^{c'}$, which is more involved than $\mathsf{BADFWD}(\mathcal{Q})$ studied in Lemma 4.3.3. Because of the condition $|\mathcal{S}_{\mathrm{inv}}| = \ell y$, there are y paths $Y_1 \to \cdots \to Y_\ell$ with $Y_i \in \boldsymbol{Z}_i$ for $i = 1, \ldots, \ell$. The adversary wins if it ever queries a value that is on any of these paths. Note that this is different from the proof of Lemma 4.3.3, where the adversary had to guess any starting point of a path. In the current setting, the attacker learns additional information of failed attempts. For example, suppose that $Z_1 \neq Z_2$ and the adversary makes a forward query $\mathcal{P}(X) = Y$, where $X \in \boldsymbol{Z}_1$ and $Y \in \boldsymbol{Z}_2$ but which does *not* set $\mathsf{BADINV}(\mathcal{Q})$. As it does not set $\mathsf{BADINV}(\mathcal{Q})$, the adversary knows that querying $\mathcal{P}(Y)$ (i.e., guessing $Y \in \boldsymbol{Z}_2$ as candidate value for a chain) is fruitless.

To simplify our reasoning, we will be more generous to the adversary, and for each query input X that the adversary makes, it receives both the forward evaluation $\mathcal{P}(X)$ and the inverse evaluation $\mathcal{P}^{-1}(X)$. Stated differently, for the current game the query direction does not matter, and for each attempt X it learns $\mathcal{P}^{-1}(X) \to X \to \mathcal{P}(X)$. The adversary wins if this is a proper subpath of any of the y target paths $X_1 \to Y_1 \to \cdots \to Y_\ell \to Y_{\ell+1}$, where $X_1 = \mathcal{P}^{-1}(Y_1)$ and $Y_{\ell+1} = \mathcal{P}(Y_\ell)$.[1]

A visualization of this game is given in Figure 4.2 for $\ell = 4$. For this example, recall that for $i = 1, 2, 3$, $\boldsymbol{Z}_i$ consists of all values $Y \in \{0,1\}^b$ such that $\mathrm{outer}_{r'}(Y) = Z_i$, and that $\boldsymbol{Z}_4$ consists of all values $Y \in \{0,1\}^b$ such that

[1]The usage of parameter X_1 in this path, as opposed to Y_0, appears illogical at first sight, but fits the parameter definitions as outlined in Figure 4.1.

$\text{outer}_s(Y) = Z_4$. By the conditioned event, there exist y paths through the sets $\boldsymbol{Z}_1, \ldots, \boldsymbol{Z}_4$. In the example, $y = 2$, hence there are two such paths. The adversary sets $\mathsf{BADINV}(\mathcal{Q})$ if and only if it ever queries one of the *at most* ℓy nodes on these lines (not including the ones on the outer shores $\{0,1\}^b$). It is noteworthy that these paths are disjoint: they never cross the same node in the same shore.

Now, suppose the adversary makes a query X, it thus results in a path $\mathcal{P}^{-1}(X) \to X \to \mathcal{P}(X)$. The query is considered a failed query if it is not a proper subpath of any of the y paths. In particular, a failed query either does not intersect with any of the y paths, *or* it intersects with one of the y paths at their very ends. In other words, a query is considered a failed one for path $X_1 \to Y_1 \to \cdots \to Y_\ell \to Y_{\ell+1}$ if and only if it intersects with either of $\varnothing, X_1, X_1 \to Y_1, Y_\ell \to Y_{\ell+1}, Y_\ell$, as illustrated in Figure 4.2 (up to symmetry). Any other intersection of the failed query result with the path is impossible, as e.g., depicted in Figure 4.2, due to the fact that $\mathcal{P}$ is a permutation.

We remark that for $0 \leq i < j \leq \ell + 1$, a query can be successful for one target path at position i, and failed for another target path at position j at the same time. In this case, the adversary is nevertheless successful. From this, we can conclude that any query attempt either is successful or it eliminates at most 3 possible values from further guessing.

In summary, to win, the adversary must make a query in $\mathcal{S}_{\text{inv}}$. This set is of size at most ℓy and is a subset of the set $\bigcup_{i=1}^{\ell} \boldsymbol{Z}_i$ of size at least $2^{c'}$.[2] Since one query eliminates at most 3 candidates and this is the only information available for the adversary, after $i-1$ unsuccessful attempts, the i^{th} attempt succeeds with probability at most $\frac{\ell y}{2^{c'}-3(i-1)}$. Thus, defining $\mathsf{BADINV}(\mathcal{Q}_0) := \bot$,

$$\begin{aligned}
&\mathbf{Pr}\left(\mathsf{BADINV}(\mathcal{Q}) \mid |\mathcal{S}_{\text{inv}}| = \ell y\right) \\
&\qquad = \sum_{i=1}^{\mathcal{N}} \mathbf{Pr}\left(\mathsf{BADINV}(\mathcal{Q}_i) \mid |\mathcal{S}_{\text{inv}}| = \ell y \wedge \neg\mathsf{BADINV}(\mathcal{Q}_{i-1})\right) \\
&\qquad \leq \sum_{i=1}^{\mathcal{N}} \frac{\ell y}{2^{c'} - 3(i-1)} \leq \frac{\ell y \mathcal{N}}{2^{c'} - 3\mathcal{N}} \leq 2\frac{\ell y \mathcal{N}}{2^{c'}}\,, \qquad (4.21)
\end{aligned}$$

where in the last inequality, we used $\mathcal{N} \leq 2^{c'-1}/3$.

Now, it remains to plug the bound into (4.20). We can copy the analysis

[2]Note that this correctly captures the case $i = \ell$, as $|\boldsymbol{Z}_\ell| = 2^{\nu-s} \geq 2^{c'}$.

of Lemma 4.3.3 verbatim and obtain

$$\begin{aligned}\mathbf{Pr}\left(\mathsf{BADINV}(\mathcal{Q})\right) &\leq 2\frac{\mathcal{N}}{2^{c'}}\sum_{y=1}^{2^{c'}}\ell y \cdot \mathbf{Pr}\left(|\mathcal{S}_{\text{inv}}| = \ell y\right)\\ &\leq 2\frac{\mathcal{N}}{2^{c'}}\mathsf{E}\left(|\mathcal{S}_{\text{inv}}|\right)\\ &\leq \frac{4\ell\mathcal{N}}{2^{\nu-r'}}\,, \qquad (4.22)\end{aligned}$$

where the last inequality uses $|\mathcal{S}_{\text{inv}}| = \ell|\mathcal{S}|$ and (4.17). This completes the proof. $\square$

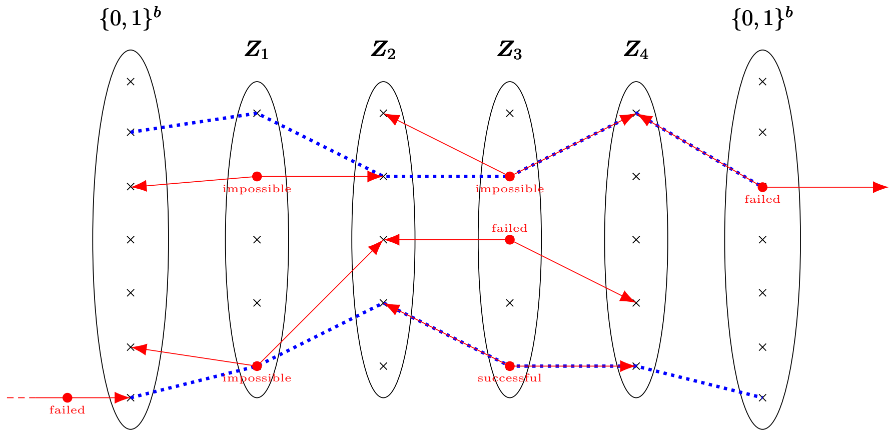


Figure 4.2: Illustration of the conditioned $\mathsf{BADINV}(\mathcal{Q})$ event for $\ell = 4$ and $y = 2$. To win, the adversary must guess any node from $\bigcup_{i=1}^{4} \boldsymbol{Z}_i$ on any of the $y = 2$ dotted blue paths. A query attempt X is successful if and only if $\mathcal{P}^{-1}(X) \to X \to \mathcal{P}(X)$ (depicted solid red) is a proper subpath of any of the blue lines. As $\mathcal{P}$ is a permutation, a failed query attempt is either non-overlapping with any of the dotted blue paths, or it may partially overlap only at the ends of a blue line, the other cases are impossible and illustrated as such.

Remark 4.3.1. *In the proof of Lemma 4.3.4, we bounded the probability that the* i^{th} *query is successful by* $\frac{y}{2^{c'}-3(i-1)}$. *There is a small loss in this bound due to various simplifications we had to make. First note that as we provide* *the adversary both directions of the queries, we slightly increase its knowledge and thus success probability. In addition, each query attempt X has its leftmost r' bits $\mathrm{outer}_{r'}(X)$ fixed, and the adversary thus commits itself to the value Z_i and thus to the position in Figure 4.2 the query could occur. However, there is no way to make use of this property, as in the general case, the values $Z_1, \ldots, Z_\ell$ may be equal, and the query can nevertheless occur at multiple positions. Finally, in the specific case where some values Z_i are mutually equal, this basically reduces the set of candidates to be in $\mathcal{S}$, thus also the set of possibly successful values, and possibly also the amount of information the adversary learns from a failed attempt.*

4.3.4 Conclusion

From (4.11), Lemma 4.3.2, Lemma 4.3.3, and Lemma 4.3.4, we obtain

$$\mathbf{Pr}\left(\mathsf{PRE}(\mathcal{Q})\right) \leq \frac{4\mathcal{N}}{2^\nu} + \min\left\{\frac{4\ell\mathcal{N}}{2^{\nu-r'}}, \frac{\mathcal{N}(\mathcal{N}-1)}{2^c} + \frac{2\mathcal{N}}{2^{c''}}\right\}, \tag{4.23}$$

and this completes the proof of Theorem 4.3.1.

4.4 Concluding Remarks

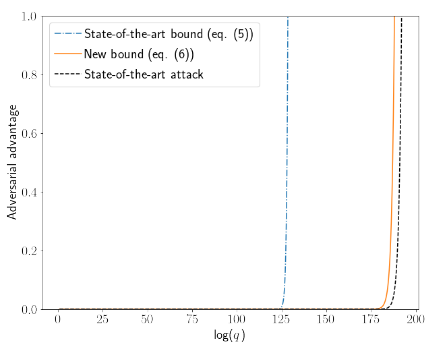
In Section 4.4.1, we compare our result with the state-of-the-art bound of (4.4) (Section 4.2.1) and the best existing attack. Finally, in Section 4.4.2 we discuss the broader applicability of our result. Throughout this section, for simplicity we restrict ourselves to the case where $r'' = r$.

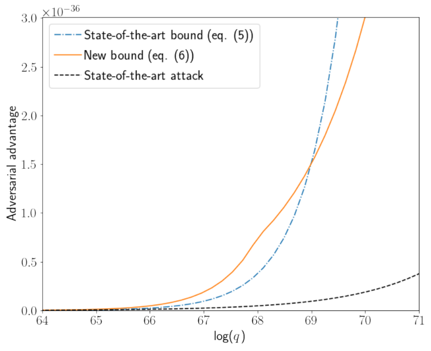
4.4.1 Comparisons

Recall that in Theorem 4.3.1, we obtained the following bound:

$$\mathbf{Adv}^{\mathrm{ePre}}_{\mathrm{Sponge}}(\mathcal{N}) \leq \frac{4\mathcal{N}}{2^\nu} + \min\left\{\frac{4\ell\mathcal{N}}{2^{\nu-r'}}, \frac{\mathcal{N}(\mathcal{N}+1)}{2^c}\right\}.$$

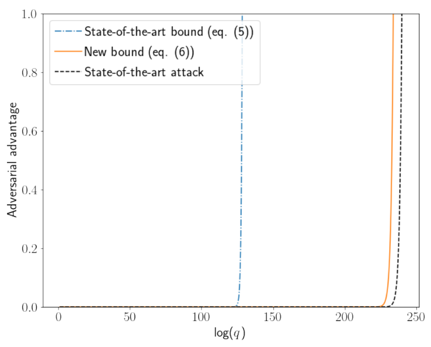
If $\ell = 1$, our security bound matches the state-of-the-art bound up to a factor of 4, while if $\ell > 1$, our bound improves the existing state of the art significantly. In both cases, the bound matches the best known attack outlined in Section 4.2.2 (up to constant). In the following, we show the improvement with the parameters used in the modes Ascon-Hash [DEMS21a] and Spongent [BKL+11].

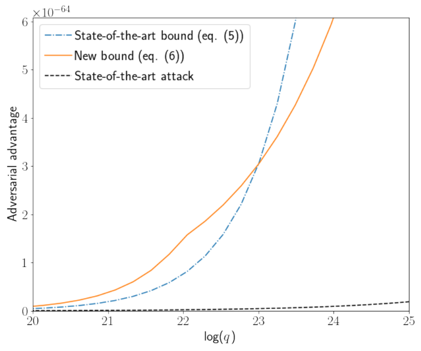




(a) Security bound (b) Close-up of intersection

Figure 4.3: Comparison of the state-of-the-art security bound, new security bound, and best known attack for the Ascon-Hash mode with parameters $(b, c, r, r', \nu) = (320, 256, 64, 64, 256)$.





(a) Security bound (b) Close-up of intersection

Figure 4.4: Comparison of the state-of-the-art security bound, new security bound, and best known attack for the Spongent mode with parameters $(b, c, r, r', \nu) = (272, 256, 16, 16, 256)$.

First consider the Ascon-Hash mode with parameters $(b, c, r, r', \nu) = (320, 256, 64, 64, 256)$. In this case, $\ell = \nu/r' = 4$. In Figure 4.3, we compare the state-of-the-art bound of Section 4.2.1, our new bound of (4.5), and the best known attack of Section 4.2.2. We observe that our new bound *significantly* improves the state-of-the-art bound starting at a very low value of $\mathcal{N}$. In detail, the adversarial advantage is approximately 1.5×10^{-36} for $\mathcal{N} \approx 2^{69}$ at the intersection point as shown in Figure 4.3b, i.e., at the point where the old bound starts to degenerate but our new bound stays low.

It is also interesting to consider the largest mode of Spongent, i.e., with parameters $(b, c, r, r', \nu) = (272, 256, 16, 16, 256)$. It has a small rate, and consequently a high value $\ell = \nu/r = 16$, but the same capacity and output size as the ones of Ascon-Hash. A comparison of the old bound, new bound, and best known attack is given in Figure 4.4. Here, the intersection point occurs at $\mathcal{N} \approx 2^{23}$, with an advantage of approximately 3×10^{-64}. Our bound thus improves even more the state-of-the-art bound to reach a preimage resistance close to 240 bits.

4.4.2 Applicability of the Result

Preimage resistance has historically always been considered to be a fundamental security notion (see, e.g., the security classification of Rogaway and Shrimpton [RS04], and the extensive literature on both proofs and attacks). While everywhere preimage resistance is important, the immediate practical applications of our result appear limited: for instance, everywhere preimage resistance suffices in proof-of-work schemes, but our result primarily benefits lightweight hash functions, which are unlikely to be deployed in such contexts. However, looking ahead, in Chapter 7, we will use the proof of Theorem 4.3.1 as a building block to evaluate the security of a system based on sponge-based hash chains. More precisely, the technical core of the proofs is the *domain-oriented* preimage resistance of the sponge, and we isolate the result below.

Domain-Oriented Preimage Resistance. Everywhere preimage resistance requires the adversary to invert *any given point* fixed at the start of the security game, even though a corresponding preimage may not necessarily exist. In contrast, many applications assume that a (secret) preimage *does* exist. This property is named domain-oriented preimage resistance (Definition 2.4.2) and changes the setting a bit. Similar to (everywhere) preimage resistance, the security bounds derived via indifferentiability (see Corollary 3.1.1.1) are lossy. As pointed out by Andreeva and Stam [AS11], in general, everywhere preimage resistance does not imply domain-oriented preimage resistance, and vice versa. Instead of looking at the specific cases where an implication holds, we can actually use a simple hybrid argument to establish domain-oriented preimage resistance of the sponge. Let $\mathcal{A}$ be an adversary with query complexity $\mathcal{N}$, let $\mathcal{P} \xleftarrow{\$} \texttt{Perm}(b)$, and h the sponge based on permutation $\mathcal{P}$, with output length fixed to ν. We have:

$$\begin{aligned}
&\mathbf{Adv}^{\mathrm{dPre}[\kappa,\nu]}_{\mathrm{Sponge}}(\mathcal{A}) = \mathbf{Pr}\left(\begin{array}{l} M \xleftarrow{\$} \{0,1\}^{\kappa}, \mathcal{A}^{\mathcal{P}}(h(M)) \to M' \\ \text{with } h(M) = h(M') \end{array}\right) \\
&\leq \left|\mathbf{Pr}\left(\begin{array}{l} M \xleftarrow{\$} \{0,1\}^{\kappa}, \mathcal{A}^{\mathcal{P}}(h(M)) \to M' \\ \text{with } h(M) = h(M') \end{array}\right) - \mathbf{Pr}\left(\begin{array}{l} Z \xleftarrow{\$} \{0,1\}^{\nu}, \mathcal{A}^{\mathcal{P}}(Z) \to M \\ \text{with } h(M) = Z \end{array}\right)\right| \\
&\quad + \mathbf{Pr}\left(\begin{array}{l} Z \xleftarrow{\$} \{0,1\}^{\nu}, \mathcal{A}^{\mathcal{P}}(Z) \to M \\ \text{with } h(M) = Z \end{array}\right) .
\end{aligned}$$

For the first term (the absolute difference), from $\mathcal{A}$ we can build a distinguisher $\mathcal{D}$ that returns 1 if $\mathcal{A}$ finds a preimage, else 0, and this transformation preserves the probabilities. Now, $\mathcal{D}$ is a distinguisher which is given access to $\mathcal{P}$, and has access either to random string or a sponge evaluation of a secret message in $\{0,1\}^{\kappa}$. In the latter case, the construction oracle that $\mathcal{D}$ interacts with is an outer-keyed sponge, described in Section 3.2.1, with capacity $\tilde{c} = \min\{c, c'\} = c'$. Here, the distinguisher has offline complexity of $\mathcal{N}$, and online complexity of $\mathcal{M} \leq \lceil \frac{\kappa}{r} \rceil + \lceil \frac{\nu}{r'} \rceil - 1$. Using (3.8) from Chapter 3 and assuming $\lceil \frac{\kappa}{r} \rceil$ is a small constant, the distinguishing advantage of $\mathcal{D}$ is of the form

$$\tilde{\mathcal{O}}\left(\frac{\mathcal{N}}{2^{\kappa}} + \frac{\mathcal{N}}{2^{c'}}\right) .$$

The second term, in turn, corresponds to finding a preimage for a random point, which can be upper bounded by the everywhere preimage resistance of the sponge. Therefore, we obtain

$$\mathbf{Adv}^{\mathrm{dPre}[\kappa,\nu]}_{\mathrm{Sponge}}(\mathcal{A}) = \tilde{\mathcal{O}}\left(\frac{\mathcal{N}}{2^{\kappa}} + \frac{\mathcal{N}}{2^{c'}} + \frac{\mathcal{N}}{2^{\nu}} + \min\left\{\frac{\mathcal{N}}{2^{\nu-r'}}, \frac{\mathcal{N}(\mathcal{N}+1)}{2^{c}}\right\}\right) . \quad (4.24)$$

In Remark 4.2.1, we discussed how the generic attack from Proposition 4.2.1 can be adapted for domain-oriented preimage resistance, and showed that an adversary can succeed with high probability once its complexity reaches

$$\mathcal{N} \approx \min\left\{2^{\kappa}, 2^{\nu}, \max\left\{\min\left\{2^{c'}, 2^{\nu-r'}\right\}, 2^{c/2}\right\}\right\} .$$

Therefore, (4.24) is tight when $c' \geq c/2$.

Sponge with Alternative Transformations. Our result can also be adapted when the underlying permutation is replaced by a different type of primitive. Foekens [Foe23] showed that if the permutation is replaced by a random transformation, preimage resistance holds up to approximately $\mathcal{N} \approx 2^{\nu}$ queries. Intuitively, this is because the adversary cannot efficiently invert a random function, effectively removing the bad event $\mathsf{BADINV}(\mathcal{Q})$ from the analysis. However, designing secure random functions is hard in practice. More recently, we showed [SLZ+25] that using a permutation combined with feed-forward during absorption similarly achieves preimage resistance up to about $\mathcal{N} \approx 2^{\nu}$ queries.

Chapter 5

Indifferentiability of the Sponge Construction with a Restricted Number of Message Blocks

5.1 Introduction

While hash functions are typically required to support inputs of arbitrary length, several applications only hash inputs of a fixed or bounded length. For example, the widely used Fiat-Shamir heuristic [FS86] allows to remove interactivity of a public-coin proof by using a hash function for which the input has a length fixed in advance. Similarly, applications such as password hashing, hash chains (see, e.g., Chapter 7), or the SAFE framework [AKMQ23] may only require hashing inputs of restricted size.

In this chapter, we explore how such an input-length restriction impacts the security of the sponge construction. In more detail, we consider the sponge construction of Algorithm 1 from Chapter 3 with $r'' = r$. Its indifferentiability bound (Theorem 3.1.1) is known to be tight in the general case, and the matching attack (see Section 5.2.3) works by finding inner collisions, and compensating for differences in the outer part using an additional absorption call. This strategy, however, requires absorbing more than $\lceil \frac{c}{2r} \rceil$ message blocks. Hence, when the number of absorbed message blocks is limited to $\ell \leq \lceil \frac{c}{2r} \rceil$, this attack no longer applies. This reveals a gap between the known upper and lower bound, which this chapter aims to address.

Contribution. In detail, we show that if the number of absorbed blocks is restricted to $\ell \leq \lceil \frac{c}{2r} \rceil$ blocks (but arbitrarily long digests can still be squeezed), then stronger security guarantees may be achieved. We prove in Theorem 5.3.1 that, when the size of the input is restricted to ℓ blocks, the sponge is indifferentiable from a random oracle up to

$$\approx \min\left\{2^{c'}, 2^{b/2}, \max\left\{2^{c/2}, 2^{b-\ell\times r}\right\}\right\}$$

queries (up to constant and logarithmic factors). We compare this result with existing bounds in Section 5.3. Moreover, as detailed in Section 5.5, our proof extends naturally to the public indifferentiability setting [YMO09, DRS09, MPS12], yielding security up to

$$\approx \min\left\{2^{b/2}, \max\left\{2^{c/2}, 2^{b-\ell\times r}\right\}\right\}$$

queries. We also show how this result closes a gap in the collision and second preimage resistance of the standard (unrestricted) sponge when $r' > r$.

Related Work. When one message block is absorbed and the digest is extracted in one squeeze call, the sponge boils down to a truncated permutation. Choi et al. [CLL19] proved that a truncated random permutation is indifferentiable from a $\mathcal{RO}$ up to

$$\approx \min\left\{2^{\frac{b+c'}{3}}, 2^{c}, 2^{c'}\right\}$$

queries. This result has been generalized to the case where the fixed prefix is randomized by Grassi and Mennink [GM22].

Outline. The rest of this chapter is organized as follows: Section 5.2 introduces the necessary preliminaries. Section 5.3 presents our indifferentiability result and its proof. Section 5.4 discusses the tightness of the bound. Section 5.5 extends the result to the public indifferentiability setting and explores implications for the standard (unrestricted) sponge construction. Finally, Section 5.6 concludes.

5.2 Preliminaries

5.2.1 Security Model

Construction Queries and Their Cost. In order to alleviate notation, we change slightly the input/output pattern of construction queries compared to Chapter 3. Rather than taking the requested output length $\nu \in \mathbb{N}^*$, we assume that the construction oracle takes as input the number of requested blocks $k \in \mathbb{N}^*$. Thus, for any $M \in \{0,1\}^*$, $\mathrm{Sponge}^{\mathcal{P}}(M,k)$ returns the first k squeezed blocks after absorbing M. Thus, in order to match the real world, we consider that the random oracle queried with inputs $M \in \{0,1\}^*$ and $k \in \mathbb{N}^*$ returns $\mathcal{RO}^{\infty}(M)[1 : r' \times k]$. As discussed in Section 3.1.2.1, we measure construction queries in terms of the number of permutation evaluations required in the real world to produce the output. More precisely, if $|pad(M)| = l \times r$, then a query with input M and k has a cost of $l + k - 1$. However, unlike the setting in Section 3.1.2.1, and as explained in Remark 3.1.1, we distinguish between construction-originated and primitive-originated queries when counting queries.

Query History. The primitive query history of the distinguisher is an ordered list denoted by $\mathcal{Q}_P$, where each element is a tuple $(X, Y, d) \in \{0,1\}^b \times \{0,1\}^b \times \{fwd, inv\}$. If $\mathcal{Q}_P[i] = (X, Y, d)$, it means the i-th primitive query was in direction d, and the distinguisher received X or Y in response, depending on the direction. The construction query history is denoted $\mathcal{Q}_C$ and consists of tuples (M, k, Z). In the real world, Z is the k^{th} squeezed block after absorbing M. In the ideal world, $Z = \mathcal{RO}^{\infty}(M)[r' \times (k-1) + 1 : r' \times k]$.

Public Indifferentiability. Plain indifferentiability is defined in Definition 2.4.3. In the indifferentiability setting, the simulator is *not* allowed to access the construction queries made by the adversary. However, in many applications, such as signature schemes or proof systems, the hash function input is public, making this model overly restrictive [BR96, BR93, FS86]. This motivated the introduction of the *public indifferentiability* framework [YMO09, DRS09, MPS12]. Public differentiability differs from indifferentiability precisely in the fact that, whenever the adversary makes a construction query, the simulator is immediately informed of the queried input. Given a simulator $\mathbf{S}$, $\mathbf{Adv}^{\mathrm{pubiff}}_{\mathrm{Sponge},\mathbf{S}}(\mathcal{D})$ is defined similarly to the indifferentiability setting. Note that public indifferentiability is a strictly weaker notion. For instance, the (plain) Merkle-Damgård construction [Mer89, Dam89] is not indifferentiable, but it is publicly indifferentiable [DRS09].

Indifferentiability of the Sponge when Restricting the Number of Message Blocks. When the number of message blocks is restricted to $\ell \in \mathbb{N}^*$ blocks, we require that construction queries with input message M satisfy $|pad(M)| \leq \ell \times r$. More formally, denote by $\texttt{RDist}[q_P, q_C, \ell]$ the set of distinguishers satisfying this restriction, making at most q_P primitive queries and construction queries with a total cost of at most q_C. The restricted indifferentiability (resp., public indifferentiability) advantage of the sponge from a random oracle with simulator $\mathbf{S}$ is defined as follows:

$$\mathbf{Adv}^{\text{R-iff}}_{\text{Sponge},\mathbf{S}}(q_P, q_C, \ell) = \sup_{\mathcal{D}\in\mathtt{RDist}[q_P,q_C,\ell]} \mathbf{Adv}^{\text{indif}}_{\text{Sponge},\mathbf{S}}(\mathcal{D}) \ ,$$
$$\mathbf{Adv}^{\text{R-pubiff}}_{\text{Sponge},\mathbf{S}}(q_P, q_C, \ell) = \sup_{\mathcal{D}\in\mathtt{RDist}[q_P,q_C,\ell]} \mathbf{Adv}^{\text{pubiff}}_{\text{Sponge},\mathbf{S}}(\mathcal{D}) \ .$$

5.2.2 Graph Construction

We adapt the graph representation from [BDPV08] to our setting. This graph is derivable from a primitive query history $\mathcal{Q}$. The nodes are all elements in $\{0,1\}^b$. For $X, Y \in \{0,1\}^b, m \in \{0,1\}^r$, we write $X \xrightarrow{m} Y$ whenever there exists $d \in \{\mathit{fwd}, \mathit{inv}\}$ such that $(X \oplus (m\|0^c), Y, d) \in \mathcal{Q}$; and $X \rightarrow Y$ whenever $X \xrightarrow{0^r} Y$. The set $\mathtt{ValidP}$ gathers the nodes involved in paths, i.e.,

$$\mathtt{ValidP} := \Big\{ IV \xrightarrow{m_1} N_{A,1} \xrightarrow{m_2} \cdots \xrightarrow{m_l} N_{S,1} \rightarrow N_{S,2} \rightarrow \cdots \rightarrow N_{S,k} \mid m_l \neq 0^r \wedge k \in \mathbb{N}^* \wedge l \leq \ell \Big\} .$$

The set $\mathtt{Rooted}$ includes all the nodes which appear in valid paths. The set $\mathtt{rRooted}$ is a compact way of characterizing the paths. Informally, if the simulator receives a forward query with X, and $(X, M, k) \in \mathtt{rRooted}$, and $\mathit{unpad}(M) \neq \perp$, then in order to produce a consistent answer with respect to the random oracle, the answer Y must satisfy

$$\text{outer}_{r'}(Y) = \mathcal{RO}^{\infty}(\mathit{unpad}(M))\,[r' \times k + 1 : r' \times (k+1)] \, .$$

Finally, we define the set $\mathtt{AbsP}$ as follows:

$$\mathtt{AbsP} = \{IV\} \cup \Big\{ Y \mid \exists l < \ell, m_1, \ldots, m_l \in \{0,1\}^r, N_{A,1}, \ldots, N_{A,l-1} \in \{0,1\}^b \text{ such that } IV \xrightarrow{m_1} N_{A,1} \xrightarrow{m_2} \cdots \xrightarrow{m_l} Y \Big\} .$$

In other words, this set gathers the nodes where absorption of a message block is still possible. Note that the obtained path is not necessarily a valid path, since the message blocks $m_1, \ldots, m_l$ can all be equal to 0^r. One important remark is that, as long as no inner collisions occur between nodes in $\mathtt{AbsP}$,

$$|\mathtt{AbsP}| \leq \min\Big\{ q+1, 2 \times 2^{(\ell-1)\times r} \Big\} \ , \tag{5.1}$$

where $q := |\mathcal{Q}|$. Looking ahead, inner collisions that may allow the adversary to distinguish can only occur at nodes within `AbsP`.

We will see later that in some situations, the simulator has a query history different from the primitive query history of the distinguisher. Therefore, when this is not clear from the context, for

$$\texttt{Set} \in \{\texttt{ValidP}, \texttt{Rooted}, \texttt{rRooted}, \texttt{AbsP}\} ,$$

we use the notation $\texttt{Set}(\mathcal{Q})$ to clarify which query history is used to build `Set`.

5.2.3 Differentiability Attack of the Sponge

In this section we describe a differentiability attack on the (plain) sponge [BDPV07]. Let $k = \frac{c}{2r}$, and for simplicity, assume that k is an integer, and that the padding function is the 10^*-padding. Consider the following distinguisher $\mathcal{D}$:

1. $\mathcal{D}$ first makes all possible k first absorb calls with primitive queries. This results in $2^{c/2}$ distinct states $(Y_i)_{i\in[\![1,2^{c/2}]\!]}$ such that
$$IV \xrightarrow{m_1} N_1 \xrightarrow{m_2} N_2 \rightarrow \cdots \xrightarrow{m_k} Y_i \, ;$$

2. If the simulator emulates well a permutation,[1] then with high probability there exist $i \neq j$ such that $Y_i \overset{c}{=} Y_j$ and neither $\text{outer}_r(Y_i)$ nor $\text{outer}_r(Y_j)$ is equal to 0^r. Let M_i (resp., M_j) be the concatenation of all message blocks used to reach Y_i (resp., Y_j);

3. The distinguisher makes the construction queries for
$$\mathit{unpad}_r^{10^*}(M_i \| \text{outer}_r(Y_i)) \quad \text{and} \quad \mathit{unpad}_r^{10^*}(M_j \| \text{outer}_r(Y_j)) \, ,$$
with sufficiently long outputs. If both outputs coincide, then the distinguisher returns "real", otherwise "ideal".

In the real world, two sequences of message blocks that lead to the same state have the same digest, while in the ideal world, the outputs are independent. $\mathcal{D}$ succeeds with high probability, and makes $2^{k\times r} = 2^{c/2}$ queries.

This attack requires at least $k+1$ absorb calls. Thus, whenever $k+1 > \ell$, this attack cannot be applied to the sponge when restricting the number of message blocks to ℓ.

[1]If this is not the case, then we can use another distinguisher which exploits this difference of behavior.

Differentiability Attack When $c' \leq c/2$. When $r' > r$, the indifferentiability bound has an additional term of the form $\frac{q \cdot \mathrm{mucol}(q, 2^{r'})}{2^{c'}}$. We can construct a distinguisher $\mathcal{D}$ with query complexity q such that $q \cdot \mathrm{mucol}(q, 2^{r'}) \approx 2^{c'}$. The distinguisher proceeds as follows:

1. $\mathcal{D}$ performs q construction queries, each on a randomly chosen message long enough to prevent the simulator from guessing the messages with exhaustive search. For each query, $\mathcal{D}$ requests a $\gamma + 1$ squeezed blocks, where γ is sufficiently large (e.g., $\gamma = b$). In the real world, this results in q states Y_i with
$$IV \xrightarrow{M} Y_i \rightarrow Y_i^{(1)} \rightarrow \cdots \rightarrow Y_i^{(\gamma)} .$$
In the ideal world, each query returns γ random blocks of size r';

2. Among the first r' bits outputted by each construction query, $\mathcal{D}$ identifies $u \in \{0,1\}^{r'}$, the outer part that occurs most frequently. This isolates $\approx \mathrm{mucol}(q, 2^{r'})$ of the queries that all share this common outer part;

3. $\mathcal{D}$ then makes q primitive queries, each with outer part fixed to u, and with random inner part. Each query is followed by a chain of γ permutation queries. This results in q values $Z_i \in \{0,1\}^b$ with $\mathrm{outer}_{r'}(Z_i) = u$ and the chains
$$Z_i \rightarrow Z_i^{(1)} \rightarrow \cdots \rightarrow Z_i^{(\gamma)} ;$$

4. In the real world, with high probability there exists $i, j \in [\![1, q]\!]$ with $Y_i = Z_i$. The adversary can detect this equality by comparing the outer parts of the γ-long chains starting from Y_i and Z_i. However, in the ideal world, the simulator has no visibility into the construction queries made in the first step. Thus, with high probability, the outputs of the construction queries will not match any chain of values computed in the previous phase.

Informally, we can see that this attack relies on the fact that the simulator is not aware of the construction queries.

5.3 Indifferentiability Result

In this section, we prove indifferentiability of the sponge when restricting the number of message blocks, as stated in Theorem 5.3.1.

Theorem 5.3.1. *Let b, c, r, c', r' with $b = r + c = r' + c'$. Consider the sponge construction of Algorithm 1 with parameters b, c, r, c', r' and $r'' = r$. There exists a simulator $\mathbf{S}$ with complexity $\tilde{\mathcal{O}}(q)$ queries such that*

$$\mathbf{Adv}^{\text{R-iff}}_{\text{Sponge},\mathbf{S}}(q_C, q_P, \ell) \leq \frac{q_P(12r' + 16)}{2^{c'}} + \frac{5q^2}{2^b} + \min\left\{\frac{10q(q+1)}{2^c}, \frac{20q}{2^{b-\ell\times r}}\right\},$$

where $q := q_P + q_C$.

The proof relies on multicollision analysis, as discussed in Section 2.5.2. For ease of expression of the bounds, we used an explicit upper bound on the expected maximal multicollision size (i.e., the second bound in Lemma 2.5.2).

Interpretation of the Bound. In the following, we ignore logarithmic factors. When the maximum number of message blocks absorbed is not a limiting factor (i.e., when $c/2 \leq r(\ell - 1)$), we obtain an upper bound of the form

$$\tilde{\mathcal{O}}\left(\frac{q}{2^{c'}} + \frac{q^2}{2^c}\right),$$

which matches the bound of Naito and Ohta [NO14]. On the other hand, when $c/2 > r(\ell - 1)$, we obtain an upper bound of the form

$$\tilde{\mathcal{O}}\left(\frac{q}{2^{c'}} + \frac{q^2}{2^b} + \frac{q}{2^{b-\ell\times r}}\right).$$

Moreover, when only one message block is absorbed (i.e., $\ell = 1$), our bound matches again the one from Naito and Ohta [NO14] by setting $r'' = r$ and $r = 0$ (recall that r'' is the rate of the first message block, and r is the rate when absorbing subsequent message blocks). To our knowledge, the bound provides new results whenever $c/2 > r \times (\ell - 1)$ and when $\ell \neq 1$.

For example, the member of the PHOTON [GPP11] sponge-based hash function family with parameters $(b, c, r, c', r') = (256, 224, 32, 224, 32)$. The standard indifferentiability bound for the sponge construction guarantees generic security up to approximately $\approx 2^{112}$ queries. However, assuming the 10*-padding, and restricting inputs to be of size at most 127 bits, our result guarantees indifferentiability from a random oracle up to 2^{128} queries.

Outline. The remainder of this section is organized as follows. Section 5.3.1 introduces the simulator used for the proof, Section 5.3.2 introduces an intermediate world, Section 5.3.3 lists the bad events, and Section 5.3.4 splits and computes the probabilities.

Algorithm 3 Simulator definition (1/2). $\mathbf{S}^{\text{init}}()$ is run once at the beginning of the game, $\mathbf{S}^{\text{fwd}}$ (resp., $\mathbf{S}^{\text{inv}}$) is the algorithm run by $\mathbf{S}$ on forward (resp., inverse) queries. The inputs X and Y are b-bit strings.

```
1: function S^init()
2:     $\mathcal{Q}_{\text{Sim}} \leftarrow \emptyset$
3:     rRooted $\leftarrow \{(IV, \epsilon, 1)\}$

4: function S^inv($Y$)
5:     if $\exists X \in \{0,1\}^b$, $d \in \{fwd, inv\}$ such that $(X, Y, d) \in \mathcal{Q}_{\text{Sim}}$
6:         return $X$
7:     $X \xleftarrow{\$} \{0,1\}^b$
8:     $\mathcal{Q}_{\text{Sim}} \leftarrow \mathcal{Q}_{\text{Sim}} \cup \{(X, Y, inv)\}$
9:     return $X$
```

5.3.1 Simulator Definition

The simulator $\mathbf{S}$ is described in Algorithms 3 and 4. On a high level view, $\mathbf{S}$ keeps track of the graph construction from Section 5.2.2 to ensure that its responses are consistent with the random oracle. To do that, it logs the query history in an ordered list called $\mathcal{Q}_{\text{Sim}}$. For $i \in \mathbb{N}^*$, $\mathcal{Q}_{\text{Sim}}[i]$ contains a tuple (X, Y, d) which corresponds to the i^{th} query. For any fresh query, $\mathbf{S}$ behaves as a random function. We could have a simulator which outputs permutation-consistent responses, but this would not improve significantly the bound. More precisely, the term $\frac{q^2}{2^b}$ in the bound of Theorem 5.3.1 does not only come from the PRP/PRF switching lemma, but appears at other places. For example, at the end of Section 5.4 we present an attack with a cost of $\approx 2^{b/2}$ queries which would succeed even when using a permutation-consistent variant of $\mathbf{S}$.

5.3.2 World Decomposition

Denote by W_R the real world, i.e., implemented by $(\text{Sponge}^{\mathcal{P}}, \mathcal{P})$ as shown in Figure 5.1a, and let W_I be the ideal world giving access to $(\mathcal{RO}, \mathbf{S})$ (see Figure 5.1c). We introduce an intermediate world W_{IM}, depicted in Figure 5.1b. This world gives access to $(\text{Sponge}^{\mathbf{S}}, \mathbf{S})$, where the simulator is based on a random oracle hidden from the adversary. This decomposition, done among others in the proof of PHOTON indifferentiability [NO14], allows to separate the quality of randomness of the simulator from the consistency of the answers.

Algorithm 4 Simulator definition (2/2).

```
10: function $\mathbf{S}^{\mathrm{fwd}}(X)$
11:     if $\exists Y \in \{0,1\}^b, d \in \{\mathit{fwd}, \mathit{inv}\}$ such that $(X, Y, d) \in \mathcal{Q}_{\mathrm{Sim}}$
12:         return $Y$
13:     // If $\exists\ IV \xrightarrow{M\|0^{(k-1)r}} X \oplus (m\|0^c)$ where $X \oplus (m\|0^c) \in \mathtt{AbsP}$
14:     if $\exists M \in \{0,1\}^*$, $m \in \{0,1\}^r \setminus \{0^r\}$, $N \in \{0,1\}^b, k \in \mathbb{N}$ such that
        $(N, M, k) \in \mathtt{rRooted}$ and $X = N \oplus (m\|0^c)$ and $k + |M|/r \leq \ell$
15:         $M' \leftarrow M\|0^{(k-1)r}\|m$
16:         $y_r \xleftarrow{\$} \{0,1\}^{r'}$
17:         if $\mathit{unpad}(M') \neq \bot$
18:             $y_r \leftarrow \mathcal{RO}^\infty(\mathit{unpad}(M'))\,[1 : r']$
19:         $y_c \xleftarrow{\$} \{0,1\}^{c'}$
20:         $Y \leftarrow y_r\|y_c$
21:         $\mathcal{Q}_{\mathrm{Sim}} \leftarrow \mathcal{Q}_{\mathrm{Sim}} \cup \{(X, Y, \mathit{fwd})\}$
22:         $\mathtt{rRooted} \leftarrow \mathtt{rRooted} \cup \{(Y, M', 1)\}$
23:         return $Y$
24:     else if $\exists M \in \{0,1\}^*, k \in \mathbb{N}$ such that $(X, M, k) \in \mathtt{rRooted}$
25:         $y_r \xleftarrow{\$} \{0,1\}^{r'}$
26:         if $\mathit{unpad}(M) \neq \bot$
27:             $y_r \leftarrow \mathcal{RO}^\infty(\mathit{unpad}(M))\,[r' \times k + 1 : r' \times (k+1)]$
28:         $y_c \xleftarrow{\$} \{0,1\}^{c'}$
29:         $Y \leftarrow y_r\|y_c$
30:         $\mathcal{Q}_{\mathrm{Sim}} \leftarrow \mathcal{Q}_{\mathrm{Sim}} \cup \{(X, Y, \mathit{fwd})\}$
31:         $\mathtt{rRooted} \leftarrow \mathtt{rRooted} \cup \{(Y, M, k+1)\}$
32:         return $Y$
33:     else
34:         $Y \xleftarrow{\$} \{0,1\}^b$
35:         $\mathcal{Q}_{\mathrm{Sim}} \leftarrow \mathcal{Q}_{\mathrm{Sim}} \cup \{(X, Y, \mathit{fwd})\}$
36:         return $Y$
```

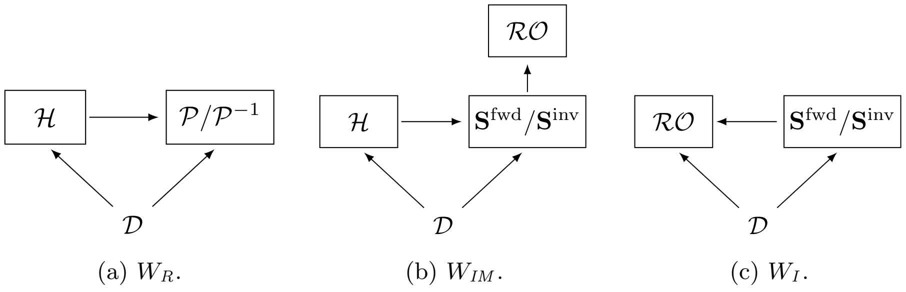

$\mathcal{H}$
$\mathcal{P}/\mathcal{P}^{-1}$
$\mathcal{D}$
(a) W_R.
$\mathcal{RO}$
$\mathcal{H}$
$\mathbf{S}^{\mathrm{fwd}}/\mathbf{S}^{\mathrm{inv}}$
$\mathcal{D}$
(b) W_{IM}.
$\mathcal{RO}$
$\mathbf{S}^{\mathrm{fwd}}/\mathbf{S}^{\mathrm{inv}}$
$\mathcal{D}$
(c) W_I.

Figure 5.1: Definition of the different worlds. $\mathcal{P}$ returns responses with lazy sampling.

W_R Versus W_{IM}. As detailed more formally in Section 5.3.4.2, the difference between W_R and W_{IM} lies in the difference of randomness. More precisely, the ideal world implements a perfectly random permutation, while the simulator behaves like a two-sided random function. This is a well known result, and is upper bounded by the PRP/PRF switching lemma.

W_{IM} Versus W_I. The simulator from world W_{IM} has a priori more knowledge than the one from W_I, since it indirectly knows the construction queries made by the adversary. In Section 5.3.3, we define a bad event GUESS which prevents the adversary from exploiting this difference of behavior. Moreover, we have to guarantee that the simulator behaves consistently according to the random oracle, i.e., whenever one can compute $\mathrm{Sponge}^{\mathcal{P}}(M, k)$ from the query history $\mathcal{Q}_P$, then it coincides with $\mathcal{RO}^{\infty}(M)[1 : r' \times k]$. We will define additional bad events to capture inconsistent answers. These bad events, including GUESS, are given in Section 5.3.3.

5.3.3 Bad Events

In this section, we formally define bad events over the query history of the simulator. For the sake of the proof, we define an enhanced simulator query history $\mathrm{Ext}(\mathcal{Q}_{\mathrm{Sim}})$ that contains additionally the origin $\mathrm{O} \in \{\mathrm{C}, \mathrm{D}\}$ of the query, i.e., elements in $\mathrm{Ext}(\mathcal{Q}_{\mathrm{Sim}})$ are of the form (X, Y, d, O), where “C” indicates that the query was made by the construction and “D” by the adversary. Note that the simulator has no access to this list, it is introduced only to rigorously analyze the security. This enhanced query history is only useful for W_{IM}. We also define $\mathrm{Fresh}(\mathcal{Q}_{\mathrm{Sim}})$, which is the query history without duplicated queries. More precisely, the i^{th} element of $\text{Fresh}(\mathcal{Q}_{\text{Sim}})$ is a tuple $(X_i, Y_i, d_i) \in \mathcal{Q}_{\text{Sim}}$ such that there is no $j < i$ such that $\text{Fresh}(\mathcal{Q}_{\text{Sim}})[j] = (X_i, Y_i, d)$ for $d \in \{\mathit{fwd}, \mathit{inv}\}$. Let σ be the total number of queries made to the simulator, so that $\sigma = q_C + q_P$ in W_{IM}, $\sigma = q_P$ in W_I. Moreover, let $\bar{\sigma}$ be the total number of *fresh* queries to the simulator, so that $\bar{\sigma} \leq \sigma$. For simplicity of notation, for $i \in [\![1, \bar{\sigma}]\!]$, (X_i, Y_i) denotes the first two elements in $\text{Fresh}(\mathcal{Q}_{\text{Sim}})[i]$. We define the following bad events:

$$\begin{aligned}
\mathsf{GUESS\_SQU} :\ & \exists i \in [\![1, \sigma]\!] \text{ such that } \text{Ext}(\mathcal{Q}_{\text{Sim}})[i] = (X, Y, \mathit{fwd}, \text{D}) \text{ and} \\
& X \notin \texttt{Rooted}(\mathcal{Q}_P[1 : i-1]) \text{ and } \exists j < i \text{ such that} \\
& \text{Ext}(\mathcal{Q}_{\text{Sim}})[j] = (X', Y', \mathit{fwd}, \text{C}) \text{ and } X = X' \\
\text{or } & \exists i \in [\![1, \sigma]\!] \text{ such that } \text{Ext}(\mathcal{Q}_{\text{Sim}})[i] = (X, Y, \mathit{inv}, \text{D}) \text{ and} \\
& \exists j < i \text{ such that } \text{Ext}(\mathcal{Q}_{\text{Sim}})[j] = (X', Y', \mathit{fwd}, \text{C}) \text{ and } Y = Y' \,, \\
\mathsf{GUESS\_ABS} :\ & \exists i \in [\![1, \sigma]\!] \text{ such that } \text{Ext}(\mathcal{Q}_{\text{Sim}})[i] = (X, Y, \mathit{fwd}, \text{D}) \text{ and} \\
& X \notin \texttt{Rooted}(\mathcal{Q}_P[1 : i-1]) \text{ and } \exists j < i \text{ such that} \\
& \text{Ext}(\mathcal{Q}_{\text{Sim}})[j] = (X', Y', \mathit{fwd}, \text{C}), Y' \in \texttt{AbsP}(\mathcal{Q}_{\text{Sim}}), \text{ and } X \overset{c}{=} Y' \,, \\
\mathsf{GUESS} :\ & \mathsf{GUESS\_SQU} \vee \mathsf{GUESS\_ABS} \,,
\end{aligned}$$

$$\begin{aligned}
\mathsf{COL} :\ & \exists i \in [\![1, \bar{\sigma}]\!] \text{ such that } \text{Fresh}(\mathcal{Q}_{\text{Sim}})[i] = (X_i, Y_i, \mathit{fwd}) \text{ and} \\
& \exists j < i \text{ such that } Y_i = Y_j \\
\text{or } & \exists i \in [\![1, \bar{\sigma}]\!] \text{ such that } \text{Fresh}(\mathcal{Q}_{\text{Sim}})[i] = (X_i, Y_i, \mathit{inv}) \text{ and} \\
& \exists j < i \text{ such that } X_i = X_j \,, \\
\mathsf{CONNECT} :\ & \exists i \in [\![1, \bar{\sigma}]\!] \text{ such that } \text{Fresh}(\mathcal{Q}_{\text{Sim}})[i] = (X_i, Y_i, \mathit{fwd}) \text{ and} \\
& \exists j < i \text{ such that } Y_i = X_j \\
\text{or } & \exists i \in [\![1, \bar{\sigma}]\!] \text{ such that } \text{Fresh}(\mathcal{Q}_{\text{Sim}})[i] = (X_i, Y_i, \mathit{inv}) \text{ and} \\
& \exists j < i \text{ such that } X_i = Y_j \,,
\end{aligned}$$

$$\begin{aligned}
\mathsf{INNER}^1 :\ & \exists i \in [\![1, \bar{\sigma}]\!], m \in \{0,1\}^r \text{ such that } \text{Fresh}(\mathcal{Q}_{\text{Sim}})[i] = (X_i, Y_i, \mathit{fwd}) \text{ and} \\
& X_i \oplus (m \| 0^c), Y_i \in \texttt{AbsP}(\text{Fresh}(\mathcal{Q}_{\text{Sim}})) \text{ and} \\
& \exists j < i \text{ such that either } Y_i \overset{c}{=} X_j \text{ or } Y_i \overset{c}{=} Y_j \,, \\
\mathsf{INNER}^2 :\ & \exists i \in [\![1, \bar{\sigma}]\!] \text{ such that } \text{Fresh}(\mathcal{Q}_{\text{Sim}})[i] = (X_i, Y_i, \mathit{fwd}) \text{ and } \exists Y \in \{0,1\}^b \\
& \text{such that } Y \in \texttt{AbsP}(\text{Fresh}(\mathcal{Q}_{\text{Sim}})[1 : i-1]) \text{ and } Y_i \overset{c}{=} Y \\
\text{or } & \exists i \in [\![1, \bar{\sigma}]\!] \text{ such that } \text{Fresh}(\mathcal{Q}_{\text{Sim}})[i] = (X_i, Y_i, \mathit{inv}) \text{ and } \exists Y \in \{0,1\}^b \\
& \text{such that } Y \in \texttt{AbsP}(\text{Fresh}(\mathcal{Q}_{\text{Sim}})[1 : i-1]) \text{ and } X_i \overset{c}{=} Y \,, \\
\mathsf{INNER} :\ & \mathsf{INNER}^1 \vee \mathsf{INNER}^2 \,.
\end{aligned}$$

Moreover, let $\mathsf{BAD} = \mathsf{GUESS} \vee \mathsf{COL} \vee \mathsf{CONNECT} \vee \mathsf{INNER}$.

Interpretation. The bad events COL, $\mathsf{CONNECT}$, and INNER concern the consistency of the answers with respect to the random oracle, i.e., when a path connects to a node or two paths collide without the simulator being able to ensure consistency. COL flags full-state collisions, and $\mathsf{CONNECT}$ is set when the order of the queries is not respected. The event INNER flags both potential full-state collisions and inconsistent queries that involve nodes in $\mathtt{AbsP}$. Note that the event of a query hitting the IV is captured by the event INNER^2. Finally, COL is also useful to bound the distance between W_R and W_{IM}.

GUESS is an event that can be set only in W_{IM}. It reflects the fact that the distinguisher is able to enter in the middle of a path without having made all the queries before that path. To do that, the adversary has two different possibilities. The first one, corresponding to $\mathsf{GUESS\_SQU}$, consists of making use of the outer parts disclosed by the construction queries. In other words, for a fixed $u \in \{0,1\}^{r'}$, if there exist x construction queries which output is u, then a primitive query with input $u\|w$ has a probability of $\approx \frac{x}{2^{c'}}$ to set $\mathsf{GUESS\_SQU}$. The second strategy corresponds to $\mathsf{GUESS\_ABS}$. It targets the nodes in $\mathtt{AbsP}$, and only concerns forward queries. In this setting, besides having access to r' bits, the adversary is allowed to overwrite up to r bits, as illustrated in Figure 5.2. As long as GUESS is not set, the adversary is not aware of paths where the simulator in W_{IM} has more knowledge than the one from W_I. This subtlety, which comes from the simulator's lack of access to construction queries, was pointed out, e.g., in [DFL12, CFN$^+$12].

5.3.4 Probability Splitting and Computation

Remember that $q := q_P + q_C$. For $i \in [\![1, \sigma]\!]$ and any event $\mathbf{evt}$, $\mathbf{evt}_i$ denotes that $\mathbf{evt}$ is triggered after the first i queries. Moreover, for $W_X \in \{W_R, W_{IM}, W_I\}$, denote by $\mathbf{Pr}_{W_X}(\mathbf{evt})$ the probability that $\mathbf{evt}$ is set in world W_X.

5.3.4.1 Outline

Let $\mathcal{D}$ be any distinguisher, we have

$$\begin{aligned}\mathbf{Adv}^{\text{indif}}_{\text{Sponge}}(\mathcal{D}) &= \left|\mathbf{Pr}\left(\mathcal{D}^{W_R} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{W_I} = 1\right)\right| \\ &\leq \left|\mathbf{Pr}\left(\mathcal{D}^{W_R} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{W_{IM}} = 1\right)\right| + \qquad (5.2)\\ &\quad \left|\mathbf{Pr}\left(\mathcal{D}^{W_{IM}} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{W_I} = 1\right)\right| . \qquad (5.3)\end{aligned}$$

Looking ahead, we will show in Section 5.3.4.3 that

$$\left|\mathbf{Pr}\left(\mathcal{D}^{W_{IM}} = 1 \mid \neg\mathsf{BAD}\right) - \mathbf{Pr}\left(\mathcal{D}^{W_I} = 1 \mid \neg\mathsf{BAD}\right)\right| = 0\,. \tag{5.4}$$

Using this fact, denoting $P := \mathbf{Pr}\left(\mathcal{D}^{W_{IM}} = 1 \mid \neg\mathsf{BAD}\right) = \mathbf{Pr}\left(\mathcal{D}^{W_I} = 1 \mid \neg\mathsf{BAD}\right)$, the distance (5.3) can be decomposed as follows:

$$\begin{aligned}
(5.3) = {} & \big|\mathbf{Pr}\left(\mathcal{D}^{W_{IM}} = 1 \wedge \mathsf{BAD}\right) - \mathbf{Pr}\left(\mathcal{D}^{W_I} = 1 \wedge \mathsf{BAD}\right) + \\
& \mathbf{Pr}\left(\mathcal{D}^{W_{IM}} = 1 \wedge \neg\mathsf{BAD}\right) - \mathbf{Pr}\left(\mathcal{D}^{W_I} = 1 \wedge \neg\mathsf{BAD}\right)\big| \\
= {} & \big|\mathbf{Pr}\left(\mathcal{D}^{W_{IM}} = 1 \wedge \mathsf{BAD}\right) - \mathbf{Pr}\left(\mathcal{D}^{W_I} = 1 \wedge \mathsf{BAD}\right) + \\
& P\left(1 - \mathbf{Pr}\left(\mathcal{D}^{W_{IM}} \text{ sets } \mathsf{BAD}\right)\right) - P\left(1 - \mathbf{Pr}\left(\mathcal{D}^{W_I} \text{ sets } \mathsf{BAD}\right)\right)\big| \\
= {} & \big|\mathbf{Pr}\left(\mathcal{D}^{W_{IM}} \text{ sets } \mathsf{BAD}\right)\left(\mathbf{Pr}\left(\mathcal{D}^{W_{IM}} = 1 \mid \mathsf{BAD}\right) - P\right) - \\
& \mathbf{Pr}\left(\mathcal{D}^{W_I} \text{ sets } \mathsf{BAD}\right)\left(\mathbf{Pr}\left(\mathcal{D}^{W_I} = 1 \mid \mathsf{BAD}\right) - P\right)\big| \\
\le {} & 2 \times \max\left\{\mathbf{Pr}\left(\mathcal{D}^{W_{IM}} \text{ sets } \mathsf{BAD}\right), \mathbf{Pr}\left(\mathcal{D}^{W_I} \text{ sets } \mathsf{BAD}\right)\right\}\,.
\end{aligned} \tag{5.5}$$

Therefore, the advantage of the adversary can be upper bounded as the sum of (5.2) and (5.5). The remainder of the proof is organized as follows. Section 5.3.4.2 is dedicated to upper bounding the distance in (5.2), then Section 5.3.4.3 shows the claim made in (5.4), i.e., the distance between W_{IM} and W_I conditioned on $\neg\mathsf{BAD}$ is zero, Section 5.3.4.4 upper bounds (5.5), and we conclude in Section 5.3.4.5.

5.3.4.2 W_R Versus W_{IM}

Because the padding is injective, the $\mathcal{RO}$ calls in lines 18 and 27 of Algorithm 4 are never accessed twice for the same entry. Therefore, the simulator behaves exactly as a random permutation as long as COL is not set in W_{IM}. Therefore, by the fundamental lemma of game playing [BR06],

$$\left|\mathbf{Pr}\left(\mathcal{D}^{W_R} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{W_{IM}} = 1\right)\right| \le \mathbf{Pr}\left(\mathcal{D}^{W_{IM}} \text{ sets } \mathsf{COL}\right)\,.$$

Moreover, for any $i \in [\![1, q]\!]$, one has

$$\mathbf{Pr}_{W_{IM}}\left(\mathsf{COL}_i \mid \neg\mathsf{COL}_{i-1}\right) \le \frac{i-1}{2^b}\,.$$

Therefore,

$$(5.2) \le \mathbf{Pr}\left(\mathcal{D}^{W_{IM}} \text{ sets } \mathsf{COL}\right) \le \frac{q(q-1)}{2^{b+1}}\,. \tag{5.6}$$

5.3.4.3 W_{IM} Versus W_I as Long as no Bad

The only difference between W_{IM} and W_I lies in the consistency of the responses: the construction oracle in W_{IM} calls the simulator, while in W_I, the answers come from a random oracle. As explained in Section 5.3.3, $\neg\mathsf{GUESS}$ prevents the distinguisher from being aware of paths where the simulator from W_{IM} has more knowledge than the one in W_I. Therefore, with $\neg\mathsf{GUESS}$, both simulators provide the same level of consistency, and w.l.o.g., we focus on the simulator from W_I. In this world, the construction queries come from a random oracle and $\mathcal{Q}_{\text{Sim}}$ coincides with the primitive query history of the distinguisher $\mathcal{Q}_P$. Let $i \in [\![1, q_P]\!]$, assume that the query number i gives $\mathcal{Q}_P[i] := (X_i, Y_i, d_i)$. Assume by contradiction that this query breaks the consistency, while BAD is not set. We split the cases depending on the direction of the query.

Forward Query. In order to break the consistency, the query must be involved in a valid path. There are two possibilities, depending on whether $X_i \in \mathtt{AbsP}$ or not:

(i) There exists a valid path

$$IV \xrightarrow{m_1} N_{A,1} \xrightarrow{m_2} \cdots \xrightarrow{m_l} N_{S,1} \to N_{S,2} \to \cdots \to N_{S,k} \to X_i \to Y_i\,,$$

with $m_l \neq 0^r$. In other words, X_i is involved in a path during the squeezing phase. Note that we can have $k = 0$. Let $k' := k + 2$.

(ii) There exists a valid path

$$IV \xrightarrow{m_1} N_{A,1} \xrightarrow{m_2} \cdots \xrightarrow{m_{l-2}} N_{A,l-2} \xrightarrow{m_{l-1}} X_i \oplus (m_l \| 0^c) \xrightarrow{m_l} Y_i$$

where $m_l \neq 0^r$. Concretely, it means that $X_i \in \mathtt{AbsP}$. In that case, let $k' := 1$.

Let $M := m_1 \| \cdots \| m_l$. In both cases, $(Y_i, M, k') \in \mathtt{rRooted}(\mathcal{Q}_P)$. Because of $\neg\mathsf{BAD}$, the existence of such a path is unique. Indeed, if X_i is involved in two different paths, then it implies that either a full-state collision or a collision with an element in $\mathtt{AbsP}$ occurred, which are prevented by respectively $\neg\mathsf{COL}$ and $\neg\mathsf{INNER}$. Thus line 14 or 24 of Algorithm 4 is satisfied for this particular M and k', and by construction of the simulator the answer Y is consistent with respect to $\mathcal{RO}$, i.e.,

$$\text{outer}_{r'}(Y) = \mathcal{RO}^{\infty}\left(\mathit{unpad}(M)\right)\left[r' \times (k'-1) + 1 : r' \times k'\right].$$

Moreover, thanks to $\neg\mathsf{CONNECT} \wedge \neg\mathsf{INNER}$, there are no edges starting from Y_i resulting to a valid path. Therefore, this query cannot break the consistency.

Inverse Query. An inverse query breaking the consistency implies that X_i connects either to the IV, or to an already existing path such that the newly created path is valid. These two cases are prevented by $\neg\mathsf{COL} \wedge \neg\mathsf{INNER} \wedge \neg\mathsf{CONNECT}$.

Conclusion. As long as BAD is not set, all queries to $\text{Sponge}^{\mathbf{S}}$ and to $\mathcal{RO}$ give the same output, and both simulators exhibit the same behavior. Therefore,

$$\left|\mathbf{Pr}\left(\mathcal{D}^{W_I} = 1 \mid \neg\mathsf{BAD}\right) - \mathbf{Pr}\left(\mathcal{D}^{W_{IM}} = 1 \mid \neg\mathsf{BAD}\right)\right| = 0\,. \qquad (5.7)$$

5.3.4.4 Probability of BAD

Basic Reasoning. The bad events are defined over the simulator table, and GUESS can only be set in W_{IM}. Moreover, the simulator receives extra queries in W_{IM}, which only increases its success probability to set BAD. We will thus evaluate the probability of BAD in W_{IM}. By basic probability, we have

$$\begin{aligned}
&\mathbf{Pr}_{W_{IM}}\left(\mathsf{BAD}\right)\\
&\le \sum_{i=1}^{q} \mathbf{Pr}_{W_{IM}}\left(\mathsf{BAD}_i \mid \neg\mathsf{BAD}_{i-1}\right)\\
&\le \sum_{i=1}^{q} \Big(\mathbf{Pr}_{W_{IM}}\left(\mathsf{GUESS}_i \mid \neg\mathsf{BAD}_{i-1}\right) + \mathbf{Pr}_{W_{IM}}\left(\mathsf{COL}_i \mid \neg\mathsf{BAD}_{i-1}\right)\\
&\quad + \mathbf{Pr}_{W_{IM}}\left(\mathsf{CONNECT}_i \mid \neg\mathsf{BAD}_{i-1}\right) + \mathbf{Pr}_{W_{IM}}\left(\mathsf{INNER}_i \mid \neg\mathsf{BAD}_{i-1}\right)\Big)\,. \qquad (5.8)
\end{aligned}$$

We evaluate each term individually.

$\mathsf{GUESS\_SQU}_i$. Setting this event is similar to a guessing game: in order to win, the adversary must be able to guess a node N in a valid path. Thanks to the construction oracle, $\mathcal{D}$ has access to the r' upper bits of q_C different nodes. Let $u \in \{0,1\}^{r'}$, we define the random variable F_u as follows:

$$F_u := \left|\{(x, y, fwd, C) \in \text{Ext}(\mathcal{Q}_{\text{Sim}}) \mid \text{outer}_{r'}(y) = u\}\right|\,,$$

i.e., F_u is the number of construction queries which outer part hit u. The distribution of the random variables $(F_u)_{u\in\{0,1\}^{r'}}$ is the same as the bins-and-balls experiment described in Lemma 2.5.2. Now, given a query $v\|w$ with $v \in \{0,1\}^{r'}$ and $w \in \{0,1\}^{c'}$, the probability that $\mathsf{GUESS\_SQU}_i$ is set, conditioned on the query history of the $i-1$ previous queries $\mathcal{Q}_{i-1}$, is upper bounded by

$$\frac{\max_{u \in \{0,1\}^{r'}} F_u}{2^{c'}} .$$

Therefore, by summing over all possible $\mathcal{Q}_{i-1}$ we obtain

$$\begin{aligned}\mathbf{Pr}_{W_{IM}}\left(\mathsf{GUESS\_SQU}_i \mid \neg\mathsf{BAD}_{i-1}\right) &\leq \frac{\mathsf{E}\left(\max_{u \in \{0,1\}^{r'}} F_u\right)}{2^{c'}} \\ &= \frac{\mathrm{mucol}(q_C, 2^{r'})}{2^{c'}} .\end{aligned}$$

Using the second bound of Lemma 2.5.2 to tame multicollisions, we obtain

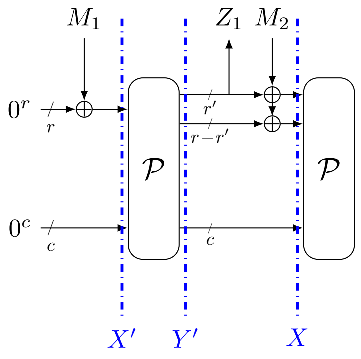
$$\mathbf{Pr}_{W_{IM}}\left(\mathsf{GUESS\_SQU}_i \mid \neg\mathsf{BAD}_{i-1}\right) \leq \frac{2q_C}{2^b} + \frac{3r'+4}{2^{c'}} . \tag{5.9}$$

$\mathsf{GUESS\_ABS}_i$. For the event $\mathsf{GUESS\_ABS}_i$, the situation is more complex. Indeed, assume first that $c = c'$. Then in order to win, the adversary has to guess c bits. The r upper bits do not matter, since the adversary can overwrite them (see Figure 5.2b, where $r' - r = 0$). Therefore, in this setting the adversary is more powerful than in the case of the event $\mathsf{GUESS\_SQU}_i$. However, the number of nodes that the adversary can guess here is upper bounded by $|\mathtt{AbsP}|$, which is much smaller than q_C in some settings. Now, Figure 5.2 illustrates the two different possibilities depending on the values of r' and r. First, if $r' < r$ (cf., Figure 5.2a), the adversary has access to and can control r' bits. It can add any bits to $r - r'$ other bits, but has no access to them. Therefore, in this setting the success probability is upper bounded by

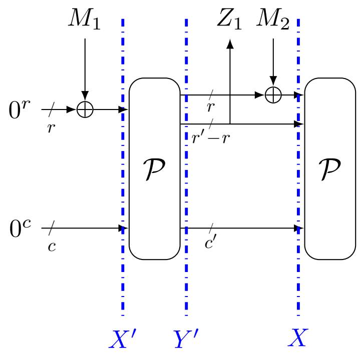
$$\frac{|\mathtt{AbsP}|}{2^{c'}} \leq \frac{|\mathtt{AbsP}|}{2^{c}} .$$

On the other hand, if $r' \geq r$ (cf., Figure 5.2b), the adversary has access and can control r bits, has access to $r' - r$ bits, and has to guess the c' remaining ones. The situation therefore again boils down to a bins-and-balls game. This time, $|\mathtt{AbsP}|$ different balls are thrown in $R := 2^{r'-r}$ bins. For $u \in [\![1, R]\!]$, let S_u be the number of balls in the bucket u. The probability to set $\mathsf{GUESS\_ABS}_i$ conditioned on the query history is upper bounded by

$$\frac{\max_{u \in [\![1,R]\!]} S_u}{2^{c'}} .$$



(a) When $r' < r$.



(b) When $r' \geq r$.

Figure 5.2: Illustration of the event $\mathsf{GUESS\_ABS}$. X' and Y' are nodes from a prior construction query, and the distinguisher makes a forward query with input X.

We obtain

$$\begin{aligned}\mathbf{Pr}_{W_{IM}}\left(\mathsf{GUESS\_ABS}_i \mid \neg\mathsf{BAD}_{i-1}\right) &\leq \frac{\mathsf{E}\left(\max_{u\in[\![1,R]\!]} S_u\right)}{2^{c'}} \\ &= \frac{\mathrm{mucol}(|\mathtt{AbsP}|, 2^{r'-r})}{2^{c'}} .\end{aligned}$$

For simplicity of result's presentation, we will use the second upper bound provided in Lemma 2.5.2 to upper bound the multicollision term. This gives

$$\mathbf{Pr}_{W_{IM}}\left(\mathsf{GUESS\_ABS}_i \mid \neg\mathsf{BAD}_{i-1}\right) \leq \frac{2\,|\mathtt{AbsP}|}{2^c} + \frac{3r'+4}{2^{c'}} . \tag{5.10}$$

COL_i and $\mathsf{CONNECT}_i$. As long as BAD_{i-1} is not set, the remaining events can be set with a non-zero probability only if the query is fresh. Here, the adversary has no access to the randomness source used by the simulator. Thus, the probabilities of the bad events COL_i and $\mathsf{CONNECT}_i$ can be upper bounded as follows:

$$\begin{aligned}\mathbf{Pr}_{W_{IM}}\left(\mathsf{COL}_i \mid \neg\mathsf{BAD}_{i-1}\right) &\leq \frac{i-1}{2^b} , \\ \mathbf{Pr}_{W_{IM}}\left(\mathsf{CONNECT}_i \mid \neg\mathsf{BAD}_{i-1}\right) &\leq \frac{i-1}{2^b} .\end{aligned} \tag{5.11}$$

INNER_i. For any query with a node in $\mathtt{AbsP}$, the probability that it sets INNER_i^1 is upper bounded by $\frac{2q}{2^c}$. Therefore,

$$\mathbf{Pr}\left(\mathsf{INNER}_i^1 \mid \neg\mathsf{BAD}_{i-1}\right) \leq \mathbf{1}_{Y_i \in \mathtt{AbsP}} \times \frac{2q}{2^c}\,.$$

Moreover, the probability to set INNER_i^2 is upper bounded as follows:

$$\mathbf{Pr}\left(\mathsf{INNER}_i^2 \mid \neg\mathsf{BAD}_{i-1}\right) \leq \frac{|\mathtt{AbsP}|}{2^c}\,.$$

Therefore,

$$\mathbf{Pr}\left(\mathsf{INNER}_i \mid \neg\mathsf{BAD}_{i-1}\right) \leq \mathbf{1}_{Y_i \in \mathtt{AbsP}} \times \frac{2q}{2^c} + \frac{|\mathtt{AbsP}|}{2^c}\,. \tag{5.12}$$

Wrap-up. Remark that the event GUESS can be set only with a primitive query from the distinguisher, so that the adversary has at most q_P attempts. Now, plugging (5.9) to (5.12) into (5.8) gives

$$\begin{aligned}
&\mathbf{Pr}_{W_{IM}}\left(\mathsf{BAD}\right)\\
&\leq \sum_{i=1}^{q_P}\left(\frac{2q_C}{2^b} + \frac{2\,|\mathtt{AbsP}|}{2^c} + \frac{6r'+8}{2^{c'}}\right) + \sum_{i=1}^{q}\left(\frac{2(i-1)}{2^b} + \mathbf{1}_{Y_i \in \mathtt{AbsP}} \times \frac{2q}{2^c} + \frac{|\mathtt{AbsP}|}{2^c}\right)\\
&\leq \frac{q_P\,(6r'+8)}{2^{c'}} + \frac{2q^2}{2^b} + \frac{5q\,|\mathtt{AbsP}|}{2^c}\\
&\leq \frac{q_P\,(6r'+8)}{2^{c'}} + \frac{2q^2}{2^b} + \min\left\{\frac{5q(q+1)}{2^c}, \frac{10q}{2^{b-\ell\times r}}\right\}\,,
\end{aligned} \tag{5.13}$$

where the penultimate inequality uses $2q_Cq_P \leq q^2$, and the last inequality uses (5.1). Note that we implicitly used $\neg\mathsf{BAD}_{i-1}$ at each step to upper bound $|\mathtt{AbsP}|$.

Finally, since the bound in (5.13) also applies in W_I, we have

$$(5.5) \leq \frac{q_P\,(12r'+16)}{2^{c'}} + \frac{4q^2}{2^b} + \min\left\{\frac{10q(q+1)}{2^c}, \frac{20q}{2^{b-\ell\times r}}\right\}\,. \tag{5.14}$$

5.3.4.5 Conclusion

Remember that in Section 5.3.4.1, we established that the distinguisher advantage is upper bounded by the sum of (5.2) and (5.5). These terms are themselves upper bounded in respectively (5.6) and (5.14), which concludes the proof. □

5.4 Tightness of the Result

Our result guarantees indifferentiability up to $\min\left\{2^{c'}, 2^{b/2}, \max\left\{2^{c/2}, 2^{b-\ell\times r}\right\}\right\}$ queries (up to logarithmic and constant factors). In the following, we show different attacks that apply to our simulator. The best strategy depends on the parameters used, as clarified in the following.

Attack in $\approx 2^{c'}$ Queries. The second attack from Section 5.2.3 applies here as well. It remains effective as long as the simulator cannot feasibly exhaust the input message space.

Attack in $\approx 2^{b/2}$ Queries. Because the simulator acts as a random function, a simple attack looking for collisions with primitive queries would give a high advantage after $2^{b/2}$ queries. Yet we want to show that, even when the simulator is permutation-consistent, going beyond the birthday barrier seems impossible. Consider the following attack:

1. Make $2^{b/2}$ different forward primitive queries (not starting from the IV) to obtain $X_i \to Y_i$ for $i \in [\![1, 2^{b/2}]\!]$;
2. Choose a message
$$M \xleftarrow{\$} \{M \in \{0,1\}^* \mid |pad_r(M) \neq \bot \text{ and } |pad_r(M)| \leq r \times \ell\} \ ,$$
and expand the path starting from the IV with M by making $2^{b/2}$ forward primitive queries, so that we obtain the following valid path:
$$IV \xrightarrow{M} N_1 \to \cdots \to N_{2^{b/2}} \ ;$$
3. If $\mathsf{CONNECT}$ is set with the node N_k hitting X_i, then make the construction query associated to $IV \xrightarrow{M} N_1 \to \cdots \to N_{k-1} \to X_i \to Y_i$. If the answer is consistent, return "real", otherwise "ideal".

This attack relies on setting the event $\mathsf{CONNECT}$. Any good simulator should set this event with probability $\approx \frac{q^2}{2^b}$ (otherwise one can distinguish). Moreover, provided the message space is too large for exhaustive search, then with high probability the simulator has not queried the random oracle with input M, and none of the answers it defined in the first step matches the stream $\mathcal{RO}^\infty(M)$ (similarly to the second attack from Section 5.2.3, $X_i \to Y_i$ can be extended to longer chains to make the success probability of the simulator negligible).

Attack in $\approx \max\left\{2^{c/2}, 2^{b-\ell\times r}\right\}$ Queries. When $c/2 \geq b - \ell r$, then it means that the number of absorbed message blocks is not limiting, so that the attack described in Section 5.2.3 applies. Otherwise, when $c/2 < b - \ell r$, consider the following attack:

1. Make all possible first $\ell - 1$ absorb calls, so that the list `AbsP` is complete. This costs $\approx 2^{r\times(\ell-1)}$ queries;

2. Choose a message $M \xleftarrow{\$} \{0,1\}^r \backslash \{0^r\}$, and expand the path starting from the IV with M by making $2^{b-\ell\times r}$ forward primitive queries, so that we obtain the following valid path:

$$IV \xrightarrow{M} N_1 \to \cdots \to N_{2^{b-\ell\times r}} \,;$$

3. With high probability there exists an inner collision on c bits between a node $Z \in$ `AbsP` $\setminus \{N_1, \ldots, N_{\ell-1}\}$ and an N_i. If such a collision is found, use the last absorb call on the node Z to transform this partial collision into a full-state collision.

As explained in Section 5.2.3, such a collision would break the consistency with high probability in the ideal world. This attack costs $\max\left\{2^{r\times(\ell-1)}, 2^{b-\ell\times r}\right\} = 2^{b-\ell\times r}$ queries. This attack resembles the classical differentiability attack from Section 5.2.3.

5.5 Public Indifferentiability and Application to Collision and Second Preimage Resistance

As detailed in Section 5.2.1, public indifferentiability is a weaker security notion, which has nevertheless concrete applications. In the setting of public indifferentiability, the simulator is aware of all construction queries made by the distinguisher. We define a simulator $\mathbf{S}'$ which, when being informed of the construction query with input (M, k), where $(M, k, Y) \in$ `rRooted`, makes the corresponding forward query to $\mathbf{S}$. In this setting, GUESS is no more a bad event, yet the remainder of Section 5.3.4.3 remains unchanged. We can copy the proof of Section 5.3.4, but without using the bad event GUESS. This gives a bound

$$\mathbf{Adv}^{\text{R-pubiff}}_{\text{Sponge}}(q_C, q_P, \ell) \leq \frac{3q(q-1)}{2^b} + \min\left\{\frac{6q(q+1)}{2^c}, \frac{12q}{2^{b-\ell\times r}}\right\} .$$

As a matter of fact, the public indifferentiability of regular sponge has not been established so far. We can adapt the proof with two simple modifications: (i) drop the condition $k + |M|/r < \ell$ of line 14 of Algorithm 4; (ii) use that $|\mathtt{AbsP}| \leq q+1$. Thus, for any adversary $\mathcal{A}$ with complexity q, we have

$$\mathbf{Adv}^{\text{pubiff}}_{\text{Sponge}}(\mathcal{A}) \leq \frac{3q(q-1)}{2^b} + \frac{6q(q+1)}{2^c}\,,$$

for *any squeezing rate* r'. This implies that, from a generic perspective, the sponge can safely squeeze over the entire state. Admittedly, due to the proof technique induced by restricted indifferentiability, the constant factors in this bound are likely loose.

In fact, collision and everywhere (second) preimage resistance do not rely on the secrecy of the inputs, so public indifferentiability can be used. Mandal et al. [MPS12, Theorem 3] showed that public indifferentiability implies correlation intractability, a notion introduced by Canetti et al. [CGH98] that is stronger than collision and (second) preimage resistance. For completeness, we revisit the implication of Andreeva et al. [AMP10b, Appendix A] (stated in Lemma 2.4.1) to derive exact collision and second preimage bounds.

Lemma 5.5.1. *Let $\mathcal{H}$ be a XOF construction with primitive set* $\mathtt{Prim}$. *Let $\mathcal{RO}$ be a random oracle, $\mathcal{F} \xleftarrow{\$} \mathtt{Prim}$, and $\mathbf{S}$ be a simulator with oracle access to $\mathcal{RO}$ and that has access to all queries made by $\mathcal{RO}$. Let $\kappa, \nu \in \mathbb{N}$. Let $\mathtt{x} \in \{\text{col}[\nu], \text{ePre}[\nu], \text{eSec}[\kappa,\nu]\}$, and let $\mathcal{A}$ be an $\mathtt{x}$ adversary such that $\mathcal{A}$ outputting a message M has made the query $\mathcal{H}^{\mathcal{F}}(M)$. There exists a public indifferentiability adversary $\mathcal{A}_1$ with respect to the simulator, and an $\mathtt{x}$ adversary $\mathcal{A}_2$ such that*

$$\mathbf{Adv}^{\mathtt{x}}_{\mathcal{H}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{pubiff}}_{\mathcal{H},\mathbf{S}}(\mathcal{A}_1) + \mathbf{Adv}^{\mathtt{x}}_{\mathcal{RO}}(\mathcal{A}_2)\,.$$

Here, $\mathbf{Adv}^{\mathtt{x}}_{\mathcal{RO}}(\mathcal{A}_2)$ slightly abuses notation as $\mathcal{A}_2$ gets direct access to $\mathcal{RO}$ and aims to break $\mathtt{x}$ security. Moreover, $\mathcal{A}_1$ has at most the same query complexity as $\mathcal{A}$, and $\mathcal{A}_2$ has at most the same query complexity as $\mathcal{A}$ and $\mathbf{S}$ combined.

Proof. Let $\mathcal{A}$ be an adversary against property $\mathtt{x}$. We have

$$\begin{aligned}\mathbf{Adv}^{\mathtt{x}}_{\mathcal{H}}(\mathcal{A}) &= \mathbf{Pr}\left(\mathcal{A}^{\mathcal{H}^{\mathcal{F}},\mathcal{F}} \text{ breaks } \mathtt{x}\right)\\ &\leq \left|\mathbf{Pr}\left(\mathcal{A}^{\mathcal{H}^{\mathcal{F}},\mathcal{F}} \text{ breaks } \mathtt{x}\right) - \mathbf{Pr}\left(\mathcal{A}^{\mathcal{RO},\mathbf{S}^{\mathcal{RO}}} \text{ breaks } \mathtt{x}\right)\right| +\\ &\quad \mathbf{Pr}\left(\mathcal{A}^{\mathcal{RO},\mathbf{S}^{\mathcal{RO}}} \text{ breaks } \mathtt{x}\right),\end{aligned}$$

where "$\mathcal{A}^W$ breaks x" means that $\mathcal{A}^W$ wins the security game associated to property x. For the first term (i.e., the absolute value), from $\mathcal{A}$ we build a distinguisher $\mathcal{A}_1$ that relays faithfully $\mathcal{A}$'s queries, and at the end of the game, $\mathcal{A}_1$ evaluates whether $\mathcal{A}$ broke x, and returns 1 if it is the case, else 0. Since by assumption $\mathcal{A}$ must make the associated hash function evaluations for any output it produces, we have

$$\left|\mathbf{Pr}\left(\mathcal{A}^{\mathcal{H}^{\mathcal{F}},\mathcal{F}} \text{ breaks } \mathtt{x}\right) - \mathbf{Pr}\left(\mathcal{A}^{\mathcal{RO},\mathbf{S}^{\mathcal{RO}}} \text{ breaks } \mathtt{x}\right)\right| = \\ \left|\mathbf{Pr}\left(\mathcal{A}_1^{\mathcal{H}^{\mathcal{F}},\mathcal{F}} \to 1\right) - \mathbf{Pr}\left(\mathcal{A}_1^{\mathcal{RO},\mathbf{S}^{\mathcal{RO}}} \to 1\right)\right| = \mathbf{Adv}^{\text{pubiff}}_{\mathcal{H},\mathbf{S}}(\mathcal{A}_1)\,.$$

For the second term, from $\mathcal{A}$ we build an adversary $\mathcal{A}_2$ that: (i) relays $\mathcal{A}$'s construction queries to the random oracle; (ii) runs locally the simulator $\mathbf{S}$ to answer $\mathcal{A}$'s primitive queries, making any necessary $\mathcal{RO}$ query itself. Since $\mathtt{x} \in \{\text{col}[\nu], \text{ePre}[\nu], \text{eSec}[\kappa, \nu]\}$, $\mathcal{A}$ is the only entity making random oracle queries. Therefore, from the query history of $\mathcal{A}$, $\mathcal{A}_2$ can exactly deduce what inputs where queried to the random oracle, and hence it can safely run the public indifferentiability simulator $\mathbf{S}$ by itself without decreasing $\mathcal{A}$'s success probability. Moreover, the query complexity of $\mathcal{A}_2$ is at most the query complexity of $\mathcal{A}$ and $\mathbf{S}$ accumulated. Hence,

$$\mathbf{Pr}\left(\mathcal{A}^{\mathcal{RO},\mathbf{S}^{\mathcal{RO}}} \text{ breaks } \mathtt{x}\right) = \mathbf{Pr}\left(\mathcal{A}_2^{\mathcal{RO}} \text{ breaks } \mathtt{x}\right) = \mathbf{Adv}^{\mathtt{x}}_{\mathcal{RO}}(\mathcal{A}_2)\,.$$

This concludes the proof. □

Note that this implication does not hold for domain-oriented preimage resistance, as the preimage is hidden from the adversary $\mathcal{A}_2$.

Application to Collision and Second Preimage Resistance. For (everywhere) preimage resistance, we already deduced tight bounds in Chapter 4, we thus focus on collision and second preimage resistance. We can deduce the following result.

Corollary 5.5.1.1. *Let $b, c, r, q \in \mathbb{N}^*$ with $b = r + c = c' + r'$. Consider the sponge construction of Algorithm 1 with parameters $b, c, r, c', r', c'' = c, r'' = r$. Let $\mathcal{A}$ be an adversary that makes q queries. We have*

$$\mathbf{Adv}^{\text{col}[\nu]}_{\text{Sponge}}(\mathcal{A}) \leq \frac{3q(q-1)}{2^b} + \frac{6q(q+1)}{2^c} + \frac{q(q-1)}{2^{\nu+1}}\,,$$
$$\mathbf{Adv}^{\text{eSec}[\kappa,\nu]}_{\text{Sponge}}(\mathcal{A}) \leq \frac{3q(q-1)}{2^b} + \frac{6q(q+1)}{2^c} + \frac{q}{2^{\nu}}\,.$$

Therefore, the generic attacks described in Section 3.1.3 are tight for any squeezing rate r'.

Remark 5.5.1. *For simplicity, this chapter focused on the setting where the first absorption rate r'' equals to the subsequent absorption rate r. However, the public indifferentiability bound can be extended to the case where $r'' \geq r$, by adapting the definitions of the bad events* COL *and* CONNECT *to account for the fact that the initial value has a smaller capacity. This would lead to a bound of the form*

$$\mathbf{Adv}^{\text{pubiff}}_{\text{Sponge}}(\mathcal{A}) = \tilde{\mathcal{O}}\left(\frac{q(q+1)}{2^c} + \frac{q}{2^{c''}}\right),$$

for any adversary $\mathcal{A}$ with complexity q.

5.6 Conclusion

In this chapter, we showed that the sponge construction may achieve an improved indifferentiability bound when restricted to inputs of limited length. Depending on the parameters, this result can be used to either increase security or allow for higher absorption rates in applications requiring fixed-length input hash functions. However, the security remains inherently limited by the birthday bound in the size of the underlying permutation.

We also established public indifferentiability in this restricted-input setting. Notably, this result applies to the (unrestricted) sponge construction as well. To the best of our knowledge, this is the first treatment focusing on the sponge's public indifferentiability. Unlike the (plain) Merkle-Damgård construction [Mer89,Dam89], which is differentiable from a random oracle [CDMP05], and which received subsequent analysis in the public indifferentiability setting [DRS09], the sponge construction's design is sound: the standard indifferentiability bound is tight, and implies a tight public indifferentiability bound whenever $c' = c$. Interestingly, we observe that the public indifferentiability bound is independent of the size of the capacity when squeezing c', suggesting that in scenarios where inputs do not need to be secret, one may generically squeeze over the entire state to improve throughout.

Chapter 6

Permutation-Based Hashing Beyond the Birthday Bound

6.1 Introduction

We already discussed in Chapter 3 that the sponge is tightly indifferentiable from a random oracle up to $\approx 2^{c/2}$ queries (see Theorem 3.1.1). In particular, in order to obtain a sponge-based hash function with k bits of security, one must use a permutation of size at least $2k+1$ bits. On the other hand, permutation-based *keyed* applications achieve a higher level of security. A long line of research on keyed applications of the sponge [BDPV07, BDPV11d, CDH$^+$12, ADMV15, GPT15, MRV15, NY16, DMP22] and duplex [BDPV11b, JLM14, SY15, MRV15, DMV17, DM19a, JLM$^+$19, CJN20, Men23] has demonstrated that typical sponge-/duplex-based encryption, authentication, and authenticated encryption can achieve around $\min\{2^c/\mathcal{M}, 2^k/\mu\}$ security, where k is the key size, μ is the number of users, and $\mathcal{M}$ the online complexity (see Sections 3.2.2 and 3.3.1.3). Clever use of domain separators can even push security to $\min\{2^c, 2^k/\mu\}$ (see, e.g., [Men23, Corollary 1], as we discussed in Section 3.2.2.3).

The gap between $2^{c/2}$ security in keyless applications and $2^c/\mathcal{M}$ security in keyed applications (assuming the key is long enough) is insignificant in case a general purpose permutation such as Keccak-f[1600] [BDPV11d] is available. However, when only small permutations are available, the situation changes drastically. Suppose, for the sake of example, we have the ISO/IEC standardized permutation P_{256} of the PHOTON family [GPP11] at our disposal (a comparable example can be given for ISO/IEC standardized Spongent [BKL$^+$11]). If the permutation is used in duplex-based authenticated encryption, we obtain generic 2^{192} security provided that the online complexity is bounded by 2^{64} blocks [Men23, Section 9]. In a keyless setting, however, in the plain sponge construction this permutation yields 2^{127} security at best! The contrast becomes even more pronounced for smaller permutations. For example, NIST Lightweight Cryptography [Nat19] finalist Elephant [BCDM20] is a permutation-based authenticated scheme instantiated with three different permutations of sizes respectively 160, 176, and 200 bits. Assuming that the online complexity is bounded by 2^{50} bytes (the limit as stated in the call for proposals of the NIST Lightweight Cryptography competition), Elephant achieves 112, 127, and 127 bits of security in the random permutation model. On the other hand, the Elephant finalist did *not* offer a hashing functionality, the reason simply being that the three permutations were too small for hashing [BCDM20, Section 1].

This issue is not unique for the sponge: alternative permutation-based hash functions such as the modes of Grindahl [KRT07], the SHA-3 finalists Grøstl [GKM$^+$11] and JH [Wu11], or the generalized parazoa framework [AMP12] achieve security up to $b/2$ bits at best (see also Table 6.2 later on).

This leads to a fundamental question as to whether it is possible to obtain more than $b/2$ bits of security for a hash function based on a permutation of size b bits, i.e., to beat the birthday bound in the underlying permutation size. Note that this is not just a theoretical question: a construction of this type would allow to hash with smaller permutations, and it would allow to reduce the gap between the required permutation sizes for keyed and keyless permutation-based constructions for a given security level.

6.1.1 Compression Functions Based on Permutations

To answer this question, one direction is to search for a secure permutation-based compression function. Indeed, to facilitate the design of hash functions, one can start from a robust compression function upon which one applies a domain extender to turn it into a hashing mode. Yet the question of finding "good" compression functions based on permutations is much more challenging than for block ciphers, since as opposed to the latter, permutations are non-compressing primitives. Stam's conjecture [Sta08], later proved by Steinberger [Ste10] and Steinberger et al. [SSY12], implies that for a compression function compressing m bits, collision resistance of 2^s requires at least $\frac{s+m}{b-s}$

different b-bit permutation calls per compression function call, which is hard to reach in practice. In addition, most of the existing literature regarding permutation-based compression functions [RS08, SS08, MP12] only focus on collision resistance and preimage resistance. In particular, no compression function construction is known to guarantee security in the indifferentiability framework. On top of that, looking at the sponge, it has an insecure compression function, suggesting that achieving security at the compression function level is overkill.

6.1.2 Double Block Length Hashing Based on a Block Cipher

Instead of focusing on designing a secure compression function, we rather take inspiration from the double block length hashing design rationale, an idea that dates back to Meyer and Schilling with the design of MDC-2 and MDC-4 [MS88]. It consists of using a state twice as large as the primitive size with at least two primitive calls per compression function call. In block cipher-based hashing, the literature studying compression functions for double block length hashing is vast [MS88, Sta08, Sta09, LM92, Hir06, Hir04, ÖS09, LS15, JÖS12, Men12, LSS11, LK11, FGL09, KP97, Men14, AFK+11, AFK+11]. All of these constructions have been proven collision/preimage resistant, but none of them is indifferentiable (or only with a weak bound), as shown in [Men13]. Thus a dedicated proof is required to prove indifferentiability at the hash function level. Examples of proven indifferentiable hash function constructions include Naito's construction using Hirose's compression function without feed-forward [Nai17], MDPH [Nai19], used by the NIST lightweight cryptography finalist Romulus [IKMP20], and EXEX-NI from Naito et al. [NSS21]. The latter scheme uses an invertible (thus insecure) and efficient compression function, and the construction is indifferentiable up to $n - \log(n)$ bits, where n denotes the block size. We will look into the same direction, i.e., aim at finding an efficient double block length hashing scheme based on permutations with an insecure compression function.

6.1.3 Our Contribution

In detail, we propose a double block length hashing mode based on two permutations of size $b = r + c$ bits, called the "double sponge". The double sponge is described in detail in Section 6.2 and depicted in Figure 6.1. The scheme can be seen as two sponges whose states are blended together after each absorption of a message block, using an efficient mixing function. Similarly to most double block length hashing schemes, the double sponge processes the same message block through two different permutation calls. As before, its security and efficiency depend on the parameters r and c, which are also called rate and capacity, respectively, as before.

An indifferentiability proof of the double sponge up to a bound of $2c/3$ bits is given in Section 6.3. The simulator introduced for this proof is designed to be indistinguishable from a random permutation up to $2^{2c/3}$ queries, as we demonstrate in Lemma 6.4.4. The major challenge in the design of the simulator is to guarantee that it provides answers matching the ones of the random oracle. Indeed, as we target a security bound beyond $2^{c/2}$, inner collisions may occur and the simulator has to deal with them. This problem is resolved mostly by allowing the simulator to selectively define certain (but a limited number of) queries in advance, and by defining sophisticated bad events taming the inner collisions. The simulator is described in detail in Section 6.3.2, including an extensive rationale.

In Section 6.5, we describe a differentiability attack with respect to our simulator which succeeds after approximately $2^{\frac{2c+r}{3}}$ queries. The attack gets close to the security bound of the double sponge, but admittedly leaves a (small) gap of $2^{\frac{r}{3}}$ queries. This gap is likely caused by restrictions we had to impose to avoid inner multi-collisions. Releasing this conditions, i.e., allowing inner multi-collisions, would incur a more complex simulator tree structure and new, even more sophisticated, bad events. In Section 6.6 we conclude the work, and in particular discuss in more detail the (im-)possibilities to improve the differentiability attack or the indifferentiability proof.

6.1.4 Comparison with Existing Hashing Modes

The double sponge allows to have a hash function using a b-bit permutation while it was not possible with existing permutation-based constructions when the target level of security κ is between $\frac{1}{2}(b-1)$ and $\frac{2}{3}(b-1)$. In particular, for $\kappa = 112$ (corresponding to NIST Lightweight Cryptography requirements), one needs a double sponge with $c \geq 168$ (as opposed to $c \geq 224$ for the plain sponge). This concretely means that if one has at their disposal only the ISO/IEC standardized 176-bit Spongent, or the 196-bit PHOTON permutation, then one can use them with the DS and attain 112 bits of security. One can even use smaller variants of these standardized primitives and still attain a decent level of security, although slightly below $\kappa = 112$. Moreover, for certain parameter sets, using a DS can improve the security, without sacrificing the number of bits compressed per permutation call. More precisely, for a plain sponge with a rate of r bits and primitive size b bits, when $r \leq \frac{b}{5}$, then using

Table 6.1: Comparison of the double sponge (DS) with several double block length hashing modes. r and c denote respectively the rate and capacity used in the DS. n and k denote respectively the block size and the key size of the block cipher-based hashing modes. The security bound is in bits, and holds in the indifferentiability framework. In the security bound of block cipher-based modes, logarithmic factors in n are omitted.

Mode	# Primitives per compression	Compression rate (ρ)	Security bound (κ)	State size	Primitive size		Note	Reference
					input	output		
Mennink's in, e.g., ChopMD	3	n	$\frac{n}{2}$	$4n$	$2n$	n	$k = n$	[Men13]
MDPH	2	$k - n$	n	$3n + k$	$n + k$	n	$k > n$	[Nai19, GIM22]
EXEX-NI	2	$k - n$	n	$n + k$	$n + k$	n	$k \geq 2n$	[NSS21]
DS	2	r	$\frac{2c}{3}$	$2(r + c)$	$r + c$	$r + c$		This work

DS with a rate of $2r$ improves the security bound. Of course, this security improvement does not come for free: our construction requires two distinct permutations,[1] a state twice as large, and additionally two multiplications by 2 and additions in $\mathrm{GF}(2^b)$ per compression function evaluation.

6.1.4.1 Comparison with Double Block Length Hashing Modes

In Table 6.1, we compare DS with the most notable double block length hashing modes that were proven to be indifferentiable from a random oracle: Mennink's indifferentiable double block length compression function in any suitable indifferentiable hashing mode [Men13], MDPH [Nai19], and EXEX-NI [NSS21]. In this discussion we exclude Naito's mode on top of Hirose's compression function [Nai17] and the MDC-4 [MS88] compression function in any suitable hashing mode [Men13] as they give worse numbers. We also do not include hashing modes that were only proven collision resistant; for this, we refer to the discussion of Naito et al. [NSS21, Table 1]. It is important to note that block ciphers are compressing primitives, and for MDPH and EXEX-NI we require the key size to be larger than the block size ($k > n$ and $k \geq 2n$, respectively). Henceforth, in our comparison we consider the primitive *input* size, which equals $k + n$ for block cipher-based modes and $r + c$ for the DS.

If we restrict our focus to a fixed security bound κ and a fixed rate ρ, MDPH and EXEX-NI require a primitive of size $\rho + 2\kappa$ bits whereas DS requires a primitive of size $\rho + \frac{3}{2}\kappa$ bits. Mennink's construction is restricted to $2\kappa = \rho = n$, in which case the primitive input size is 4κ as opposed to $\frac{7}{2}\kappa$ for DS.

[1]However, we discuss the possibilities of using a single permutation in Section 6.3.5.

Table 6.2: Comparison of the double sponge (DS) with several permutation-based hashing modes. r and c denote respectively the rate and capacity used in the plain sponge. The best known attack and security bound are in bits, and the latter holds in the indifferentiability framework.

Mode	# Primitives per compression	Compression rate (ρ)	Security bound (κ)	Security attack (λ)	State size	Primitive size	Reference
Sponge	1	r	$\frac{c}{2}$	$\frac{c}{2}$	$r+c$	$r+c$	[BDPV07, BDPV08]
Grøstl	2	b	$\frac{b}{4}$	$\frac{b}{2}$	$3b$	b	[GKM+11, AMP10a]
JH	1	$\frac{b}{2}$	$\frac{b}{4}$	$\frac{b}{2}^{(*)}$	b	b	[Wu11, BMN10, MPS16]
DS	2	r	$\frac{2c}{3}$	$\frac{2c+r}{3}$	$2(r+c)$	$r+c$	This work

$^{(*)}$: [MPS16] mentions that the indifferentiability bound on JH is not tight and suggests that it could possibly be improved further to $\frac{b}{2}$ bits.

The gain comes at a cost in state size. Again fixing κ and ρ, EXEX-NI requires a state size of $\rho+2\kappa$ bits (that mode is specifically designed for having a low memory size) as opposed to $2\rho+3\kappa$ for DS. MDPH has a state size of $\rho+4\kappa$, which is better than DS only if $\rho \geq \kappa$.

6.1.4.2 Comparison with Permutation-Based Hashing Modes

We replicate our analysis in Table 6.2, focusing this time on permutation-based hashing modes that have a proven indifferentiability bound. We draw the comparison between DS on the one hand and the sponge [BDPV07, BDPV08], Grøstl [GKM+11, AMP10a], and JH [Wu11, BMN10, MPS16] on the other hand. We do not include the Grindahl [KRT07,AMP12] and PHOTON [GPP11, NO14] modes, as they have identical metrics to the sponge, nor the general parazoa framework [AMP12].

It is important to note that for DS, as well as for Grøstl and JH, the proven security bound κ is not tightly matching the best attack λ. Nevertheless, regardless of whether we consider the the best known attack or the security bound, the DS outperforms the sponge in terms of permutation size. In detail:

- for fixed security bound κ and a fixed rate ρ, the sponge requires a primitive of size $\rho+2\kappa$ bits whereas DS requires a primitive of size $\rho+\frac{3}{2}\kappa$ bits;
- for fixed security attack λ and a fixed rate ρ, the sponge requires a primitive of size $\rho+2\lambda$ bits whereas DS requires a primitive of size $\frac{\rho}{2}+\frac{3}{2}\lambda$ bits.

The improvement of DS over the sponge comes at a cost in extra state size: DS operates on a state of $2\rho + 3\kappa$ bits as opposed to $\rho + 2\kappa$ for the sponge.

Grøstl is restricted to $\rho = 4\kappa$, in which case the permutation size is 4κ too, as opposed to $\frac{11}{2}\kappa$. Thus, for the restricted regime of Grøstl, Grøstl outperforms the other schemes. JH is restricted to $2\kappa = \rho = \frac{b}{2}$, in which case the permutation size is 4κ as opposed to $\frac{7}{2}\kappa$ for DS. If we consider instead the best known attack λ, JH is restricted to $\lambda = \rho$, in which case both JH and the DS require a primitive of size 2λ bits. Therefore, there is little to no difference between JH and DS. These results may not come as a surprise as both Grøstl and JH have a large absorption rate, and cannot be proven secure beyond the birthday bound in the permutation size.

6.1.5 Notation

If $\boldsymbol{x}$ is a vector with k coordinates, then for any $i \in \{1, \ldots, k\}$, $\boldsymbol{x}_i$ is the i^{th} element of $\boldsymbol{x}$. If S is a set of size 2, then for $s \in S$, $\bar{s}$ is the unique element in S different from s. Finally, given a dictionary `dict` with keys in a set S, $\mathbf{Img}(\texttt{dict}) := \{\texttt{dict}[x] \mid x \in S\} \setminus \{\perp\}$, $\mathbf{Dom}(\texttt{dict}) := \{x \in S \mid \texttt{dict}[x] \neq \perp\}$.

6.2 Double Sponge

In this section we describe our hashing mode, and prove its indifferentiability in Section 6.3.

6.2.1 Description

Let $b, r, c \in \mathbb{N}^*$, with $b = r + c$. On a high level view, the double sponge can be seen as two sponges with a state mixing at each iteration. The construction operates on a state of size $2b$ bits, and requires two cryptographic permutations $\mathcal{P}_{top}$ and $\mathcal{P}_{bot}$ over b bits. One absorption call enables to process r bits of the padded message, and during the squeezing phase, r bits of the digest are extracted at a time. The top and bottom permutations $\mathcal{P}_{top}$ and $\mathcal{P}_{bot}$ will be considered distinct, independent, and uniformly random in the indifferentiability proof. The mixing is performed by applying to the state a 2×2 matrix with coefficients over $\mathrm{GF}(2^b)$, defined as follows:

$$MIX = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}. \tag{6.1}$$

Given $\mathcal{P} \in (\texttt{Perm}(b))^2$, $\mathrm{DS}^{\mathcal{P}}$ denotes DS instantiated with the permutations $\mathcal{P}$. Our construction is defined in Algorithm 5 and depicted in Figure 6.1.

Algorithm 5 The double sponge construction based on $\mathcal{P} := (\mathcal{P}_{top}, \mathcal{P}_{bot})$. The algorithm takes as input the message to hash M and the number of bits of the stream to extract ν. *pad* is an injective padding with the same restrictions as for the sponge construction (Figure 3.1). IV^{top} and IV^{bot} are two fixed initialization vectors.

```
function $\mathrm{DS}^{\mathcal{P}}(M, \nu)$
    // The state is initialized, the message is padded
    $(S^{top}, S^{bot}) \leftarrow (IV^{top}, IV^{bot})$
    $M_1 \| \cdots \| M_k \leftarrow pad_r(M)$
    // Absorption
    for $i = 1, \ldots, k$ do
        $(S^{top}, S^{bot}) \leftarrow MIX\left(\mathcal{P}_{top}(S^{top} \oplus (M_i \| 0^c)), \mathcal{P}_{bot}(S^{bot} \oplus (M_i \| 0^c))\right)$
    // Squeezing
    for $i = 1, \ldots, \lceil \nu / r \rceil$ do
        $Z_i \leftarrow \mathrm{outer}_r((S^{top}, S^{bot}))$
        $(S^{top}, S^{bot}) \leftarrow MIX\left(\mathcal{P}_{top}(S^{top}), \mathcal{P}_{bot}(S^{bot})\right)$
    return $(Z_1 \| \ldots \| Z_{\lceil \nu / r \rceil})[1 : \nu]$
```

6.2.2 Design Rationale

For simplicity of the proof, the double sponge uses two independent permutations, but we discuss in Section 6.3.5 a tweak of the double sponge to use the same permutation. The top and bottom parts of the double sponge should be entangled at the compression function level, as otherwise one can exploit the independency of the top and bottom parts, and find attacks like the ones on the iterated concatenated combiners [Jou04]. For that reason, we require the *MIX* matrix to be MDS, and such that the two coefficients in both rows are not equal. Given these constraints, we opted for a simple choice, i.e., (6.1). Note that the coefficient 2 in this matrix can be any non-zero coefficient in $\mathrm{GF}(2^b)$ different from 1.

The message is absorbed at both the top *and* bottom parts of the state. This ensures that it influences both top and bottom input permutation calls. It is not possible to absorb two different message blocks during one compression function call, since from i top and bottom queries, one can then build i^2 different states (i.e., rooted nodes, see Section 6.2.3). The probability of find-

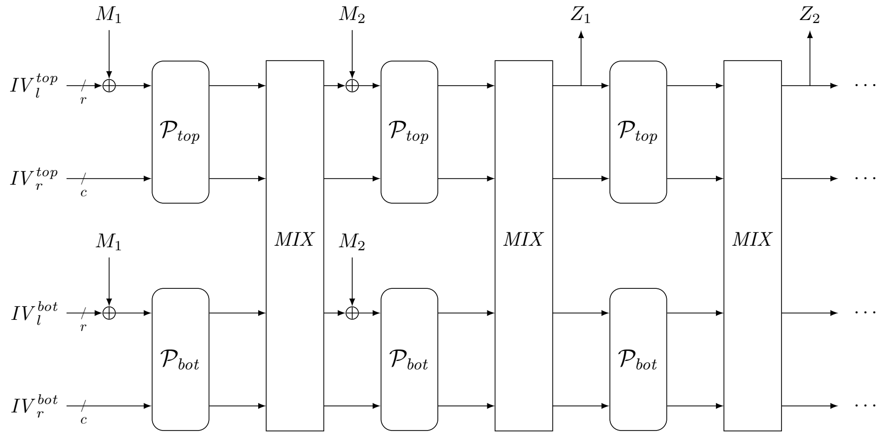


Figure 6.1: The double sponge. In this example, the input message M is padded as $M_1 \| M_2 \leftarrow pad_r(M)$. The initial values $IV^{top}, IV^{bot} \in \{0,1\}^b$ are split as $IV_l^{top} \| IV_r^{top}$ and $IV_l^{bot} \| IV_r^{bot}$ with $IV_l^{top}, IV_l^{bot} \in \{0,1\}^r$, $IV_r^{top}, IV_r^{bot} \in \{0,1\}^c$.

ing full-state collisions would increase to the birthday bound in the capacity. Finally, it is not possible either to squeeze both the top and bottom parts, because looking ahead, doing so makes the linear equation generated by the simulator unsolvable, see point 2 of Section 6.3.2.

6.2.3 Graph Representation

In this section we provide a graph representation of the states of the double sponge similarly to the sponge construction. Assume in the following that the permutation outputs are lazily sampled, so that we only know a fraction of the permutation evaluations. The set of nodes is $V = \{0,1\}^{2b}$, and elements are represented using the notation (A^{top}, A^{bot}), where $A^{top} \in \{0,1\}^b$ is called the top part and $A^{bot} \in \{0,1\}^b$ the bottom part. For $(A^{top}, A^{bot}), (B^{top}, B^{bot}) \in V$, and $m \in \{0,1\}^r$, we add the edge $(A^{top}, A^{bot}) \xrightarrow{m} (B^{top}, B^{bot})$ whenever

$$MIX\left(\mathcal{P}_{top}(A^{top} \oplus (m \| 0^c)), \mathcal{P}_{bot}(A^{bot} \oplus (m \| 0^c))\right) = (B^{top}, B^{bot}),$$

and the two underlying permutation evaluations are defined. A special subset of the nodes is the set of *rooted nodes*, where $(A^{top}, A^{bot}) \in$ `Rooted` whenever (A^{top}, A^{bot}) is accessible from (IV^{top}, IV^{bot}). Given a rooted node (A^{top}, A^{bot}), one can retrieve the unique[2] XOF call $\mathrm{DS}(M)[i : i+r-1]$ which is associated to this state. Finally, we refer to *partial edges* when there exists $(A^{top}, A^{bot}) \in \texttt{Rooted}, m \in \{0,1\}^r$, and $s \in \{top, bot\}$ such that $\mathcal{P}_s(A^s \oplus (m\|0^c))$ is known, but $\mathcal{P}_{\bar{s}}(A^{\bar{s}} \oplus (m\|0^c))$ is not known.

6.3 Indifferentiability of the Double Sponge

In this section, we prove that the double sponge is indifferentiable from a $\mathcal{RO}$ with a bound of $\tilde{\mathcal{O}}\left(q^{\frac{3}{2}}/2^c\right)$, as stated in Theorem 6.3.1. In Section 6.3.5 we discuss an extension of our result when using a single permutation.

Construction Queries and Their Cost. As discussed in Section 3.1.2.1, we measure construction queries in terms of the number of permutation evaluations required in the real world to produce the output. More precisely, if $|pad_r(M)| = l \times r$, then a query with input M and k has a cost of $l + k - 1$. However, unlike the setting in Section 3.1.2.1, and similarly to Chapter 5, we distinguish between construction-originated and primitive-originated queries when counting queries. We will use q_P to denote the number of primitive queries and q_C to denote the query cost of constructions queries.

6

Theorem 6.3.1. *Let* DS *be the construction from Section 6.2. Let* $q := q_C + q_P$*. Let* $\mathcal{D}$ *be an adversary with complexity* (q_P, q_C) *If* $30 < 2^c$ *and* $3q < 2^c$*. There exists a simulator* $\mathbf{S}$ *such that*

$$\mathbf{Adv}^{\mathrm{indif}}_{\mathrm{DS},\mathbf{S}}(\mathcal{D}) \leq \frac{40q^{\frac{3}{2}}}{2^c - 3q}\,.$$

Moreover, $\mathbf{S}$ *makes a total of at most* $3q_P$ *random oracle queries, and its overall memory storage and runtime are linear in* q_P*.*

The proof of this theorem is given in the remainder of this section.

6.3.1 General Setting of the Proof and Outline

The adversary must distinguish between the real world $W_R := (\mathrm{DS}^{\mathcal{P}}, \mathcal{P}, \mathcal{P}^{-1})$ and the ideal world $W_I := (\mathcal{RO}, \mathbf{S}^{\mathrm{fwd}}, \mathbf{S}^{\mathrm{inv}})$. To do so, it can perform construction queries by using the procedure `ConsQuery`(), which takes as input

[2]This call is unique as long as there is no collision on $2c + r$ bits between two rooted nodes.

a message $M \in \{0,1\}^*$, and the index of the stream $k \in \mathbb{N}$, corresponding to $\mathcal{RO}^\infty(M)[k \times r + 1 : (k+1)r]$ in the ideal world or the $(k+1)^{\text{th}}$ squeeze call after absorbing M in the real world. Forward (resp., inverse) queries are made via the interface `FwdPQuery` () (resp., `InvPQuery` ()). Both take as input an element $s \in \{top, bot\}$ and the element to query in $\{0,1\}^b$. To prove indifferentiability, we use the simulator described in Section 6.3.2 which defines on-the-fly permutation-consistent answers and logs its responses in two dictionaries $\mathtt{tabP}_{top}$ and $\mathtt{tabP}_{bot}$. From these two tables, it is able to keep track of the graph representation defined in Section 6.2.3. For simplicity of the proof, we require that the simulator's graph does not contain partial edges. In other words, if (A^{top}, A^{bot}) is a rooted node, and there exists $m \in \{0,1\}^r, s \in \{top, bot\}$ such that $\mathtt{tabP}_s[A^s \oplus (m\|0^c)] \neq \perp$, then *necessarily*, $\mathtt{tabP}_{\bar{s}}[A^{\bar{s}} \oplus (m\|0^c)] \neq \perp$. In the real world, we assume that the primitive oracle decides outputs using lazy sampling. Thus, in order to ease the comparison between the real world and the ideal world, we introduce world W_{IM2} illustrated in Figure 6.3c (see Figure 6.3 for a depiction of all worlds considered in this proof), which implements the real world with a supplementary layer called `GraphProc`, ensuring that the graph deducible from the query history of `GraphProc` does not contain partial edges either. In worlds W_{IM2} and W_I, we implicitly consider that the collateral permutation outputs are given *for free*. Now, W_{IM2} and W_I can be differentiated in two ways: consistency with $\mathcal{RO}$, and the statistical difference between the simulator's answers with the ones of a random primitive. Each of these points is addressed separately, thanks to the introduction of another intermediate world W_{IM1}, as illustrated in Figure 6.3b. A similar game splitting was done among others in the indifferentiability proof of PHOTON [NO14]. Section 6.3.3 introduces bad events for the $\mathcal{RO}$-consistency. Section 6.3.4 explains more rigorously the intermediate worlds W_{IM1} and W_{IM2}. This part also splits the underlying probability computation, which is addressed in detail in Section 6.4. This eventually leads to the proof's conclusion.

6.3.2 Simulator Definition

In this section we provide the high level definition of our simulator, with its formal procedures provided in Algorithm 6.

6.3.2.1 On Forward Query

For any query `FwdPQuery` (top, X) or query `FwdPQuery` (bot, X), the simulator goes through several phases, as explained in the following.

Algorithm 6 Simulator **S** main functions. `UpdateGraph` () takes care of updating the set `Rooted` according to $\texttt{tabP}_{top}$ and $\texttt{tabP}_{bot}$. `LinSolve` () is described in paragraph 2 from Section 6.3.2.

```
function Init()
    $\texttt{Rooted} \leftarrow \{(IV^{top}, IV^{bot})\}$
    $\texttt{tabP}_{top} \leftarrow \texttt{EmptyDict}\,()$; $\texttt{tabP}_{bot} \leftarrow \texttt{EmptyDict}\,()$
function $\mathbf{S}^{\mathrm{fwd}}(s, X)$
    input: $(s, X) \in \{top, bot\} \times \{0,1\}^b$
    output: $Y \in \{0,1\}^b \cup \{\perp\}$
    if $\texttt{tabP}_s[X] \neq \perp$
        return $\texttt{tabP}_s[X]$
    NodesToAns $\leftarrow \texttt{NodeCollection}\,(s, X)$
    // If query is not on the graph
    if NodesToAns $= \emptyset$
        $\texttt{tabP}_s[X] \xleftarrow{\$} \{0,1\}^b \setminus \mathbf{Img}(\texttt{tabP}_s)$
        return $\texttt{tabP}_s[X]$
    $\texttt{DictOutParts}_{top}, \texttt{DictOutParts}_{bot} \leftarrow \texttt{LinSolve}\,(\mathbf{NodesToAns})$
    // If linear system solving failed
    if $\exists s, \texttt{DictOutParts}_s = \perp$
        return $\perp$
    $\texttt{PermConsOutputTop}\,(\texttt{DictOutParts}_{top})$
    $\texttt{PermConsOutputBot}\,(\texttt{DictOutParts}_{bot})$
    UpdateGraph ()
    EnsureNoPartialEdges ()
    return $\texttt{tabP}_s[X]$

function $\mathbf{S}^{\mathrm{inv}}$(s,Y)
    input: $(s, Y) \in \{top, bot\} \times \{0,1\}^b$
    output: $X \in \{0,1\}^b \cup \{\perp\}$
    if $\exists X \in \{0,1\}^b, \texttt{tabP}_s[X] = Y$
        return $X$
    $X \xleftarrow{\$} \{0,1\}^b \setminus \mathbf{Dom}(\texttt{tabP}_s)$
    $\texttt{tabP}_s[X] = Y$
    UpdateGraph ()
    EnsureNoPartialEdges ()
    return $X$
```

1. Collection of Nodes to Expand. The simulator first collects all rooted nodes *after absorption of a message block* where it needs to provide $\mathcal{RO}$-consistent answers. The goal of this procedure is to guarantee that at the end of the query execution, no partial edges exist on the simulator graph. Because the number of queries can go beyond $2^{c/2}$, it is not unlikely that some rooted nodes collide on their top or bottom inner part, thus more than one node may be collected during this step. Example 6.3.1 gives a minimal example of such a case.

Example 6.3.1. *Let* $(A^{top}, A^{bot}), (B^{top}, B^{bot}) \in \texttt{Rooted}$ *be two distinct nodes such that* $A^{bot} \overset{c}{=} B^{bot}$. *Assume that the query is* $\texttt{FwdPQuery}\,(top, A^{top} \oplus (m\|0^c))$ *for* $m \in \{0,1\}^r$, *and let* $m' := m \oplus \text{outer}_r(A^{bot} \oplus B^{bot})$. *Then the simulator needs to complete an edge from* (A^{top}, A^{bot}) *with the message* m, *i.e., decide on* $\texttt{tabP}_{top}[A^{top} \oplus (m\|0^c)]$ *and* $\texttt{tabP}_{bot}[A^{bot} \oplus (m\|0^c)]$. *Nevertheless, since* $A^{bot} \oplus (m\|0^c) = B^{bot} \oplus (m'\|0^c)$, *the query will then also fix* $\texttt{tabP}_{bot}[B^{bot} \oplus (m'\|0^c)]$, *so that to avoid a partial edge starting from* (B^{top}, B^{bot}), *the simulator also needs to decide* $\texttt{tabP}_{top}[B^{top} \oplus (m'\|0^c)]$.

On a high level view, this procedure can be depicted using a depth-first search algorithm. Indeed, consider a graph (maintained by the simulator, different from the graph representation) where the nodes V' are

$$V' = \left\{(A^{top} \oplus (m\|0^c), A^{bot} \oplus (m\|0^c)) \mid m \in \{0,1\}^r \wedge (A^{top}, A^{bot}) \in \texttt{Rooted}\right\},$$

i.e., the rooted nodes after absorbing all possible message blocks. Then for $s \in \{top, bot\}$, we add the edge $(X_1^{top}, X_1^{bot}) \xrightarrow{s} (X_2^{top}, X_2^{bot})$ whenever $X_1^s \overset{c}{=} X_2^s$. If the query was $\texttt{FwdPQuery}\,(s, X)$, let $\textbf{NToVisit} = \{(Y^{top}, Y^{bot}) \in V' \mid Y^s = X\}$. The node collection phase then computes all the nodes accessible from **NToVisit**, and returns in the set **NodesToAns** all of the nodes found.

To summarize, for every $(X^{top}, X^{bot}) \in \textbf{NodesToAns}$, the simulator needs to determine $\texttt{tabP}_{top}[X^{top}]$ and $\texttt{tabP}_{bot}[X^{bot}]$. If the query does not impact any node (i.e., $\textbf{NodesToAns} = \emptyset$), the simulator simply returns a uniform permutation-consistent answer. In Algorithm 6, the node collection procedure is invoked in line 7, and the full procedure is described in Algorithm 7.

2. Linear System Solving. Let $(X_1^{top}, X_1^{bot}), (X_2^{top}, X_2^{bot}), \ldots, (X_m^{top}, X_m^{bot})$ be all nodes in **NodesToAns**. By construction of the latter set, whenever $m > 1$, every (X_i^{top}, X_i^{bot}) is colliding on its top or bottom part to another (X_j^{top}, X_j^{bot}) in this set. Thanks to the last message block not being equal to zero due to padding, and as long as no full-state collision occurred, for any $(X^{top}, X^{bot}) \in \textbf{NodesToAns}$, the simulator knows exactly what $\mathcal{RO}$

Algorithm 7 Simulator sub-routine to collect the nodes.

```
1:  function NodeCollection$(s, x)$
        input: $(s,x) \in \{top, bot\} \times \{0,1\}^b$
        output: Set, containing nodes of the form $(x^{top}, x^{bot})$
2:      NToVisit $\leftarrow \{(A^{top} \oplus (m\|0^c), A^{bot} \oplus (m\|0^c)) \mid (A^{top}, A^{bot}) \in$ Rooted $\wedge\, m \in \{0,1\}^r \wedge A^s \oplus (m\|0^c) = x\}$
3:      NodesToAns $\leftarrow \emptyset$
4:      while NToVisit $\neq \emptyset$ do
5:          // Pick any node $(X^{top}, X^{bot}) \in$ NToVisit
6:          Let $(X^{top}, X^{bot}) \in$ NToVisit
7:          NodesToAns $\leftarrow$ NodesToAns $\cup \{(X^{top}, X^{bot})\}$
8:          // Collect all rooted nodes which display an $u$-inner collision with
               $(X^{top}, X^{bot})$
9:          NToVisit $\leftarrow$ NToVisit $\cup \{(A^{top} \oplus (m\|0^c), A^{bot} \oplus (m\|0^c)) \mid m \in \{0,1\}^r \wedge (A^{top}, A^{bot}) \in$ Rooted $\wedge\, \exists u \in \{top, bot\}, A^u \oplus (m\|0^c) = X^u\}$
10:         NToVisit $\leftarrow$ NToVisit $\setminus$ NodesToAns
11:     return NodesToAns
```

calls are necessary to provide consistent answers, i.e., $M \in \{0,1\}^*, \alpha \in \mathbb{N}^*$ such that in the real world, the knowledge of $\mathcal{P}_{top}(X^{top})$ and $\mathcal{P}_{bot}(X^{bot})$ gives $\mathrm{DS}^{\mathcal{P}}(M)[\alpha : \alpha + r]$. Let $h_1, \ldots, h_m \in \{0,1\}^r$ be the $\mathcal{RO}$ answers corresponding to the nodes in **NodesToAns**. Then, the simulator aims at providing answers that satisfy

$$\begin{aligned}
\mathrm{outer}_r(\mathtt{tabP}_{top}[X_1^{top}] \oplus 2\mathtt{tabP}_{bot}[X_1^{bot}]) &= h_1\,,\\
\mathrm{outer}_r(\mathtt{tabP}_{top}[X_2^{top}] \oplus 2\mathtt{tabP}_{bot}[X_2^{bot}]) &= h_2\,,\\
&\vdots\\
\mathrm{outer}_r(\mathtt{tabP}_{top}[X_m^{top}] \oplus 2\mathtt{tabP}_{bot}[X_m^{bot}]) &= h_m\,.
\end{aligned}$$

The answers are $\mathcal{RO}$-consistent if and only if they satisfy the equations above. For every $i \in \{1, \ldots, m\}$, either X_i^{top} or X_i^{bot} appears in at least two equations, and therefore it is not always obvious whether the system of linear equations derived has a solution. In examples 6.3.2 and 6.3.3 below, we assume that $b > 2$, and the X_i^s denote values in $\{0,1\}^b$ where the X_i^{top}'s (resp., X_i^{bot}'s) are all distinct.

Example 6.3.2. *Assume that*

$$\textbf{NodesToAns} = \{(X_1^{top}, X_1^{bot}), (X_1^{top}, X_2^{bot}), (X_1^{top}, X_3^{bot})\}\,.$$

Then, after associating variable a_i^{top} *(resp.,* a_i^{bot}*) to* $\mathrm{outer}_r(\mathtt{tabP}_{top}[X_i^{top}])$ *(resp.,* $\mathrm{outer}_r(2\,\mathtt{tabP}_{bot}[X_i^{bot}])$*), the system of consistency equations becomes*

$$\begin{aligned} a_1^{top} \oplus a_1^{bot} &= h_1\,, \\ a_1^{top} \oplus a_2^{bot} &= h_2\,, \\ a_1^{top} \oplus a_3^{bot} &= h_3\,. \end{aligned}$$

Using a matrix notation, we obtain $\boldsymbol{Ma} = \boldsymbol{h}$*, with*

$$\boldsymbol{M} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}, \qquad \boldsymbol{a} = \begin{pmatrix} a_1^{top} \\ a_1^{bot} \\ a_2^{bot} \\ a_3^{bot} \end{pmatrix}, \qquad \boldsymbol{h} = \begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix}.$$

The kernel of $\boldsymbol{M}$ *is spanned by the vector* $\langle(\mathbf{1},\mathbf{1},\mathbf{1},\mathbf{1})\rangle$*, so* $\boldsymbol{M}$ *has rank 3. Therefore for every* $\boldsymbol{h} \in (\{0,1\}^r)^3$*, the equation has a solution (Note that the values* a_i^s *do not need to be distinct).*

6

Example 6.3.3. *Assume that*

$$\textbf{NodesToAns} = \{(X_1^{top}, X_1^{bot}), (X_1^{top}, X_2^{bot}), (X_2^{top}, X_1^{bot}), (X_2^{top}, X_2^{bot})\}\,.$$

With the same notation as in example 6.3.2, the system of consistency equations becomes

$$\begin{aligned} a_1^{top} \oplus a_1^{bot} &= h_1\,, \\ a_1^{top} \oplus a_2^{bot} &= h_2\,, \\ a_2^{top} \oplus a_1^{bot} &= h_3\,, \\ a_2^{top} \oplus a_2^{bot} &= h_4\,. \end{aligned}$$

Using a matrix notation, we obtain $\boldsymbol{Ma} = \boldsymbol{h}$*, with*

$$\boldsymbol{M} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}, \qquad \boldsymbol{a} = \begin{pmatrix} a_1^{top} \\ a_2^{top} \\ a_1^{bot} \\ a_2^{bot} \end{pmatrix}, \qquad \boldsymbol{h} = \begin{pmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{pmatrix}.$$

This square matrix has a kernel of dimension 1, it is thus non-invertible. In other words, for most vectors $\boldsymbol{h}$*, the linear system is unsolvable.*

In Section 6.3.3, we define a BAD event such that its absence always guarantees solvable linear systems with *uniform* solutions with *fresh* randomness. In this constrained setting, the simulator fixes the variables appearing in more than one equation to random values.

Example 6.3.4. *Going back to example 6.3.2, the simulator chooses* $a_1^{top} \xleftarrow{\$} \{0,1\}^r$ *and sets*

$$\begin{aligned} a_1^{bot} &= h_1 \oplus a_1^{top} , \\ a_2^{bot} &= h_2 \oplus a_1^{top} , \\ a_3^{bot} &= h_3 \oplus a_1^{top} . \end{aligned}$$

Because $\boldsymbol{h}$ *contains fresh random values and* a_1^{top} *was sampled uniformly at random, the elements* a_k^s *all follow the uniform distribution and are independent.*

The associated procedure is called in line 12 of Algorithm 6. It returns two elements $\texttt{DictOutParts}_{top}$ and $\texttt{DictOutParts}_{bot}$ which, upon success, map the elements $X_i^s \in \{0,1\}^b$ to the associated r-bit constraint of $\texttt{tabP}_s[X_i^s]$. In more detail, for top outputs, this constraint is $\text{outer}_r(\texttt{tabP}_{top}[X_i^{top}])$ while for bottom outputs this is $\text{outer}_r(2\texttt{tabP}_{bot}[X_i^{bot}])$. When the simulator fails to solve the linear system, it returns $\perp$, therefore allowing the adversary to distinguish easily.

We henceforth assume that the simulator was able to solve the linear system.

Algorithm 8 Simulator sub-routine to choose the permutation outputs.

```
1: // $\texttt{DictOutParts}_{top}$ (resp. $\texttt{DictOutParts}_{bot}$) is a dictionary where its
       elements are of form $(x \in \{0,1\}^b \to z \in \{0,1\}^r)$, meaning that the
       outer part of $\texttt{tabP}_{top}[x]$ (resp., $2\texttt{tabP}_{bot}[x]$) is fixed to $z$
2: function PermConsOutputTop($\texttt{DictOutParts}_{top}$)
3:     for all $x^{top} \to y_r^{top} \in \texttt{DictOutParts}_{top}$ do
4:         $y^{top} \xleftarrow{\$} \{y \in \{0,1\}^b \mid \text{outer}_r(y) = y_r^{top}\} \setminus \textbf{Img}(\texttt{tabP}_{top})$
5:         $\texttt{tabP}_{top}[x^{top}] = y^{top}$

6: function PermConsOutputBot($\texttt{DictOutParts}_{bot}$)
7:     for all $x^{bot} \to y_r^{bot} \in \texttt{DictOutParts}_{bot}$ do
8:         $y^{bot} \xleftarrow{\$} \{y \in \{0,1\}^b \mid \text{outer}_r(2y) = y_r^{bot}\} \setminus \textbf{Img}(\texttt{tabP}_{bot})$
9:         $\texttt{tabP}_{bot}[x^{bot}] = y^{bot}$
```

3. Choice of the Permutation Outputs. In this step, the simulator samples permutation-consistent answers among the ones satisfying the constraint found in the previous step. The procedure is invoked in lines 16 and 17 of Algorithm 6 and presented in Algorithm 8, but is thoroughly explained in this paragraph. After the simulator fixed $y_r := \text{outer}_r(\texttt{tabP}_{top}[x])$ in the previous step, it samples the answer from the set

$$\{y \in \{0,1\}^b \mid \text{outer}_r(y) = y_r\} \setminus \textbf{Img}(\texttt{tabP}_{top}) \tag{6.2}$$

and similarly for bottom answers with $y_r := \text{outer}_r(2\texttt{tabP}_{bot}[x])$ and $\text{outer}_r(2y) = y_r$. Note that at this stage, the simulator is *always* able to provide permutation-consistent answers (i.e., the set appearing in (6.2) is never empty), since the total number of queries is smaller than 2^c.

4. Extension to Several Iterations. The simulator is not done yet. It is possible that the newly created edges point towards nodes with one of their inner parts appearing in the query history. In this case, the simulator needs to complete the partial edges with $\mathcal{RO}$-consistent answers. In the worse case where *both* parts appear in the query history (modulo xoring by a message block), then the simulator has lost. The underlying procedure, called `EnsureNoPartialEdges` () is invoked in line 19 of Algorithm 6 and presented in Algorithm 9. It is similar to steps 1, 2, and 3 combined, unless that the consistency equation has some of its variables instantiated with particular values. One of the bad events defined in Section 6.3.3 ensures that the simulator does not require more than two iterations of `EnsureNoPartialEdges` () per query.

6.3.2.2 On Inverse Query

On an inverse query, the simulator provides an answer uniformly at random among the permutation-consistent ones. If this query happens to hit a rooted node, then the simulator has to run the same procedure as the one described in step 4. It is noteworthy that the probability that an inverse query hits the tree is the same as the probability that a fresh node has one of its inner part appearing in the query history.

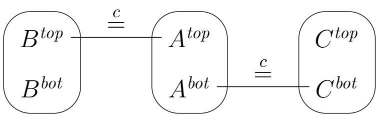
6.3.3 Bad Events and Discussion

Two main undesirable behaviors can occur when running the simulator algorithm. The first one happens when the linear system derived does not have a solution. In this case, regardless of the simulator's answer, the adversary can

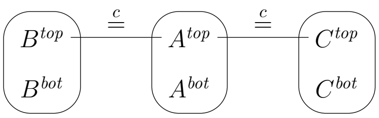
Algorithm 9 Function ensuring that there is no partial edge.

```
1:  function EnsureNoPartialEdges ()
2:      // Identify the nodes with an inner part in the query history
3:      PartialTop ← $\{A^{top} \oplus (m\|0^c) \mid m \in \{0,1\}^r \wedge \exists A^{bot} \in \{0,1\}^b : (A^{top}, A^{bot}) \in \texttt{Rooted} \wedge \texttt{tabP}_{top}[A^{top} \oplus (m\|0^c)] = \perp \wedge \texttt{tabP}_{bot}[A^{bot} \oplus (m\|0^c)] \neq \perp\}$
4:      PartialBot ← $\{A^{bot} \oplus (m\|0^c) \mid m \in \{0,1\}^r \wedge \exists A^{top} \in \{0,1\}^b : (A^{top}, A^{bot}) \in \texttt{Rooted} \wedge \texttt{tabP}_{bot}[A^{bot} \oplus (m\|0^c)] = \perp \wedge \texttt{tabP}_{top}[A^{top} \oplus (m\|0^c)] \neq \perp\}$
5:      if PartialTop = PartialBot = $\emptyset$
6:          return // Nothing to take care of
7:      // Node collection
8:      NodesToAns ← {}
9:      for all $X^{top}$ ∈ PartialTop do
10:         NodesToAns = NodesToAns ∪ NodeCollection ($top, X^{top}$)
11:     for all $X^{bot}$ ∈ PartialBot do
12:         NodesToAns = NodesToAns ∪ NodeCollection ($bot, X^{bot}$)
13:     // Note that the linear system solving takes into account the already
            defined permutation outputs
14:     $\texttt{DictOutParts}_{top}, \texttt{DictOutParts}_{bot}$ ← LinSolve (NodesToAns)
15:     if $\exists s, \texttt{DictOutParts}_s = \perp$
16:         return // Linear system solving failed
17:     // Permutation consistency, graph update
18:     PermConsOutputTop ($\texttt{DictOutParts}_{top}$)
19:     PermConsOutputBot ($\texttt{DictOutParts}_{bot}$)
20:     UpdateGraph ()
21:     EnsureNoPartialEdges () // Recursive call
```

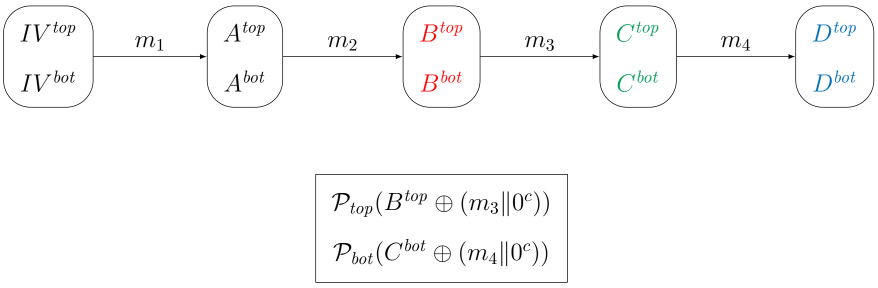
differentiate easily between W_I and W_{IM1}, since the difference between these two worlds lies in the $\mathcal{RO}$-consistency (see Section 6.3.4). The second undesirable behavior is when the simulator runs over a large number of iterations, i.e., step 4 is repeated many times. Besides the algorithm termination, this is a problem in the sense that every subsequent permutation output decided by the simulator is considered to be given *for free*, giving therefore a large factor in the upper bounds. In the following we define bad events which ensure that the simulator does not run into these problems.



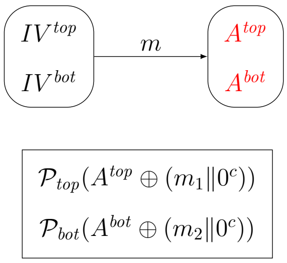
(a) DOUBLE2COL.



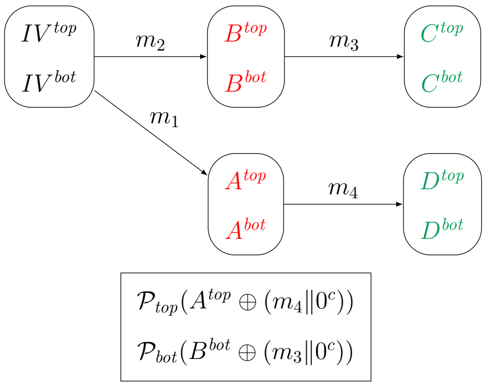
(b) 3COL.



(c) THREE_ROUNDS.


IV^{top}
IV^{bot}
m
A^{top}
A^{bot}
$\mathcal{P}_{top}(A^{top} \oplus (m_1\|0^c))$
$\mathcal{P}_{bot}(A^{bot} \oplus (m_2\|0^c))$

(d) BAD_NODE.



(e) AUX^3.

Figure 6.2: Illustration of the main BAD events. In Figures 6.2a and 6.2b, the drawn nodes are rooted. In Figures 6.2c to 6.2e, one color represents one iteration of the simulator and the rectangles contain particular permutation outputs already known by the adversary before the query.

6.3.3.1 Events on Edge Addition

We first start by defining bad events that can be triggered at any iteration, whenever an edge is added to the graph.

i) DOUBLE2COL: there exist $(A^{top}, A^{bot}), (B^{top}, B^{bot}), (C^{top}, C^{bot}) \in \texttt{Rooted}$ such that $(A^{top}, A^{bot}) \neq (B^{top}, B^{bot})$, $(A^{top}, A^{bot}) \neq (C^{top}, C^{bot})$, and

$$\begin{aligned} \text{inner}_c(A^{top}) &= \text{inner}_c(B^{top}) \\ \text{and } \text{inner}_c(A^{bot}) &= \text{inner}_c(C^{bot}) \,. \end{aligned}$$

This event is illustrated in Figure 6.2a;

ii) 3COL: there exist distinct $(A^{top}, A^{bot}), (B^{top}, B^{bot}), (C^{top}, C^{bot}) \in \texttt{Rooted}$ such that

$$\begin{aligned} \text{inner}_c(A^{top}) &= \text{inner}_c(B^{top}) = \text{inner}_c(C^{top}) \\ \text{or } \text{inner}_c(A^{bot}) &= \text{inner}_c(B^{bot}) = \text{inner}_c(C^{bot}) \,. \end{aligned}$$

This event is illustrated in Figure 6.2b;

iii) BAD_NODE: there exist $i \in \{1, \ldots, q\}, m, m_1, m_2 \in \{0,1\}^r$, (A^{top}, A^{bot}), $(B^{top}, B^{bot}) \in \texttt{Rooted}$, such that (i) before query i, $(B^{top}, B^{bot}) \notin \texttt{Rooted}$, (ii) during query i, the edge $(A^{top}, A^{bot}) \xrightarrow{m} (B^{top}, B^{bot})$ is produced, (iii) *both* $B^{top} \oplus (m_1 \| 0^c)$ and $B^{bot} \oplus (m_2 \| 0^c)$ were already in the top (resp., bottom) query history *before* the edge was produced. This event is illustrated in Figure 6.2d;

iv) BAD_NODE_S: there exist $i \in \{1, \ldots, q\}, m, m' \in \{0,1\}^r$, and (A^{top}, A^{bot}), $(B^{top}, B^{bot}) \in \texttt{Rooted}$ such that (i) before query i, $(B^{top}, B^{bot}) \notin \texttt{Rooted}$, (ii) during query i, the edge $(A^{top}, A^{bot}) \xrightarrow{m} (B^{top}, B^{bot})$ is produced, (iii) *both* $B^{top} \oplus (m' \| 0^c)$ and $B^{bot} \oplus (m' \| 0^c)$ were already in the top (resp., bottom) query history *before* the edge was produced. In short, BAD_NODE_S is set when a new edge is added from (B^{top}, B^{bot}) to the graph without the simulator being able to guarantee $\mathcal{RO}$-consistency.

6.3.3.2 Second-Iteration Event

We define a bad event THREE_ROUNDS specific to the second iteration of the algorithm, i.e., the second (resp., third) call of `EnsureNoPartialEdges` () in step 4 for a forward (resp. inverse) query. At a high level view, this event means that one single query triggers the addition of three consecutive edges $(A^{top}, A^{bot}) \xrightarrow{m} (B^{top}, B^{bot}) \xrightarrow{m'} (C^{top}, C^{bot}) \xrightarrow{m''} (D^{top}, D^{bot})$, where $(A^{top}, A^{bot}) \in$ `Rooted`. For a forward query `FwdPQuery` (s, X), it means that there exist $i \in \{1, \ldots, q\}$, $(A^{top}, A^{bot}) \in$ `Rooted`, $(B^{top}, B^{bot}), (C^{top}, C^{bot}) \in \{0,1\}^{2b}$, $m, m', m'' \in \{0,1\}^r$ such that at query i, the following sequence of events happens:

1. $X = A^s \oplus (m\|0^c)$, so that the first iteration of `NodeCollection` () gives $(A^{top} \oplus (m\|0^c), A^{bot} \oplus (m\|0^c)) \in$ **NodesToAns**;
2. The edge $(A^{top}, A^{bot}) \xrightarrow{m} (B^{top}, B^{bot})$ is added, making (B^{top}, B^{bot}) a rooted node;
3. $B^{top} \oplus (m'\|0^c)$ or $B^{bot} \oplus (m'\|0^c)$ was already in the top (resp., bottom) query history, so that another iteration of `NodeCollection` () is necessary;
4. The edge $(B^{top}, B^{bot}) \xrightarrow{m'} (C^{top}, C^{bot})$ is added, making (C^{top}, C^{bot}) a rooted node;
5. $C^{top} \oplus (m''\|0^c)$ or $C^{bot} \oplus (m''\|0^c)$ appears in the top (resp., bottom) query history, so that another iteration of `NodeCollection` () is needed.

6

For an inverse query `InvPQuery` (s, Y), additionally to the sequence of events above, the response of the inverse query must beforehand hit one of the inner parts of (A^{top}, A^{bot}). THREE_ROUNDS is illustrated in Figure 6.2c.

6.3.3.3 Auxiliary Events

Finally, we introduce the following events, which allow to simultaneously eliminate rare but undesirable cases and simplify the proof.

i) AUX^1: There exist $i \in \{1, \ldots, q\}$, and distinct $(A^{top}, A^{bot}), (B^{top}, B^{bot}) \in \{0,1\}^{2b}$ such that the query i adds (A^{top}, A^{bot}) and (B^{top}, B^{bot}) to the set `Rooted` and such that $A^s \overset{c}{=} B^s$ for some $s \in \{top, bot\}$;

ii) AUX^2: There exist $s \in \{top, bot\}$, $i \in \{1, \ldots, q\}$, $X \in \{0,1\}^b$, and distinct (A^{top}, A^{bot}), $(B^{top}, B^{bot}) \in$ `Rooted` such that query i is of the form `InvPQuery` (s, X) and the answer Y is such that $Y \overset{c}{=} A^s$ and (A^{top}, A^{bot}) displays an inner collision with (B^{top}, B^{bot}) on their s part for $s \in \{top, bot\}$;

iii) AUX^3: There exists $i \in \{1, \ldots, q\}$ such that query i requires a second iteration of the simulator during which the set **NodesToAns** in Algorithm 9 has a size greater than 1. This event is illustrated in Figure 6.2e.

6.3.3.4 Interpretation

Intuitively, $\mathsf{DOUBLE2COL}$ and $\mathsf{3COL}$ ensure that the set **NodesToAns** is of size at most 2 and that the linear system always has a solution for first iteration nodes. For the subsequent iterations, we additionally need the event $\mathsf{BAD\_NODE\_S}$. However, $\mathsf{BAD\_NODE}$ is implied by $\mathsf{BAD\_NODE\_S}$, and the former event is useful for the other parts of the proof, thus we will only make use of $\mathsf{BAD\_NODE}$. Note that $\mathsf{DOUBLE2COL}$ and $\mathsf{3COL}$ are suboptimal for the linear system solvability, e.g., in example 6.3.2, the linear system is solvable while $\mathsf{3COL}$ is set. The real bad event here would be to have a cycle in the graph defined in step 1, Section 6.3.2. Nevertheless, $\mathsf{DOUBLE2COL}$ and $\mathsf{3COL}$ are very useful for the probability computation of $\mathsf{THREE\_ROUNDS}$, which allows to upper bound the size of the tree. Note that when assuming $\neg\mathsf{AUX}_i^3$, $\mathsf{THREE\_ROUNDS}_i$ is the only scenario where the simulator requires more than two iterations of `NodeCollection`() during query i. Finally, AUX^1, AUX^2, and AUX^3 are bad events that alleviate the proof complexity and eliminate bad cases. In the following, let

$$\mathsf{BAD} := \mathsf{DOUBLE2COL} \vee \mathsf{3COL} \vee \mathsf{BAD\_NODE} \vee \mathsf{THREE\_ROUNDS} \vee \bigvee_j \mathsf{AUX}^j\,.$$

If **evt** denotes any of those aforementioned events, for any $i \in \{1, \ldots, q\}$, $\mathbf{evt}_i$ denotes that **evt** is triggered after i queries.

6.3.4 World Decomposition

Remember that the ideal world W_I consists of $(\mathcal{RO}, \mathbf{S}^{\mathrm{fwd}}, \mathbf{S}^{\mathrm{inv}})$, and the real world W_R consists of $(\mathrm{DS}^{\mathcal{P}}, \mathcal{P}, \mathcal{P}^{-1})$. Our indifferentiability proof uses two intermediate worlds as shown in Figure 6.3. The first one, called W_{IM1}, gives access to $(\mathrm{DS}^{\mathbf{S}}, \mathbf{S}^{\mathrm{fwd}}, \mathbf{S}^{\mathrm{inv}})$, where $\mathbf{S}$ relies on a random oracle. It is a natural intermediate step between the real and the ideal world, since it allows to separate the $\mathcal{RO}$-consistency (c.f., Definition 2.4.4) from the simulator's quality of randomness. Note that since the BAD event is defined on the simulator $\mathbf{S}$, the former therefore also applies in W_{IM1}. However, while the distance from W_I to W_{IM1} is now easier to analyze, this is not the case for the distance between W_{IM1} and W_R. Indeed, in W_R, the answers are returned using lazy sampling, so that the notion of "iterations" does not exist in this world, thus preventing the usage of the bad event $\mathsf{THREE\_ROUNDS}$. For that reason, we introduce another intermediate world W_{IM2}, which implements the real world with a supplementary layer mimicking the simulator's node collection phase and step 4 of Section 6.3.2. This layer is called `GraphProc` and

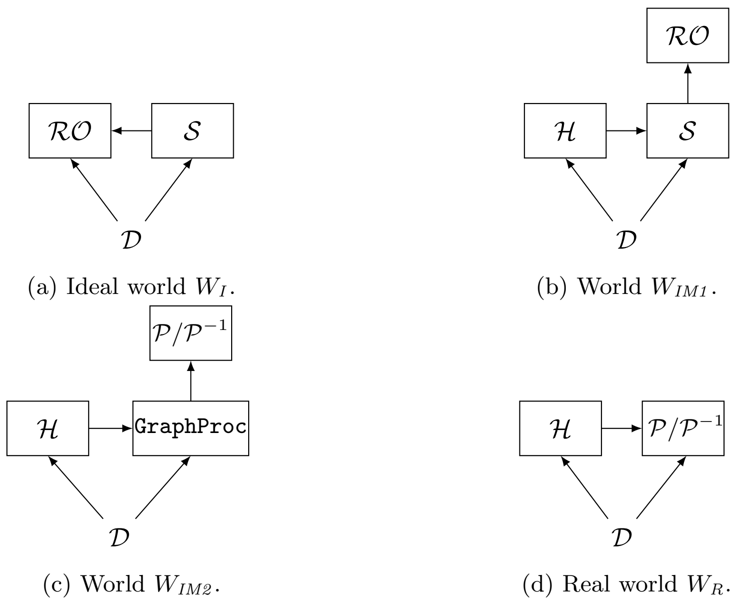


Figure 6.3: Definition of the different worlds. $\mathcal{P}$ returns responses with lazy sampling.

is described as follows: on a forward query or inverse query, an answer is drawn using lazy sampling. Then, the procedure `EnsureNoPartialEdges` () is run, with `PermConsOutputTop` () and `PermConsOutputBot` () replaced by a permutation-consistent lazy sampling.

For $X \in \{I, IM1, IM2, R\}$, let $\mathcal{D}^{W_X}$ denote the fact that distinguisher $\mathcal{D}$ is in world W_X. Recall that our goal is to bound $\mathbf{Adv}^{\text{indif}}_{\text{DS},\mathbf{S}}(\mathcal{D})$. By the triangle inequality:

$$\begin{aligned}\mathbf{Adv}^{\text{indif}}_{\text{DS},\mathbf{S}}(\mathcal{D}) &= \left|\mathbf{Pr}\left(\mathcal{D}^{W_I}=1\right) - \mathbf{Pr}\left(\mathcal{D}^{W_R}=1\right)\right| \\ &\leq \left|\mathbf{Pr}\left(\mathcal{D}^{W_I}=1\right) - \mathbf{Pr}\left(\mathcal{D}^{W_{IM2}}=1\right)\right| \qquad (6.3a)\\ &+ \left|\mathbf{Pr}\left(\mathcal{D}^{W_{IM2}}=1\right) - \mathbf{Pr}\left(\mathcal{D}^{W_R}=1\right)\right| . \qquad (6.3b)\end{aligned}$$

For the distance in (6.3a):

$$\begin{aligned}(6.3a) = \Big|&\mathbf{Pr}\left(\mathcal{D}^{W_I}=1\wedge \mathsf{BAD}\right) + \mathbf{Pr}\left(\mathcal{D}^{W_I}=1\wedge \neg\mathsf{BAD}\right) \\ &- \mathbf{Pr}\left(\mathcal{D}^{W_{IM1}}=1\wedge \neg\mathsf{BAD}\right) + \mathbf{Pr}\left(\mathcal{D}^{W_{IM1}}=1\wedge \neg\mathsf{BAD}\right)\end{aligned}$$

$$\begin{aligned}
&- \mathbf{Pr}\left(\mathcal{D}^{W_{IM2}} = 1 \wedge \mathsf{BAD}\right) - \mathbf{Pr}\left(\mathcal{D}^{W_{IM2}} = 1 \wedge \neg\mathsf{BAD}\right)\Big| \\
\leq\ & \Big|\mathbf{Pr}\left(\mathcal{D}^{W_I} = 1 \wedge \mathsf{BAD}\right) - \mathbf{Pr}\left(\mathcal{D}^{W_{IM2}} = 1 \wedge \mathsf{BAD}\right)\Big| \\
+\ & \Big|\mathbf{Pr}\left(\mathcal{D}^{W_I} = 1 \wedge \neg\mathsf{BAD}\right) - \mathbf{Pr}\left(\mathcal{D}^{W_{IM1}} = 1 \wedge \neg\mathsf{BAD}\right)\Big| \\
+\ & \Big|\mathbf{Pr}\left(\mathcal{D}^{W_{IM1}} = 1 \wedge \neg\mathsf{BAD}\right) - \mathbf{Pr}\left(\mathcal{D}^{W_{IM2}} = 1 \wedge \neg\mathsf{BAD}\right)\Big| \\
\leq\ & \Big|\mathbf{Pr}\left(\mathcal{D}^{W_I} \text{ sets } \mathsf{BAD}\right)\mathbf{Pr}\left(\mathcal{D}^{W_I} = 1 \mid \mathsf{BAD}\right) - \\
& \qquad \mathbf{Pr}\left(\mathcal{D}^{W_{IM2}} \text{ sets } \mathsf{BAD}\right)\mathbf{Pr}\left(\mathcal{D}^{W_{IM2}} = 1 \mid \mathsf{BAD}\right)\Big| && \text{(6.4a)} \\
+\ & \Big|\mathbf{Pr}\left(\mathcal{D}^{W_I} = 1 \mid \neg\mathsf{BAD}\right) - \mathbf{Pr}\left(\mathcal{D}^{W_{IM1}} = 1 \mid \neg\mathsf{BAD}\right)\Big| && \text{(6.4b)} \\
+\ & \Big|\mathbf{Pr}\left(\mathcal{D}^{W_{IM1}} = 1 \mid \neg\mathsf{BAD}\right) - \mathbf{Pr}\left(\mathcal{D}^{W_{IM2}} = 1 \mid \neg\mathsf{BAD}\right)\Big|. && \text{(6.4c)}
\end{aligned}$$

The distance in (6.4a) can eventually be bounded as follows:

$$\text{(6.4a)} \leq \max\left\{\mathbf{Pr}\left(\mathcal{D}^{W_I} \text{ sets } \mathsf{BAD}\right), \mathbf{Pr}\left(\mathcal{D}^{W_{IM2}} \text{ sets } \mathsf{BAD}\right)\right\}. \qquad (6.5)$$

6

In conclusion, we find that $\mathbf{Adv}^{\text{indif}}_{\text{DS},\mathbf{S}}(\mathcal{D})$ of (6.3) is bounded by the sum of the terms in (6.5), (6.4b), (6.4c), and (6.3b):

$$\begin{aligned}
\mathbf{Adv}^{\text{indif}}_{\text{DS},\mathbf{S}}(\mathcal{D}) \leq\ & \max\left\{\mathbf{Pr}\left(\mathcal{D}^{W_I} \text{ sets } \mathsf{BAD}\right), \mathbf{Pr}\left(\mathcal{D}^{W_{IM2}} \text{ sets } \mathsf{BAD}\right)\right\} && \text{(6.6a)} \\
+\ & \Big|\mathbf{Pr}\left(\mathcal{D}^{W_I} = 1 \mid \neg\mathsf{BAD}\right) - \mathbf{Pr}\left(\mathcal{D}^{W_{IM1}} = 1 \mid \neg\mathsf{BAD}\right)\Big| && \text{(6.6b)} \\
+\ & \Big|\mathbf{Pr}\left(\mathcal{D}^{W_{IM1}} = 1 \mid \neg\mathsf{BAD}\right) - \mathbf{Pr}\left(\mathcal{D}^{W_{IM2}} = 1 \mid \neg\mathsf{BAD}\right)\Big| && \text{(6.6c)} \\
+\ & \Big|\mathbf{Pr}\left(\mathcal{D}^{W_{IM2}} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{W_R} = 1\right)\Big|. && \text{(6.6d)}
\end{aligned}$$

We will discuss these two probabilities and three distances below. In the following, let $q := q_C + q_P$.

Bad Events. We upper bound the BAD event probabilities in Lemma 6.4.2 (Section 6.4.1) and obtain the upper bound

$$(6.6a) \leq \frac{249q^3}{(2^c - 3q)^2} + \frac{21q}{2^c - 3q}.$$

Ideal World W_I Versus W_{IM1} as Long as no BAD. Here, the adversary has access to $(\mathcal{RO}, \mathbf{S}^{\text{fwd}}, \mathbf{S}^{\text{inv}})$ or $(\text{DS}^{\mathbf{S}}, \mathbf{S}^{\text{fwd}}, \mathbf{S}^{\text{inv}})$, where in W_{IM1}, $\mathbf{S}^{\text{fwd}}$

and $\mathbf{S}^{\text{inv}}$ are built upon a random oracle. Therefore, as long as the simulator is $\mathcal{RO}$-consistent with respect to the double sponge, the two worlds are indistinguishable. The $\mathcal{RO}$-consistency is guaranteed by the success of the linear system solving phase of the simulator. More formally, in Lemma 6.4.3 (Section 6.4.2), we prove that as long as $\neg\mathsf{BAD}$ holds, this phase never fails. The proof relies on a simple enumeration of all possible cases regarding the structure of the set **NodesToAns**. Therefore,

$$(6.6b) = 0\,.$$

W_{IM1} **Versus** W_{IM2} **as Long as no** BAD**.** Here, the adversary has access to $(\mathrm{DS}^{\mathbf{S}}, \mathbf{S}^{\text{fwd}}, \mathbf{S}^{\text{inv}})$ or $(\mathrm{DS}^{\mathcal{P}}, \mathcal{P}, \mathcal{P}^{-1})$. Now, since the construction component is the same in both worlds, the distinguisher can convert the construction queries into primitive queries. The game thus reduces to a distinguishing game between $(\mathbf{S}^{\text{fwd}}, \mathbf{S}^{\text{inv}})$ and $(\mathcal{P}, \mathcal{P}^{-1})$. The full proof is in Lemma 6.4.4 (Section 6.4.3), and it gives an upper bound

$$(6.6c) \leq \frac{3q^{\frac{3}{2}}}{2^c - 3q}\,.$$

W_{IM2} **Versus Real World** W_R**.** Similarly to the real world, world W_{IM2} implements a double sponge using a permutation, the only difference lies in the timing at which the permutation samplings are performed. A permutation with answers being lazily sampled is equivalent to a permutation drawn at the beginning of the game. These two worlds are thus equivalent, and hence

$$(6.6d) = 0\,.$$

From (6.6), we eventually obtain

$$\begin{aligned}\mathbf{Adv}^{\text{indif}}_{\mathrm{DS},\mathbf{S}}(q_P, q_C) &\leq \frac{249q^3}{(2^c - 3q)^2} + \frac{21q}{2^c - 3q} + \frac{3q^{\frac{3}{2}}}{2^c - 3q}\\ &\leq \frac{19q^{\frac{3}{2}}}{2^c - 3q} + \frac{21q}{2^c - 3q}\\ &\leq \frac{40q^{\frac{3}{2}}}{2^c - 3q}\,,\end{aligned}$$

which concludes Theorem 6.3.1.

6.3.5 Extension to Single Permutation

One natural question arises on whether it is possible to extend the double sponge construction by replacing $\mathcal{P}_{top}$ and $\mathcal{P}_{bot}$ by the same permutation, but with different initialization vectors IV^{top} and IV^{bot}. From a proof perspective, such an extension would significantly complicate the analysis, and particularly increase drastically the number of cases to be considered. On the other hand, using a single permutation is desirable in practice. Thus, as a trade-off, we propose the following modification to the double sponge construction: we replace the permutations $\mathcal{P}_{top}$ and $\mathcal{P}_{bot}$ by respectively $\mathcal{P}(0\|\cdot)[2:b]$ and $\mathcal{P}(1\|\cdot)[2:b]$. In that setting, the impact on the simulator definition, bad event analysis, and statistical distance computation is relatively mild.

The presence of domain separator bits allows the simulator to determine whether it needs to consider the top or bottom part of the nodes during a forward query. As a result, the procedure `NodeCollection` () does not change, apart from a parsing adjustment. The procedure `LinSolve` () does not need any changes. The procedures `PermConsOutputTop` () and `PermConsOutputBot` () can be merged into a single procedure. Next, if (X^{top}, X^{bot}) is a newly produced node (after absorption of a message block), then the updated version of `EnsureNoPartialEdges` () runs a new iteration only when either $0\|X^{top}$ or $1\|X^{bot}$ appears in the query history. Note that having $1\|X^{top}$ or $0\|X^{bot}$ in the query history does not require one other iteration, since if X^{top} (resp., X^{bot}) appears later as a bottom (resp., top) node, it will be generated from a fresh source of randomness. Finally, the algorithm run by the simulator on an inverse query does not change.

Regarding the upper bounding of distances, only the bad event analysis (cf., (6.6a)) and the quality of randomness of the simulator (cf., (6.6c)) are affected. In the bad event analysis, the nodes are now linear combinations of truncated permutation outputs instead of permutation outputs, but this modification does not affect the upper bounds derived in the analysis.[3] On the other hand, we need an additional bad event which states that one query to the simulator results in the addition of two rooted nodes (A^{top}, A^{bot}) and (B^{top}, B^{bot}) such that $A^{top} \overset{c}{=} B^{bot}$. This is a rare event, and the auxiliary bad events ensure that the simulator generates no more than three nodes in a single query. Therefore, this event occurs with a probability of form $\tilde{\mathcal{O}}\left(\frac{q}{2^c}\right)$. Thus, modulo the loss of one bit and small constant factors, the bad event upper bounding does not change significantly. Finally, the permutation's randomness is exhausted twice as fast, resulting in further constant factor losses.

[3]To be precise, the bounding in equation (6.9) still holds.

6.4 Probability Computation

In this section, we establish upper bounds for the three terms from equations (6.6*a*), (6.6*b*), and (6.6*c*) in Sections 6.4.1, 6.4.2, and 6.4.3 respectively.

6.4.1 BAD Event Probability Analysis

Before stating an upper bound on the BAD probabilities in Lemma 6.4.2, we define the trimmed tree in Definition 6.4.1, and state a result on the size of the latter. This result ensures the algorithm termination and allows to upper bound the number of queries given for free by the simulator.

Definition 6.4.1. *The trimmed tree is a subset of* `Rooted` *defined in an inductive way:*

- *Initialization:* `Trimmed` $\leftarrow \{(IV^{top}, IV^{bot})\}$*;*
- *On query i, whenever $(A^{top}, A^{bot}) \xrightarrow{m} (B^{top}, B^{bot})$ is added to the graph with $(A^{top}, A^{bot}) \in$* `Trimmed`*; if (B^{top}, B^{bot}) does not trigger* BAD*, then* `Trimmed` $\leftarrow$ `Trimmed` $\cup \{(B^{top}, B^{bot})\}$*.*

In other words, the trimmed tree is the set of rooted nodes deprived of the ones (and their descendants) that would trigger BAD *if added.*

Lemma 6.4.1. *For any $i \in \{0, \dots, q\}$, the size of the trimmed tree after i queries is upper bounded by $3i + 1$.*

Proof. First observe that the nodes returned by `NodeCollection`() must display a collision on their top or bottom part with another node in the same set. Therefore, having more than two nodes returned implies that there is either a 3-collision or a double collision among the rooted nodes, which correspond to respectively 3COL and DOUBLE2COL. Therefore, as long as no BAD happens, `NodeCollection`() returns at most two nodes each time.

Now, the trimmed tree is initialized with a single node (IV^{top}, IV^{bot}), thus has size of 1. Then, at query i, if the procedure `NodeCollection`() is run using `Trimmed` instead of `Rooted`, we know from the above observation that |**NodesToAns**| is at most 2. In addition, because of THREE_ROUNDS, the nodes produced after more than two iterations are not added to the trimmed tree. Finally, AUX^3 guarantees that at most one node has two consecutive edges added. We conclude that in total, at most 3 nodes are added to the trimmed tree per query. □

Lemma 6.4.2. *For any distinguisher $\mathcal{D}$ making q_P primitive queries and construction queries which would correspond to a total of q_C primitive queries in world W_R, if $30 < 2^c$, one has*

$$\mathbf{Pr}\left(\mathcal{D}^{W_I} \textit{ sets } \mathsf{BAD}\right) \leq \frac{249q_P^3}{(2^c - 3q_P)^2} + \frac{21q_P}{2^c - 3q_P}\,,$$
$$\mathbf{Pr}\left(\mathcal{D}^{W_{IM2}} \textit{ sets } \mathsf{BAD}\right) \leq \frac{249q^3}{(2^c - 3q)^2} + \frac{21q}{2^c - 3q}\,,$$

where $q := q_C + q_P$.

Proof. For an event $\mathbf{E}$, we denote by $\mathbf{Pr}_I(\mathbf{E})$ (resp., $\mathbf{Pr}_{IM2}(\mathbf{E})$) the probability that $\mathbf{E}$ occurs in world W_I (resp., W_{IM2}). Let q be the total number of primitive queries made, so that $q = q_P$ in world W_I, and $q \leq q_C + q_P$ in world W_{IM2}. Let

$$\begin{aligned} S_{\mathsf{2COL}} :=& \big\{(A^{top}, A^{bot}) \in \texttt{Trimmed} \mid \exists (B^{top}, B^{bot}) \in \texttt{Trimmed} \setminus \{(A^{top}, A^{bot})\}, \\ & \quad s \in \{top, bot\} \text{ s.t., } A^s \overset{c}{=} B^s\big\}\,, \\ N_{\mathsf{2COL}} :=& |S_{\mathsf{2COL}}|\,. \end{aligned}$$

In other words, N_{2COL} counts the number of nodes displaying an inner collision on their top or bottom part in the trimmed tree. Note that, in particular, N_{2COL} cannot be larger than $3q + 1$.

Remark 6.4.1. *As long as BAD is not set, S_{2COL} can be split as the disjoint union of $S_{\mathsf{2COL}}(top)$ and $S_{\mathsf{2COL}}(bot)$ defined as follows for $s \in \{top, bot\}$.*

$$\begin{aligned} S_{\mathsf{2COL}}(s) = \big\{&(A^{top}, A^{bot}) \in \texttt{Trimmed} \mid \\ & \exists (B^{top}, B^{bot}) \in \texttt{Trimmed} \setminus \{(A^{top}, A^{bot})\} \textit{ s.t., } A^s \overset{c}{=} B^s\big\}\,. \end{aligned}$$

As long as BAD does not occur, the size of S_{2COL} is the sum of the two set sizes.

Now, by basic probability theory, for $X \in \{I, IM2\}$,

$$\mathbf{Pr}_X(\mathsf{BAD}) \leq \sum_{m=0}^{3q+1} \mathbf{Pr}_X(\mathsf{BAD} \mid N_{\mathsf{2COL}} = m) \cdot \mathbf{Pr}_X(N_{\mathsf{2COL}} = m) \tag{6.7}$$

$$\leq \sum_{m=0}^{3q+1} \sum_{i=1}^{q} \mathbf{Pr}_X(\mathsf{BAD}_i \mid N_{\mathsf{2COL}} = m \wedge \neg\mathsf{BAD}_{i-1}) \cdot \mathbf{Pr}_X(N_{\mathsf{2COL}} = m)\,, \tag{6.8}$$

where we remind that BAD_i denotes that BAD is set after i queries.

We start with $\mathbf{Pr}_X\left(\mathsf{BAD}_i \mid N_{\mathsf{2COL}} = m \wedge \neg\mathsf{BAD}_{i-1}\right)$ for $X \in \{I, \mathit{IM2}\}$ and any $i \in \{1, \ldots, q\}$ and $m \in \{0, \ldots, 3q+1\}$. An important observation is that, because of $\neg\mathsf{BAD}_{i-1}$, the trimmed tree recoverable from the first $i-1$ queries and the actual tree coincide. Therefore, from $N_{\mathsf{2COL}} = m$ and $\neg\mathsf{3COL}_{i-1} \wedge \neg\mathsf{DOUBLE2COL}_{i-1}$, we know that the tree has exactly $m/2$ colliding pairs on their inner top or bottom part.

Remark 6.4.2. *In world W_{IM2}, the responses are drawn uniformly at random from a set of size at least $2^b - 3i$, while in the ideal world W_I, the response Y is generated as follows: (i) either Y is sampled from a set of size at least 2^b-3i; or (ii) r bits of Y (or $2Y$) are derived from a random oracle call and the remaining c bits are chosen at random to remain permutation-consistent. Therefore, for any forward query with answer Y, $y \in \{0,1\}^b$, and $X \in \{I, \mathit{IM2}\}$,*

$$\mathbf{Pr}_X\left(Y = y\right) \leq \frac{1}{2^r(2^c - 3q)}, \qquad \mathbf{Pr}_X\left(2Y = y\right) \leq \frac{1}{2^r(2^c - 3q)}. \tag{6.9}$$

Looking ahead, the probability computation will give the same upper bound in both worlds.

Since BAD groups several sub-events, we split accordingly the probability computation, and when it is meaningful each sub-event is studied as many times as the number of iterations of the simulator that we consider. For instance, remember that $\mathsf{DOUBLE2COL}_i$ is an event that can occur when an edge is added. It is split into $(\mathsf{DOUBLE2COL}_i[k])_{k\in\mathbb{N}\setminus\{0\}}$, where $\mathsf{DOUBLE2COL}_i[k]$ means that the k^{th} iteration of the simulator during the query i sets the event $\mathsf{DOUBLE2COL}$. Note that the order in which we consider the bad events matters. In more detail, if we compute the probability of event $\mathbf{E}$, then $\mathbf{F}$, and then $\mathbf{G}$, in the last computation $\neg\mathbf{E} \wedge \neg\mathbf{F}$ is implicitly assumed.

The remainder of the proof is organized as follows. Sections 6.4.1.1 to 6.4.1.3 are dedicated to forward queries events, where each section concerns one iteration of the simulator. Section 6.4.1.4 looks at inverse queries events. Finally, Section 6.4.1.5 gathers the found upper bounds and plugs them into (6.8).

6.4.1.1 First Iteration Events, Forward Query

The first iteration on a forward query is the easiest case, since the newly created edges provide the maximal possible randomness. Because of $\neg\mathsf{BAD}_{i-1}$, and from the remark at the beginning of the proof of Lemma 6.4.1, the node collection phase returns at most two nodes (after absorption). In the following, we focus on the hardest case where $|\mathbf{NodesToAns}| = 2$, so that $\mathbf{NodesToAns} :=$

$\{(X_1^{top}, X_1^{bot}), (X_2^{top}, X_2^{bot})\}$, where $X_1^s = X_2^s$ for $s \in \{top, bot\}$. Thus, three permutation outputs need to be fixed. For $k \in \{1, 2\}$ and $s \in \{top, bot\}$, let Y_k^s be the random variable equal to $\mathtt{tabP}_s[X_k^s]$. After the first iteration, the graph has two more rooted nodes that we denote by $(A^{top}, A^{bot}), (B^{top}, B^{bot})$. Now we can look at the probability that (A^{top}, A^{bot}) and/or (B^{top}, B^{bot}) triggers BAD using the condition $N_{\mathsf{2COL}} = m$.

1. AUX^1**.** This event implies that

$$\begin{aligned} Y_1^{top} + 2Y_1^{bot} &\overset{c}{=} Y_2^{top} + 2Y_2^{bot} \\ \text{or}\ \ 2Y_1^{top} + Y_1^{bot} &\overset{c}{=} 2Y_2^{top} + Y_2^{bot}\,, \end{aligned}$$

where we know that $Y_1^{top} = Y_2^{top}$ or $Y_1^{bot} = Y_2^{bot}$, but the other elements are distinct. By Remark 6.4.2, in both worlds, this event happens with probability at most $\frac{2}{2^c - 3q}$.

Now, $\neg\mathsf{AUX}_i^1$ allows to study separately the probability that the node (A^{top}, A^{bot}) (resp., (B^{top}, B^{bot})) triggers a first iteration event. Since the probabilities are the same, in the following we focus only on (A^{top}, A^{bot}).

2. AUX^2**.** The event AUX^2 only concerns inverse queries, thus it does not apply here.

3. AUX^3**.** The event AUX^3 only concerns the second iteration, thus it does not apply here.

4. $\mathsf{3COL}$**.** This event means that there exist distinct $(N_1^{top}, N_1^{bot}), (N_2^{top}, N_2^{bot}) \in \mathtt{Rooted}$ such that

$$\begin{aligned} A^{top} \overset{c}{=} N_1^{top} \overset{c}{=} N_2^{top} \\ \text{or}\ \ A^{bot} \overset{c}{=} N_1^{bot} \overset{c}{=} N_2^{bot}\,. \end{aligned}$$

Because of the condition $N_{\mathsf{2COL}} = m$ and since $\neg\mathsf{BAD}_{i-1}$ holds, there are $m/2$ tuples $\{(N_1^{top}, N_1^{bot}), (N_2^{top}, N_2^{bot})\}$. From Remark 6.4.1, we know that the latter are disjointly split into $S_{\mathsf{2COL}}(bot)$ and $S_{\mathsf{2COL}}(top)$. Therefore the probability is upper bounded by $\frac{m}{2(2^c - 3q)}$.

5. $\mathsf{DOUBLE2COL}$**.** There are two possibilities:

- The inserted node *displays* a double collision. In other words, there exist (N_1^{top}, N_1^{bot}), $(N_2^{top}, N_2^{bot}) \in \texttt{Rooted}$ (not necessarily distinct) such that

$$\begin{aligned} A^{top} &\overset{c}{=} N_1^{top} \\ \text{and } A^{bot} &\overset{c}{=} N_2^{bot} . \end{aligned}$$

Let

$$I_1^{top} = \text{inner}_c(N_1^{top}) , \qquad I_2^{bot} = \text{inner}_c(N_2^{bot}) .$$

Then, this means that there must exist $O_1, O_2 \in \{0,1\}^r$ such that

$$(Y_1^{top}, Y_1^{bot}) = MIX^{-1}(O_1 \| I_1^{top}, O_2 \| I_2^{bot}) . \tag{6.10}$$

Now, because of $\neg\mathsf{BAD}_{i-1}$, there are at most $(3(i-1)+1)^2 \le (3i)^2$ tuples $(N_1^{top}, N_1^{bot}), (N_2^{top}, N_2^{bot}) \in \texttt{Rooted}$, and for each tuple, (6.10) is satisfied with probability at most $\left(\frac{1}{2^c - 3q}\right)^2$. Therefore, this event happens with probability at most $\left(\frac{3i}{2^c-3q}\right)^2$.

- The inserted node is *part of* a double collision: in other words there exist $(N_1^{top}, N_1^{bot}), (N_2^{top}, N_2^{bot}) \in \texttt{Rooted}$ such that

$$\begin{cases} N_1^{top} \overset{c}{=} A^{top} , \\ N_1^{bot} \overset{c}{=} N_2^{bot} , \end{cases} \quad \text{or} \quad \begin{cases} N_1^{top} \overset{c}{=} N_2^{top} , \\ N_1^{bot} \overset{c}{=} A^{bot} . \end{cases}$$

Similarly to $\mathsf{3COL}$, this gives a probability of $\frac{m}{2^c-3q}$.

6. $\mathsf{BAD\_NODE}$. This event means that there exist $x^{top} \in \mathbf{Dom}(\texttt{tabP}_{top})$ and $x^{bot} \in \mathbf{Dom}(\texttt{tabP}_{bot})$ such that

$$\begin{aligned} A^{top} &\overset{c}{=} x^{top} \\ \text{and } A^{bot} &\overset{c}{=} x^{bot} . \end{aligned}$$

For every fixed tuple (x^{top}, x^{bot}), this happens with probability at most $\left(\frac{1}{2^c-3q}\right)^2$, and there are at most $9i^2$ such tuples. Therefore we obtain the upper bound $\left(\frac{3i}{2^c-3q}\right)^2$.

Now we are done with the first iteration events. We can then compute the same probabilities for second-iteration events (if a second iteration is required) and compute the probability that a third iteration is needed.

6.4.1.2 Second-Iteration Events, Forward Query

Thanks to $\neg\mathsf{BAD\_NODE}$ for the first iteration nodes, neither (A^{top}, A^{bot}) nor (B^{top}, B^{bot}) have the inner part of *both* their top and bottom part in the query history. Nevertheless, it is possible that only their top or bottom part displays an inner collision with the query history, and this is not a bad event. Since the size of the top (resp., bottom) query history is upper bounded by $3i$, and four different cases can happen, i.e., with node (A^{top}, A^{bot}) or (B^{top}, B^{bot}), on top or bottom part, this event happens with probability at most $\frac{12i}{2^c-3q}$. In the following, we first get rid of the case where the simulator or `GraphProc` has to take care of more than one node.

1. AUX^3. This event can be achieved in two different ways:

- *Both* (A^{top}, A^{bot}) and (B^{top}, B^{bot}) have their upper or lower part appearing in the query history. The probability analysis is similar to the one made at the first paragraph of Section 6.4.1.2, and we obtain an upper bound of $\left(\frac{6i}{2^c-3q}\right)^2$;
- (A^{top}, A^{bot}) (or (B^{top}, B^{bot})) has its upper or lower part appearing in the query history *and* displays an inner collision with another rooted node. For $s \in \{top, bot\}$, let

$$\xi^s = \left\{j \in \{1, \ldots, q\} \mid (A_j^{top}, A_j^{bot}) \in S_{\mathsf{2COL}}(s) \vee (B_j^{top}, B_j^{bot}) \in S_{\mathsf{2COL}}(s)\right\}\ , \tag{6.11}$$

  where we abuse notation for (A_j^{top}, A_j^{bot}) and (B_j^{top}, B_j^{bot}) to refer to the nodes produced during the first iteration of query j. Then, as long as BAD does not occur, ξ^{top} and ξ^{bot} are disjoint and the size of $\xi^{top} \cup \xi^{bot}$ is at most m. Now, this case of AUX^3 happens with a non-zero probability if and only if $i \in \xi^s$ for $s \in \{top, bot\}$, in which case the probability can be upper bounded by $\frac{6i}{2^c-3q}$.

We henceforth assume $\neg\mathsf{AUX}_i^3$, so that the procedure `NodeCollection()` returns only one node (X^{top}, X^{bot}). In the ideal world, the simulator has no problem to solve the consistency equation, and inserts to the graph one more node (C^{top}, C^{bot}). Now, (C^{top}, C^{bot}) does not contain as much randomness as the first iteration nodes, since one of the permutation calls is already fixed. However, we are going to make use of the fact that second iteration edges are *rarely* produced. W.l.o.g., we can assume that X^{top} appeared in the query history, and let $y^{top} := \mathtt{tabP}_{top}[X^{top}]$, and $Y^{bot} := \mathtt{tabP}_{bot}[X^{bot}]$; in particular, the only fresh random value is Y^{bot}. In the following we compute the probability that this node triggers BAD using the condition $N_{\mathsf{2COL}} = m$.

2. AUX^1**.** This event implies that (C^{top}, C^{bot}) collides with (A^{top}, A^{bot}) or (B^{top}, B^{bot}), thus

$$y^{top} + 2Y^{bot} \overset{c}{=} Y_k^{top} + 2Y_k^{bot}$$
$$\text{or } 2y^{top} + Y^{bot} \overset{c}{=} 2Y_k^{top} + Y_k^{bot}$$

for some $k \in \{1,2\}$, where we remind that Y_k^s is the random variable equal to $\mathtt{tabP}_s[X_k^s]$, fixed during the first node collection. This event happens with probability at most $\frac{4}{2^c-3q}$.

3. AUX^2**.** The event AUX^2 only concerns inverse queries, thus it does not apply here.

4. $\mathsf{3COL}$**.** This event means that there exist $(N_1^{top}, N_1^{bot}), (N_2^{top}, N_2^{bot}) \in$ `Rooted` distinct such that

$$C^{top} \overset{c}{=} N_1^{top} \overset{c}{=} N_2^{top}$$
$$\text{or } C^{bot} \overset{c}{=} N_1^{bot} \overset{c}{=} N_2^{bot}\,,$$

or in more detail,

$$y^{top} + 2Y^{bot} \overset{c}{=} N_1^{bot} \overset{c}{=} N_2^{bot}$$
$$\text{or } 2y^{top} + Y^{bot} \overset{c}{=} N_1^{top} \overset{c}{=} N_2^{top}\,.$$

Similarly to the first iteration nodes, this event happens with probability at most $\frac{m}{2(2^c-3q)}$.

5. $\mathsf{DOUBLE2COL}$**.** Again, there are two possibilities:

- (C^{top}, C^{bot}) *displays* a double collision: there exist (N_1^{top}, N_1^{bot}), (N_2^{top}, N_2^{bot}) $\in$ `Rooted` (not necessarily distinct) such that

$$C^{top} \overset{c}{=} N_1^{top}$$
$$\text{and } C^{bot} \overset{c}{=} N_2^{bot}\,.$$

Let $I_1^{top} = \text{inner}_c(N_1^{top})$, $I_2^{bot} = \text{inner}_c(N_2^{bot})$. This means that there must exist $O_1, O_2 \in \{0,1\}^r$ such that

$$(y^{top}, Y^{bot}) = MIX^{-1}(O_1 \| I_1^{top}, O_2 \| I_2^{bot})\,. \tag{6.12}$$

Now, the upper b bits of the equation do not provide any randomness. Nevertheless, DOUBLE2COL with a second iteration node implies that the following two sub-events happened:

- **E**$_1$: There exist (N_1^{top}, N_1^{bot}), $(N_2^{top}, N_2^{bot}) \in$ `Rooted`, $O_1, O_2 \in \{0,1\}^r$ such that the node (A^{top}, A^{bot}) or (B^{top}, B^{bot}) has its top part already in the query history and the associated primitive output y^{top} satisfies (6.12). This happens with a probability of at most $\frac{2^r \cdot 2(3i)^2}{2^c - 3q}$;
- **E**$_2$: Y^{bot} satisfies (6.12), for given (N_1^{top}, N_1^{bot}), $(N_2^{top}, N_2^{bot}) \in$ `Rooted`. This happens with probability at most $\frac{1}{2^r(2^c-3q)}$.

We need to consider the symmetric case where the bottom part of the node (A^{top}, A^{bot}) or (B^{top}, B^{bot}) was already in the query history. Therefore, this case of DOUBLE2COL happens with probability at most $\left(\frac{6i}{2^c-3q}\right)^2$.

- (C^{top}, C^{bot}) is *part* of a double collision, i.e., there exist (N_1^{top}, N_1^{bot}), $(N_2^{top}, N_2^{bot}) \in$ `Rooted` (not necessarily distinct) such that

$$\begin{cases} N_1^{top} \overset{c}{=} C^{top}\,, \\ N_1^{bot} \overset{c}{=} N_2^{bot}\,, \end{cases} \quad \text{or} \quad \begin{cases} N_1^{top} \overset{c}{=} N_2^{top}\,, \\ N_1^{bot} \overset{c}{=} C^{bot}\,. \end{cases}$$

Similarly to the first iteration nodes, this gives a probability of at most $\frac{m}{2^c-3q}$.

6. BAD_NODE. Since BAD_NODE$_i$ on the node (C^{top}, C^{bot}) implies THREE_ROUNDS$_i$, we can skip this probability computation.

6.4.1.3 Third Iteration Events, Forward Query

Now, to guarantee $\neg$THREE_ROUNDS$_i$, we need that (C^{top}, C^{bot}) (if it exists) does not have any of its inner parts appearing in the query history. Again, this event implies the following two sub-events:

- An inner part of a first iteration node appears in the query history: by the first paragraph of Section 6.4.1.2, this event happens with probability at most $\frac{12i}{2^c-3q}$;
- An inner part of a second iteration node appears in the query history: since $\neg\mathsf{AUX}_i^3$ guarantees that at most one node was produced during the second iteration, this event happens with probability at most $\frac{6i}{2^c-3q}$.

Both of these events are independent, therefore $\mathsf{THREE\_ROUNDS}_i$ happens with probability at most $2\left(\frac{6i}{2^c-3q}\right)^2$.

6.4.1.4 The Case of Inverse Queries

This case is similar to the second-iteration events, since inverse queries *rarely* hit the tree, i.e., they hit one of the inner parts of the tree with probability at most $\frac{6i}{2^c-3q}$. Moreover, the event AUX^2 is set with probability at most $\frac{m}{2^c-3q}$. Assuming then $\neg\mathsf{AUX}^2$, **NodesToAns** contains only one node and the behavior of the simulator and `GraphProc` is the same as if a second iteration node was produced with one of its top part in the query history, and the probability computation is the same. Consequently,

$$\begin{aligned}\mathbf{Pr}\left(\mathsf{BAD}_i \mid N_{\mathsf{2COL}} = m \wedge \neg\mathsf{BAD}_{i-1} \wedge \neg\mathsf{AUX}^2 \wedge \text{query } i \text{ is inverse}\right) \leq \\ \mathbf{Pr}\left(\mathsf{BAD}_i \mid N_{\mathsf{2COL}} = m \wedge \neg\mathsf{BAD}_{i-1} \wedge \text{query } i \text{ is forward}\right).\end{aligned}$$

6.4.1.5 Conclusion

From Sections 6.4.1.1 to 6.4.1.4, we know that

$$\begin{aligned}&\mathbf{Pr}_X\left(\mathsf{BAD}_i \mid N_{\mathsf{2COL}} = m \wedge \neg\mathsf{BAD}_{i-1}\right) \\ &\leq \begin{cases} 180\left(\frac{i}{2^c-3q}\right)^2 + \frac{6}{2^c-3q} + \frac{9m}{2(2^c-3q)} & \text{if } i \notin \xi^{top} \cup \xi^{bot}, \\ 180\left(\frac{i}{2^c-3q}\right)^2 + \frac{6}{2^c-3q} + \frac{9m}{2(2^c-3q)} + \frac{6q}{2^c-3q} & \text{otherwise}, \end{cases}\end{aligned} \tag{6.13}$$

where we remind that ξ^s is the number of queries such that there exists a node produced during the first iteration in $S_{\mathsf{2COL}}(s)$ (see (6.11)). The conditions under which we study the corresponding BAD events guarantee that ξ^{top} and ξ^{bot} are disjoint, and $|\xi^{top}| + |\xi^{bot}| = m$. Therefore, this gives

$$\mathbf{Pr}_X\left(\mathsf{BAD} \mid N_{\mathsf{2COL}} = m\right) \leq \sum_{i=1}^{q}\left(180\left(\frac{i}{2^c-3q}\right)^2 + \frac{6}{2^c-3q} + \frac{9m}{2(2^c-3q)}\right) +$$

$$\sum_{i\in\xi^{top}\cup\xi^{bot}} \frac{6q}{2^c-3q}$$
$$\leq \frac{60q^3}{(2^c-3q)^2}+\frac{90q^2}{(2^c-3q)^2}+\frac{30q}{(2^c-3q)^2}+\frac{6q}{2^c-3q}+ \frac{21mq}{2(2^c-3q)}\,.$$

Now, plugging this equation into (6.7) gives

$$\begin{aligned}\mathbf{Pr}_X(\mathsf{BAD}) &\leq \frac{60q^3}{(2^c-3q)^2}+\frac{90q^2}{(2^c-3q)^2}+\frac{30q}{(2^c-3q)^2}+\frac{6q}{2^c-3q}+ \\ &\quad \sum_{m=0}^{3q+1}\mathbf{Pr}_X(N_{\mathsf{2COL}}=m)\frac{21mq}{2(2^c-3q)} \\ &\leq \frac{60q^3}{(2^c-3q)^2}+\frac{90q^2}{(2^c-3q)^2}+\frac{30q}{(2^c-3q)^2}+\frac{6q}{2^c-3q}+ \\ &\quad \frac{21q}{2(2^c-3q)}\mathsf{E}(N_{\mathsf{2COL}})\,. \end{aligned} \tag{6.14}$$

It now remains to compute $\mathsf{E}(N_{\mathsf{2COL}})$. For $i\in\{1,\ldots,|\texttt{Trimmed}|\}$, denote by (N_i^{top},N_i^{bot}) the i^{th} element in $\texttt{Trimmed}$. Moreover, we define the Bernoulli variable V_i equal to 1 if and only if $(N_i^{top},N_i^{bot})\in S_{\mathsf{2COL}}$. Then,

$$\begin{aligned}\mathsf{E}(N_{\mathsf{2COL}}) &= \mathsf{E}\left(\sum_{i=1}^{|\texttt{Trimmed}|}V_i\right)=\sum_{i=1}^{|\texttt{Trimmed}|}\mathsf{E}(V_i) \\ &= \sum_{i=1}^{|\texttt{Trimmed}|} 2\mathbf{Pr}\left(\exists j<i, s\in\{top,bot\}\text{ such that } N_i^s \overset{c}{=} N_j^s\right) \\ &\leq 2\sum_{i=1}^{|\texttt{Trimmed}|}\frac{2(i-1)}{2^c-3q} \\ &= \frac{4}{2^c-3q}\frac{|\texttt{Trimmed}|\,(|\texttt{Trimmed}|-1)}{2} \\ &\leq \frac{18q^2}{2^c-3q}+\frac{6q}{2^c-3q}\,, \end{aligned}\tag{6.15}$$

where the last inequality uses $|\texttt{Trimmed}| \leq 3q+1$. Finally, plugging (6.15) into (6.14), we obtain

$$\begin{aligned}\mathbf{Pr}_X(\mathsf{BAD}) &\leq \frac{60q^3}{(2^c-3q)^2} + \frac{90q^2}{(2^c-3q)^2} + \frac{30q}{(2^c-3q)^2} + \frac{6q}{2^c-3q} + \\ &\frac{21q}{2(2^c-3q)}\left(\frac{18q^2}{2^c-3q} + \frac{6q}{2^c-3q}\right) \\ &\leq \frac{249q^3}{(2^c-3q)^2} + \frac{153q^2}{(2^c-3q)^2} + \frac{30q}{(2^c-3q)^2} + \frac{6q}{2^c-3q} \\ &\leq \frac{249q^3}{(2^c-3q)^2} + \frac{30q}{(2^c-3q)^2} + \frac{19q}{2^c-3q}\,.\end{aligned}$$

If we assume that $30 < 2^c$, then we obtain the upper bound

$$\mathbf{Pr}_X(\mathsf{BAD}) \leq \frac{249q^3}{(2^c-3q)^2} + \frac{21q}{2^c-3q}\,.$$

When $X = I$, $q = q_P$, while when $X = \mathit{IM2}$, $q \leq q_C + q_P$, hence the result. □

6

6.4.2 World W_I Versus World W_{IM1} as Long as no BAD

Lemma 6.4.3 argues that as long as no BAD happens, the ideal and intermediate worlds are indistinguishable.

Lemma 6.4.3. *For any distinguisher $\mathcal{D}$,*

$$\left|\mathbf{Pr}\left(\mathcal{D}^{W_I} = 1 \mid \neg\mathsf{BAD}\right) - \mathbf{Pr}\left(\mathcal{D}^{W_{IM1}} = 1 \mid \neg\mathsf{BAD}\right)\right| = 0\,.$$

Proof. The only difference between the worlds W_I and W_{IM1} lies in the consistency of the responses. In more detail, as long as the simulator is $\mathcal{RO}$-consistent with respect to DS (c.f., Definition 2.4.4), then W_I and W_{IM1} are perfectly indistinguishable. We stress that for $s \in \{top, bot\}$, an s inverse query hitting the tree does not necessarily break the consistency, since the simulator uses the $\bar{s}$ part to guarantee consistency. Therefore, the $\mathcal{RO}$-consistency boils down to ensuring that the procedure `LinSolve` () never returns $\perp$, which we argue in the following.

From the remark at the beginning of the proof of Lemma 6.4.1, we know that as long as no BAD happens, $|\mathbf{NodesToAns}| \leq 2$ for any iteration of the simulator. Moreover, any node in **NodesToAns** has at least b bits (i.e., its top or bottom part) undetermined. When $|\mathbf{NodesToAns}| = 1$, if BAD_NODE is not set, the linear system comprises only one equation with at least one unknown, thus the simulator can always output consistent answers. Therefore, we henceforth consider the case where $|\mathbf{NodesToAns}| = 2$. W.l.o.g., assume that $\mathbf{NodesToAns} = \{(X_1^{top}, X_1^{bot}), (X_2^{top}, X_2^{bot})\}$ where $X_1^{top} = X_2^{top}$. Now, the form of the underlying linear system depends on the direction of the query and the iteration run by the simulator. In the following we study each case.

Forward Query, First Iteration. The consistency equation is of the form

$$\begin{aligned} a_1^{top} \oplus a_1^{bot} &= h_1 \,, \\ a_1^{top} \oplus a_2^{bot} &= h_2 \,, \end{aligned} \tag{6.16}$$

where a_1^{top} is the variable corresponding to

$$\mathrm{outer}_r(\mathtt{tabP}_{top}[X_1^{top}]) = \mathrm{outer}_r(\mathtt{tabP}_{top}[X_2^{top}]) \,,$$

a_1^{bot} to $\mathrm{outer}_r(\mathtt{tabP}_{bot}[2X_1^{bot}])$, and a_2^{bot} to $\mathrm{outer}_r(\mathtt{tabP}_{bot}[2X_2^{bot}])$. This equation has a solution, since the simulator can choose $a_1^{top} \overset{\$}{\leftarrow} \{0,1\}^r$, and infer a_1^{bot}, a_2^{bot}.

6

Inverse Query, First Iteration. In this case, since BAD (and in particular AUX^2) does not happen, $|\mathbf{NodesToAns}| \leq 1$.

Second Iteration. Because of $\neg\mathsf{AUX}^3$, $|\mathbf{NodesToAns}| \leq 1$.

Therefore, the simulator can always produce consistent answers. □

6.4.3 World W_{IM1} Versus World W_{IM2} as Long as no BAD

In this section we upper bound the distinguisher's advantage between W_{IM1} and W_{IM2} without BAD.

Lemma 6.4.4. *For any distinguisher $\mathcal{D}$ making q_P primitive queries and construction queries which would correspond to a total of q_C primitive queries in world W_R, one has*

$$\left|\mathbf{Pr}\left(\mathcal{D}^{W_{IM1}} = 1 \mid \neg\mathsf{BAD}\right) - \mathbf{Pr}\left(\mathcal{D}^{W_{IM2}} = 1 \mid \neg\mathsf{BAD}\right)\right| \leq \frac{3q^{\frac{3}{2}}}{2^c - 3q} \,,$$

where $q := q_C + q_P$.

The proof is deferred to Section 6.4.3.3. First, we introduce a distinguishing game useful for the proof in Section 6.4.3.1, and upper bound the statistical distance between the two underlying distributions in Section 6.4.3.2.

6.4.3.1 A Simpler Distinguishing Game

We first introduce a simple distinguishing game whose statistical distance computation is crucial for the proof of Lemma 6.4.4. In the following, let $\phi : \{0,1\}^b \to \{0,1\}^b$ be a bijection.

World W_1. In this world, the elements in $\{0,1\}^b$ are partitioned into 2^r bins $(B_k)_{k=0,\dots,2^r-1}$ in the following way:

$$\forall x \in \{0,1\}^b,\, x \in B_k \iff \mathrm{outer}_r(\phi(x)) = k\,.$$

Each bucket contains 2^c elements. Then, on a forward query, an element $k \in \{0,1\}^r$ is drawn uniformly at random, and the answer is the result of a sampling without replacement in bucket B_k. On an inverse query with $y \in \{0,1\}^b$, a permutation-consistent element with lazy sampling is returned, and y is withdrawn from bucket $B_{\mathrm{outer}_r(y)}$.

World W_2. This world implements a random permutation over $\{0,1\}^b$, where the elements are returned using lazy sampling.

Intuitively, W_2 corresponds to the permutations $\mathcal{P}_{top}$ and $\mathcal{P}_{bot}$, and we will argue later that W_1 is an abstraction of the simulator way of sampling the responses. For $x \in \{1,2\}$, $\mathbf{Pr}_x\left(\mathbf{evt}\right)$ denotes the probability that $\mathbf{evt}$ is set in the world W_x.

6.4.3.2 Upper Bounding the Statistical Distance Using the χ^2 Method

Lemma 6.4.5 upper bounds the statistical distance between W_1 and W_2. The proof uses the χ^2 method by Dai et al. [DHT17] that we describe in the following. Let $\Omega := \{0,1\}^b$ be a sample space, and consider two probability distributions P_1 and P_2 with values in Ω^q. Let $\boldsymbol{x} := (x_1,\dots,x_q)$ be a random vector following P_1, and for any $i \in \{1,\dots,q\}$, let $\boldsymbol{x}^i := (x_1,\dots,x_i)$. The χ^2 divergence is defined as follows:

$$\chi^2(\boldsymbol{x}^{i-1}) := \sum_{\substack{x\in\Omega,\\ \mathbf{Pr}_2\left(x_i=x \mid \boldsymbol{x}^{i-1}\right)\neq 0}} \frac{\left(\mathbf{Pr}_1\left(x_i = x \mid \boldsymbol{x}^{i-1}\right) - \mathbf{Pr}_2\left(x_i = x \mid \boldsymbol{x}^{i-1}\right)\right)^2}{\mathbf{Pr}_2\left(x_i = x \mid \boldsymbol{x}^{i-1}\right)}\,.$$

Moreover, assume that the support of P_1 is included in the support of P_2. Then the χ^2 method upper bounds the statistical distance between P_1 and P_2

as follows:

$$\mathbf{Adv}^{\text{ind}}\left(P_1, P_2\right)(q) \leq \left(\frac{1}{2}\sum_{i=1}^{q} \mathsf{E}\left(\chi^2\left(\boldsymbol{x}^{i-1}\right)\right)\right)^{\frac{1}{2}}, \tag{6.17}$$

where we abuse notation for $\mathbf{Adv}^{\text{ind}}\left(P_1, P_2\right)(q)$.

Lemma 6.4.5. *For any* $q \leq 2^c$,

$$\mathbf{Adv}^{\text{ind}}\left(W_1, W_2\right)(q) \leq \frac{1}{2}\frac{q^{\frac{3}{2}}}{2^c - q}.$$

Proof. In the following, if $\boldsymbol{x}$ is a vector of length $k > 1$ with values in $\{0,1\}^b$, let

$$\mathcal{P}\mathrm{Cons}(\boldsymbol{x}) := \{y \in \{0,1\}^b \mid \forall i \in \{1,\dots,k\}, y \neq \boldsymbol{x}_i\}.$$

Because the inverse queries in W_1 and W_2 trigger the same sampling procedure (thus a zero term in the chi-squared divergence), we can w.l.o.g., focus on forward queries. Denote by P_1 (resp., P_2) the associated distribution in W_1 (resp., W_2). In both cases, let `tabP` be the dictionary that logs the query history, i.e., if $\texttt{tabP}[x] = y$, then a forward query with x gives y, and vice versa for inverse queries. For $i \in \{1,\dots,q\}$, let $\boldsymbol{x}^{i-1}$ be a random vector of $i-1$ elements sampled according to P_1. Let $x \in \mathcal{P}\mathrm{Cons}(\boldsymbol{x}^{i-1})$, we then define $S(x)$ as follows:

$$S(x) = \left|\{y \in \mathbf{Img}(\texttt{tabP}) \mid \mathrm{outer}_r(y) = \mathrm{outer}_r(\phi(x))\}\right|.$$

i.e., the number of elements that have already been withdrawn from bucket $B_{\mathrm{outer}_r(\phi(x))}$. Now, in order to obtain x in W_1, the adversary must first sample $\mathrm{outer}_r(\phi(x))$, and then draw the appropriate element in a bucket of size $2^c - S(x)$. Thus

$$\mathbf{Pr}_1\left(x_i = x \mid \boldsymbol{x}^{i-1}\right) = \frac{1}{2^r}\frac{1}{2^c - S(x)},$$
$$\mathbf{Pr}_2\left(x_i = x \mid \boldsymbol{x}^{i-1}\right) = \frac{1}{2^b - (i-1)}.$$

Therefore, using that $0 \leq S(x) \leq i-1 \leq q \leq 2^c$,

$$\begin{aligned}
&\left|\frac{\mathbf{Pr}_1\left(x_i = x \mid \boldsymbol{x}^{i-1}\right) - \mathbf{Pr}_2\left(x_i = x \mid \boldsymbol{x}^{i-1}\right)}{\mathbf{Pr}_2\left(x_i = x \mid \boldsymbol{x}^{i-1}\right)}\right| \\
&= \left|\frac{2^b - (i-1) - 2^b + 2^r S(x)}{2^b - 2^r S(x)}\right| \\
&= \left|\frac{2^r S(x) - (i-1)}{2^b - 2^r S(x)}\right| \\
&\leq \frac{i-1}{2^c - q} .
\end{aligned}$$

Using the obtained equation, we can compute the χ^2 divergence:

$$\begin{aligned}
\chi^2(\boldsymbol{x}^{i-1}) &= \sum_{x \in \mathcal{P}\mathrm{Cons}(\boldsymbol{x}^{i-1})} \mathbf{Pr}_2\left(x_i = x \mid \boldsymbol{x}^{i-1}\right) \left(\frac{\mathbf{Pr}_1\left(x_i = x \mid \boldsymbol{x}^{i-1}\right) - \mathbf{Pr}_2\left(x_i = x \mid \boldsymbol{x}^{i-1}\right)}{\mathbf{Pr}_2\left(x_i = x \mid \boldsymbol{x}^{i-1}\right)}\right)^2 \\
&\leq \sum_{x \in \mathcal{P}\mathrm{Cons}(\boldsymbol{x}^{i-1})} \mathbf{Pr}_2\left(x_i = x \mid \boldsymbol{x}^{i-1}\right) \left(\frac{i-1}{2^c - q}\right)^2 \\
&= \left(\frac{i-1}{2^c - q}\right)^2 .
\end{aligned}$$

Now, we can apply the χ^2 technique to obtain

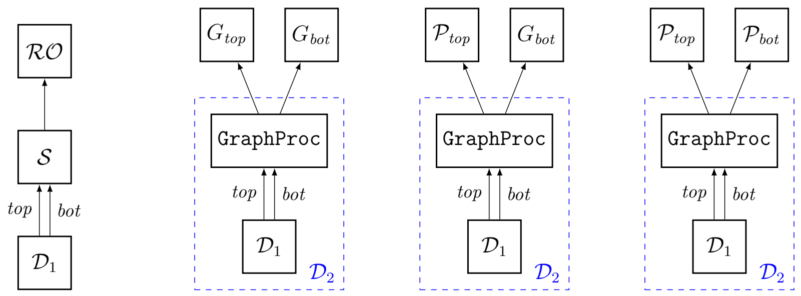
$$\begin{aligned}
\mathbf{Adv}^{\mathrm{ind}}\left(W_1, W_2\right)(q) &\leq \left(\frac{1}{2}\sum_{i=1}^{q} \mathsf{E}\left(\chi^2\left(\boldsymbol{x}^{i-1}\right)\right)\right)^{\frac{1}{2}} \\
&\leq \left(\frac{1}{2}\sum_{i=1}^{q} \left(\frac{i-1}{2^c - q}\right)^2\right)^{\frac{1}{2}} \\
&\leq \frac{1}{2}\frac{q^{\frac{3}{2}}}{2^c - q} .
\end{aligned}$$

□

Remark 6.4.3. *This distinguishing problem is similar to the one of Bhattacharya and Nandi's work [BN18, Theorem 2]. They have an upper bound of the form*

$$\tilde{\mathcal{O}}\left(\frac{q}{2^{\frac{b+c}{2}}}\right) .$$

However, in their setting the adversary only makes forward queries. In our case, allowing inverse queries introduces a supplementary bias in the distribution that can be upper bounded by $\frac{q^{\frac{3}{2}}}{2^c}$.



(a) W_a (b) W_b (c) W_c (d) W_d

Figure 6.4: World decomposition for the proof of Lemma 6.4.4. W_a and W_d correspond respectively to W_{IM1} and W_{IM2} (without the construction oracle). $\mathcal{D}_2$ is a distinguisher built from $\mathcal{D}_1$ when distinguishing W_b, W_c, and W_d.

6.4.3.3 Proof of Lemma 6.4.4

Using Lemma 6.4.5, we can now prove Lemma 6.4.4. Consider $\mathcal{D}$ which aims at distinguishing W_{IM1} and W_{IM2} without BAD. We decompose the q_P primitive queries into q_P^{top} top queries and q_P^{bot} bottom queries, so that $q_P = q_P^{top} + q_P^{bot}$. From $\mathcal{D}$, we build another distinguisher $\mathcal{D}_1$ which converts all the construction queries into the appropriate primitive queries, so that $\mathcal{D}_1$ makes at most $q_C/2 + q_P^{top}$ (resp., $q_C/2 + q_P^{bot}$) top (resp., bottom) primitive queries. Since the construction component is the same in both worlds, doing this provides at least the same amount of information to the adversary. Define W_a as W_{IM1} without the component DS, and similarly, let W_d comprise W_{IM2} without the component DS. Then, we have

$$\left|\mathbf{Pr}\left(\mathcal{D}^{W_{IM1}} = 1 \mid \neg\mathsf{BAD}\right) - \mathbf{Pr}\left(\mathcal{D}^{W_{IM2}} = 1 \mid \neg\mathsf{BAD}\right)\right| \leq \left|\mathbf{Pr}\left(\mathcal{D}_1^{W_a} = 1 \mid \neg\mathsf{BAD}\right) - \mathbf{Pr}\left(\mathcal{D}_1^{W_d} = 1 \mid \neg\mathsf{BAD}\right)\right| .$$

Now, we remark that the simulator embeds two distinct primitives, each one simulating the top or bottom permutation. Given the current simulator definition, it is not clear whether these two primitives are independent. Indeed, a top (resp., bottom) query can trigger a bottom (resp., top) output being defined, and the outer parts are chosen according to an equation involving top and bottom outer parts. To address these points, we introduce two intermediate worlds W_b and W_c, as illustrated in Figure 6.4.

World W_b. W_b differs from W_a in the way the dictionaries $\texttt{DictOutParts}_{top}$ and $\texttt{DictOutParts}_{bot}$ are generated. Namely, the function `LinSolve` () is replaced by a uniform sampling of the outer parts. Given this description, it is now clear that the two primitives have independent sampling procedures. In Figure 6.4b, they are represented by the components G_{top} and G_{bot} which take as parameter a bit $\beta \in \{0,1\}$, and for every fresh query, G_s operates as follows:

- If $\beta = 0$ or the query is in the inverse direction, it returns a permutation-consistent answer uniformly at random;
- Otherwise it samples $h \xleftarrow{\$} \{0,1\}^r$, and returns a permutation-consistent answer y uniformly at random with the restriction that

$$h = \begin{cases} \text{outer}_r(y) & \text{if } s = top\,, \\ \text{outer}_r(2y) & \text{if } s = bot\,. \end{cases}$$

The component `GraphProc` invokes the procedures G_{top} and G_{bot} with $\beta = 0$ whenever the node collection phase returns an empty set (corresponding to Section 6.3.2.1 of Algorithm 6). Looking ahead, G_{top} and G_{bot} with β always equal to 1 correspond to W_1 defined in Section 6.4.3.1.

Now, recall that in W_a the outer parts returned by `LinSolve` () come from $\mathcal{RO}$ calls combined with uniform sampling. In particular, because no BAD event occurs, these values are all freshly random, so that W_b is just a rearrangement of W_a. Therefore,

$$\left|\mathbf{Pr}\left(\mathcal{D}_1^{W_a} = 1 \mid \neg\mathsf{BAD}\right) - \mathbf{Pr}\left(\mathcal{D}_1^{W_b} = 1 \mid \neg\mathsf{BAD}\right)\right| = 0\,.$$

New Distinguisher. From $\mathcal{D}_1$ we build yet another distinguisher $\mathcal{D}_2$ which obtains for free all the subsequent queries defined by the simulator or the primitives G_{top} and G_{bot}. In other words, $\mathcal{D}_2$ runs the component `GraphProc` on its own (see Figure 6.4). Since no BAD happens, $\mathcal{D}_2$ makes at most $3(q_C/2 + q_P^{top})$ (resp., $3(q_C/2 + q_P^{bot})$) top (resp., bottom) primitive queries, and clearly,

$$\left|\mathbf{Pr}\left(\mathcal{D}_1^{W_b} = 1 \mid \neg\mathsf{BAD}\right) - \mathbf{Pr}\left(\mathcal{D}_1^{W_d} = 1 \mid \neg\mathsf{BAD}\right)\right| \leq \\ \left|\mathbf{Pr}\left(\mathcal{D}_2^{W_b} = 1 \mid \neg\mathsf{BAD}\right) - \mathbf{Pr}\left(\mathcal{D}_2^{W_d} = 1 \mid \neg\mathsf{BAD}\right)\right|\,.$$

World W_c. In this world, the left sampler G_{top} is replaced by an ideal permutation. Since G_{top} and G_{bot} are independent and $\mathcal{D}_2$ runs `GraphProc` by itself, we obtain

$$\left|\mathbf{Pr}\left(\mathcal{D}_2^{W_b} = 1 \mid \neg\mathsf{BAD}\right) - \mathbf{Pr}\left(\mathcal{D}_2^{W_c} = 1 \mid \neg\mathsf{BAD}\right)\right| \leq \\ \mathbf{Adv}^{\text{ind}}\left(G_{top}, \mathcal{P}_{top}\right)\left(3(q_C/2 + q_P^{top})\right).$$

Likewise, in W_d, the right sampler G_{bot} is replaced by an ideal permutation, and we obtain

$$\left|\mathbf{Pr}\left(\mathcal{D}_2^{W_c} = 1 \mid \neg\mathsf{BAD}\right) - \mathbf{Pr}\left(\mathcal{D}_2^{W_d} = 1 \mid \neg\mathsf{BAD}\right)\right| \leq \\ \mathbf{Adv}^{\text{ind}}\left(G_{bot}, \mathcal{P}_{bot}\right)\left(3(q_C/2 + q_P^{bot})\right).$$

Conclusion. Using the triangle inequality and the results above,

$$\left|\mathbf{Pr}\left(\mathcal{D}^{W_{IM1}} = 1 \mid \neg\mathsf{BAD}\right) - \mathbf{Pr}\left(\mathcal{D}^{W_{IM2}} = 1 \mid \neg\mathsf{BAD}\right)\right| \leq \\ \mathbf{Adv}^{\text{ind}}\left(G_{top}, \mathcal{P}_{top}\right)\left(3(q_C/2 + q_P^{top})\right) + \mathbf{Adv}^{\text{ind}}\left(G_{bot}, \mathcal{P}_{bot}\right)\left(3(q_C/2 + q_P^{bot})\right).$$

We remark that the chi-squared divergence of $G_{top}/\mathcal{P}_{top}$ (resp. $G_{bot}/\mathcal{P}_{bot}$) is always smaller than the one of W_1/W_2 from Section 6.4.3.1. Indeed, in the distinguishing game $G_s/\mathcal{P}_s$, when $\beta = 1$, the conditional probabilities are the same ones as in the game W_1/W_2, while when $\beta = 0$, the conditional probabilities in G_{top} and $\mathcal{P}_{top}$ are equal, giving a zero term in the chi-squared divergence. Therefore, we can use the result of Lemma 6.4.5, and obtain

$$\begin{aligned}\mathbf{Adv}^{\text{ind}}\left(G_{top}, \mathcal{P}_{top}\right)\left(3(q_C/2 + q_P^{top})\right) &\leq \frac{1}{2}\frac{(3(q_C/2 + q_P^{top}))^{\frac{3}{2}}}{2^c - (3(q_C/2 + q_P))},\\ \mathbf{Adv}^{\text{ind}}\left(G_{bot}, \mathcal{P}_{bot}\right)\left(3(q_C/2 + q_P^{bot})\right) &\leq \frac{1}{2}\frac{(3(q_C/2 + q_P^{bot}))^{\frac{3}{2}}}{2^c - (3(q_C/2 + q_P))},\end{aligned}$$

Finally, defining $q := q_C + q_P$, this gives

$$\left|\mathbf{Pr}\left(\mathcal{D}^{W_{IM1}} = 1 \mid \neg\mathsf{BAD}\right) - \mathbf{Pr}\left(\mathcal{D}^{W_{IM2}} = 1 \mid \neg\mathsf{BAD}\right)\right| \leq \frac{3q^{\frac{3}{2}}}{2^c - 3q},$$

which concludes the proof.

6.5 Tightness of the Bound

In this section, we argue about the tightness of the bound of Theorem 6.3.1 by providing two attacks. The first one, described in Section 6.5.1, has an advantage of $\tilde{\mathcal{O}}\left(\frac{q^2}{2^{2c+r}}\right)$. This attack is not specific to our simulator, since it can be used to mount collision and second preimages (which can then be used to differentiate). The second attack, as explained in Section 6.5.2, has an advantage of $\Omega\left(\frac{q^3}{2^{2c+r}}\right)$ and only works against our simulator of Section 6.3.2.

6.5.1 Simple Birthday Attack

The birthday attack is similar to the best-known differentiability attack of the sponge construction [BDPV07]. It consists of finding a full-state collision in the nodes, i.e., find (A^{top}, A^{bot}), $(B^{top}, B^{bot}) \in \texttt{Rooted}$, $m \in \{0,1\}^r$ such that

$$(A^{top}, A^{bot}) = (B^{top} \oplus (m\|0^c), B^{bot} \oplus (m\|0^c))\,. \tag{6.18}$$

To do so, the distinguisher performs q top and bottom queries to expand rooted paths, so that it obtains a set of rooted nodes $\{(N_i^{top}, N_i^{bot})\}_{i=1,\ldots,q}$, and searches for (A^{top}, A^{bot}), (B^{top}, B^{bot}) in this list that satisfy (6.18). The probability to find such nodes is then $\approx \frac{q^2}{2^{2c+r}}$. Once this is done, the distinguisher adds one edge from (A^{top}, A^{bot}) with a zero message. Consequently, an edge from (B^{top}, B^{bot}) with the message m is also added. The distinguisher can then retrieve the unique messages and indexes M_A, k_A and M_B, k_B associated to (A^{top}, A^{bot}) and $(B^{top} \oplus (m\|0^c), B^{bot} \oplus (m\|0^c))$. Finally, it queries $\texttt{ConsQuery}\,(M_A, k_A)$ and $\texttt{ConsQuery}\,(M_B, k_B)$ and checks whether the two responses are equal. If this is the case, then the distinguisher returns W_R, otherwise it returns W_I. Indeed, consistency is always satisfied in the real world, while in the ideal world, with high probability the two $\mathcal{RO}$ calls are distinct. This attack succeeds with high probability when $q = 2^{c+r/2}$. Note that this attack works against any simulator, and therefore one cannot get better than $c + r/2$ bits of security.

6.5.2 A Simulator-Specific Attack

We show a differentiability attack with an advantage of $\approx \left(1 - \frac{1}{2^r}\right)\frac{q^3}{2^{2c+r}}$. Consider the following distinguisher $\mathcal{D}$:

1. $\mathcal{D}$ first starts by making q forward top and bottom primitive queries and obtains for $i \in \{1, \ldots, q\}$ the queries $X_i^{top} \to Y_i^{top}$ and $X_i^{bot} \to Y_i^{bot}$;

2. Then it makes q top and bottom primitive queries to expand rooted paths. The top queries are of the form $(A_j^{top} + (m||0^c))$, and the bottom queries of form $(A_j^{bot} + (m||0^c))$, where $(A_j^{top}, A_j^{bot}) \xleftarrow{\$}$ `Rooted`, and $m \xleftarrow{\$} \{0,1\}^r$. After this step, $\mathcal{D}$ obtains q rooted notes, and for any $i \in \{1, \ldots, q\}$, the i^{th} node is denoted by (A_i^{top}, A_i^{bot});
3. The distinguisher checks whether the event $\mathsf{BAD\_NODE\_S}$ is set, i.e., there exist $m \in \{0,1\}^r$, $i, j, k \in \{1, \ldots, q\}$ such that $A_i^{top} = X_j^{top} \oplus (m\|0^c)$ and $A_i^{bot} = X_k^{bot} \oplus (m\|0^c)$;
4. If such a collision is found, the distinguisher makes the construction query associated to $(A_i^{top} \oplus (m\|0^c), A_i^{bot} \oplus (m\|0^c))$, and returns 1 if the answer is consistent and 0 otherwise;
5. Otherwise, if $\mathsf{BAD\_NODE\_S}$ is not set, $\mathcal{D}$ returns 0.

In the following, we abbreviate $\mathsf{BAD\_NODE\_S}$ to BNS. $\mathcal{D}$ makes at most $4q$ primitive queries and one construction query. In the real world, the answers are always consistent, and in the ideal world, the answer is consistent with probability $\frac{1}{2^r}$, therefore

$$\begin{aligned}\mathbf{Pr}\left(\mathcal{D}^{W_R} = 1 \mid \mathsf{BNS}\right) &= 1\,,\\ \mathbf{Pr}\left(\mathcal{D}^{W_I} = 1 \mid \mathsf{BNS}\right) &= \frac{1}{2^r}\,.\end{aligned}$$

Moreover, in W_R and W_I, the probability that one fresh edge sets BNS_i is $\approx \frac{q^2}{2^{2c+r}}$, thus

$$\mathbf{Pr}\left(\mathcal{D}^{W_I} \text{ sets } \mathsf{BNS}\right) \approx \mathbf{Pr}\left(\mathcal{D}^{W_R} \text{ sets } \mathsf{BNS}\right) \approx \frac{q^3}{2^{2c+r}}\,.$$

Therefore, noting that $\mathcal{D}$ outputs 0 when BNS is not set,

$$\begin{aligned}\mathbf{Pr}\left(\mathcal{D}^{W_R} = 1\right) &= \mathbf{Pr}\left(\mathcal{D}^{W_R} = 1 \mid \mathsf{BNS}\right)\mathbf{Pr}\left(\mathcal{D}^{W_R} \text{ sets } \mathsf{BNS}\right) \approx \frac{q^3}{2^{2c+r}}\,,\\ \mathbf{Pr}\left(\mathcal{D}^{W_I} = 1\right) &= \mathbf{Pr}\left(\mathcal{D}^{W_I} = 1 \mid \mathsf{BNS}\right)\mathbf{Pr}\left(\mathcal{D}^{W_I} \text{ sets } \mathsf{BNS}\right) \approx \frac{1}{2^r}\frac{q^3}{2^{2c+r}}\,.\end{aligned}$$

So that

$$\begin{aligned}\mathbf{Adv}_{\mathrm{DS}}^{\mathrm{indif}}(\mathcal{D}) &= \left|\mathbf{Pr}\left(\mathcal{D}^{W_R} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{W_I} = 1\right)\right|\\ &\approx \frac{q^3}{2^{2c+r}}\left(1 - \frac{1}{2^r}\right)\,.\end{aligned}$$

Setting $q = 2^{\frac{2c+r}{3}}$ thus gives a high probability success. This means that our simulator cannot give a security bound better than $\frac{q^3}{2^{2c+r}}$.

Note that there is a (small) gap of $r/3$ bits of security between this attack and our proof. This means that either the attack can be improved or (more likely) our simulator has a better security bound. However, proving a better security bound would be extremely challenging. Indeed, in our proof, we do allow inner collisions (as we aim for security beyond the birthday bound), but restrict ourselves to nodes which do not have more than one single collision on their inner part. This was possible thanks to the introduction of dedicated bad events (namely DOUBLE2COL and 3COL) which are set after approximately $2^{2c/3}$ queries. In order to obtain beyond $2c/3$ bits of security, we would have to release this restriction, thus allow higher degree collisions, and possibly allow partial edges to be added to the graph. This would incur a complex tree structure and new, sophisticated bad events. Furthermore, a better bound on the simulator's quality of randomness must also be established.

6.6 Conclusion

In this work, we propose a new permutation-based hashing mode that achieves security beyond the birthday bound in the capacity. It also achieves security beyond the birthday bound in the permutation size if $c > 3b/4$.

We proved $2c/3$ bits of security in the powerful indifferentiability framework, while Section 6.5.2 discussed the maximal achievable security bound with respect to our simulator. As explained in Section 6.5.2, reaching a tight bound with respect to our simulator is extremely hard. This is because inner collisions of degree more than two are difficult to deal with. It should be noted that the attack of Section 6.5.2 is for our specific simulator; it may be that there exists a simulator which defeats this attack.[4] However, the challenges involved in describing such simulator and proving its security make it a difficult undertaking. Indeed, in this case the higher-degree inner collisions would not only become intractable from a proof side (as already explained in Section 6.5.2), but also from the simulator side, as the latter has to keep track of them.

At the time of writing the thesis, a recent work of Leurent and Mathéus [LM25b] introduced, among other results, generic indifferentiability attacks on the double sponge construction exploiting 4-sums, which are independent on the simulator used. Their attack achieves a complexity of $\mathcal{O}(2^c)$ for any mixing layer,

[4]More precisely, we can tweak our simulator $\mathbf{S}$ in a way such that its correctness does not depend on $\neg$BNS, but on a more specific event.

and $\mathcal{O}(\max\{2^{3c/4}, 2^{c-r/3}\})$ against our mixing matrix. In particular, it outperforms our simulator-specific attack when $r > c/2$. Therefore, the current state of affairs for the double sponge is as follows: when $r \leq c/2$, the maximum security achievable with respect to our simulator is $2c/3 + r/3$ bits, and the maximum security achievable by any simulator is now bounded by $c - r/3$ bits. When $r > c/2$, the maximum security achievable for any simulator is $\min\{3c/4, c - r/3\}$ bits.

In Section 6.5.1, we described a collision attack with complexity $q \approx 2^{c+r/2}$ queries, and this is the best collision attack we found. It would therefore be interesting to explore in more depth whether the collision resistance of our construction can indeed exceed the indifferentiability bound.

The double sponge construction may also be relevant in light of recent international efforts to standardize stronger hash functions. In March 2025, the Chinese National Institute of Commercial Cryptography Standards issued a call for hash functions providing 1024-bit preimage and second preimage security with 1024-bit digests [NIC25]. Existing sponge functions, such as SHA-3 ($b = 1600$) cannot achieve more than $c/2$ bits of second preimage security, which corresponds to less than 800 bits for SHA-3 (see Proposition 3.1.1). In a work not presented in this thesis, we showed [SLZ+25] that a minor modification to the sponge can, when instantiated with $b = 1600$, boost first and second preimage security to 1024 bits, but leaves collision security at 800 bits at best. As a more robust alternative, one may consider the double sponge construction: with $b = 1600$ and $r = 64$, it attains 1024 bits of indifferentiability security. Although this is inefficient, especially given the permutation and state size, we believe this is a promising direction for developing permutation-based hash functions that exceed the birthday bound in *overall* security.

Ideally, one would like to achieve c bits of security with two permutation calls per compression function call, similarly to what has been achieved for double block length hashing schemes based on block ciphers. However, contrary to block ciphers, permutations are non-compressing primitives, making this direction challenging. One potential future work would be to consider an n-sponge based on n independent permutations, augmented with a well-chosen mixing layer. This direction is comparable to provably secure confusion-diffusion networks [DSSL16], and may allow to surpass $2c/3$ bits of security.

Part III

Application-Specific Permutation-Based Hashing

Chapter 7

Permutation-Based Hash Chains with Application to Password Hashing

7.1 Introduction

The far majority of internet services rely on passwords for authentication. However, despite their broad usage, they introduce significant security weaknesses. For example, major security problems have appeared in the context of static passwords over the last years. These problems have lead the more security-sensitive services to move to two-factor authentication, where an additional device, app, or something else is used for the user to authenticate themselves. The FIDO Alliance has made significant efforts to standardize and promote secure authentication [FID25].

A second authentication factor can take many forms, such as biometrics, email verification, hardware tokens, or smartphone applications. Among these, time-based one-time password schemes are commonly used. For example, Microsoft Authenticator uses the TOTP scheme [MMPR11] internally. TOTP generates a time-based one-time password by performing an HMAC evaluation [BCK96, Nat08] over a time-derived counter, thereby limiting the validity period of each password. TOTP requires the client and server to share a secret key, which can be problematic in certain situations, especially when it comes to secure key storage on the server.

To avoid storing secret keys on the server, an alternative approach is to use hash chains. In a nutshell, the core idea of a hash chain, which dates back to Lamport [Lam81] and is specified in an RFC as S/Key [Hal95], is the following. Let $h : \{0,1\}^n \to \{0,1\}^n$ be a cryptographic hash function. As initialization, the client generates a random password x of n bits, and sends $x_K := h^K(x)$ securely to the server, where h^K is the K-fold evaluation of h. Then, during authentication round k, for $k \in \{1,\dots,K\}$, the client sends $x_{K-k} := h^{K-k}(x)$ to the server, which verifies that x_{K-k} is indeed a preimage of x_{K-k+1} through h. The server updates its stored value to x_{K-k}. Although this construction is very simple, it suffers from two main weaknesses: passwords remain valid indefinitely (at least, until the next authentication round takes place), and the repeated iteration of the same one-way function makes it weaker than a single iteration [BDD$^+$17].

To address these limitations, Kogan et al. introduced a time-based one-time password scheme named T/Key [KMB17]. T/Key solves the validity issue by including timestamps in the iterated evaluations of h and the multi-iteration problem by using domain separation. It is inspired by the aforementioned TOTP scheme, with crucial difference that TOTP relies on a shared secret key whereas T/Key does not. In detail, in T/Key the hash chain transition from password x_{K-k} to x_{K-k+1} for $k \in \{1,\dots,K\}$ is computed as

$$x_{K-k+1} := h(\langle \mathrm{ctr}_{K-k+1}\rangle_t \parallel id \parallel x_{K-k})\,, \tag{7.1}$$

where $\langle \mathrm{ctr}_k\rangle_t$ encodes injectively the timestamp, and id is a random salt used to avoid pre-computation attacks. In other words, the initialization computes x_K from a random x as

$$x_K := h_K^{id} \circ h_{K-1}^{id} \circ \cdots \circ h_1^{id}(x)\,,$$

where $h_k^{id}(\cdot) := h(\langle \mathrm{ctr}_k\rangle_t \parallel id \parallel \cdot)$ for brevity. Kogan et al. proved that T/Key is a secure hash chain based password system under the assumption that the hash function is a random oracle [BR93]. In detail, assuming that h is a random oracle, the functions $h_1,\dots,h_K$ can be considered perfectly random and independent, and breaking the hash chain then corresponds to inverting any point on the chain. An adversary can only succeed in this with probability at most approximately

$$\frac{2q+2K}{2^n}\,, \tag{7.2}$$

where q is the number of random oracle queries and n the output size of the random oracle.

However, the security model in which the result (7.2) was obtained is rather basic, and in particular does not accurately capture the adversarial power. Specifically, two issues arise: (i) the model does not refactor the adversarial complexities into offline and online time, and (ii) it does not consider the multi-user setting. The former issue is particularly relevant in the context of T/Key where time frames are rather short – the developers suggest time frames of 30 seconds. Short time frames mean that the adversary can only make a limited amount of "live" authenticity attempts after it has learned the value to target, but it has had a much larger pre-computation phase where it could have made a retrospectively lucky hash function evaluation. The latter issue on the multi-user setting is important as T/Key would not be used in isolation but rather by thousands of users at the same time. In addition, dedicated multi-user analysis (as opposed to using a generic single-user to multi-user reduction [Bih02, BBM00]) typically allows for more detailed security bounds, and thus, potentially, a longer lifetime of hash chains.

On top of this, the result (7.2) only holds in the random oracle model. In order to then use T/Key (or any other hash chain based system), one first has to instantiate the random oracle with an actual hash function. In detail, if one would instantiate the random oracle as a chop-Merkle-Damgård construction [Mer89, Dam89, CDMP05] (used in SHA-224, SHA-384, SHA-512/224, and SHA-512/256) or a sponge construction [BDPV07] (used in SHA3-{224,256,384,512}, SHAKE128, and SHAKE256), it is possible to rely on the indifferentiability security level [MRH04, CDMP05] of said constructions [CDMP05, CN08, BDPV08], which implies that the constructions "behave like" random oracles,[1] but these analyses are not tailored towards the current use case: they do not consider different types of attack phases and different users at the same time. Note that Kogan et al. instantiate T/Key with SHA-256, with the output truncated to 130 bits, de facto making it a chop-Merkle-Damgård construction.

7.1.1 Universal Construction and Generalized Model

We consider a slight abstraction of T/Key, which applies to any hash chain where a transition from one password to the next one is computed using

$$h(\langle \mathrm{ctr}_k \rangle_t \parallel id \parallel x)\,,$$

where $\langle \mathrm{ctr}_k \rangle_t$ encodes injectively a counter or timestamp, and id is a random salt (identical throughout the entire hash chain) used to avoid pre-computation

[1]Refer to the discussions in Section 2.4.1.2 for further discussions on the indifferentiability framework.

attacks. For brevity, we denote this slightly more universal hash chain based password system by U/Key. A detailed formalism of the construction is given in Section 7.2.1.

We then introduce in Section 7.2.2 a new security model for hash chain based password systems of this type. By design of U/Key, this model formalizes the release pattern unique to hash chain based password systems, where the adversary is successively given elements from a chain, and has a limited number of queries to invert each of them. In addition, the model does not restrict itself to the random oracle model; to the contrary, h can be any hash function construction based on an idealized underlying primitive. This, in particular, makes the model directly applicable to popular hash function constructions such as chop-Merkle-Damgård, Merkle-Damgård with a prefix-free padding [Mer89, Dam89, CDMP05], and the sponge [BDPV07], but also any other construction which as a hash function is not indifferentiable from a random oracle. (As a matter of fact, the construction that we consider in Section 7.4.1 is, for some parameter set, differentiable, while still maintaining a negligible adversary success probability.)

The model also allows for a security analysis in a more fine-grained modeling of the adversary than before: the model refactors the adversarial complexity into offline queries and online queries, a strategy previously explored in a general setting by Ghoshal and Tessaro [GT23]. In addition, it is defined in the multi-user setting. With these refinements, our model is a strict generalization of the model of Kogan et al. [KMB17].

7.1.2 Refined Analysis of U/Key

Having settled the universal construction U/Key and the novel security model, we perform an updated analysis of hash chain based password systems fitting to the U/Key specification in the random oracle model, thus covering more fine-grained adversaries. This involves analyzing cascades of random oracles in a complex setting, notably accounting for the online and offline phases and the multi-user setting. We start with U/Key in the random oracle model in Section 7.3.1 and prove that it is secure up to a bound of the order

$$\tilde{\mathcal{O}}\left(\frac{\mu}{2^s} \cdot \frac{q_{\mathit{off}}}{2^n} + \max\left\{\frac{\mu}{2^s}, 1\right\} \cdot \frac{q_{on}}{2^n}\right), \tag{7.3}$$

where n is the length of the passwords, μ is the number of users, s the size of the salts, and q_{off} and q_{on} denote respectively the number of offline and online queries. Most notably, this bound indicates that in the multi-user setting, security degrades (i) proportionally to the number of users regarding the offline phase and (ii) proportionally to $\max\left\{\frac{\mu}{2^s}, 1\right\}$ regarding the online phase. A technical difficulty in the proof is caused by the fact that a special treatment is needed on whether an earlier offline query is “useful” later on; this treatment differs depending on the number of users.

Next, in Section 7.3.3 we investigate the implication of this result in case the random oracle is instantiated with a hash function construction that is indifferentiable from a random oracle. In these analyses, we focus on a total query complexity $q = q_{off} + q_{on}$.

Merkle-Damgård. First, in Section 7.3.3.1 we compose the hash chain based password systems with the Merkle-Damgård construction [Mer89,Dam89], remarking that in the context of U/Key, the inputs to the Merkle-Damgård construction are prefix-free, which gives a bound of the order [CDMP05]

$$\tilde{\mathcal{O}}\left(\frac{q}{2^n} + \frac{q^2}{2^b}\right),$$

where b denotes the output size of the compression function.

Sponges. Then, in Section 7.3.3.2, we analyze the security of U/Key when the construction is instantiated with a sponge [BDPV08], and we obtain a bound of the order

$$\tilde{\mathcal{O}}\left(\frac{q}{2^n} + \frac{q^2}{2^c}\right),$$

where c denotes the capacity. This result is particularly interesting because SHA-3 as well as the NIST Lightweight Cryptography winner Ascon [DEMS21a, DEMS21b] are based on the sponge construction.

Note that this bound implies that security cannot be guaranteed beyond half of the permutation size. However, the security game underlying the security of U/Key is in fact a complexified variant of (domain-oriented) preimage resistance, and we already saw in Chapter 4 that the preimage resistance of the sponge can, in fact, significantly surpass $c/2$ bits of security. These two observations suggest that the above bounding is lossy. Inspired by this, we derive in Section 7.4 a dedicated security proof. The complexity of this proof arises from the simultaneous presence of multiple chaining values that can be inverted. Additionally, due to the sponge evaluating the digests in multiple rounds, accounting for the release pattern dictated by U/Key becomes markedly difficult. As a matter of fact, the security proof may be considered of independent interest, as internally it solves the problem of understanding the preimage resistance of a cascaded evaluation of the sponge construction. In the single-user setting, restricted to the simpler T/Key, we derive a bound of the order

$$\tilde{\mathcal{O}}\left(\frac{K\ell(q_{off} + q_{on})}{2^n} + \frac{K\ell^2(q_{off} + q_{on})}{2^c} + \min\left\{ \frac{\ell q_{on}}{2^{n-r}}, \frac{(q_{off} + q_{on})q_{on}}{2^c} \right\} \right), \tag{7.4}$$

where $r = b - c$, and $\ell := \lceil \frac{n}{r} \rceil$. Our analysis applies not only to the native sponge construction, but also to an optimization of the sponge in the context of U/Key. In this optimization, the state of the sponge is initialized directly with the counter and the salt, which allows to reduce the number of permutation calls required during the absorption phase (see Section 7.4.1 for more details). Moreover, our proof is modular and can be easily extended to any keyed construction which has been proven to be indistinguishable from a random function.

Truncated Permutation. Looking at these sponge functions in detail, it can be observed that in the context of hash chains, they can be evaluated very efficiently. For example, a typical instantiation of T/Key operates on counters of 32 bits, salts of 80 bits, and a password of 130 bits [KMB17], thus requiring a hash function from 242 to 130 bits. SHA-3 internally uses a 1600-bit permutation, and processes messages and squeezes digests $1600 - 2k$ bits at a time, where k is the security level. This means that if SHA-3 (any of the four instances) is used in the T/Key hash chain, a hash function evaluation simply consists of a truncated permutation evaluation. With this in mind, we additionally consider the security of U/Key if the random oracle is instantiated with a truncated permutation in Section 7.3.3.3. This construction is also proven to be indifferentiable from a random oracle [CLL19], and we obtain a bound of the order

$$\tilde{\mathcal{O}}\left(\frac{q}{2^n} + \frac{q^{3/2}}{2^{\frac{2b-n}{2}}} + \frac{q}{2^{b-(n+t)}} \right), \tag{7.5}$$

where b denotes the permutation size and t the counter size.

However, as before, indifferentiability composition is still a bit lossy, as the indifferentiability analyses of these constructions have not been tailored towards the specific use case of hash chains: in particular, the bound is expressed in the *total* query complexity q only. Therefore, in Section 7.5, we take a closer look at U/Key using a truncated permutation and derive a *dedicated* security proof. The proof is of a high level of technicality and contains various subtleties, but on the upside, it is much better than the results obtained with the more general sponge in (7.4), or by combining (7.3) with the indifferentiability bound of truncation. In detail, we derive a bound of the order

$$\tilde{\mathcal{O}}\left(\frac{\mu q_{\mathit{off}}}{2^{n+s}} + \frac{K\mu q_{\mathit{off}}}{2^b} + \max\left\{\frac{\mu}{2^s}, 1\right\} \cdot \frac{q_{on}}{2^n} + \max\left\{\frac{K\mu}{2^n}, 1\right\} \cdot \frac{q_{on}}{2^{b-n}}\right) .$$

Concretely, if we consider counters of $t = 32$ bits, and a setting with less than $\mu = 10^{12}$ users, then one can use permutations *as small as* $b = 200$ *bits*, while still having security guaranteed up to 2^{100} and 2^{128} queries in the online and offline phases, respectively. This improves significantly over the bound in (7.5), which, in that setting, cannot guarantee more than 84 bits of security (even during the offline phase) in the single-user setting. Note that the obtained security bound is better than that of U/Key instantiated with a (more general) sponge construction, which is essentially caused by the fact that a hash function attacker has fewer freedom in attacking a single truncated permutation than attacking the sponge.

7.1.3 Application

The results of U/Key in the random oracle model allow to derive more fine-grained bounds that directly apply to the instantiation suggested by the designers of T/Key. More precisely, Kogan et al. instantiate the hash function with SHA-256, which itself is based on the Merkle-Damgård construction. However, given their parameter sets with counters of 32 bits, salts of 80 bits, and passwords of 130 bits, the 512-bit block size of SHA-256 allows to process all data of a chaining value in a single compression function call. Thus, the result from (7.3) applies, and suggests that the password size could be lowered to 100 bits, providing 100 and 128 bits of security in respectively.

On the other side, U/Key with a small truncated permutation can offer efficiency advantages over U/Key based on SHA-2. For instance, U/Key based on Keccak-f[200] operates on a state of 200 bits. In that case, setting $n = 100$ and $s = 68$ provides 100 and 128 bits of security in the online and offline phases, respectively, as long as the number of users stays below 2^{40}.

7.1.4 Outline

The generalized hash chain based password system U/Key that we consider in this work is formalized in Section 7.2.1 and the novel model is presented in Section 7.2.2. The security of U/Key in the new model is derived in Section 7.3:

first, a new random oracle model result is derived in Section 7.3.1, and then this result is combined with indifferentiability security results on (prefix-free) Merkle-Damgård, the sponge, and a truncated permutation in Section 7.3.3. Sections 7.4 and 7.5 present improved dedicated security results on U/Key with a sponge and a truncated permutation, respectively. Finally, Section 7.6 concludes.

7.2 Hash Chain Based Password System

The generic construction is described in Section 7.2.1 and the security model in Section 7.2.2.

7.2.1 Construction

We will consider a construction that is a bit more abstract than T/Key, which in turn was already a generalization of S/Key. For brevity, we will dub our construction U/Key, where U stands for "universal". After the description, we will discuss in more detail how T/Key fits this construction.

Let K, n, s, t be integers, and $h : \{0,1\}^{n+s+t} \to \{0,1\}^n$ a cryptographic hash function. We will describe a hash chain based password system based on h, U/Key$[h]$, with a chain length of K, a password size of n bits, salts of s bits, and a counter over t bits. It is defined through an initialization phase, a chain computation phase, and an authentication phase:

1. Initialization: the client draws a password $x_0 \xleftarrow{\$} \{0,1\}^n$ and a salt $id \xleftarrow{\$} \{0,1\}^s$;

2. Chain computation: for $k = 1, \ldots, K$, the client evaluates $x_k = h_k\left(x_{k-1}\right)$, where h_k is defined as follows:

$$h_k(x) = h\left(\langle \mathrm{ctr}_k \rangle_t \parallel id \parallel x\right) ,$$

   where ctr_k is some counter value that may be specific to the actual system. After this computation, the client sends id and x_K to the server;

3. Authentication: if the client wants to authenticate at round k, and the last authentication was at round $k' > k$,[2] it sends the password x_k to the

[2]Typically, $k' = k+1$, but for a time-based system such as T/Key, this is not necessarily the case.

server. The server on its side retrieves the latest password it has stored, $x_{k'}$, and verifies if

$$h_{k'}(h_{k'-1}(\cdots(h_{k+1}(x_k))\cdots)) = x_{k'}$$

holds. It updates the stored round to k and password to x_k. Note that, depending on the concrete underlying scheme, the counter k may need to be sent along with the password x_k. In the case of T/Key, there is a resynchronization in place based on different successful attempts. This could be used as alternative solution.

Note that, for authentication, the client does not necessarily have to store the whole chain on its side. Instead, it can opt to store certain "checkpoints" [KMB17], or perform an efficient pebbling method [Sch16]. These approaches offer a trade-off between memory and computation time required to evaluate the hashes.

T/Key. The T/Key system [KMB17], specifically, is covered by above description of U/Key with two refinements. First, upon initialization, the client takes the current timestamp t_i and takes as counter values $\mathrm{ctr}_k = t_\mathrm{i} + K - k$, and the value of t_i is sent to the server along with the salt *id* and the last chaining value x_K. The chain is processed in a pre-described time-based manner, where any transition happens each predetermined period. This means that upon authentication the difference between k' and k depends on the time the user authenticated last.

For reference, the designers of T/Key suggest using this construction with the following typical parameters:

$$K = 2 \times 10^6\,, \qquad n = 130\,, \qquad s = 80\,, \qquad t = 32\,.$$

Furthermore, each password is valid for 30 seconds (so the chain is reversed each 30 seconds).

7.2.2 Security Model

We will describe security of U/Key of Section 7.2.1. It will not be defined for a hash function h but rather for a hash function construction $\mathcal{H}$ based on idealized primitive $\mathcal{F} \in \texttt{Prim}$: we denote this by U/Key$[\mathcal{H}^\mathcal{F}]$. The security game is defined in Algorithm 10. The attack game consists of two phases. In the first one, called the *offline phase*, the adversary $\mathcal{A}$ is allowed to make pre-computation queries, which translates to $\mathcal{A}$ being able to make at most

q_{off} queries to $\mathcal{F}$. In the second phase, called the *online phase*, the adversary $\mathcal{A}$ has additional access to $\mu \geq 1$ parallel runs of the construction, where each hash chain has a length K. The online phase itself is subdivided into K rounds. For $k \in [\![1, K]\!], m \in [\![1, \mu]\!]$, let $x_{k,m}$ be the chaining value number k of user number m, and let $\mathcal{H}^{\mathcal{F}}_{k,m}(x) = \mathcal{H}^{\mathcal{F}}(\langle \mathrm{ctr}_k \rangle_t \parallel \mathit{id}_m \parallel x)$. At the beginning of round number k, all $x_{k,m}$'s are released to the adversary, which is then allowed to make $q_{on,k}$ queries to $\mathcal{F}$. At the end of the round, $\mathcal{A}$ may submit $x \in \{0,1\}^n$, and wins if x is a preimage of one of the $x_{k,m}$ by $\mathcal{H}^{\mathcal{F}}_{k,m}$. Without loss of generality, we assume that the adversary proposing x as a preimage of $x_{k,m}$ has made all necessary primitive queries to evaluate $\mathcal{H}^{\mathcal{F}}_{k,m}(x)$. The total number of online queries is denoted by $q_{on} := \sum_{k=1}^{K} q_{on,k}$.

For simplicity, we assume that the initializations of all sessions are performed at the same time, and so are the traversals through the chains. Note that in reality the offline and online phases of the different users are not necessarily synchronized, as online queries of one user could be made during the offline phase of another user. Thus, q_{off} represents the maximum, over all users, of offline queries. Therefore, we believe that enforcing that all users start at the same time in the model does not result in a loss of generality. We define $\mathbf{Adv}_{\mathrm{U/Key}[\mathcal{H}]}\left(q_{\mathit{off}}, (q_{on,k})_{k\in[\![1,K]\!]}, \mu\right)$ to be the maximum success probability in SecGame$(\mathcal{A}, \mathcal{H}, q_{\mathit{off}}, (q_{on,k})_{k\in[\![1,K]\!]}, \mu)$ of Algorithm 10, maximized over all adversaries $\mathcal{A}$. Note, when the security bound is independent on the way the online queries $(q_{on,k})_k$ are distributed (which is the case for our random oracle model analysis and that of the truncated permutation construction), we rather use the notation $\mathbf{Adv}_{\mathrm{U/Key}[\mathcal{H}]}\left(q_{\mathit{off}}, q_{on}, \mu\right)$.

Adversarial Resources. The adversary is bounded by a certain offline complexity q_{off} and by online complexities $q_{on,k}$ for $k \in [\![1, K]\!]$. Note that the offline complexity may be rather high. The adversary can do pre-computations (possibly on more powerful computers), and we would typically aim at around 128-bit security, which is the level of generic security guaranteed by, e.g., SHA-256 or SHA3-256. This means that q_{off} may get as high as 2^{128}. (Of course, storage may then be a problem for the adversary; refer to Section 7.6.2 for further discussion regarding this.)

The fine-grained definition of the online complexities allows for analyzing the hash chain based system U/Key of Section 7.2.1 in full generality, noting that if a user leaves a big gap between two authentication phases, the adversary has more time to attack the scheme *in that round*. That said, for T/Key each period is of the same length, e.g., 30 seconds as suggested by its designers. In this case, all online complexities $q_{on,k}$ are (roughly) identical. A rough bound for the online complexities in U/Key can be derived from the computations for the Bitcoin blockchain. It is estimated [Blo25] that currently all Bitcoin miners in total perform $\approx 2^{70}$ SHA-256 evaluations per second. In particular, 2^{100} evaluations would require more than 35 years of computation. Although this is a very rough upper bound, in particular if a different hash function is selected, we can use it to upper bound $q_{on,k}$.

Algorithm 10 Security model for U/Key based on the construction $\mathcal{H}$, itself relying on a primitive $\mathcal{F} \in \texttt{Prim}$. We write $\mathcal{H}^{\mathcal{F}}_{k,m}(x) := \mathcal{H}^{\mathcal{F}}(\langle \mathrm{ctr}_k \rangle_t \parallel id_m \parallel x)$ for brevity.

```
1:  function SecGame($\mathcal{A}, \mathcal{H}, q_{off}, (q_{on,k})_{k \in [\![1,K]\!]}, \mu$)
2:      $\mathcal{F} \xleftarrow{\$} \texttt{Prim}$
3:      OfflinePhase($\mathcal{A}, \mathcal{F}$)
4:      OnlinePhase($\mathcal{A}, \mathcal{H}, \mathcal{F}$)

5:  function OfflinePhase($\mathcal{A}, \mathcal{F}$)
6:      $\mathcal{A}$ is allowed to make at most $q_{off}$ queries to $\mathcal{F}$

7:  function OnlinePhase($\mathcal{A}, \mathcal{H}, \mathcal{F}$)
8:      for $m = 1, \dots, \mu$ do
9:          // consider $\mu$ independent initializations
10:         $id_m, (x_{0,m}, x_{1,m}, \dots, x_{K,m}) \leftarrow$ U/Key[$\mathcal{H}^{\mathcal{F}}$]
11:     $k \leftarrow K$
12:     $\mathcal{A}$ is given $id_m$ for all $m \in [\![1, \mu]\!]$
13:     while $k > 0$ do
14:         $\mathcal{A}$ is given $x_{k,m}$ for all $m \in [\![1, \mu]\!]$
15:         $\mathcal{A}$ is allowed to make at most $q_{on,k}$ queries to $\mathcal{F}$
16:         PreimageGuess($\mathcal{A}, \mathcal{H}, \mathcal{F}, x_{k,1}, \dots, x_{k,\mu}$)
17:         $k = k - 1$

18: function PreimageGuess($\mathcal{A}, \mathcal{H}, \mathcal{F}, x_{k,1}, \dots, x_{k,\mu}$)
19:     $\mathcal{A}$ is allowed to submit $y \in \{0,1\}^n$ and wins if there exists $m \in [\![1, \mu]\!]$
        such that $\mathcal{H}^{\mathcal{F}}_{k,m}(y) = x_{k,m}$
```

7.3 Random Oracle Results with U/Key

Given the novel security model with refined treatment of adversarial power, it makes sense to have a look at U/Key based on a random oracle and to supersede the original security result on T/Key in the new model. We do so in Section 7.3.1, with the proof in Section 7.3.2. In this section, we also particularly elaborate on the impact of the split of adversarial complexity into offline and online complexity.

Then, in Section 7.3.3 we investigate the meaning of the result if the random oracle is instantiated with some well-established construction. In detail, we consider the instantiation with (prefix-free) Merkle-Damgård in Section 7.3.3.1, with the sponge in Section 7.3.3.2, and with the truncated permutation in Section 7.3.3.3.

7.3.1 U/Key Security in the New Model

In Theorem 7.3.1 we state the security of U/Key in the new security model when instantiated with a random oracle.

Theorem 7.3.1. *Let $\mathcal{RO} : \{0,1\}^{n+s+t} \to \{0,1\}^n$ be a random oracle, and consider the hash function $\mathcal{H}^{\mathcal{RO}} : \{0,1\}^{n+s+t} \to \{0,1\}^n$ as $\mathcal{H}^{\mathcal{RO}}(x) = \mathcal{RO}(x)$. Then, we have*

$$\mathbf{Adv}_{\mathrm{U/Key}[\mathcal{H}]}(q_{\mathit{off}}, q_{\mathit{on}}, \mu) \leq \mathrm{mucol}(\mu, 2^s) \cdot \left(\min\left\{ \frac{2\mu}{2^s}, 1 \right\} \cdot \frac{2q_{\mathit{off}}}{2^n} + \frac{2q_{\mathit{on}}}{2^n} \right) ,$$

where $\mathrm{mucol}(\cdot,\cdot)$ *is defined in Lemma 2.5.2.*

The proof is given in Section 7.3.2.

Interpretation of the Bound. Remember that the total number of queries is given by $q_{\mathit{off}} + q_{\mathit{on}}$. If $\mu \leq 2^{s-1}$, the bound is of the order

$$\tilde{\mathcal{O}}\left(\frac{\mu}{2^s} \cdot \frac{q_{\mathit{off}}}{2^n} + \frac{q_{\mathit{on}}}{2^n} \right) .$$

Thus in the multi-user setting, the offline phase displays a security loss proportional to μ, while there is no security loss for the online phase. This is explained by the fact that in the offline phase, the adversary is not provided with the salts and must thus correctly guess them. Increasing the number of users increases the adversary probability to make a query with one correct salt by a factor of μ. On the other hand, if $\mu \geq 2^{s-1}$, then the bound is of the order

$$\tilde{\mathcal{O}}\left(\frac{\mu}{2^s} \cdot \left(\frac{q_{\mathit{off}} + q_{on}}{2^n}\right)\right) .$$

Indeed, when $\mu \gg 2^s$, for a well-chosen adversarial strategy, then (almost) all queries of the offline phase are expected to correspond to some existing salt. Moreover in this setting, some salts are expected to collide, thus one query from the adversary can target several chaining values at the same time, hence the factor of $\frac{\mu}{2^s} > 1$.

The security bound allows for many different interpretations due to its flexibility in the number of users and the number of offline and online queries. For example, for a typical instantiation with $q_{\mathit{off}} \ll 2^{128}$, $K = 2^{21}$, and $q_{on} \ll 2^{100}$, with a salt size $s = 80$ and password size $n = 100$, one can guarantee security as long as the number of users is at most $\mu \leq 2^{52}$. These results support the generic security of the instantiation by the designers of T/Key with SHA-256 [KMB17], as one iteration of the compression function suffices to process the salt, the counter, and the chaining value.

7.3.2 Proof of Theorem 7.3.1

For $k \in [\![1, K]\!], m \in [\![1, \mu]\!]$, let $x_{k,m}$ be the chaining value number k of user number m ($x_{0,m}$ denotes the root passwords). Without loss of generality, we assume that the salts are drawn before the offline phase. We define a random variable Coll_s that determines how the salts are colliding together. This variable is represented as a vector with μ coordinates, each element taking values in $[\![1, \mu]\!]$. When two elements at indices i and j share the same value, it means that the salts of users i and j collide. Fixing Coll_s determines the number of distinct salts, that we will denote as $\tilde{\mu}$. Conditioning on this variable does not provide information about the randomness of the $\tilde{\mu}$ distinct salt values, except for the fact that they are sampled without replacement. Moreover, let NC_s be a random variable counting the maximum number of colliding salts, i.e.,

$$\mathrm{NC}_s = \max_{id} \# \left\{ m \in [\![1, \mu]\!] \;\middle|\; id_m = id \right\} .$$

In particular, the value of Coll_s determines the one of NC_s. We have

$$\mathsf{E}\left(\mathrm{NC}_s\right) = \mathrm{mucol}(\mu, 2^s) , \tag{7.6}$$

where $\mathrm{mucol}(\cdot, \cdot)$ is defined in Lemma 2.5.2. Assume that the salts are colliding according to a certain $\mathrm{coll}_s \in ([\![1, \mu]\!])^{\mu}$. Fixing this random variable fixes the number of distinct salts, that we denote by $\tilde{\mu} \leq \mu$. Moreover, let $k \in [\![1, K]\!]$, and $\tilde{m} \in [\![1, \tilde{\mu}]\!]$. Define $h_{k,\tilde{m}}$ to be $h\left(\langle \mathrm{ctr}_k \rangle_t \parallel id_{\tilde{m}} \parallel \cdot\right)$, where $id_{\tilde{m}}$ is the $\tilde{m}^{\text{th}}$ distinct salt.

The adversary's success in winning the security game relies on finding a preimage of one of the chaining values, which can only be obtained through $h_{k,\tilde{m}}$-queries. Since h is a random oracle, having knowledge of other outputs that are not obtained through $h_{k,\tilde{m}}$-queries does not enhance the adversary's probability of success. Therefore, such queries are considered "useless". In particular, during the offline phase the adversary has no access to the salts, thus it has to guess them, and whenever $\mu \ll 2^s$, this significantly lowers the adversarial success probability. To capture this phenomenon, we use the random variables $Q_{\mathit{off}}^{(k,\tilde{m})}$ which counts the number of useful $h_{k,\tilde{m}}$-queries during the offline phase. Moreover, let $q_{\mathit{on}}^{(k,\tilde{m})}$ be the number of $h_{k,\tilde{m}}$-queries during the online phase. Note that this quantity is not a random variable, since the adversary is given the salts during the online phase. Moreover, we partition the offline (resp., online) queries according to the counter value queried, i.e., $q_{\mathit{off}}^{(k)}$ (resp., $q_{\mathit{on}}^{(k)}$) counts the number of queries of the form $h(\langle \mathrm{ctr}_k \rangle_t \parallel \cdot \parallel \cdot)$. Finally, let

$$Q^{(k,\tilde{m})} = Q_{\mathit{off}}^{(k,\tilde{m})} + q_{\mathit{on}}^{(k,\tilde{m})} . \tag{7.7}$$

Then, it holds that

$$\begin{aligned} \sum_{k=1}^{K} \sum_{m=1}^{\tilde{\mu}} Q_{\mathit{off}}^{(k,\tilde{m})} &\leq \sum_{k=1}^{K} q_{\mathit{off}}^{(k)} \leq q_{\mathit{off}} , \\ \sum_{k=1}^{K} \sum_{m=1}^{\tilde{\mu}} q_{\mathit{on}}^{(k,\tilde{m})} &\leq \sum_{k=1}^{K} q_{\mathit{on}}^{(k)} \leq q_{\mathit{on}} , \\ \sum_{k=1}^{K} \sum_{m=1}^{\tilde{\mu}} Q^{(k,\tilde{m})} &\leq q_{\mathit{off}} + q_{\mathit{on}} . \end{aligned} \tag{7.8}$$

Furthermore, when $\mu \leq 2^{s-1}$, we have

$$\mathsf{E}\left(Q_{\mathit{off}}^{(k,\tilde{m})} \mid \mathrm{Coll}_s = \mathrm{coll}_s\right) \leq \frac{q_{\mathit{off}}^{(k)}}{2^s - \mu} \leq \frac{2q_{\mathit{off}}^{(k)}}{2^s} . \tag{7.9}$$

The adversary $\mathcal{A}$ wins the game whenever there exists $k \in [\![1, K]\!], \tilde{m} \in [\![1, \tilde{\mu}]\!]$ such that $\mathcal{A}$ found a preimage of one of the x_{k,m_i}, where all m_i's have the same salt equal to the $\tilde{m}^{\text{th}}$ distinct salt (corresponding thus to a $h_{k,\tilde{m}}$-query). Denote this event by $\mathsf{HIT}[k,\tilde{m}]$. We have

$$\mathbf{Adv}_{\mathrm{U/Key}[\mathcal{H}]}\left(q_{\mathit{off}}, q_{on}, \mu\right) \leq \sum_{\mathrm{coll}_s} \mathbf{Pr}\left(\bigvee_{k,\tilde{m}} \mathsf{HIT}[k,\tilde{m}] \wedge \mathrm{Coll}_s = \mathrm{coll}_s\right)$$

$$\leq \sum_{\mathrm{coll}_s} \sum_{\tilde{m}=1}^{\tilde{\mu}} \sum_{k=1}^{K} \underbrace{\mathbf{Pr}\left(\mathsf{HIT}[k,\tilde{m}] \wedge \bigwedge_{k'<k} \neg\mathsf{HIT}[k',\tilde{m}] \wedge \mathrm{Coll}_s = \mathrm{coll}_s\right)}_{(10)} . \qquad (7.11)$$

Here, we introduce the condition $\bigwedge_{k'<k} \neg\mathsf{HIT}[k',\tilde{m}]$, since if an adversary hits an earlier chaining value, it can set $\mathsf{HIT}[k,\tilde{m}]$ by making cascaded evaluations. Now, we can split (10) according to the value of $Q^{(k,\tilde{m})}$. Therefore,

$$(10) \leq \mathbf{Pr}\left(\mathrm{Coll}_s = \mathrm{coll}_s\right) \sum_{q^{(k,\tilde{m})}} \mathbf{Pr}\left(Q^{(k,\tilde{m})} = q^{(k,\tilde{m})} \mid \mathrm{Coll}_s = \mathrm{coll}_s\right) \times$$

$$\mathbf{Pr}\left(\mathsf{HIT}[k,\tilde{m}] \mid \bigwedge_{k'<k} \neg\mathsf{HIT}[k',\tilde{m}] \wedge \mathrm{Coll}_s = \mathrm{coll}_s \wedge Q^{(k,\tilde{m})} = q^{(k,\tilde{m})}\right) .$$

For a given coll_s, let nc_s be the corresponding value of the random variable NC_s. There are two possibilities for the adversary to set the conditioned $\mathsf{HIT}[k,\tilde{m}]$:

- The adversary guesses one *exact* preimage, or in other words, makes a $h_{k,\tilde{m}}$-query with input x_{k-1,m_i}, where user m_i has the salt number $\tilde{m}$. In that case, since the functions $h_{k,\tilde{m}}$ are independent random oracles, each preimage is sampled uniformly at random, and knowledge of the values $x_{k'',m}$ for $k'' > k$ does not help the adversary. One $h_{k,\tilde{m}}$-query can target simultaneously at most nc_s different chaining values. Therefore, the conditioned probability of this case of $\mathsf{HIT}[k,\tilde{m}]$ is upper bounded by $\frac{\mathrm{nc}_s q^{(k,\tilde{m})}}{2^n}$;

- The adversary finds a preimage, which is not the *exact* one. Again, one $h_{k,\tilde{m}}$-query can target simultaneously at most nc_s different chaining values, and each new query results in a uniformly random output. Thus, the conditioned probability of this case of $\mathsf{HIT}[k,\tilde{m}]$ is upper bounded by $\frac{\mathrm{nc}_s q^{(k,\tilde{m})}}{2^n}$.

Therefore,

$$
\begin{aligned}
(7.11) \leq & \sum_{\mathrm{coll}_s} \mathbf{Pr}\left(\mathrm{Coll}_s = \mathrm{coll}_s\right) \times \sum_{\tilde{m}=1}^{\tilde{\mu}} \sum_{k=1}^{K} \sum_{q^{(k,\tilde{m})}} \mathbf{Pr}\left(Q^{(k,\tilde{m})} = q^{(k,\tilde{m})} \mid \mathrm{Coll}_s = \mathrm{coll}_s\right) \\
& \qquad \times \frac{2\mathrm{nc}_s q^{(k,\tilde{m})}}{2^n} \\
\leq & \sum_{\mathrm{coll}_s} \mathbf{Pr}\left(\mathrm{Coll}_s = \mathrm{coll}_s\right) \sum_{\tilde{m}=1}^{\tilde{\mu}} \sum_{k=1}^{K} \frac{2\mathrm{nc}_s}{2^n} \mathsf{E}\left(Q^{(k,\tilde{m})} \mid \mathrm{Coll}_s = \mathrm{coll}_s\right) \\
\leq & \sum_{\mathrm{coll}_s} \frac{2\mathrm{nc}_s}{2^n} \mathbf{Pr}\left(\mathrm{Coll}_s = \mathrm{coll}_s\right) \mathsf{E}\left(\sum_{\tilde{m}=1}^{\tilde{\mu}} \sum_{k=1}^{K} Q^{(k,\tilde{m})} \mid \mathrm{Coll}_s = \mathrm{coll}_s\right) . \qquad (7.12)
\end{aligned}
$$

We derive two different upper bounds for the expectation. Both always hold, but the best upper bound depends on the value of $\frac{\mu}{2^s}$.

Case 1. This case will give the best bound when $\mu \leq 2^{s-1}$. From (7.7) and (7.9), we obtain

$$
\mathsf{E}\left(Q^{(k,\tilde{m})} \mid \mathrm{Coll}_s = \mathrm{coll}_s\right) \leq q_{on}^{(k,\tilde{m})} + \frac{2q_{\mathit{off}}^{(k)}}{2^s} .
$$

Therefore, using (7.8),

$$
\mathsf{E}\left(\sum_{\tilde{m}=1}^{\tilde{\mu}} \sum_{k=1}^{K} Q^{(k,\tilde{m})} \mid \mathrm{Coll}_s = \mathrm{coll}_s\right) \leq q_{on} + \frac{2\mu}{2^s} q_{\mathit{off}} . \qquad (7.13)
$$

Case 2. This case will give the best upper bound when $\mu \geq 2^{s-1}$. In this setting, the salts are expected to cover a large portion of the space $\{0,1\}^s$, therefore trying to count the offline queries that are useful to the adversary does not give an improved probability. We instead use (7.8) to obtain

$$
\mathsf{E}\left(\sum_{\tilde{m}=1}^{\tilde{\mu}} \sum_{k=1}^{K} Q^{(k,\tilde{m})} \mid \mathrm{Coll}_s = \mathrm{coll}_s\right) \leq q_{on} + q_{\mathit{off}} . \qquad (7.14)
$$

Conclusion. Combining (7.6), (7.13) and (7.14) into (7.12) gives

$$\begin{aligned}
&\mathbf{Adv}_{\text{U/Key}[\mathcal{H}]}\left(q_{\mathit{off}}, q_{on}, \mu\right) \\
&\leq \sum_{\text{coll}_s} \frac{2\text{nc}_s}{2^n} \cdot \mathbf{Pr}\left(\text{Coll}_s = \text{coll}_s\right) \left(q_{on} + \min\left\{\frac{2\mu}{2^s}, 1\right\} \cdot q_{\mathit{off}}\right) \\
&\leq \frac{2}{2^n} \cdot \mathsf{E}\left(\text{NC}_s\right) \cdot \left(q_{on} + \min\left\{\frac{2\mu}{2^s}, 1\right\} \cdot q_{\mathit{off}}\right) \\
&\leq \text{mucol}(\mu, 2^s) \cdot \left(\min\left\{\frac{2\mu}{2^s}, 1\right\} \cdot \frac{2q_{\mathit{off}}}{2^n} + \frac{2q_{on}}{2^n}\right) .
\end{aligned}$$

This concludes the proof. □

7.3.3 Composition With Indifferentiable Hash Functions

In this section we study the security of U/Key instantiated with several indifferentiable hash function constructions in the single-user setting, i.e., if $\mu = 1$. Bounds in the multi-user setting can be obtained by using generic single-user to multi-user reductions [Bih02, BBM00], which result in multiplying all bounds by a factor of μ. However, since our goal is to have a first idea of the tightness of the bounds, for simplicity we only consider the single-user setting.

If a hash function is indifferentiable from a random oracle, this means that it behaves as such and that it can replace a random oracle in almost any practical use case. Andreeva et al. [AMP10b] made the composition explicit. Translated to our case, their reduction means that given a hash function construction $\mathcal{H}$, and for any simulator $\mathbf{S}$, the advantage of the adversary in breaking the security game from Algorithm 10 can be upper bounded as follows:

$$\begin{aligned}
&\mathbf{Adv}_{\text{U/Key}[\mathcal{H}]}\left(q_{\mathit{off}}, q_{on}, 1\right) \leq \\
&\qquad \mathbf{Adv}_{\text{U/Key}[\mathcal{RO}]}\left(q_{\mathit{off}}, q_{on}, 1\right) + \mathbf{Adv}^{\text{iff}}_{\mathcal{H},\mathbf{S}}\left(q_{\mathit{off}} + q_{on}\right) , \qquad (7.15)
\end{aligned}$$

where we abuse notation and use $\mathbf{Adv}_{\text{U/Key}[\mathcal{RO}]}\left(q_{\mathit{off}}, q_{on}, 1\right)$ to refer to the advantage of an adversary in breaking Algorithm 10 when the hash function is a random oracle. In other words, this term is bounded in Theorem 7.3.1, and all we need to do is to obtain the indifferentiability bound for the hash function construction $\mathcal{H}$.

Admittedly, looking ahead, this generic reduction is not tight. The reason is that the indifferentiability term $\mathbf{Adv}^{\text{iff}}_{\mathcal{H},\mathbf{S}}(q)$ takes as security parameter the total number of primitive evaluations $q = q_{\mathit{off}} + q_{on}$ rather than its separation into offline and online queries. This already suggests that a direct analysis will likely give a better bound (and we will do so in Sections 7.4 and 7.5). Yet, there is still value in investigating the guaranteed security through composition, which we will do for (prefix-free) Merkle-Damgård in Section 7.3.3.1, the sponge in Section 7.3.3.2, and truncated permutation in Section 7.3.3.3. However, as in this reasoning the indifferentiability bounds will be the dominating factors anyway, we will simplify the result of Theorem 7.3.1 and simply use that

$$\mathbf{Adv}_{\text{U/Key}[\mathcal{RO}]}\left(q_{\mathit{off}}, q_{\mathit{on}}, 1\right) = \tilde{\mathcal{O}}\left(\frac{q}{2^n}\right) . \tag{7.16}$$

7.3.3.1 Merkle-Damgård

Let $b, u, n \in \mathbb{N}^*$ such that $n \leq b$. The Merkle-Damgård (or MD for short) construction operates on top of a compression function $\mathcal{F} : \{0,1\}^u \times \{0,1\}^b \to \{0,1\}^b$. The input message $M \in \{0,1\}^*$ is first injectively padded into u-bit blocks as $m_1 \parallel \cdots \parallel m_\ell$. Let $IV \in \{0,1\}^b$ be a fixed initialization vector. Then, MD computes its output as

$$\text{outer}_n(\mathcal{F}\left(m_\ell, \mathcal{F}\left(m_{\ell-1}, \mathcal{F}\left(\cdots \mathcal{F}\left(m_1, IV\right)\right)\right)\right)) .$$

When n is sufficiently small, the resulting construction (commonly named chop-MD) is indifferentiable [CDMP05, CN08]. However, in the context of U/Key, the number of compression function calls required for each chaining value remains unchanged, so that the inputs to the MD construction form a prefix-free set. It was proven by Coron et al. [CDMP05] that, as long as the inputs are padded in a prefix-free manner, the obtained construction is indifferentiable up to bound

$$\mathbf{Adv}^{\text{iff}}_{\text{PF-MD},\mathbf{S}}\left(q\right) = \tilde{\mathcal{O}}\left(\frac{q^2}{2^b}\right) ,$$

for some simulator $\mathbf{S}$, and where b is the output length of the compression function. The obtained bound is at least as good as that of chop-MD. From (7.15) and (7.16), we obtain that U/Key instantiated with the MD construction achieves the following level of security:

$$\mathbf{Adv}_{\text{U/Key[ChopMD]}}\left(q_{\mathit{off}}, q_{\mathit{on}}, 1\right) = \tilde{\mathcal{O}}\left(\frac{q}{2^n}\right) + \tilde{\mathcal{O}}\left(\frac{q^2}{2^b}\right) .$$

7.3.3.2 Sponge

We consider the sponge construction as defined in Section 3.1.1 with $r = r' = r''$. For completeness, we briefly recall its inner workings. Let $b, c, r \in \mathbb{N}^*$ such that $b = c + r$. The sponge construction operates on top of a permutation $\mathcal{P} : \{0,1\}^b \to \{0,1\}^b$. The input message $M \in \{0,1\}^*$ is first injectively padded into r-bit blocks as $m_1 \parallel \cdots \parallel m_{\ell'}$, under the condition that the last block is non-zero. Let $IV \in \{0,1\}^b$ be a fixed initialization vector. The sponge absorbs its message blocks as

$$S = (m_{\ell'} \parallel 0^c) \oplus \mathcal{P}\Big((m_{\ell'-1} \parallel 0^c) \oplus \mathcal{P}\big(\cdots \oplus \mathcal{P}\left((m_1 \parallel 0^c) \oplus IV\right)\big)\Big),$$

and squeezes its n-bit digest r bits at a time as

$$\mathrm{outer}_n(\mathrm{outer}_r(\mathcal{P}(S)) \parallel \mathrm{outer}_r(\mathcal{P}^2(S)) \parallel \cdots \parallel \mathrm{outer}_r(\mathcal{P}^\ell(S)))\,,$$

where $\ell = \lceil \frac{n}{r} \rceil$.

The sponge has been proven to be indifferentiable from a random oracle up to bound [BDPV08]

$$\mathbf{Adv}^{\mathrm{iff}}_{\mathrm{Sponge},\mathbf{S}}(q) = \tilde{\mathcal{O}}\left(\frac{q^2}{2^c}\right)\,,$$

for some simulator $\mathbf{S}$. From (7.15) and (7.16), we obtain that U/Key instantiated with the sponge construction achieves the following level of security:

$$\mathbf{Adv}_{\mathrm{U/Key[Sponge]}}(q_{\mathit{off}}, q_{\mathit{on}}, 1) = \tilde{\mathcal{O}}\left(\frac{q}{2^n}\right) + \tilde{\mathcal{O}}\left(\frac{q^2}{2^c}\right)\,.$$

In particular, one cannot have a better security than half of the permutation size.

With $t = 32$, and a security goal of 128 bits (thus $n \geq 128$), in the single-user case the minimal permutation size is 257 bits, but this requires at least 288 primitive evaluations to compute one element on the chain as it processes with rate $r = 1$. Taking a slightly larger primitive size, such as $b = 320$ of Ascon-Hash [DEMS21a, DEMS21b], allows for a rate of $r = 64$ bits, and one requires 5 permutation calls per hash function evaluation (3 for absorbing, and 2 for squeezing). If we extend the analysis to multiple users and to a salt of size 80 bits, the numbers scale to 368 primitive evaluations, or $4 + 2$ primitive evaluations, respectively.

7.3.3.3 Truncated Permutation

As the hash function is typically used for relatively small data, one might instead consider the truncated permutation construction. As we will see now, this significantly improves the security bound.

Let $b \in \mathbb{N}^*$ such that $b \geq n+s+t$. The truncated permutation construction operates on top of a permutation $\mathcal{P} : \{0,1\}^b \to \{0,1\}^b$. Due to the condition that $b \geq n+s+t$, we can absorb all data in one permutation call. In detail, for an arbitrary message $M \in \{0,1\}^{n+s+t}$, the truncated permutation computes its output as

$$\mathrm{outer}_n(\mathcal{P}(M \parallel 0^{b-n-s-t}))\,.$$

Choi et al. [CLL19] showed that the truncated permutation construction for fixed salt (i.e., in the single-user setting) is indifferentiable from a random oracle up to bound

$$\mathbf{Adv}^{\mathrm{iff}}_{\mathrm{TruncP}}(q) = \tilde{\mathcal{O}}\left(\frac{q^{3/2}}{2^{\frac{2b-n}{2}}} + \frac{q}{2^{b-(n+t)}}\right)\,.$$

(In the multi-user setting, when there are multiple random salts, one can rely on the result of Grassi and Mennink [GM22].) From (7.15) and (7.16), we obtain that U/Key instantiated with the truncated permutation construction achieves the following level of security:

$$\mathbf{Adv}_{\mathrm{U/Key[TruncP]}}(q_{\mathit{off}}, q_{\mathit{on}}, 1) = \tilde{\mathcal{O}}\left(\frac{q}{2^n}\right) + \tilde{\mathcal{O}}\left(\frac{q^{3/2}}{2^{\frac{2b-n}{2}}} + \frac{q}{2^{b-(n+t)}}\right)\,.$$

Observe that, comparing the bounds of Section 7.3.3.2 and 7.3.3.3, using a truncated permutation typically gives a better choice than the sponge whenever $n + t \leq \frac{b}{2}$, since this choice allows to minimize the number of primitive calls and maximizes the security at the same time.

With $t = 32$, and a security goal of 128 bits (thus $n \geq 128$), the minimal permutation size is 288 bits and a single permutation evaluation is made. In other words, for the current use case, a truncated permutation is superior over the sponge whenever the used permutation is at least 288 bits. If one uses a smaller permutation, one cannot achieve 128-bit security using the truncated permutation construction, while it may still be possible using the sponge (provided $b \geq 257$) at the cost of extra permutation evaluations.

7.4 Dedicated Security Proof of U/Key with a Sponge

From the results of Section 7.3.3 it is clear that indifferentiability is a too strong security property for all of these constructions. For the case of the sponge, on the one hand its indifferentiability bound is undeniably tight: as discussed in Section 3.1.2, inner collisions within the sponge states can be found in approximately $2^{c/2}$ queries, and those inner collisions can be used among others to find collisions and second preimages. On the other hand, we showed in Theorem 4.3.1 that finding a preimage (according to Definition 2.4.1) of a value cannot be done in less than 2^{n-r} queries. In particular, when $n - r > c/2$, this gives a better bound than indifferentiability, thus suggesting that U/Key could as well benefit from this security bound. However, preimage resistance is not the exact security property that we are seeking. Firstly, this is because the target value comes itself from a previous sponge call, which does not correspond to the setting of everywhere preimage.[3] Additionally, we are not aware of results that account for the singular nature of the cascaded hash function evaluations or the release pattern of the chaining values. Therefore, a dedicated security proof seems unavoidable.

The remainder of this section is organized as follows. Section 7.4.1 describes the exact scheme that we study and the rationale behind it. Section 7.4.2 defines a variant of PRF security needed for our proof. Finally, Section 7.4.3 is dedicated to the main result of this section, and Sections 7.4.4 and 7.4.5 to the corresponding proof.

7.4.1 Optimization of Sponge-Based Instantiation

According to the description of U/Key, the chaining value, the counter, as well as the salt are seen as inputs to the hash function. In the context of a sponge (cf., Section 7.3.3.2), this means that the string $\langle \mathrm{ctr}_{K-k+1} \rangle_t \parallel id \parallel x_{K-k}$ is padded into blocks of r bits, and each of these blocks is added one by one to the outer part of the state of the sponge. For a typical instantiation of U/Key, where the counter and salt can be of total size 112 bits, this induces a significant overhead regarding the number of required permutation evaluations if the rate is small. On the other hand, from a security perspective, this overhead appears to be unnecessary. Indeed, from the perspective of the security proof, there is no distinction between absorbing the counter and the salt into the

[3]However, in Section 4.4.2, we extended the result to the setting where a random preimage is guaranteed to exist, which gets closer to this use case.

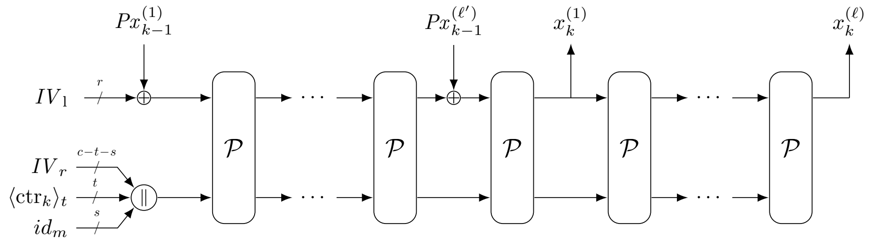


Figure 7.1: U/Key with a tweaked version of the sponge. The figure illustrates the computation of $x_k = \mathrm{outer}_n(x_k^{(1)} \parallel \cdots \parallel x_k^{(\ell)})$ from x_{k-1}. Here, $Px_{k-1}^{(1)} \parallel \cdots \parallel Px_{k-1}^{(\ell')} = pad(x_{k-1})$, where $\ell \leq \ell'$, and $IV := IV_l \parallel IV_r$ is fixed.

sponge by blocks of r bits or directly initializing the state with these values. Strictly speaking, in the former case, the initial states are randomized, but this modification does not affect the proof.

Therefore, we consider a family of sponges with initial states of the form $IV_{k,id}$, where

$$IV_{k,id} = IV \parallel \langle \mathrm{ctr}_{K-k+1} \rangle_t \parallel id_m \,,$$

and $IV \in \{0,1\}^{b-t-s}$ can be any value fixed in advance. Note that this is possible only if the capacity is large enough (i.e., $c \geq t + s$). However, looking forward to the main result in this section, namely Theorem 7.4.1, for a typical use case of $t = 32, s = 80$, a capacity less than 112 bits does not provide a decent level of security anyway. One round of our refined scheme is depicted in Figure 7.1. For simplicity, we consider the simple 10*-padding, which appends to the message a 1 and as much 0's as necessary to obtain a padded message with a length multiple of r. Therefore, the number of rounds necessary to absorb the ℓ message blocks is equal to ℓ or $\ell + 1$.

Note that the construction that is instantiated in U/Key is similar to the sponge variant as used in PHOTON [GPP11], where the initial absorbing rate r'' can also be larger (in our case, $r'' = r + t + s$). However, that construction keeps an indifferentiability result comparable to the sponge as long as $r'' < r + c/2$ [NO14], but we will show that even beyond this threshold security is still achieved. We stress that the security bound obtained in Theorem 7.4.1 holds both for this optimized version of the sponge and the plain sponge construction. In the remainder of this section, we will abuse notation and keep calling this optimized construction the sponge.

7.4.2 Key-Reveal PRF Security

Looking ahead, the main tool used in the proof will be the PRF security of the outer-keyed sponge (OKS), which was described in Section 3.2.1. In fact, we will rely on a slight variant of PRF security, namely key-reveal PRF security. The key-reveal PRF security game is exactly as the PRF security game, except that the key is revealed at the end of the interaction, right before the distinguisher outputs its decision bit (this will be a dummy key in the ideal world). Consequently, key-reveal PRF security is strictly stronger than PRF security, as the adversary can opt to ignore the obtained information. While introducing (yet) another security definition might initially appear perplexing, it will later be essential for the reduction made in the proof of Theorem 7.4.1. Moreover, a closer examination of the actual PRF security proof of the outer-keyed sponge [ADMV15, NY16, Men18] reveals an interesting aspect: *the authors actually established key-reveal PRF security of the sponge.* This phenomenon arises because, in proofs utilizing techniques such as the H-coefficient technique [Pat08b], the keys are often assumed to be revealed at the end of the interaction for the sake of simplicity.

That said, we now require a formal definition of key-reveal PRF security tailored to OKS. Let $\mathcal{RO}^*$ be a random oracle, $\mathcal{K} \xleftarrow{\$} \{0,1\}^\kappa$, and $\mathcal{P} \xleftarrow{\$} \texttt{Perm}(b)$. The adversary is given access to three oracles:

- $\overline{\mathcal{K}}_{\mathcal{K}}$: when queried, it returns $\mathcal{K}$;
- $\overline{\mathcal{C}}^{\mathcal{C}}$ for $\mathcal{C} \in \{\mathrm{OKS}_{\mathcal{K}}^{\mathcal{P}}, \mathcal{RO}^*\}$: if $\overline{\mathcal{K}}_{\mathcal{K}}$ has not been queried yet, this oracle relays the query to $\mathcal{C}$, otherwise, it returns $\perp$;
- $\overline{\mathcal{P}}^{\mathcal{P}}$: if $\overline{\mathcal{K}}_{\mathcal{K}}$ has not been queried yet, this oracle relays the query to $\mathcal{P}$, otherwise, it returns $\perp$.

The key-reveal PRF advantage of an adversary $\mathcal{A}$ against OKS is defined as follows:

$$\mathbf{Adv}_{\mathrm{OKS}}^{\text{prf-krev}}(\mathcal{A}) = \left|\mathbf{Pr}\left(\mathcal{A}\left[\overline{\mathcal{K}}_K, \overline{\mathcal{C}}^{\mathrm{OKS}_{\mathcal{K}}^{\mathcal{P}}}, \overline{\mathcal{P}}^{\mathcal{P}}\right] = 1\right) - \mathbf{Pr}\left(\mathcal{A}\left[\overline{\mathcal{K}}_{\mathcal{K}}, \overline{\mathcal{C}}^{\mathcal{RO}^*}, \overline{\mathcal{P}}^{\mathcal{P}}\right] = 1\right)\right| .$$

$\mathbf{Adv}_{\mathrm{OKS}}^{\text{prf-krev}}(\mathcal{M}, \mathcal{N})$ denotes the maximum of $\mathbf{Adv}_{\mathrm{OKS}}^{\text{prf-krev}}(\mathcal{A})$, among all adversaries allowed to make at most $\mathcal{N}$ permutation queries and construction queries with a cost of $\mathcal{M}$. (The cost of construction queries is determined in terms of the total number of primitive calls induced by construction calls with OKS, as detailed in Section 3.2.1.2.)

7.4.3 Security Result

Now, we state the security of U/Key instantiated with the sponge construction in Theorem 7.4.1. At a high level, the security proof relies on several key observations. First, the random secret password x_0 remains undisclosed to the adversary throughout the process. This enables us to treat the evaluation of the sponge on x_0 as an outer-keyed sponge. Consequently, we can replace x_1 with a random value by leveraging the PRF advantage of the outer-keyed sponge. By repeating this process K times, we find ourselves with K instances of the PRF security of the outer-keyed sponge and K instances of a variant of the domain-oriented preimage of the sponge (cf., Definition 2.4.2). These variants involve access to chaining values following the release pattern of hash chain protocols. Moreover, in each of these variants, we can use the key-reveal PRF advantage, which allows to conclude that, during most of the attack phase, the adversary does not learn any information about the target to invert. We consider the single-user setting. Given the proof technique that we currently employ, it does not seem that considering a multi-user setting would improve over a generic single-user to multi-user reduction [Bih02, BBM00].

Theorem 7.4.1. *Let* Sponge *denote the construction from Section 7.4.1. Assuming that $(\ell-1)^2 < 2^b$, and $q_{off} + q_{on} + (\ell+\ell')K \leq 2^c/6$, we have*

$$\mathbf{Adv}_{\text{U/Key[Sponge]}}\left(q_{off}, (q_{on,k})_{k\in[\![1,K]\!]}, 1\right) \leq \sum_{k=1}^{K} \left(\mathbf{Adv}^{\text{prf}}_{\text{OKS}}(\ell'+\ell, q_k) + \frac{8\ell q_k}{2^n} + \min\left\{\frac{4\ell q_{on,k}}{2^{n-r}}, \frac{2q_{on,k}\cdot q_k}{2^c}\right\} + \mathbf{Adv}^{\text{prf-krev}}_{\text{OKS}}(\ell+\ell', 2\ell(q_k - q_{on,k})) \right),$$

where $q_k = (\ell+\ell')(K-k) + q_{off} + \sum_{k'\geq k} q_{on,k'}$. The result also holds when considering the plain sponge construction from Section 7.3.3.2.

The proof is given in Section 7.4.4.

Interpretation of the Bound. We focus on the setting where all online phases have the same cost (i.e., $\forall k, k', q_{on,k} = q_{on,k'}$). In this case, the bound simplifies to

$$\mathbf{Adv}_{\text{U/Key[Sponge]}}(q_{off}, q_{on}, 1) \leq \frac{8\ell K q_1}{2^n} + \min\left\{\frac{4\ell q_{on}}{2^{n-r}}, \frac{2q_{on}q_1}{2^c}\right\} + 2K\cdot\mathbf{Adv}^{\text{prf-krev}}_{\text{OKS}}(\ell+\ell', 2\ell q_1).$$

Moreover, as discussed in Section 3.2.1.2, assuming that $\lceil \frac{n}{r} \rceil$ is a small constant, we have

$$\mathbf{Adv}^{\text{prf-krev}}_{\text{OKS}}(\mathcal{M}, \mathcal{N}) = \tilde{\mathcal{O}}\left(\frac{\mathcal{M}\mathcal{N}}{2^c} + \frac{\mathcal{N}}{2^n}\right).$$

If we additionally assume that $\ell K \ll q_{\mathit{off}} + q_{\mathit{on}}$, we obtain

$$\mathbf{Adv}_{\text{U/Key[Sponge]}}(q_{\mathit{off}}, q_{\mathit{on}}, 1) = \tilde{\mathcal{O}}\Bigg(\frac{K\ell(q_{\mathit{off}} + q_{\mathit{on}})}{2^n} + \frac{K\ell^2(q_{\mathit{off}} + q_{\mathit{on}})}{2^c} + \min\left\{\frac{\ell q_{\mathit{on}}}{2^{n-r}}, \frac{(q_{\mathit{off}} + q_{\mathit{on}})q_{\mathit{on}}}{2^c}\right\}\Bigg). \quad (7.17)$$

Consider the typical parameters for T/Key, so that $K \leq 2^{21}$, and suppose we aim for 100 bits of security in the online phase, and 128 bits in the offline phase. Because of the first term in (7.17), this implies $n \geq 149$. Consider the Spongent [BKL+11] permutation of size $b = 176$. With $n = 150, c = 150$, we have $r = 26, \ell = \ell' = 6$, and the security requirements are met in the single-user setting.

We note that if the permutation is larger than 200 bits, any sponge evaluation in the context of U/Key consists of only one permutation call. In this case, it makes more sense to look at U/Key instantiated with a truncated permutation, and we do so in Section 7.5.

7.4.4 Proof of Theorem 7.4.1

Denote by $x_0, \ldots, x_K \in \{0,1\}^n$ the chaining values, and id the salt of the user. Let $\mathcal{A}$ be an adversary for the security game of Algorithm 10, allowed to make at most q_{off} offline queries and a total of q_{on} queries during the online phase, where $q_{\mathit{on},k}$ queries are allowed after the release of x_k and before the release of x_{k-1}. For simplicity, we assume that after each release of x_k, the primitive queries that transition from x_k to x_{k+1} are given for free to the adversary. This gives a total number of queries equal to $q_{\mathit{on},1}$.

We will use a hybrid argument with K distinct security games. For each $k \in [\![0, K]\!]$, we define S_k to be the following sampling procedure:

- $\forall k' \leq k, x_{k'} \xleftarrow{\$} \{0,1\}^n$,
- $\forall k' > k, x_{k'} \leftarrow \text{Sponge}^{\mathcal{P}}_m(IV_{\text{ctr}_{k'}, id}, x_{k'-1})$,

where by abuse of notation $\text{Sponge}_m^{\mathcal{P}}(IV, \cdot)$ denotes the sponge with its state initialized with IV. Note that S_0 corresponds to the original sampling procedure used in the security game. For $k \in [\![1, K]\!]$, we define $G(k)$ to be the security game described in Algorithm 10, but it is aborted when the counter equals k in Section 7.2.2. In other words, $G(k)$ stops right before x_{k-1} is given to the adversary. In particular, in $G(k)$, the adversary has no opportunities to provide a preimage for $x_{k'}$ for any $k' < k$. Note that, according to this formalism, $G(1)$ corresponds to the full security game.

In the proof, we will transition through these different security games, but the chaining values may not necessarily follow the sampling procedure S_0. The specific sampling procedure will be indicated as a subscript in the probabilities.

Initial Step. We start with the first step of our reasoning:

$$\begin{aligned}&\mathbf{Adv}_{\text{U/Key[Sponge]}}(\mathcal{A}) = \mathbf{Pr}_{S_0}(\mathcal{A} \text{ wins } G(1)) \\ &\leq |\mathbf{Pr}_{S_0}(\mathcal{A} \text{ wins } G(1)) - \mathbf{Pr}_{S_1}(\mathcal{A} \text{ wins } G(1))| + \mathbf{Pr}_{S_1}(\mathcal{A} \text{ wins } G(1)) \ .\end{aligned} \tag{7.18}$$

The only difference between S_0 and S_1 is in the way in which x_1 is sampled: either randomly or via an evaluation of the sponge. From $\mathcal{A}$, we build a distinguisher $\mathcal{A}'_1$ which returns 1 whenever $\mathcal{A}$ wins the game $G(1)$. Since x_0 is never revealed to the adversary, it can be seen as a secret key. Therefore, $\mathcal{A}'_1$ is a distinguisher in the PRF security game of the outer-keyed sponge initialized with the state $IV_{1,id}$. Its resources are upper bounded by $\ell' + \ell$ construction queries and q_1 primitive queries. Therefore, we obtain from (7.18):

$$\mathbf{Adv}_{\text{U/Key[Sponge]}}(\mathcal{A}) \leq \mathbf{Adv}_{\text{OKS}}^{\text{prf}}(\ell' + \ell, q_1) + \mathbf{Pr}_{S_1}(\mathcal{A} \text{ wins } G(1)) \ . \tag{7.19}$$

Inductive Reasoning. We proceed with the inductive step to upper bound the term $\mathbf{Pr}_{S_k}(\mathcal{A} \text{ wins } G(k))$ for any $k \in [\![1, K-1]\!]$. Recall that in S_k, the values $x_1, \ldots, x_k$ are sampled uniformly at random, and the remaining chaining values are derived from the hash chain protocol with the root x_k. In particular, in $G(k)$, the adversary is not able to see any inconsistency in the fact that x_k is not the image of x_{k-1} by a sponge call, and finding a preimage of any of these earlier $x_{k'}$'s does not trigger a win.

This means that there are two possibilities for the adversary to win $G(k)$: either $\mathcal{A}$ finds a preimage for x_k by the sponge with the initial state fixed to $IV_{k,id}$, which we abbreviate as "$\mathcal{A}$ solves pre(x_k)", or $\mathcal{A}$ wins $G(k+1)$. Indeed, if the adversary wins $G(k)$ without discovering a preimage for x_k, it indicates that it was already successful in the game before x_k was revealed, thus during $G(k+1)$. We thus have:

$$\begin{aligned}
&\mathbf{Pr}_{S_k}\left(\mathcal{A} \text{ wins } G(k)\right) \\
&\leq \mathbf{Pr}_{S_k}\left(\mathcal{A} \text{ solves pre}(x_k) \text{ in } G(k)\right) + \mathbf{Pr}_{S_k}\left(\mathcal{A} \text{ wins } G(k+1)\right) \\
&\leq \mathbf{Pr}_{S_k}\left(\mathcal{A} \text{ solves pre}(x_k) \text{ in } G(k)\right) \\
&+ \left|\mathbf{Pr}_{S_k}\left(\mathcal{A} \text{ wins } G(k+1)\right) - \mathbf{Pr}_{S_{k+1}}\left(\mathcal{A} \text{ wins } G(k+1)\right)\right| \\
&+ \mathbf{Pr}_{S_{k+1}}\left(\mathcal{A} \text{ wins } G(k+1)\right) .
\end{aligned}$$

Then, from $\mathcal{A}$ we can build a distinguisher $\mathcal{A}'_{k+1}$ returning 1 if and only if $\mathcal{A}$ wins $G(k+1)$. Because S_k and S_{k+1} only differ in the sampling of x_{k+1}, and x_k is never given to the adversary in $G(k+1)$, we can again use the PRF advantage of the sponge. The resources of $\mathcal{A}'_{k+1}$ are upper bounded by $\ell' + \ell$ construction queries and q_{k+1} primitive queries. Thus

$$\begin{aligned}
\mathbf{Pr}_{S_k}\left(\mathcal{A} \text{ wins } G(k)\right) \leq \mathbf{Pr}_{S_k}\left(\mathcal{A} \text{ solves pre}(x_k) \text{ in } G(k)\right) \\
+ \mathbf{Adv}^{\text{prf}}_{\text{OKS}}\left(\mathcal{A}'_{k+1}\right) + \mathbf{Pr}_{S_{k+1}}\left(\mathcal{A} \text{ wins } G(k+1)\right) . \quad (7.20)
\end{aligned}$$

We upper bound the quantity $\mathbf{Pr}_{S_k}\left(\mathcal{A} \text{ solves pre}(x_k) \text{ in } G(k)\right)$ for any $k \in [\![1, K]\!]$ in Lemma 7.4.2.

Lemma 7.4.2. *If $(\ell-1)^2 < 2^b$, and $q_k \leq 2^c/6$, we have*

$$\begin{aligned}
\mathbf{Pr}_{S_k}\left(\mathcal{A} \textit{ solves pre}(x_k) \textit{ in } G(k)\right) \leq \frac{8\ell q_k}{2^n} + \min\left\{\frac{4\ell q_{on,k}}{2^{n-r}}, \frac{2q_{on,k}\cdot q_k}{2^c}\right\} \\
+ \mathbf{Adv}^{\text{prf-krev}}_{\text{OKS}}(\ell+\ell', 2\ell(q_k - q_{on,k})) .
\end{aligned}$$

The full proof follows the approach that we used in Chapter 4, and is given in Section 7.4.5.

Now, we can plug the bound derived from Lemma 7.4.2 into (7.20) and obtain

$$\begin{aligned}
\mathbf{Pr}_{S_k}\left(\mathcal{A} \text{ wins } G(k)\right) \leq \frac{8\ell q_k}{2^n} + \min\left\{\frac{4\ell q_{on,k}}{2^{n-r}}, \frac{2q_{on,k}\cdot q_k}{2^c}\right\} + \mathbf{Adv}^{\text{prf}}_{\text{OKS}}(\mathcal{A}'_{k+1}) \\
+ \mathbf{Adv}^{\text{prf-krev}}_{\text{OKS}}(\ell+\ell', 2\ell(q_k - q_{on,k})) + \mathbf{Pr}_{S_{k+1}}\left(\mathcal{A} \text{ wins } G(k+1)\right) . \quad (7.21)
\end{aligned}$$

Conclusion. We remark that, eventually, $\mathbf{Pr}_{S_K}\left(\mathcal{A} \text{ wins } G(K)\right)$ is equal to $\mathbf{Pr}_{S_K}\left(\mathcal{A} \text{ solves pre}(x_K) \text{ in } G(K)\right)$. Therefore, by applying the steps taken in

(7.20) and (7.21) inductively, we obtain from (7.19):

$$\begin{aligned}
&\mathbf{Adv}_{\mathrm{U/Key[Sponge]}}(\mathcal{A}) \\
&\leq \sum_{k=1}^{K} \mathbf{Pr}_{S_k}(\mathcal{A} \text{ solves } \mathrm{pre}(x_k) \text{ in } G(k)) + \sum_{k=1}^{K} \mathbf{Adv}_{\mathrm{OKS}}^{\mathrm{prf}}(\ell' + \ell, q_k) \\
&\leq \sum_{k=1}^{K} \Bigg(\frac{8\ell q_k}{2^n} + \min\left\{ \frac{4\ell q_{on,k}}{2^{n-r}}, \frac{2q_{on,k} \cdot q_k}{2^c} \right\} + \\
&\qquad \mathbf{Adv}_{\mathrm{OKS}}^{\text{prf-krev}}(\ell + \ell', 2\ell(q_k - q_{on,k})) + \mathbf{Adv}_{\mathrm{OKS}}^{\mathrm{prf}}(\ell' + \ell, q_k) \Bigg) .
\end{aligned}$$

This concludes the proof. □

7.4.5 Proof of Lemma 7.4.2

In this section, we provide a self-contained proof of Lemma 7.4.2. The proof closely follows the one of Chapter 4. Lemma 7.4.2 aims to bound the probability that an adversary finds a preimage for x_k in $G(k)$. Remember that, for $k' < K$, when $x_{k'}$ is given to the adversary, the permutation evaluations that allow to compute $x_{k'+1}$ from $x_{k'}$ are given for free to the adversary. Therefore, in the game $G(k)$, the adversary was granted a total of $q^{(k)} = (\ell + \ell')(K - k) + q_{\mathit{off}} + \sum_{k' \geq k} q_{on}^{(k')}$ queries. We remark that the phase *before* the release of x_k is basically the offline phase for the particular security game $G(k)$. We can therefore distinguish between three different stages for the game $G(k)$:

- The offline phase, during which the adversary can make q_{off} queries;
- The online phase up to the release of x_k (free queries included). During that phase, the adversary can make $(\ell + \ell')(K - k) + \sum_{k' > k} q_{on}^{(k')}$ queries;
- The online phase after the release of x_k and before the release of x_{k+1} during which the adversary can make $q_{on,k}$ queries.

We refer to the two first stages as the k-offline phase, and the last as the k-online phase.

Setup. We start by establishing the notation used in this proof. In the probabilities, when no subscript is mentioned, the distribution S_k is implicitly used. Let $0 < s \leq r$ be such that $n = (\ell - 1) \times r + s$, or in words, s is the length of x_k^ℓ. The adversarial query history is denoted as $\mathcal{Q}$. It contains tuples of the form (X, Y, d), indicating that $\mathcal{P}(X) = Y$ and that the query was made in the direction $d \in \{\mathit{fwd}, \mathit{inv}\}$. Given a bad event BAD, for any $i \in [\![1, q_k]\!]$, BAD_i denotes that BAD is triggered after the first i queries. Finally, if E is an event, $\mathbf{1}_E$ is the Bernoulli variable equal to 1 if and only if E occurs.

In order to find a preimage, the adversary must complete two steps: i) Find a well-formed state such that its cascade of $\ell - 1$ consecutive permutation evaluations produces the outer parts that match exactly the components $x_k^{(l)}$; ii) Connect this state to $IV_{k,id}$ with message blocks. Let $\mathcal{S}$ be the following set:

$$\mathcal{S} = \left\{ y \in \{0,1\}^b \mid \forall l \in [\![1, \ell]\!], \operatorname{outer}_r(\mathcal{P}^l(y)) = x_k^{(l)} \right\} .$$

Moreover, define $\mathcal{S}[l]$ to be $\mathcal{P}^l(\mathcal{S})$. In words, $\mathcal{S}$ captures all the states right before the squeezing phase which cascade successively to produce the desired ℓ outer parts, and $\mathcal{S}[l]$ specifically captures the states associated with squeezing $x_k^{(l)}$. Since $\mathcal{P}$ is a permutation, these sets all have the same size. Moreover, given $0 \leq a \leq b \leq \ell$, $\mathcal{S}[a : b]$ is the multi-set equal to $\bigcup_{l=a}^{b} \mathcal{S}[l]$.

Event Splitting. If the adversary manages to find a preimage for x_k, this implies that there exist $a_1, \ldots, a_{\ell'} \in \{0,1\}^r$, $N_0, \ldots N_{\ell'}, \in \{0,1\}^b$, and $d_1, \ldots, d_{\ell'} \in \{\mathit{fwd}, \mathit{inv}\}$ such that

- $N_0 = IV_{k,id}$;
- $\forall i \in [\![1, \ell']\!], (N_{i-1} \oplus (a_i \parallel 0^c), N_i, d_i) \in Q$;
- $N_{\ell'} \in \mathcal{S}[1]$.

We denote this bad event by PRE. At this stage, we dropped the requirement on the message blocks to correspond to a valid padding. In order to avoid the complexity of reasoning about the entire graph, we can focus on specific trigger points. The pivot taken here is the direction of the query $(N_{\ell'-1} \oplus (a_{\ell'} \parallel 0^c), N_{\ell'}, d_{\ell'})$.

Let BADFWD and, for $l \in [\![1, \ell]\!]$, BADINV^l be defined as follows:

$$\begin{aligned} \mathsf{BADFWD} &: \exists (X, Y, \mathit{fwd}) \in \mathcal{Q} \text{ such that } X \in \mathcal{S}[0] \,, \\ \mathsf{BADINV}^l &: \exists (S_{\mathit{fwd}}, S_{\mathit{inv}}, d) \in \mathcal{Q} \text{ such that } S_d \in \mathcal{S}[l] \,. \end{aligned}$$

Moreover, let $\mathsf{BADINV} := \bigvee_{l=1}^{\ell} \mathsf{BADINV}^{l}$. BADFWD (resp., BADINV) corresponds to the pivot query made in the forward (resp., inverse) direction. Intuitively, with BADFWD, the adversary cannot freely choose the outer parts of the states it queries. On the other hand, if BADINV is set during the k-online phase, the adversary can freely choose the outer parts of the query it makes, allowing it to set this event with higher probability. However, in the k-offline phase, the adversary additionally needs to guess the outer parts.

Now, if the adversary finds a well-formed state through BADINV, it must further connect this state to the initial state of the sponge $IV_{k,id}$ *after having triggered* BADINV. To capture that, we will consider a slightly more complicated version of bad event INNER from Chapter 4. More precisely, we parametrize INNER by a query index i. This enforces that a fresh inner collision should have been found starting from the query i, and previously found inner collisions do not play a role. This parametrization proves to be valuable, as triggering BADINV during the k-online phase has a lower success probability than during the k-offline phase. $\mathsf{INNER}(i)$ is defined as follows:

$$\begin{aligned}\mathsf{INNER}(i) : \exists (X, Y, \mathit{fwd}) \in \mathcal{Q}, (X', Y', \mathit{inv}) \in \mathcal{Q}[i : q] \\ \text{such that } \mathrm{inner}_c(Y) = \mathrm{inner}_c(X') \,.\end{aligned}$$

In Chapter 4, we argued the following splitting:

$$\mathsf{PRE} \implies \mathsf{BADFWD} \vee (\mathsf{BADINV} \wedge \mathsf{INNER}(1)) \,,$$

which led to the following bound:

$$\mathbf{Pr}\left(\mathsf{PRE}\right) \leq \mathbf{Pr}\left(\mathsf{BADFWD}\right) + \min\left\{\mathbf{Pr}\left(\mathsf{INNER}(1)\right), \mathbf{Pr}\left(\mathsf{BADINV}\right)\right\} \,.$$

However, this approach is too coarse for our setting, and we need to further split the events to take into account the different phases. More precisely, in order to trigger PRE, there are three possibilities:

- The adversary triggers BADFWD at some point;
- The adversary triggers BADINV during the k-offline phase, and $\mathsf{INNER}(1)$ at some point;
- The adversary triggers BADINV and finds inner collisions, both during the k-online phase.

Therefore,

$$\begin{aligned}\mathbf{Pr}\left(\mathsf{PRE}\right) \leq{}& \mathbf{Pr}\left(\mathsf{BADFWD}\right) + \min\left\{\mathbf{Pr}\left(\mathsf{BADINV}_{q_k - q_{on,k}}\right), \mathbf{Pr}\left(\mathsf{INNER}(1)\right)\right\} \\ &+ \min\left\{\mathbf{Pr}\left(\mathsf{BADINV} \wedge \neg\mathsf{BADINV}_{q_k - q_{on,k}}\right), \mathbf{Pr}\left(\mathsf{INNER}(q_k - q_{on,k})\right)\right\} \,,\end{aligned}$$

where we remind that BAD_i denotes that BAD is triggered after the first i queries; while $\mathsf{INNER}(i)$ denotes that a new inner collision has been found starting from the i^{th} query. Looking ahead, $\mathbf{Pr}\left(\mathsf{BADINV}_{q_k - q_{on,k}}\right)$ will have a bound similar to $\mathbf{Pr}\left(\mathsf{BADFWD}\right)$. This allows us to eliminate the term involving $\mathsf{INNER}(1)$, resulting in the following expression:

$$\begin{aligned}\mathbf{Pr}\left(\mathsf{PRE}\right) \leq \mathbf{Pr}\left(\mathsf{BADFWD}\right) &+ \mathbf{Pr}\left(\mathsf{BADINV}_{q_k - q_{on,k}}\right) \\ &+ \min\left\{\mathbf{Pr}\left(\mathsf{BADINV} \wedge \neg\mathsf{BADINV}_{q_k - q_{on,k}}\right), \mathbf{Pr}\left(\mathsf{INNER}(q_k - q_{on,k})\right)\right\} .\end{aligned} \tag{7.22}$$

Now, we can evaluate the probabilities individually. Intuitively, for the events BADFWD and BADINV, we adopt the same approach as in Chapter 4, which involves conditioning on $|\mathcal{S}|$. The main novelty in our approach consists of showing that setting BADINV during the k-offline phase is (almost) as hard as setting BADFWD.

Probability of BADFWD. We have

$$\begin{aligned}\mathbf{Pr}\left(\mathsf{BADFWD}\right) &\leq \sum_{i=1}^{q_k}\sum_{y=1}^{2^c} \mathbf{Pr}\left(\mathsf{BADFWD}_i \wedge \neg\mathsf{BADFWD}_{i-1} \wedge |S[0]| = y\right) \\ &\leq \sum_{i=1}^{q_k}\sum_{y=1}^{2^c} \mathbf{Pr}\left(\mathsf{BADFWD}_i \;\middle|\; \neg\mathsf{BADFWD}_{i-1} \wedge |S[0]| = y\right) \times \\ &\qquad\qquad \mathbf{Pr}\left(|S[0]| = y\right) .\end{aligned}$$

Triggering BADFWD is similar to a guessing game. The adversary must make a query such that the answer lies in the set $\mathcal{S}[1]$. As explained in more detail in Chapter 4, the values in $\mathcal{S}[0]$ are defined via inverse $\mathcal{P}$-calls, thus random. Moreover, one failed attempt with the query (X, Y, fwd) from the adversary only removes X from the set of possibilities. Therefore,

$$\begin{aligned}\mathbf{Pr}\left(\mathsf{BADFWD}\right) &\leq \sum_{i=1}^{q_k}\sum_{y=1}^{2^c} \frac{y}{2^b - (i-1)} \mathbf{Pr}\left(|S[0]| = y\right) \\ &\leq \frac{2q_k}{2^b} \mathsf{E}\left(|S[0]|\right) ,\end{aligned} \tag{7.23}$$

where we used that $q_k \leq 2^{b-1}$.

Probability of $\mathsf{BADINV} \wedge \neg\mathsf{BADINV}_{q_k - q_{on,k}}$. Note that we can assume that $\ell > 1$, otherwise this event can be set with probability 1 and the probability will not dominate the "min". Again, by basic probability, we have

$$\begin{aligned}
&\mathbf{Pr}\left(\mathsf{BADINV} \wedge \neg\mathsf{BADINV}_{q_k - q_{on,k}}\right)\\
&\le \sum_{i=q_k - q_{on,k}+1}^{q_k} \sum_{y=1}^{2^c} \sum_{l=1}^{\ell} \mathbf{Pr}\left(\mathsf{BADINV}^l[i] \mid \neg\mathsf{BADINV}_{i-1} \wedge |S[1:\ell]| = \ell y\right) \times \\
&\qquad\qquad\qquad\qquad\quad \mathbf{Pr}\left(|S[1:\ell]| = \ell y\right) .
\end{aligned}$$

The idea used in the proof in Chapter 4 for the conditioned BADINV involves a close examination of the paths induced by the elements in $\mathcal{S}[1:\ell]$. Here, we only focus on one fixed outer part at a time for the individual bad events BADINV^l. For simplicity, when the adversary makes a query with input Z, both $\mathcal{P}(Z)$ and $\mathcal{P}^{-1}(Z)$ are assumed to be given to the adversary. Therefore, the adversary wins if $\mathcal{P}^{-1}(Z) \to Z \to \mathcal{P}(Z)$ coincides with a path $Y_{l-1} \to Y_l \to Y_{l+1}$ with $Y_l \in \mathcal{S}[l]$ and $Y_{l\pm1} \in \mathcal{P}^{\pm1}(\mathcal{S}[l])$. $\mathcal{S}[l]$ is a set of size y, and is a subset of a set of size at least 2^c (note that this statement is also valid when $l = \ell$, since $2^{b-s} \ge 2^c$). At a high level, either a query sets BADINV, or it fails, and in the latter case, the only information that the adversary learns is that neither Z, $\mathcal{P}(Z)$, nor $\mathcal{P}^{-1}(Z)$ are in $\mathcal{S}[1:\ell]$. Therefore, for any i, y, and l,

$$\mathbf{Pr}\left(\mathsf{BADINV}^l[i] \mid \neg\mathsf{BADINV}_{i-1} \wedge |S[1:\ell]| = \ell y\right) \le \frac{y}{2^c - 3(i-1)} \le \frac{2y}{2^c} ,$$

where we used that $q_k \le 2^c/6$. Therefore,

$$\begin{aligned}
\mathbf{Pr}\left(\mathsf{BADINV} \wedge \neg\mathsf{BADINV}_{q_k - q_{on,k}}\right) &\le \sum_{i=q_k - q_{on,k}+1}^{q_k} \sum_{y=1}^{2^c} \frac{2\ell y}{2^c} \mathbf{Pr}\left(|S[1:\ell]| = \ell y\right) \\
&\le \frac{2q_{on,k}}{2^c} \mathsf{E}\left(|S[1:\ell]|\right) . \qquad (7.24)
\end{aligned}$$

Probability of $\mathsf{BADINV}_{q_k - q_{on,k}}$. Setting this event implies among others that the adversary wins *before* x_k is released. As a first step, from $\mathcal{A}$, we want to build a distinguisher $\mathcal{D}$ that returns 1 if and only if $\mathcal{A}$ sets BADINV at the end of the k-offline phase. For $\mathcal{D}$ to verify the aforementioned event, we need to provide it additional information at the end of the k-offline phase. As the first piece of additional information, we provide x_k, which can be seen as a key for the outer-keyed sponge during the k-offline phase. Moreover, for every existing

$(X, Y, d) \in \mathcal{Q}[1 : q_k - q_{on,k}]$, we give to $\mathcal{D}$ the queries $(\mathcal{P}^l(Y), \mathcal{P}^{l+1}(Y))$, and $(\mathcal{P}^{-l-1}(X), \mathcal{P}^{-l}(X))$ for all $l \in [\![0, \ell - 2]\!]$. Now, $\mathcal{D}$ is a distinguisher in the key-reveal PRF security game of the sponge. The resources of $\mathcal{D}$ can be upper bounded by $\ell + \ell'$ construction queries, and $2\ell(q_k - q_{on,k})$ primitive queries. Therefore,

$$\begin{aligned}
&\mathbf{Pr}_{S_k}\left(\mathsf{BADINV}_{q_k - q_{on,k}}\right) \\
&\le \left|\mathbf{Pr}_{S_k}\left(\mathsf{BADINV}_{q_k - q_{on,k}}\right) - \mathbf{Pr}_{S_{k+1}}\left(\mathsf{BADINV}_{q_k - q_{on,k}}\right)\right| \\
&+ \mathbf{Pr}_{S_{k+1}}\left(\mathsf{BADINV}_{q_k - q_{on,k}}\right) \\
&\le \mathbf{Adv}^{\text{prf-krev}}_{\text{OKS}}(\ell + \ell', 2\ell(q_k - q_{on,k})) + \mathbf{Pr}_{S_{k+1}}\left(\mathsf{BADINV}_{q_k - q_{on,k}}\right) . \qquad (7.25)
\end{aligned}$$

Now, we can upper bound the term $\mathbf{Pr}_{S_{k+1}}\left(\mathsf{BADINV}_{q_k - q_{on,k}}\right)$. We have

$$\begin{aligned}
&\mathbf{Pr}_{S_{k+1}}\left(\mathsf{BADINV}_{q_k - q_{on,k}}\right) \\
&\le \sum_{y=1}^{2^c} \sum_{i=1}^{q_k - q_{on,k}} \sum_{l=1}^{\ell} \mathbf{Pr}_{S_{k+1}}\left(\mathsf{BADINV}^l[i] \mid \neg\mathsf{BADINV}_{i-1} \wedge \mathcal{S}[1 : \ell] = \ell y\right) \times \\
&\qquad\qquad\qquad \mathbf{Pr}\left(\mathcal{S}[1 : \ell] = \ell y\right) .
\end{aligned}$$

Remember that the values $x_k^{(l)}$ are sampled at random, but are not given to the adversary. Therefore, the difficulty of guessing an element in $\mathcal{S}[l]$ is augmented by the fact that the adversary must guess $x_k^{(l)}$. Given $i \in [\![1, q_k]\!]$ and $l \in [\![1, \ell]\!]$, we define the bad events GUESS_i^l as follows:

$$\mathsf{GUESS}_i^l : \exists (S_{fwd}, S_{inv}, d) \in \mathcal{Q}[i] \text{ such that } x_k^{(l)} = \begin{cases} \text{outer}_r(S_d) & \text{if } l < \ell\,, \\ \text{outer}_s(S_d) & \text{if } l = \ell\,. \end{cases}$$

Moreover, define the random variables $\mathcal{G}^l$ as follows:

$$\mathcal{G}^l = \left|\left\{i \in [\![1, q_k - q_{on,k}]\!] \mid \mathsf{GUESS}_i^l \text{ holds}\right\}\right| .$$

In other words, for each l, $\mathcal{G}^l$ counts the number of queries during the k-offline phase that have their outer part equal to $x_k^{(l)}$. Note that contrary to the case of BADINV during the k-online phase, we do have to consider the last chaining value separately. Indeed, when $s < r$, guessing $x_k^{(\ell)}$ becomes easier due to the smaller size of the last message block, but given this correct guess, guessing an element in $\mathcal{S}[\ell]$ becomes harder, since $\mathcal{S}[\ell]$ is a set of size y, and is a subset of a set of size at least 2^{b-s}. We have

$$\mathsf{E}_{S_{k+1}}\left(\mathcal{G}^{l}\right) \leq \begin{cases} \frac{(q_k - q_{on,k})}{2^r} & \text{if } l < \ell\,, \\ \frac{(q_k - q_{on,k})}{2^s} & \text{if } l = \ell\,. \end{cases}$$

Note that this bounding is independent of the events $\neg\mathsf{BADINV}_{i-1}$ and $|\mathcal{S}[1:\ell]| = y$. Now, for the probability that the i^{th} query lies in $\mathcal{S}[l]$, conditioned on GUESS_i^l, the reasoning used for the analysis of BADINV during the k-online phase still applies. However, we need to use a more accurate bounding for the size of the superset of $\mathcal{S}[\ell]$. We have

$$\begin{aligned} &\mathbf{Pr}_{S_{k+1}}\left(\mathsf{BADINV}_{q_k - q_{on,k}}\right) \\ &\leq \sum_{y=1}^{2^c} \sum_{i=1}^{q_k - q_{on,k}} \mathbf{Pr}\left(\mathcal{S}[1:\ell] = \ell y\right) \times \\ &\quad \left[\sum_{l=1}^{\ell-1} \mathbf{Pr}\left(\mathsf{GUESS}_i^l\right) \frac{y}{2^c - 3(i-1)} + \mathbf{Pr}\left(\mathsf{GUESS}_i^\ell\right) \frac{y}{2^{b-s} - 3(i-1)}\right] \\ &\leq \sum_{y=1}^{2^c} \mathbf{Pr}\left(\mathcal{S}[1:\ell] = \ell y\right) \left[\sum_{l=1}^{\ell-1} \left(\mathsf{E}\left(\mathcal{G}^l\right) \frac{2y}{2^c}\right) + \mathsf{E}\left(\mathcal{G}^\ell\right) \frac{2y}{2^{b-s}}\right] \\ &\leq \frac{2(q_k - q_{on,k})}{2^b} \mathsf{E}\left(|\mathcal{S}[1:\ell]|\right)\,, \end{aligned} \tag{7.26}$$

where we used that $q_k \leq 2^c/6 \leq 2^{b-s}/6$. Therefore, combining (7.25) and (7.26), we obtain

$$\begin{aligned} \mathbf{Pr}\left(\mathsf{BADINV}_{q_k - q_{on,k}}\right) \leq\ & \mathbf{Adv}_{\text{OKS}}^{\text{prf-krev}}(\ell + \ell', 2\ell(q_k - q_{on,k})) \\ & + \frac{2(q_k - q_{on,k})}{2^b} \mathsf{E}\left(|\mathcal{S}[1:\ell]|\right)\,. \end{aligned} \tag{7.27}$$

Probability of $\mathsf{INNER}(q_k - q_{on,k})$. For $i \in [\![q_k - q_{on,k}, q_k]\!]$, conditioned on the fact that $\mathsf{INNER}(q_k - q_{on,k})$ has not been set beforehand, the probability that the i^{th} query triggers $\mathsf{INNER}(q_k - q_{on,k})$ is upper bounded by $\frac{i2^r}{2^b - i}$. Therefore, using that $q_k \leq 2^{b-1}$,

$$\mathbf{Pr}\left(\mathsf{INNER}(q_k - q_{on,k})\right) \leq \sum_{i=q_k - q_{on,k}+1}^{q_k} \frac{2i}{2^c} \leq \frac{2q_k q_{on,k}}{2^c}\,. \tag{7.28}$$

Conclusion. Plugging (7.23), (7.24), (7.27), and (7.28) into (7.22), we obtain

$$\begin{aligned}\mathbf{Pr}\,(\mathsf{PRE}) \leq{}& \frac{2q_k}{2^b}\mathsf{E}\,(|S[0]|) + \frac{2(q_k - q_{on,k})}{2^b}\mathsf{E}\,(|\mathcal{S}[1:\ell]|)\\ &+ \mathbf{Adv}^{\text{prf-krev}}_{\text{OKS}}(\ell+\ell', 2\ell(q_k - q_{on,k}))\\ &+ \min\left\{\frac{2q_{on,k}}{2^c}\mathsf{E}\,(|S[1:\ell]|)\,,\frac{2q_{on,k}\cdot q_k}{2^c}\right\}. \qquad (7.29)\end{aligned}$$

Now, it remains to compute the expectation of the sets $\mathcal{S}[0]$ ans $\mathcal{S}[1:\ell]$. Again, we use the same approach as in Chapter 4, and use the linearity of the expectation. We have

$$\begin{aligned}\mathsf{E}\,(\mathcal{S}[1]) &= \sum_{y\in\{0,1\}^b} \mathsf{E}\left(\mathbf{1}_{Y\in\mathcal{S}[1]}\right)\\ &\leq \sum_{\substack{y\in\{0,1\}^b \text{ s.t.,}\\ \text{outer}_r(y)=x_k^{(1)}}} \mathbf{Pr}\,(Y\in\mathcal{S}[1])\\ &\leq \sum_{\substack{y\in\{0,1\}^b \text{ s.t.,}\\ \text{outer}_r(y)=x_k^{(1)}}} \frac{2^c}{2^b}\frac{2^c}{2^b-1}\cdots\frac{2^{b-s}}{2^b-(\ell-2)}\\ &= \frac{2^c(2^c)^{\ell-2}\cdot 2^{c-s}}{[2^b]_{\ell-1}}\\ &\leq 2\frac{2^b}{2^n}\,,\end{aligned}$$

where we used that $2[2^b]_{\ell-1} \geq (2^b)^{\ell-1}$ if $(\ell-1)^2 \leq 2^b$ (see Section 4.3.1). Because $\mathcal{P}$ is a permutation, we have for any $l \in [\![0,\ell]\!]$, $\mathcal{S}[l] = \mathcal{S}[1]$, therefore $\mathcal{S}[1:\ell] = \ell\times\mathcal{S}[1]$. Thus,

$$\begin{aligned}\mathsf{E}\,(\mathcal{S}[0]) &\leq 2\frac{2^b}{2^n}\,,\\ \mathsf{E}\,(\mathcal{S}[1:\ell]) &\leq 2\ell\frac{2^b}{2^n}\,.\end{aligned}$$

Plugging these bounds into (7.29) gives

$$\begin{aligned}\mathbf{Pr}_{S_k}\,(\mathcal{A}\text{ solves pre}(x_k)\text{ in }G(k)) \leq{}& \frac{8\ell q_k}{2^n} + \min\left\{\frac{4\ell q_{on,k}}{2^{n-r}}, \frac{2q_{on,k}\cdot q_k}{2^c}\right\}\\ &+ \mathbf{Adv}^{\text{prf-krev}}_{\text{OKS}}(\ell+\ell', 2\ell(q_k - q_{on,k}))\,. \qquad \square\end{aligned}$$

7.5 Improved Multi-User Security Proof of U/Key with a Truncated Permutation

For most use cases, U/Key instantiated with a sponge can be evaluated in one permutation call per sponge, making the underlying construction a truncated permutation. Also in this case, the indifferentiability result studied in Section 7.3.3.3 remains a too strong security property. Indeed, the hash function is required to take $n + s + t$ bits as input, but not all of these inputs can be manipulated freely by the adversary. In the case of a truncated permutation, fixing the salt and the counter fixes the user and thus the elements on the chains that are targeted. Therefore, fixing x_k determines the counters and the users that are targeted. Moreover, the separation between the offline and online phase is not taken into account when using a generic result.

Relying on the simplicity of the construction and its one-pass feature, we obtain in Theorem 7.5.1 below a tight bound using a dedicated multi-user proof that additionally accurately captures offline and online time. Section 7.5.1 describes the construction in more detail, as well as some additional notation used. Section 7.5.2 is dedicated to the security bound and Section 7.5.3 to the proof.

7.5.1 Description of Scheme

We describe the construction that we consider in this section in more detail. The scheme is used simultaneously by μ users, and the user number m runs the hash chain with a random salt id_m. The chaining value number k of user number m is denoted by $x_{k,m}$. In particular, $x_{0,m}$ denote the root passwords. Moreover, let $b > c \geq s+t$, and let $IV \in \{0,1\}^{c-s-t}$ be any fixed initialization vector. (The initialization vector does not play a role in the security.) The state of the permutation is split as $b = n + c$, where n bits are used for the password, and c plays a role similar to the capacity of the sponge. In particular, c includes the counter and the salt. The construction then traverses through the hash chain as

$$x_k = \text{outer}_n(\mathcal{P}(x_{k-1} \parallel \langle \text{ctr}_k \rangle_t \parallel id_m \parallel IV)) \,,$$

for $k \in [\![1, K]\!]$ and $m \in [\![1, \mu]\!]$. The sequence $(\langle \text{ctr}_k \rangle_t \parallel id_m \parallel IV)_{k,m}$ can be seen as a family of fixed prefixes. The scheme is illustrated in Figure 7.2.

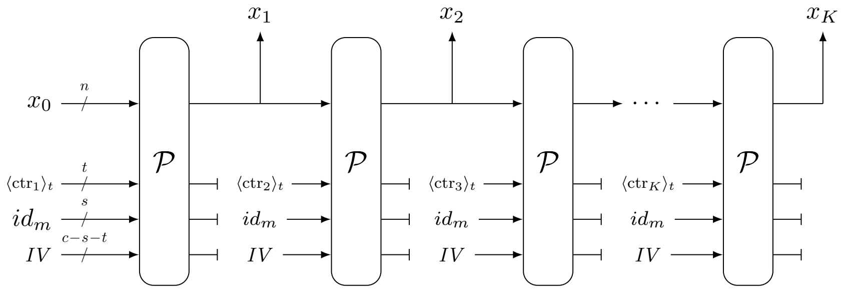

x_1
x_2
x_K
x_0
n
t
s
$c-s-t$
$\langle \mathrm{ctr}_1 \rangle_t$
id_m
IV
$\mathcal{P}$
$\langle \mathrm{ctr}_2 \rangle_t$
$\langle \mathrm{ctr}_3 \rangle_t$
$\langle \mathrm{ctr}_K \rangle_t$
$\cdots$

Figure 7.2: U/Key with a truncated permutation of size $b = n + c$ for user number m.

7.5.2 Security Result

In Theorem 7.5.1 we state the multi-user security of U/Key on top of the truncated permutation construction.

Theorem 7.5.1. *Let* TruncP *denote the construction from Section 7.5.1. Assuming that $K\mu + q_{\mathit{off}} + q_{on} \leq 2^{b-1}$, we have*

$$\begin{aligned}\mathbf{Adv}_{\text{U/Key[TruncP]}}\left(q_{\mathit{off}}, q_{on}, \mu\right) \leq \operatorname{mucol}(\mu, 2^s) \cdot \left(\min\left\{\frac{2\mu}{2^s}, 1\right\} \frac{4q_{\mathit{off}}}{2^n} + \frac{4q_{on}}{2^n}\right) \\ + \operatorname{mucol}(K\mu, 2^n) \cdot \left(\min\left\{\frac{2K\mu}{2^n}, 1\right\} \frac{4q_{\mathit{off}}}{2^c} + \frac{4q_{on}}{2^c}\right),\end{aligned}$$

where $\operatorname{mucol}(\cdot, \cdot)$ *is defined in Lemma 2.5.2.*

Interpretation of the Bound. The bound includes multiple terms, and we will provide key points that can help to interpret it. The bound is a sum of two main terms. The first (resp., second) one corresponds to a strategy where the adversary only makes forward (resp., inverse) queries. Now, regarding offline queries, whenever $\mu < 2^s$ (resp., $K\mu < 2^n$), the number of forward (resp., inverse) offline queries that are "useful" is multiplied by $\frac{\mu}{2^s}$ (resp., $\frac{K\mu}{2^n}$). If we then denote by q the (expected) number of useful queries, the bound of Theorem 7.5.1 simplifies to

$$\tilde{\mathcal{O}}\left(\frac{q}{2^n} + \frac{q}{2^c} + \frac{q\mu}{2^{s+n}} + \frac{qK\mu}{2^b}\right).$$

Moreover, as long as $\mu = \tilde{\mathcal{O}}\left(2^s\right)$, this bound is of the order

$$\tilde{\mathcal{O}}\left(\frac{q}{2^n} + \frac{q}{2^c} + \frac{qK\mu}{2^b}\right) .$$

Just like for the interpretation of Theorem 7.3.1, the security bound allows for many different interpretations. For example, if we take $s = 80$ and $t = 32$, and target a security goal of 128 bits, we must take $n \geq 128$ and $c \geq 128$, and thus require a permutation of size at least 256 bits. For more specific settings, we can make use of the separation between offline and online queries. For example, if we assume $\mu \leq 10^{12}$, $s = 68$, keep t the same, and assume $q_{on} \ll 2^{100}$, then we can take $n = c = 100$, thus having a permutation of size 200 bits, while still achieving 128-bit offline security.

7.5.3 Proof of Theorem 7.5.1

The proof shares similarities with the proof of Theorem 7.3.1, particularly in the way that forward queries to $\mathcal{P}$ are treated. A difference is in the fact that, in current setting, the adversary can also make inverse evaluations of $\mathcal{P}^{-1}$.

Setup. We split q_{on} as $q_{fwd,on} + q_{inv,on}$, depending on the direction of the queries, and similarly split $q_{off} := q_{fwd,off} + q_{inv,off}$. We first define useful notation for the treatment of forward queries. As in the proof of Theorem 7.3.1, let Coll_s be a random variable with sample space $(\llbracket 1, \mu \rrbracket)^\mu$ that determines how are the salts colliding together. More precisely, when two elements at position i and j share the same value, it means that the salts of users i and j collide. Fixing Coll_s determines the number of distinct salts, that we will denote as $\tilde{\mu}$.

Moreover, let NC_s be a random variable defined as follows:

$$\mathrm{NC}_s = \max_{id} \# \left\{ m \mid id_m = id \right\} .$$

NC_s counts the maximal number of jointly colliding salts. The value of Coll_s determines the one of NC_s. We saw in (7.6) that

$$\mathsf{E}\left(\mathrm{NC}_s\right) = \mathrm{mucol}(\mu, 2^s) , \tag{7.30}$$

where $\mathrm{mucol}(\cdot, \cdot)$ is defined in Lemma 2.5.2. Assume that Coll_s is fixed to a certain coll_s, which gives $\tilde{\mu} \leq \mu$ distinct salts, and fixes NC_s to nc_s. Let $k \in \llbracket 1, K \rrbracket$, and $\tilde{m} \in \llbracket 1, \tilde{\mu} \rrbracket$. Note that we can partition the forward queries according to the associated values of $(\tilde{m}, k)$. In particular, a forward permutation query with the unique salt number $\tilde{m}$ and counter k is called a $\mathcal{P}_{k,\tilde{m}}$-query.

During the offline phase, whenever $\mu \ll 2^s$, then with high probability a large portion of the queries do not correspond to $\mathcal{P}_{k,\tilde{m}}$-queries. Let $Q^{(k,\tilde{m})}_{\mathit{fwd,off}}$ be a random variable counting the number of useful $\mathcal{P}_{k,\tilde{m}}$-queries during the offline phase, and let $q^{(k,\tilde{m})}_{\mathit{fwd,on}}$ be the number of useful $\mathcal{P}_{k,\tilde{m}}$ queries during the online phase. Let

$$Q_{\mathit{fwd}} := \sum_{k=1}^{K} \sum_{\tilde{m}=1}^{\tilde{\mu}} Q^{(k,\tilde{m})}_{\mathit{fwd,off}} + q^{(k,\tilde{m})}_{\mathit{fwd,on}} ,$$

so that Q_{fwd} counts the total number of useful forward queries. Similarly to (7.13) and (7.14), we have

$$\mathsf{E}\left(Q_{\mathit{fwd}} \mid \mathrm{Coll}_s = \mathrm{coll}_s\right) \leq \min\left\{\frac{2\mu}{2^s}, 1\right\} \cdot q_{\mathit{fwd,off}} + q_{\mathit{fwd,on}} \,. \tag{7.31}$$

Now, we introduce terminology for the treatment of inverse queries. We define $\mathrm{Coll}_{\mathrm{CV}}$ and $\mathrm{NC}_{\mathrm{CV}}$ respectively, that play a role similar to Coll_s and NC_s, respectively. $\mathrm{Coll}_{\mathrm{CV}}$ determines how the chaining values $x_{k,m}$ from distinct permutation calls are colliding, and is a pair of random variables with sampling space $(\llbracket 1, \mu \rrbracket)^{\mu} \times (\llbracket 1, K\mu \rrbracket)^{K\mu}$. The first component of $\mathrm{Coll}_{\mathrm{CV}}$ corresponds to the random variable Coll_s, and the second one can be represented as a matrix with K rows and μ columns, each element taking values in $\llbracket 1, K\mu \rrbracket$. If the element at row k and column m equals the element at row k' and column m', it means that $x_{k,m}$ collides with $x_{k',m'}$. This random variable needs to keep track of Coll_s, as there might be cases where two different salts id_m and $id_{m'}$ collide, and $x_{k,m} = x_{k,m'}$ due to a lucky collision. In that case (and only in that case), the permutation calls to compute $x_{k+1,m}$ and $x_{k+1,m'}$ will be the same. However, we want to avoid counting these cases as collisions between chaining values. We define the random variable $\mathrm{NC}_{\mathrm{CV}}$ as the maximum number of jointly chaining values that come from *distinct* permutation evaluations. Note that fixing the value of $\mathrm{Coll}_{\mathrm{CV}}$ also fixes the value of $\mathrm{NC}_{\mathrm{CV}}$. Then, each chaining value (without counting repeated permutation evaluations) is the result of a sampling in $\{0,1\}^b$ without replacement. We have

$$\mathsf{E}\left(\mathrm{NC}_{\mathrm{CV}}\right) \leq \mathrm{mucol}(K\mu, 2^n) \,. \tag{7.32}$$

Now, assume that $\mathrm{Coll}_{\mathrm{CV}}$ is set to a certain value $\mathrm{coll}_{\mathrm{CV}}$. Given this fixed instance, we can infer the number of distinct chaining values that we denote by $\mathrm{dist}_{\mathrm{CV}}$. Given $u \in \llbracket 1, \mathrm{dist}_{\mathrm{CV}} \rrbracket$, let $Q^{(u)}_{\mathit{inv,off}}$ (resp., $Q^{(u)}_{\mathit{inv,on}}$) be a random variable which counts the number of inverse queries during the offline (resp., online) phase that have their n leftmost bits fixed to the u^{th} distinct chaining value. Note that this quantity is also a random variable for the online phase, since the adversary might make lucky inverse queries with $x_{k,m}$ *before* the value is revealed. Moreover, let

$$Q_{inv} = \sum_u Q^{(u)}_{inv,\mathit{off}} + Q^{(u)}_{inv,on} \,.$$

We now derive an upper bound for $\mathsf{E}\left(Q_{inv} \mid \mathrm{Coll}_{\mathrm{CV}} = \mathrm{coll}_{\mathrm{CV}}\right)$. We first consider the case where $K\mu \leq 2^{n-1}$. Let $z \in \{0,1\}^n$. Conditioning on the random variable $\mathrm{Coll}_{\mathrm{CV}}$ introduces a supplementary bias in the sampling of the values $x_{k,m}$. Indeed, if the first u distinct chaining values have already been sampled, the $(u+1)^{\text{th}}$ distinct chaining value cannot have its n leftmost bits equal to any of the u previous chaining values. Consequently, the u^{th} distinct chaining value is the result of a sampling from a set of size at least $2^b - u2^c \geq 2^b - K\mu \cdot 2^c$, among which at most 2^c values have their n leftmost bits equal to z. Hence, the probability that one of the $\mathrm{dist}_{\mathrm{CV}}$ chaining values equals to z can be upper bounded by

$$\frac{\mathrm{dist}_{\mathrm{CV}} \cdot 2^c}{2^b - K\mu \cdot 2^c} \leq \frac{2K\mu}{2^n} \,,$$

where we used $K\mu \leq 2^{n-1}$.

In the case where $K\mu \geq 2^{n-1}$, we upper bound $\mathsf{E}\left(Q_{inv} \mid \mathrm{Coll}_{\mathrm{CV}} = \mathrm{coll}_{\mathrm{CV}}\right)$ simply by $q_{inv,\mathit{off}} + q_{inv,on}$. Therefore,

$$\begin{aligned}
\mathsf{E}\left(Q_{inv} \mid \mathrm{Coll}_{\mathrm{CV}} = \mathrm{coll}_{\mathrm{CV}}\right) &= \mathsf{E}\left(\sum_u Q^{(u)}_{inv,\mathit{off}} + Q^{(u)}_{inv,on} \;\middle|\; \mathrm{Coll}_{\mathrm{CV}} = \mathrm{coll}_{\mathrm{CV}}\right) \\
&\leq \min\left\{\frac{2K\mu}{2^n}, 1\right\} \cdot q_{inv,\mathit{off}} + q_{inv,on} \,. \qquad (7.33)
\end{aligned}$$

Bad Event and Probability Splitting. Let $\mathcal{Q}_k$ be the query history of the adversary *before* the $x_{k,m}$'s are revealed. The adversary winning the security game implies that the following bad event happened:

$$\begin{aligned}
\mathsf{BAD} : \exists m \in [\![1,\mu]\!], k \in [\![1,K]\!], d \in \{\mathit{fwd}, \mathit{inv}\}, y \in \{0,1\}^n, z \in \{0,1\}^c \\
\text{such that } \Big((y \parallel \mathrm{ctr}_k \parallel \mathit{id}_m \parallel \mathit{IV}), (x_{k,m} \parallel z), d \Big) \in \mathcal{Q}_{k-1} \,.
\end{aligned}$$

In other words, the adversary must make a query which collides with one of the μ chains. Now, we split BAD as $\mathsf{BADFWD} \vee \mathsf{BADINV}$, depending on the direction of the query triggering BAD. Without loss of generality, we can stop the security game whenever the adversary has set BAD. Therefore, the bad events BADFWD and BADINV are disjoint, so that

$$\mathbf{Pr}\left(\mathsf{BAD}\right) \leq \mathbf{Pr}\left(\mathsf{BADFWD} \wedge \neg\mathsf{BADINV}\right) + \mathbf{Pr}\left(\mathsf{BADINV} \wedge \neg\mathsf{BADFWD}\right) . \tag{7.34}$$

Probability of BADFWD**.** This probability can be upper bounded in a similar way to what has been done in Theorem 7.3.1. We have

$$\begin{aligned}&\mathbf{Pr}\left(\mathsf{BADFWD} \wedge \neg\mathsf{BADINV}\right)\\ &\leq \sum_{\mathrm{coll}_s}\sum_{q_{fwd}} \mathbf{Pr}\left(\mathsf{BADFWD} \;\middle|\; \mathrm{Coll}_s = \mathrm{coll}_s \wedge Q_{fwd} = q_{fwd} \wedge \neg\mathsf{BADINV}\right) \times\\ &\qquad\qquad \mathbf{Pr}\left(Q_{fwd} = q_{fwd} \;\middle|\; \mathrm{Coll}_s = \mathrm{coll}_s\right) \times \mathbf{Pr}\left(\mathrm{Coll}_s = \mathrm{coll}_s\right) .\end{aligned} \tag{7.35}$$

The conditioned event BADFWD can be decomposed in a query-wise fashion. Indeed, due to the condition $Q_{fwd} = q_{fwd}$, the total number of queries likely to trigger BADFWD with a non-zero probability is equal to q_{fwd}. Now, assuming that BADFWD was not set before the useful query number i, then BADFWD can be set in two different ways during the query number i:

- The adversary guesses one *exact* preimage. In that case each of the chaining values are random, with a small bias due to the permutation.[4] More precisely, for each $k \in [\![0, K]\!]$, $m \in [\![1, \mu]\!]$, and $z \in \{0,1\}^n$,

  $$\mathbf{Pr}\left(x_{k,m} = z\right) \leq \frac{2^c}{2^b - K\mu} \leq \frac{2}{2^n} ,$$

  where we used $K\mu \leq 2^{b-1}$. Moreover, one attempt of the adversary targets at most nc_s preimages at the same time, thus its success probability is upper bounded by $\frac{2\mathrm{nc}_s}{2^n}$;
- The adversary guesses a preimage which is not the *exact* one. In that case, we can use directly the randomness of the permutation at the time of the query. For a given forward query, the answer is drawn from a set of size at least $2^b - K\mu - q$, and among them at most $\mathrm{nc}_s \cdot 2^c$ values hit the targeted chaining values. Therefore, one attempt of the adversary has a success probability of at most $\frac{2\mathrm{nc}_s}{2^n}$, where we used $K\mu + q \leq 2^{b-1}$.

[4]Actually, the root passwords $x_{0,m}$ are uniformly random, but this does not change the upper bounding.

Therefore,

$$\mathbf{Pr}\left(\mathsf{BADFWD} \;\middle|\; \mathrm{Coll}_s = \mathrm{coll}_s \wedge Q_{fwd} = q_{fwd} \wedge \neg\mathsf{BADINV}\right) \leq \frac{4\mathrm{nc}_s \cdot q_{fwd}}{2^n} \,.$$

Now plugging this upper bound into (7.35) gives

$$\begin{aligned}
&\mathbf{Pr}\left(\mathsf{BADFWD} \wedge \neg\mathsf{BADINV}\right) \\
&\leq \sum_{\mathrm{coll}_s}\sum_{q_{fwd}} \frac{4\mathrm{nc}_s \cdot q_{fwd}}{2^n} \mathbf{Pr}\left(Q_{fwd} = q_{fwd} \;\middle|\; \mathrm{Coll}_s = \mathrm{coll}_s\right) \cdot \mathbf{Pr}\left(\mathrm{Coll}_s = \mathrm{coll}_s\right) \\
&= \frac{4}{2^n} \sum_{\mathrm{coll}_s} \mathsf{E}\left(Q_{fwd} \;\middle|\; \mathrm{Coll}_s = \mathrm{coll}_s\right) \cdot \mathrm{nc}_s \cdot \mathbf{Pr}\left(\mathrm{Coll}_s = \mathrm{coll}_s\right) \,.
\end{aligned}$$

Plugging (7.30) and (7.31) into the equation above gives

$$\begin{aligned}
&\mathbf{Pr}\left(\mathsf{BADFWD} \wedge \neg\mathsf{BADINV}\right) \\
&\leq \frac{4}{2^n}\left(\min\left\{\frac{2\mu}{2^s}, 1\right\} \cdot q_{fwd,off} + q_{fwd,on}\right) \cdot \sum_{\mathrm{coll}_s} \mathrm{nc}_s \cdot \mathbf{Pr}\left(\mathrm{Coll}_s = \mathrm{coll}_s\right) \\
&= \frac{4}{2^n}\left(\min\left\{\frac{2\mu}{2^s}, 1\right\} \cdot q_{fwd,off} + q_{fwd,on}\right) \mathsf{E}\left(\mathrm{NC}_s\right) \\
&= \mathrm{mucol}(\mu, 2^s) \cdot \left(\min\left\{\frac{2\mu}{2^s}, 1\right\} \cdot \frac{4q_{fwd,off}}{2^n} + \frac{4q_{fwd,on}}{2^n}\right) \,. \qquad (7.36)
\end{aligned}$$

Probability of BADINV. It remains to upper bound the second term of (7.34). We have

$$\begin{aligned}
&\mathbf{Pr}\left(\mathsf{BADINV} \wedge \neg\mathsf{BADFWD}\right) \\
&\leq \sum_{\mathrm{coll}_{\mathrm{CV}}}\sum_{q_{inv}} \mathbf{Pr}\left(\mathsf{BADINV} \mid \mathrm{Coll}_{\mathrm{CV}} = \mathrm{coll}_{\mathrm{CV}} \wedge Q_{inv} = q_{inv} \wedge \neg\mathsf{BADFWD}\right) \times \\
&\qquad \mathbf{Pr}\left(Q_{inv} = q_{inv} \mid \mathrm{Coll}_{\mathrm{CV}} = \mathrm{coll}_{\mathrm{CV}}\right) \times \mathbf{Pr}\left(\mathrm{Coll}_{\mathrm{CV}} = \mathrm{coll}_{\mathrm{CV}}\right) \,.
\end{aligned} \qquad (7.37)$$

Now, regarding the conditioned BADINV, we can reason in a query-wise fashion. The only queries that can trigger BADINV with a non-zero probability are inverse queries with their leftmost bits fixed to one of the $\mathrm{dist}_{\mathrm{CV}}$ chaining values. For a useful query to succeed, there are two possibilities:

- The adversary guesses the *exact* state. In particular, it should guess the full inner part of c bits. These c bits are random, so that for any

$y \in \{0,1\}^c$, $m \in [\![1, \mu]\!]$, $k \in [\![1, K]\!]$,

$$\mathbf{Pr}\left(\mathrm{inner}_c(\mathcal{P}(x_{k-1,m} \parallel \mathrm{ctr}_k \parallel id_m \parallel IV)) = y\right) \le \frac{2^n}{2^b - K\mu} \le \frac{2}{2^c}\,.$$

Moreover, among the chaining values, there are at most $\mathrm{nc}_{\mathrm{CV}}$ different c-bit states which are attached to the same chaining value $x_{k,m}$. Therefore, one attempt succeeds with probability at most $\frac{2\mathrm{nc}_{\mathrm{CV}}}{2^c}$;

- The adversary guesses a preimage which does not correspond to the *exact* state. In that case we can use directly the randomness of the permutation. The inverse query answer is drawn from a set of size at least $2^b - K\mu - q_{on} - q_{off} \ge 2^{b-1}$, and among them at most $\mathrm{nc}_{\mathrm{CV}} \times 2^n$ elements set BADINV. Therefore, one attempt to set this event happens with probability at most $\frac{2\mathrm{nc}_{\mathrm{CV}}}{2^c}$.

Plugging these bounds into (7.37) gives

$$\begin{aligned}
&\mathbf{Pr}\left(\mathsf{BADINV} \wedge \neg\mathsf{BADFWD}\right)\\
&\le \sum_{\mathrm{coll}_{\mathrm{CV}}} \sum_{q_{inv}} \frac{4q_{inv} \cdot \mathrm{nc}_{\mathrm{CV}}}{2^c} \mathbf{Pr}\left(Q_{inv} = q_{inv} \mid \mathrm{Coll}_{\mathrm{CV}} = \mathrm{coll}_{\mathrm{CV}}\right) \cdot \mathbf{Pr}\left(\mathrm{Coll}_{\mathrm{CV}} = \mathrm{coll}_{\mathrm{CV}}\right)\\
&= \frac{4}{2^c} \sum_{\mathrm{coll}_{\mathrm{CV}}} \mathsf{E}\left(Q_{inv} \,\middle|\, \mathrm{Coll}_{\mathrm{CV}} = \mathrm{coll}_{\mathrm{CV}}\right) \cdot \mathrm{nc}_{\mathrm{CV}} \cdot \mathbf{Pr}\left(\mathrm{Coll}_{\mathrm{CV}} = \mathrm{coll}_{\mathrm{CV}}\right)\,.
\end{aligned}$$

We can use the inequality from (7.33) and (7.32) to obtain

$$\begin{aligned}
&\mathbf{Pr}\left(\mathsf{BADINV} \wedge \neg\mathsf{BADFWD}\right)\\
&\le \frac{4}{2^c} \sum_{\mathrm{coll}_{\mathrm{CV}}} \left(\min\left\{\frac{2K\mu}{2^n}, 1\right\} q_{inv,off} + q_{inv,on}\right) \cdot \mathrm{nc}_{\mathrm{CV}} \cdot \mathbf{Pr}\left(\mathrm{Coll}_{\mathrm{CV}} = \mathrm{coll}_{\mathrm{CV}}\right)\\
&= \mathsf{E}\left(\mathrm{NC}_{\mathrm{CV}}\right) \left(\min\left\{\frac{2K\mu}{2^n}, 1\right\} \cdot \frac{4q_{inv,off}}{2^c} + \frac{4q_{inv,on}}{2^c}\right)\\
&\le \mathrm{mucol}(K\mu, 2^n) \cdot \left(\min\left\{\frac{2K\mu}{2^n}, 1\right\} \cdot \frac{4q_{inv,off}}{2^c} + \frac{4q_{inv,on}}{2^c}\right)\,. \qquad (7.38)
\end{aligned}$$

Conclusion. We conclude by plugging (7.36) and (7.38) into (7.34). □

7.6 Conclusion

One-time passwords using hash chains are a viable option if two-factor authentication is not suitable. With their introduction of T/Key, Kogan et al. [KMB17] made a great effort to improve its state of the art. However, their analysis was in a relatively basic model, and with our novel model we have shown that it is possible to argue security in a more fine-grained treatment of the adversarial resources. In particular, we split the adversarial capacity into offline and online computation, and allow for analysis in the multi-user setting. We demonstrated the relevance of our model on our slightly more abstract construction U/Key.

7.6.1 Impact of Online Query Complexity

The separation of offline and online queries allows to more accurately determine the adversarial success probability in terms of the parameters of the scheme. For example, taking shorter timeframes implies that $q_{on,k}$ will be lower, though on the other hand K may need to be higher. For a T/Key construction where the timeframes are all of the same size, this does not make a difference as the bound is determined by $q_{on} = Kq_{on,k}$, but a difference may be present for more general constructions as covered by our new U/Key. More noticeable is the role of the number of users, as we also already showed in the interpretation of the bounds of Theorem 7.3.1 and Theorem 7.5.1. For example, depending on whether μ exceeds 2^s, the bound differs and one can even achieve higher security than the password size n provided $\mu \ll 2^s$.

7.6.2 Memory Storage

In our current security model, we assume that the adversary can store all queries made during the offline phase. However, in real life, there is a tradeoff between memory and computational power, which can result in suboptimal bounds. To address this, we can adopt an approach similar to the one of Kogan et al. [KMB17], i.e., split the adversary $\mathcal{A}$ of Algorithm 10 into two: $\mathcal{A}_1$ runs the offline phase and passes an advice string S to $\mathcal{A}_2$, which runs the online phase. The adversary $\mathcal{A}_1$ is typically unrestricted, but the advice string S has an upper bound on its size. This approach, however, has a downside: indifferentiability composition does not apply since the adversary is two-stage [RSS11]. Therefore, a dedicated proof is needed for any security bound at the construction level, and memory-bounded proofs in this context are particularly challenging [TT18, JT19, Din20].

Chapter 8

To Pad or not to Pad? Padding-Free Arithmetization-Oriented Sponges and Duplexes

8.1 Introduction

Prior to this chapter, most use cases of the sponge construction were studied in the binary setting. In this context, the message expansion incurred by the padding is not so much of an efficiency penalty. Indeed, the minimum injective padding, or 10^*-padding, adds a single 1 and a sufficient number of 0s so that the message M is of size a multiple of r bits. For a sponge evaluation, this incurs an extra permutation evaluation *only if* the unpadded M turns out to have a size divisible by r. Even if a slightly more involved padding is employed (e.g., the SHA-3 hash function standard [Nat15b] first pads M with 01 to separate plain hashing from XOFing, and then adopts the 10^*1-padding), the expected amount of permutation calls incurred purely due to the padding is negligible. In other words, padding is not the bottleneck in bit-oriented cryptographic schemes.

However, the situation drastically changes when we consider schemes that are evaluated on prime field elements rather than bits, so-called *arithmetization-oriented* or *finite-field friendly* schemes. The rise of these applications in the last years, notably in zero-knowledge proof systems [FS86], fully homomorphic encryption [Gen09], and multiparty computation in the head (MPCiTH) [KKW18], has lead to a large amount of cryptographic hash functions, such as Poseidon [GKR+21], Poseidon2 [GKS23], Anemoi and Jive [BBC+23], MiMC [AGR+16], Reinforced Concrete [GKL+22], XHash [ABK+23], Tip5 [SLS+23], and Monolith [GKL+24]. Some of these designs propose permutations that operate over a large field, $\mathbb{F}_p$ for a large p, and are instantiations of the sponge.

As the security proof of the sponge construction is field-agnostic, the bound carries over mutatis mutandis. Notably: if the permutation does not operate on $\mathbb{F}_2$ but rather $\mathbb{F}_p$, the construction is secure as long as the number of permutation evaluations is at most $p^{c/2}$. Naturally, in field-oriented designs, b, c, and r are smaller. For example, Poseidon [GKR+21] takes $p \geq 2^{32}$ (requiring $c = 8$ to achieve approximately 128-bit security), XHash [ABK+23] takes $p \approx 2^{64}$, and Reinforced Concrete [GKL+22] even operates with $p \approx 2^{256}$ and operates on a state of $b = 3$ elements, split into a capacity of $c = 1$ and a rate of $r = 2$. If such a function is used for general-purpose hashing, a superfluous permutation evaluation just to process the padding has to be made for half of the messages on average. This can be quite costly [HBHW23].

8.1.1 The Rise of SAFE

To solve the issue of padding, and various other issues in arithmetization-oriented hashing, Aumasson et al. [AKMQ23] proposed SAFE: Sponge API for Field Elements. SAFE is a generic API for sponge functions specifically tailored towards its use on field elements. In a nutshell, SAFE requires the user to first *hash* the input-output pattern IO of the sponge evaluation into a digest, that can be placed in the inner part of the initial state, and subsequently it allows for element-wise absorption and squeezing as long as these operations obey to the initial input-output pattern IO. The hash function evaluation can then additionally hash certain other bit-based data or a domain separator D. This construction was recently proven secure up to $p^{c/2}$ queries by Khovratovich et al. [KBM23], and SAFE has been implemented in the Neptune hash framework and some zero-knowledge proof projects [Set22, MK22]. A comparable construction has recently been introduced by Ashur and Singh Bhati [AB24].

The core idea of the input-output pattern is that, in many applications, such as commitment schemes, Fiat-Shamir protocols, and verifiable encryption of cryptocurrency transactions, the exact input-output pattern is known and fixed in advance, and even stays constant for many evaluations. By thus hashing this pattern into the inner state, one de facto considers sponge hashing with a prefix-free padding, and no 10^*-padding needs to be performed. Of course, for general-purpose applications, using the input-output pattern, or prefix-free padding in general, is not an ideal solution.

8.1.2 Our Contributions

In this work, we will consider an alternative approach: we will investigate the use of non-cryptographic permutations (NCPs) to indicate end-of-message and to perform domain separation. The approach takes inspiration of Hirose [Hir18], who is itself inspired from the Merkle-Damgård with permutation construction [HPY07]. In a nutshell, Hirose et al. proposed to transform the state of a Merkle-Damgård before the last compression function call using an NCP to indicate the end-of-message, and to use two different NCPs depending on whether the unpadded message was full (i.e., of size a multiple of the message block length) or partial (i.e., not full). One may observe similar appearance of NCPs together with secret primitives. For example, the SUNDAE authenticated encryption scheme [BBLT18] multiplies the state by 2 or 4 depending on whether data is full or partial. Likewise, CMAC [IK03, Dwo05] blinds the final state differently depending on whether data is full or partial.

We will adopt this idea to the sponge and discuss its potential in full generality, including a discussion on the requirements on the NCPs in various sponge-based schemes. In detail, we will consider two main types of construction in this work:

Hashing (Section 8.2). We consider the sponge hash function, or more specifically the optimized version described in Section 3.1 with increased initial absorption and increased squeezing, but with an NCP to transform the inner part of the state (different NCPs depending on whether the message is full or partial). This contribution comes closest to the idea of Hirose, but differs in that we apply it to the optimized sponge design. In addition, we consider two variants: Sponge-pi that allows the initial state to be selected from a restricted set of IVs and Sponge-pi\$ that uses an external random oracle to hash certain bit-based input into the initial state (akin to SAFE [AKMQ23] and the work of Grassi and Mennink [GM22]). We demonstrate that security is achieved up to a constant factor 7 loss in the bound, *regardless of the size of the field.* In a bit more detail, we prove that the two constructions are indifferentiable from a random oracle up to approximately $p^{c/2}/7$ queries.

Duplex (Section 8.3). We consider the use of NCPs in the duplex construction [BDPV11b]. This generalization of the duplex to a padding-free setting is significantly complex, the main reason being that the duplex is general on purpose and individual duplexing calls can have different roles in a bigger construction. Because of this reason, the duplex may potentially require an NCP in every duplexing call, and these NCPs also need to be different. This brings us to two variants Duplex-pi and Duplex-pi$, again differing in whether a hash function is used to hash certain bit-based input into the initial state, and we prove that these two achieve indifferentiability security up to approximately $\big(p^c/(\eta^2-\eta+1)\big)^{1/2}$ queries, where η denotes the number of NCPs.

8.1.3 Comparison

Overall, the use of NCPs does not come for free. In particular, for Sponge-pi we get a constant factor loss, and for Duplex-pi with η different NCPs we lose a factor $(\eta^2-\eta+1)$. However, different from simply carrying over the bit-oriented result to finite fields, this loss remains constant *regardless of the size of the field.* This actual loss furthermore highly depends on the actual number of NCPs in use. As we demonstrate in the applications, the Duplex-pi construction can be used to implement the Sponge-pi construction with $\eta=3$ NCPs, as we basically consider three NCPs (the identity, and two different ones to separate between full and partial plaintext). This also matches above-mentioned constant loss 7.

One can consider our results for $p=2$ to be a generalization of padding into the inner part, and which technically is equivalent to reducing the capacity by 1, which also means a constant factor loss. In this bit-wise setting, it is worth noting that there have been schemes that already adopted this technique to a certain extent. In particular, as we discuss in our applications in Section 8.5, the ESCH hash function of the NIST lightweight cryptography competition SPARKLE [BBC+19] is a special case of Sponge-pi, taking bit rotations with a different offset as NCPs. Our result on Sponge-pi, as a corollary, implies security of the ESCH mode. Likewise, one can implement SpongeWrap-pi as an authenticated encryption scheme on top of Duplex-pi with 3 NCPs (as we also discuss in Section 8.5).

However, yet, the true power of our schemes only becomes apparent when operating on bigger fields. Indeed, taking for example $p\approx 2^{64}$, one could typically take a sponge construction with $c=4$ and $r=1$ to achieve approximately 128-bit security and absorb with one element of approximately 64 bits at a time. With normal padding, each invocation then takes an extra permutation call, and padding into the inner part would reduce the inner part from $c=4$ to $c=3$, for which the conventional proofs only guarantee approximately 96-bit security. If we go for the extreme case of Reinforced

Concrete (with $b = 3$, $c = 1$, and $r = 2$), padding would result in a superfluous permutation call for half of the messages (as $r = 2$), an element flip in the capacity would turn the existing security bound to $p^{c-1}/2 = \tilde{\mathcal{O}}(1)$ (as $c - 1 = 0$), but our solution still gives a very decent security bound of $p^{c/2}/7$, yielding approximately 128-bit security.

8.1.4 Outline

We introduce further notation and definitions in Section 8.1.5. Our finite-field friendly sponges Sponge-pi and Sponge-pi$ are presented in Section 8.2, and the duplexes Duplex-pi and Duplex-pi$ in Section 8.3. The security proofs are gathered in Section 8.4. Section 8.5 includes example applications of our constructions, and we conclude in Section 8.6.

8.1.5 Notation and Definitions

The notation introduced in Section 2.1 naturally extends to the finite-field setting, but we make it explicit here for clarity. We denote $\mathbb{F}_p^{\leq a} = \bigcup_{i=0}^{a} \mathbb{F}_p^i$, and $\mathbb{F}_p^* = \bigcup_{i=0}^{\infty} \mathbb{F}_p^i$. Moreover, we denote by $\mathbb{F}_p^\infty$ the set of all infinite tuples of elements of $\mathbb{F}_p$. Note that $\mathbb{F}_p^*$ contains the empty tuple, which we also denote ϵ. Let $s = (s_1, s_2, \ldots, s_n)$ and $s' = (s'_1, s'_2, \ldots, s'_m) \in \mathbb{F}_p^*$, and $\ell, \ell' \in \mathbb{N}$ with $1 \leq \ell \leq \ell'$. We denote by $s \| s'$ the tuple $(s_1, \ldots, s_n, s'_1, \ldots, s'_m)$. In particular, if $s = (1)$ and s' is of the form 0^α, we abbreviate the concatenation as 10^α. Moreover, if $n = m$, we denote $s \oplus s'$ as the element-wise addition of s and s'. Moreover, $s[\ell : \ell']$ refers to the empty tuple if $\ell > n$, and $(s_\ell, \ldots, s_{\max\{\ell', n\}})$ otherwise. If $\ell \leq n$, we define $\mathrm{outer}_\ell(s)$ to be $s[1 : \ell]$ and $\mathrm{inner}_\ell(s)$ to be $s[n - \ell : n]$.

8

Padding. In our constructions, the padding will be done via a function called $\mathit{padBlock}_{r'',r}(\cdot)$. It takes as input a mesage $M \in \mathbb{F}_p^*$, and returns a tuple of field blocks to be absorbed into the state, additionally with a label $d \in \{p, f\}$ helping to choose which NCP to apply. It internally uses a function $\mathit{cutBlock}_{r'',r}(\cdot)$ that splits the input into blocks. $\mathit{cutBlock}_{r'',r}(\cdot)$ and $\mathit{padBlock}_{r'',r}(\cdot)$ are defined in Algorithm 11. Moreover, we will overload the notation by denoting $\mathit{cutBlock}_r(\cdot) = \mathit{cutBlock}_{r,r}(\cdot)$ and $\mathit{padBlock}_r(\cdot) = \mathit{padBlock}_{r,r}(\cdot)$.

Algorithm 11 Definition of *cutBlock* and *padBlock*

```
1:  function $cutBlock_{r'',r}(M)$
        input: $M \in \mathbb{F}_p^*$
        output: $\overline{M} \in \mathbb{F}_p^{\leq r''} \times (\mathbb{F}_p^{\leq r})^*$
2:      $\ell \leftarrow 1 + \lceil (|M| - r'')/r \rceil$
3:      $M_1 \leftarrow M[1 : r'']$
4:      for $1 \leq i \leq \ell - 1$ do
5:          $M_{i+1} \leftarrow M[r''+(i-1)r+1 : r'' + ir]$
6:      return $(M_1, \ldots, M_\ell)$
```

```
1:  function $padBlock_{r'',r}(M)$
        input: $M \in \mathbb{F}_p^*$
        output: $(\overline{M}, d) \in (\mathbb{F}_p^{r''} \times (\mathbb{F}_p^r)^*) \times \{p, f\}$
2:      if $|M| < r''$
3:          $M \leftarrow M \| 10^{-|M|-1 \bmod r''}$
4:          $d \leftarrow p$
5:      else if $|M| - r'' \bmod r > 0$
6:          $M \leftarrow M \| 10^{-(|M|-r'')-1 \bmod r}$
7:          $d \leftarrow p$
8:      else
9:          $d \leftarrow f$
10:     return $(cutBlock_{r'',r}(M), d)$
```

8.2 Arithmetization-Oriented Sponges

We describe our two functions Sponge-pi and Sponge-pi$ in Section 8.2.1, and discuss their security in Section 8.2.2.

8.2.1 Specification of Sponge-pi and Sponge-pi$

In a nutshell, Sponge-pi and Sponge-pi$ are both based on the sponge construction [BDPV07], but are (i) optimized in the absorption rate at initialization and in the squeezing rate as described in Chapter 3, and (ii) adopting Hirose's trick [Hir18] by transforming the inner state before squeezing using an NCP. Both constructions separate between full and partial message, by the use of the NCP. The difference between the two constructions is in the initialization: in Sponge-pi, the initial value is selected in advance from a family of μ predetermined initial values, whereas in Sponge-pi$, they are outputs of a random oracle, comparable to how it was done in [AKMQ23, GM22].

Setup. Let $b, r'', c'', r, c, r', c' \in \mathbb{N}^*$ such that $b = r'' + c'' = r + c = r' + c'$ and $c'' \leq c$. Both constructions operate on top of a cryptographic permutation $\mathcal{P} : \mathbb{F}_p^b \to \mathbb{F}_p^b$. The Sponge-pi construction is defined for a predetermined set of initial values $(IV_m)_{m \in [\![1,\mu]\!]}$, whereas Sponge-pi$ operates on a hash function $h : \mathcal{D} \to \mathbb{F}_p^{c''}$ such that $[\![1, \mu]\!] \subseteq \mathcal{D}$.

Algorithm 12 Sponge-pi and Sponge-pi\$ constructions.

```
1: function CoreSponge($IV, M, \nu$)
2:     $S \leftarrow IV$
3:     $Z \leftarrow \epsilon$
4:     $(M_1, \ldots, M_\ell), d \leftarrow padBlock_{r'',r}(M)$
5:     for $1 \leq i \leq \ell - 1$ do
6:         $S \leftarrow \mathcal{P}\left(S \oplus (M_i \| 0^*)\right)$
7:     $S \leftarrow \mathcal{P}\left(\pi^d\left(S \oplus (M_\ell \| 0^*)\right)\right)$
8:     for $1 \leq i \leq \lceil \frac{\nu}{r'} \rceil$ do
9:         $Z \leftarrow Z \| \mathrm{outer}_{r'}(S)$
10:        $S \leftarrow \mathcal{P}(S)$
11:    return $Z[1:\nu]$
```

```
1: function Sponge-pi($m, M, \nu$)
       input: $(m, M, \nu) \in [\![1, \mu]\!] \times \mathbb{F}_p^* \times \mathbb{N}^*$
2:     $IV \leftarrow 0^{r''} \| IV_m$
3:     return CoreSponge($IV, M, \nu$)
```

```
1: function Sponge-pi$($m, M, \nu$)
       input: $(m, M, \nu) \in [\![1, \mu]\!] \times \mathbb{F}_p^* \times \mathbb{N}^*$
2:     $IV \leftarrow 0^{r''} \| h(m)$
3:     return CoreSponge($IV, M, \nu$)
```

Consider two NCPs:

$$\pi^p : \mathbb{F}_p^{c''} \to \mathbb{F}_p^{c''}, \qquad \pi^f : \mathbb{F}_p^{c''} \to \mathbb{F}_p^{c''}.$$

By abuse of notation, given $A \in \mathbb{F}_p^*$ and $B \in \mathbb{F}_p^{c''}$, for $d \in \{p, f\}$ we use $\pi^d(A \| B)$ to refer to $A \| \pi^d(B)$.

The two NCPs are required to satisfy two constraints. The first constraint ensures no collision between the NCPs and no fixed-point for π^f (note that we do not need to avoid a fixed-point for π^p, because partial messages always get padded to at least one full block):

Constraint 1. *For any $x \in \mathbb{F}_p^{c''}$, we must have*

$$\pi^p(x) \neq \pi^f(x) \text{ and } x \neq \pi^f(x).$$

The second constraint is only relevant in case the user can choose the IVs (i.e., for Sponge-pi), and requires constraint 1 to apply even over different IVs:

Constraint 2. *For any distinct $m_1, m_2 \in [\![1, \mu]\!]$, we must have*

$$\left\{IV_{m_1}, \pi^f(IV_{m_1}), \pi^p(IV_{m_1})\right\} \cap \left\{IV_{m_2}, \pi^f(IV_{m_2}), \pi^p(IV_{m_2})\right\} = \emptyset.$$

A simple example that satisfies constraint 1 is to take the identity permutation for π^p and addition by a non-zero constant C on the rightmost c'' field elements for π^f. In the context of Sponge-pi, if $c'' > 1$, then in order to satisfy constraint 2, the initial values could be encoded on the leftmost $c'' - 1$ field

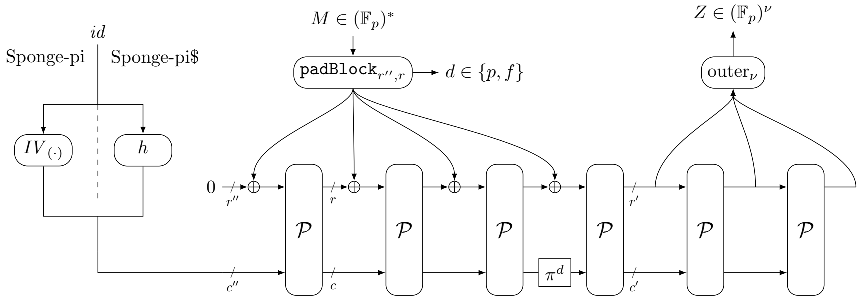


Figure 8.1: Sponge-pi and Sponge-pi$ constructions, with a requested output of ν field elements. The two functions differ in the selection of the IV. For Sponge-pi, on input of $m \in [\![1, \mu]\!]$, the function "$IV_{(\cdot)}$" selects the initial value IV_m, whereas for Sponge-pi$ the initial value is generated by an evaluation of h.

elements, while the constant C is added only to the rightmost field element. If instead $c'' = 1$ and the IVs are generated dynamically (e.g., via a counter), the procedure for generating IVs can be adapted to ensure that any new initial value IV must differ from a prior IV', $IV' + C$, and $IV' - C$. On the other hand, if a predefined, arbitrary, list of unique IVs is provided independently of the construction, then selecting NCPs to meet constraint 2 is challenging, and in that case, using the Sponge-pi$ variant may be more practical.

Constructions. The two constructions are described in Algorithm 12 and illustrated in Figure 8.1.

8.2.2 Security of Sponge-pi and Sponge-pi$

We will prove the security of both constructions. Here, we will quantify the complexity of the distinguisher by the number of permutation evaluations required to perform all computations in the real world, as done by the designers of sponge and explained in Section 3.1.2.1. This quantity is denoted by $\mathcal{N}$. Moreover, with the construction Sponge-pi$, we denote by q_H the number of queries made to the hash function h on the input domain $[\![1, \mu]\!]$. We have the following results:

Theorem 8.2.1. *Let $\mathcal{N} \in \mathbb{N}^*$ and $(IV_m)_{m \in [\![1,\mu]\!]} \subset \mathbb{F}_p^{c''}$. Let π^p and π^f be two NCPs satisfying constraints 1 and 2. Let $\mathcal{C}$ denote the* Sponge-pi *construction based on a random permutation $\mathcal{P}$. There exists a simulator $\mathbf{S}$ with complexity $\tilde{\mathcal{O}}(\mathcal{N})$ queries such that, for any distinguisher $\mathcal{D}$ making at most $\mathcal{N}$ permutation evaluations,*

$$\mathbf{Adv}^{\text{indif}}_{\mathcal{C},\mathbf{S}}(\mathcal{D}) \leq \frac{7\mathcal{N}(\mathcal{N}-1)}{p^c} + \frac{7\mu\mathcal{N}}{2p^{c''}} + \frac{\mathcal{N}(\mathcal{N}-1)}{2p^b} + \frac{\text{mucol}(\mathcal{N}, p^{r'-r})\mathcal{N}}{p^{c'}},$$

where mucol$(\cdot,\cdot)$ *is defined in Lemma 2.5.2.*

Theorem 8.2.2. *Let $\mathcal{N}, q_H \in \mathbb{N}^*$. Let π^p and π^f be two NCPs satisfying constraint 1. Let $\mathcal{C}\$$ denote the* Sponge-pi\$ *construction based on a random oracle h and a random permutation $\mathcal{P}$. There exists a simulator $\mathbf{S}$ with complexity $\tilde{\mathcal{O}}(\mathcal{N}+q_H)$ queries such that, for any distinguisher $\mathcal{D}$ making at most $\mathcal{N}$ permutation evaluations and q_H random oracle evaluations on the input domain $[\![1,\mu]\!]$,*

$$\mathbf{Adv}^{\text{indif}}_{\mathcal{C}\$,\mathbf{S}}(\mathcal{D}) \leq \frac{7\mathcal{N}(\mathcal{N}-1)}{p^c} + \frac{22\mathcal{N}q_H}{p^{c''}} + \frac{7q_H(q_H-1)}{2p^{c''}} + \frac{\mathcal{N}(\mathcal{N}-1)}{2p^b} + \frac{\text{mucol}(\mathcal{N}, p^{r'-r})\mathcal{N}}{p^{c'}},$$

where mucol$(\cdot,\cdot)$ *is defined in Lemma 2.5.2.*

Even though we stated security of the two constructions in separate theorems (to make the conditions explicit), the proofs are in fact very similar, and we prove Theorems 8.2.1 and 8.2.2 simultaneously in Section 8.4.1.

8

Interpretation of the Bounds. Ignoring logarithmic factors, the bounds are of the form

$$\mathbf{Adv}^{\text{indif}}_{\text{Sponge-pi},\mathbf{S}}(\mathcal{D}) = \tilde{\mathcal{O}}\left(\frac{\mathcal{N}^2}{p^c} + \frac{\mu\mathcal{N}}{p^{c''}} + \frac{\mathcal{N}}{p^{c'}}\right),$$
$$\mathbf{Adv}^{\text{indif}}_{\text{Sponge-pi\$},\mathbf{S}}(\mathcal{D}) = \tilde{\mathcal{O}}\left(\frac{\mathcal{N}^2}{p^c} + \frac{q_H\mathcal{N}}{p^{c''}} + \frac{\binom{q_H}{2}}{p^{c''}} + \frac{\mathcal{N}}{p^{c'}}\right).$$

By putting $\mu = 1$, i.e., allowing a single initial value, the former bound is identical to that of Naito and Ohta [NO14]. The second bound is identical, noting that $q_H \leq \mu$. The multi-user setting, where the distinguisher may choose from μ initial values, linearly scales the second term for Sponge-pi, de facto reducing the initial capacity c'' by a logarithmic factor (in $\log_p$). For Sponge-pi\$, this multi-user setting leads to an extra term covering collisions between hash function calls, similar to [AKMQ23, GM22].

In general, to get a balanced bound, one needs $p^{c''}/\mu \approx p^{c/2}$ in the case of Sponge-pi. Assume we have a field size around 2^{64}, security goal of 2^{128}, and around 2^{64} users. Then, we need $b \geq 5$ and $c = 4$, and we can lower c'' to 3 and c' to around 2. For the case of Sponge-pi\$, if the application accepts any kind of input to h, it is better to take $r = r''$. However, still the squeezing rate can be larger (i.e., $r' \approx r + c/2$) without any significant loss in the security.

8.3 Arithmetization-Oriented Duplex

We describe our two functions Duplex-pi and Duplex-pi\$ in Section 8.3.1, and discuss their security in Section 8.3.2.

8.3.1 Specification of Duplex-pi and Duplex-pi\$

The Duplex-pi and Duplex-pi\$ constructions are based on the simple duplex construction [BDPV11b], but employ NCPs per duplexing call in lieu of padding. The case is significantly more subtle than the case of Sponge-pi and Sponge-pi\$, because duplexing calls can have various different roles depending on the use case, and the NCPs need to account for that. The difference between the two constructions is, just like in Section 8.2, in the initialization: in Duplex-pi, the initial value is selected in advance from a family of μ predetermined initial values, whereas in Duplex-pi\$, they are outputs of a random oracle.

Setup. Let $b, r'', c'', r, c, r', c' \in \mathbb{N}^*$ such that $b = r'' + c'' = r + c = r' + c'$ and $c'' \leq c$. Both constructions operate on top of a cryptographic permutation $\mathcal{P} : \mathbb{F}_p^b \to \mathbb{F}_p^b$. The Duplex-pi construction is defined for a predetermined set of initial values $(IV_m)_{m \in [\![1,\mu]\!]}$, whereas Duplex-pi\$ operates on a hash function $h : \mathcal{D} \to \mathbb{F}_p^{c''}$ such that $[\![1, \mu]\!] \subseteq \mathcal{D}$.

The duplex constructions differ from the sponge constructions in that every permutation evaluation is a duplex call that absorbs data and squeezes data. Depending on the application, individual duplexing calls may have different roles and we need to accommodate for this (e.g., domain separation) by using more NCPs. In detail, we consider the following family of NCPs:

$$\left(\pi_i^d : \mathbb{F}_p^{c''} \to \mathbb{F}_p^{c''}\right)_{i\in[\![1,\zeta]\!],\, d\in\{p,f\}} .$$

This family of NCPs is required to satisfy two constraints. In detail, below constraint 3 applies to both constructions and constraint 4 applies only to the case where the user can choose the IVs (i.e., for Duplex-pi).

Constraint 3. *For any* $x \in \mathbb{F}_p^{c''}$ *and any distinct* $(i_1, d_1), (i_2, d_2) \in [\![1, \zeta]\!] \times \{p, f\}$, *we must have*

$$\pi_{i_1}^{d_1}(x) \neq \pi_{i_2}^{d_2}(x) .$$

A simple example that satisfies constraint 3 is to encode each (i, d) with a unique constant $c_{i,d}$ and then take $\pi_i^d(x) = x \oplus c_{i,d}$.

Constraint 4. *For any distinct* $m_1, m_2 \in [\![1, \mu]\!]$ *and any (not necessarily distinct)* $(i_1, d_1), (i_2, d_2) \in [\![1, \zeta]\!] \times \{p, f\}$, *we must have*

$$\pi_{i_1}^{d_1}(IV_{m_1}) \neq \pi_{i_2}^{d_2}(IV_{m_2}) .$$

Constructions. A duplex is a stateful object, and has two interfaces, named "`initialize`" and "`duplexing`". There have been various suggestions on what exactly would be captured by an `initialize` (e.g., should the first `duplexing` call be part of an `initialize` or not?), and how a `duplexing` call should be defined (e.g., should a `duplexing` call start at absorbing plaintext, or at permuting the state?). Mennink gave a treatment on how the phasing of the keyed duplex evolved [Men23, Section 3.4]. However, as we focus on the unkeyed duplex (and we will in fact prove its indifferentiability), we will adopt the convention that a `duplexing` call consists of absorbing-permuting-squeezing (as in the original duplex proposal [BDPV11b] but also in the work of Degabriele et al. [DFG23]) and that an `initialize` call initializes the state *and* does the initial `duplexing` call (which happens at a different rate):

- "`initialize`" initializes an instance of the duplex object. It takes as input the value m to initialize the state (here, the two constructions Duplex-pi and Duplex-pi\$ differ) with the correct IV at its inner part. It additionally takes as input a message block $M \in \mathbb{F}_p^*$ with $|M| \leq r''$ and a domain separator i. It pads the message block *if needed*, using $padBlock_{r''}$, absorbs it into the state, applies the appropriate NCP, permutes and returns the leftmost r' field elements of the state;

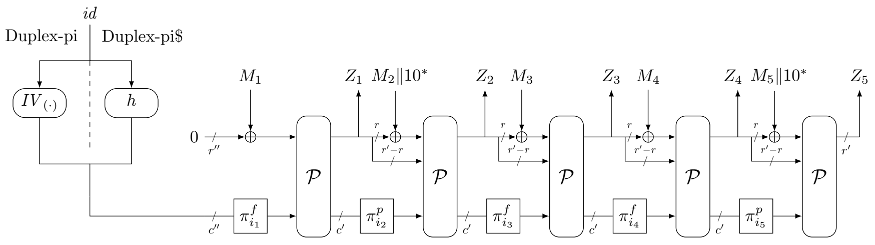


Figure 8.2: Duplex-pi and Duplex-pi$ constructions. Note that the padding function *padBlock* is integrated inside the figure for visibility. The two functions differ in the selection of the IV. For Duplex-pi, on input of $m \in [\![1, \mu]\!]$, the function "$IV_{(\cdot)}$" selects the initial value IV_m, whereas for Duplex-pi$ the initial value is generated by an evaluation of h.

- "`duplexing`" duplexes the state. It takes as input a message block $M \in \mathbb{F}_p^*$ with $|M| \leq r$ and a domain separator i. It pads the message block *if needed*, using $padBlock_r$, absorbs it into the state, applies the appropriate NCP, permutes and returns the leftmost r' field elements of the state.

The scheme can be parallelized over different instances by incorporating a unique identifier u alongside a state that gets initialized. We admit that for every duplexing call a NCP is applied on the inner part of the state. This might seem as a lot of overhead at first sight, but these NCPs can be implemented with constant additions, and the NCP that is expected to be used most often can be set to the identity (in fact, the original duplex construction [BDPV11b] did 10^*-padding for every `duplexing` call). Finally, we remark that, if $M_1, M_2 \in \mathbb{F}_p$, then absorbing first M_1 and then M_2 does not necessarily give the same output as absorbing $M_1 \| M_2$ together. However, this constraint is not so confusing in practice, since typically, before a domain separator is applied, the message blocks are first of full length, and only the last block is potentially not full.

The two constructions are described in Algorithm 13 and illustrated in Figure 8.2. Here, in the illustration, the `duplexing`-wise padding is integrated in the figure for visibility.

Algorithm 13 Duplex-pi and Duplex-pi$ constructions.

```
1: function Duplex-pi.initialize($(m, M, i)$)
       input: $(m, M, i) \in [\![1, \mu]\!] \times \mathbb{F}_p^{\leq r''} \times [\![1, \zeta]\!]$
       output: $Z \in \mathbb{F}_p^{r'}$
2:     $IV \leftarrow 0^{r''} \| IV_m$
3:     return CoreDuplex.initialize($(IV, M, i)$)
4: function Duplex-pi.duplexing($(M, i)$)
5:     return CoreDuplex.duplexing($(M, i)$)
```

```
1: function Duplex-pi$.initialize($(m, M, i)$)
       input: $(m, M, i) \in [\![1, \mu]\!] \times \mathbb{F}_p^{\leq r''} \times [\![1, \zeta]\!]$
       output: $Z \in \mathbb{F}_p^{r'}$
2:     $IV \leftarrow 0^{r''} \| h(m)$
3:     return CoreDuplex.initialize($(IV, M, i)$)
4: function Duplex-pi$.duplexing($(M, i)$)
5:     return CoreDuplex.duplexing($(M, i)$)
```

```
1: function CoreDuplex.initialize($(IV, M, i)$)
2:     $S \leftarrow IV$
3:     $\overline{M}, d \leftarrow padBlock_{r''}(M)$
4:     $S \leftarrow \mathcal{P}\left(\pi_i^d\left(S \oplus (\overline{M} \| 0^*)\right)\right)$
5:     $Z \leftarrow \text{outer}_{r'}(S)$
6:     return $Z$
7: function CoreDuplex.duplexing($(M, i)$)
8:     $\overline{M}, d \leftarrow padBlock_{r}(M)$
9:     $S \leftarrow \mathcal{P}\left(\pi_i^d\left(S \oplus (\overline{M} \| 0^*)\right)\right)$
10:    $Z \leftarrow \text{outer}_{r'}(S)$
11:    return $Z$
```

Algorithm 14 ORO.

```
1: function ORO.initialize($m, M, i$)
2:     path ← $(m, M, i)$
3:     $Z \leftarrow \mathcal{RO}(\mathtt{path}, r')$
4:     return $Z$
```

```
1: function ORO.duplexing($M, i$)
2:     path ← path.append($M, i$)
3:     $Z \leftarrow \mathcal{RO}(\mathtt{path}, r')$
4:     return $Z$
```

8.3.2 Security of Duplex-pi and Duplex-pi$

We prove security of both constructions in this section. Before doing so, we remark that security of the original duplex was derived from that of the sponge [BDPV11b], and one might wonder whether we can likewise reduce the security of Duplex-pi to Sponge-pi (or Duplex-pi$ to Sponge-pi$). However, our description of Duplex-pi, and in particular the use of various NCPs to capture domain separators, is much more general. Stated differently, Sponge-pi and Sponge-pi$ have a fixed pattern of absorb-then-squeeze, which means that one can get away with three NCPs (counting the identity as one), but in the duplex more NCPs may be required and the conditions are also more restrictive.

To formally argue indifferentiability of the duplex constructions, we have to compare its construction with a special random object, an online random oracle, or ORO. It extends the notion of a random oracle in that it is stateful and repeats outputs when the sequence of duplexing calls is repeated. We take the formalism of Degabriele et al. [DFG23], which is a keyless version of the ideal extendable input function (IXIF) of Daemen et al. [DMV17]. The ORO is defined in Algorithm 14. Here, the function append takes as input a list and an object and appends the object to the list.

Before going to the final theorems, we wish to note that the counting of the distinguisher's resources specified in Section 8.2.2 still applies, though it becomes slightly more delicate. In a nutshell, for each duplexing call, we can define a path that lead to that duplexing call (this is exactly the path *path* in the ORO functionality of Algorithm 14). $\mathcal{N}$ then counts the number of permutation evaluations as well as the number of unique paths in the evaluation of the duplex. In other words, every initialize or duplexing call counts as 1 in the number of queries, except if

- in the real world, a permutation call repeats because the associated path had already occurred before, or
- in the ideal world, the associated path had already been queried to $\mathcal{RO}$ before.

This method of counting might appear counter intuitive, and has led to misinterpretations (as we saw, e.g., in the matching attack described in Section 3.3.1.3). However, doubly counting repeated paths, given the current exiting proof techniques, seems to unavoidably lead to a loss in the tightness of the bound. The underlying reason is that current security proofs consider bad events that are triggered by fresh randomness only (i.e., only $\mathcal{RO}$ or permutation calls with new inputs), and they do not keep track of repeated paths.

We are now ready to prove security of Duplex-pi and Duplex-pi\$.

Theorem 8.3.1. *Let $\mathcal{N} \in \mathbb{N}^*$. Let $\left(\pi_i^d : \mathbb{F}_p^{c''} \to \mathbb{F}_p^{c''}\right)_{i\in[\![1,\zeta]\!],\, d\in\{p,f\}}$ be a family of NCPs satisfying constraints 3 and 4, and $(IV_m)_{m\in[\![1,\mu]\!]} \subset \mathbb{F}_p^{c''}$. Let $\mathcal{C}$ denote the* Duplex-pi *construction based on a random permutation $\mathcal{P}$. There exists a simulator $\mathbf{S}$ with complexity $\tilde{\mathcal{O}}(\mathcal{N})$ queries such that, for any distinguisher $\mathcal{D}$ making at most $\mathcal{N}$ permutation evaluations,*

$$\mathbf{Adv}_{\mathcal{C},\mathbf{S}}^{\text{indif}}(\mathcal{D}) \leq \frac{(4\zeta^2-2\zeta+1)\mathcal{N}(\mathcal{N}-1)}{p^c} + \frac{(4\zeta^2-2\zeta+1)\mu\mathcal{N}}{2p^{c''}} + \frac{\mathcal{N}(\mathcal{N}-1)}{2p^b} + \frac{\text{mucol}(\mathcal{N},p^{r'-r})\mathcal{N}}{p^{c'}},$$

where $\text{mucol}(\cdot,\cdot)$ *is defined in Lemma 2.5.2.*

Theorem 8.3.2. *Let $\mathcal{N}, q_H \in \mathbb{N}^*$. Let $\left(\pi_i^d : \mathbb{F}_p^{c''} \to \mathbb{F}_p^{c''}\right)_{i\in[\![1,\zeta]\!],\, d\in\{p,f\}}$ be a family of NCPs satisfying constraint 3. Let $\mathcal{C}\$$ denote the* Duplex-pi\$ *construction based on a random oracle h and a random permutation $\mathcal{P}$. There exists a simulator $\mathbf{S}$ with complexity $\tilde{\mathcal{O}}(\mathcal{N}+q_H)$ queries such that, for any distinguisher $\mathcal{D}$ making at most $\mathcal{N}$ permutation evaluations and q_H random oracle evaluations,*

$$\mathbf{Adv}_{\mathcal{C}\$,\mathbf{S}}^{\text{indif}}(\mathcal{D}) \leq \frac{(4\zeta^2-2\zeta+1)\mathcal{N}(\mathcal{N}-1)}{p^c} + \frac{(4\zeta^2-2\zeta+1)q_H(q_H-1)}{2p^{c''}} + \frac{3(4\zeta^2-2\zeta+1)\mathcal{N}q_H}{p^{c''}} + \frac{\mathcal{N}(\mathcal{N}-1)}{2p^b} + \frac{\text{mucol}(\mathcal{N},p^{r'-r})\mathcal{N}}{p^{c'}},$$

where $\text{mucol}(\cdot,\cdot)$ *is defined in Lemma 2.5.2.*

The proofs of Theorems 8.3.1 and 8.3.2 are very similar to each other, and are jointly given in Section 8.4.2.

Interpretation of the Bounds. Ignoring logarithmic factors, the bounds are of the form

$$\mathbf{Adv}^{\mathrm{indif}}_{\text{Duplex-pi}, \mathbf{S}}(\mathcal{D}) = \tilde{\mathcal{O}}\left(\frac{\zeta^2 \mathcal{N}^2}{p^c} + \frac{\zeta^2 \mu \mathcal{N}}{p^{c''}} + \frac{\mathcal{N}}{p^{c'}}\right),$$
$$\mathbf{Adv}^{\mathrm{indif}}_{\text{Duplex-pi\$}, \mathbf{S}}(\mathcal{D}) = \tilde{\mathcal{O}}\left(\frac{\zeta^2 \mathcal{N}^2}{p^c} + \frac{\zeta^2 q_H(q_H-1)}{p^{c''}} + \frac{\zeta^2 q_H \mathcal{N}}{p^{c''}} + \frac{\mathcal{N}}{p^{c'}}\right).$$

Compared to the bounds of Sponge-pi and Sponge-pi\$, we obtain a loss quadratic in the number of NCPs. However, the actual number of required NCPs is fixed by the applications and is typically small. For example, in Section 8.5, we show that $\zeta = 2$ suffices for authenticated encryption. In this case, with appropriately chosen parameters c'' and c' (as discussed in Section 8.2.2), Duplex-pi and Duplex-pi\$ achieve indifferentiability security up to approximately $\approx p^{c/2}$ queries.

8.4 Security Proofs

8.4.1 Proofs of Theorems 8.2.1 and 8.2.2

In this section we will provide a joint proof for Theorems 8.2.1 and 8.2.2. We begin by introducing some useful notation and defining the simulator. Then, we introduce an intermediate world, decompose the distance, and bound each component.

Setup. The simulator, named **S** for Sponge-pi and **S**\$ for Sponge-pi\$, stores its primitive queries in a table `TabP`, which comprises tuples of the form (X, Y, d), where the image of X by $\mathbf{S}_P$ is defined to be Y, and the query was made in the direction $d \in \{\mathit{fwd}, \mathit{inv}\}$. Moreover, for Sponge-pi\$, the simulator has an additional oracle $\mathbf{S}\$_H$, that produces uniformly random answers, just like a random oracle h. The simulator further stores these hash queries in a table `TabH` that contains tuples of the form (m, h), where the image of m by the oracle is h. Moreover, let

$$\mathtt{Tab} = \begin{cases} \mathtt{TabP} & \text{for Sponge-pi}\,, \\ (\mathtt{TabP}, \mathtt{TabH}) & \text{for Sponge-pi\$}\,. \end{cases}$$

Let us further define a function `ValidIV()`, as defined in Algorithm 15 for both simulators. This function, when called, returns a set of tuples of the

Algorithm 15 Setup of the simulators **S** and **S$** for Sponge-pi and Sponge-pi$.

```
1: function S.Initialize()
2:     S.TabP ← $\emptyset$
3:     S.IV ← $\{(IV_m, m)\}_{m \in [\![1,\mu]\!]}$

4: function S.ValidIV()
5:     $S \leftarrow \emptyset$
6:     for all $(IV_{id}, id) \in$ S.IV do
7:         $S \leftarrow S \cup \{(0^{r''} \| IV_m, m)\}$
8:     return $S$
```

```
1: function S$.Initialize()
2:     S$.TabP ← $\emptyset$
3:     S$.TabH ← $\emptyset$

4: function S$.ValidIV()
5:     $S \leftarrow \emptyset$
6:     for all $(m, h) \in$ S$.TabH do
7:         $S \leftarrow S \cup \{(0^{r''} \| h, m)\}$
8:     return $S$
```

form (IV, m) such that $IV \in \mathbb{F}_p^b$ corresponds to a valid initial sponge state. In the case of the construction Sponge-pi, IV can be of the form $0^{r''} \| IV_m$ for all $id \in [\![1, \mu]\!]$ while for Sponge-pi$, the (IV, m)s are of the form $(0^{r''} \| h, m)$, for all $(m, h) \in$ `TabH` at the moment of the query. To keep consistency, m will be used as a label for the $\mathcal{RO}$ call that the simulator makes.

Graph Notation. From its table `Tab`, the simulator derives a tree construction. The roots are of the form $[m]$, where there exists $(IV, m) \in$ `ValidIV()`. The other nodes are elements in $\mathbb{F}_p^b$. Given $X, Y \in \mathbb{F}_p^b$, $m_p \in (\mathbb{F}_p)^{\leq r-1} \setminus \{\epsilon\}$, and $m_f \in (\mathbb{F}_p)^r$ we define four kinds of edges:

- $X \xrightarrow{m_f/a} Y$ denotes that $(X \oplus (m_f \| 0^c), Y, d) \in$ `TabP`;
- $X \longrightarrow Y$ is a special case of the above, and it denotes that $(X, Y, d) \in$ `TabP`;
- $X \xrightarrow{m_p/p} Y$ denotes that $(\pi^p (X \oplus (m_p \| 10^*)), Y, d) \in$ `TabP`;
- $X \xrightarrow{m_f/f} Y$ denotes that $(\pi^f (X \oplus (m_f \| 0^c)), Y, d) \in$ `TabP`.

The symbol a stands for absorb, while p and f indicate a last message with respectively a partial and a full block. Similarly, given $Y \in \mathbb{F}_p^b$, $(IV, m) \in$ `ValidIV()`, $m_p \in (\mathbb{F}_p)^{\leq r''-1}$, and $m_f \in (\mathbb{F}_p)^{r''}$, we define three kinds of edges between a root node $[m]$ and Y as follows:

- $[m] \xrightarrow{m_f/a} Y$ denotes that $(IV \oplus (m_f \| 0^{c''}), Y, d) \in$ `TabP`;
- $[m] \xrightarrow{m_p/p} Y$ denotes that $(\pi^p (IV \oplus (m_p \| 10^*)), Y, d) \in$ `TabP`;

- $[m] \xrightarrow{m_f/f} Y$ denotes that $\left(\pi^f \left(IV \oplus (m_f \| 0^{c''})\right), Y, d\right) \in \texttt{TabP}$.

The set of *absorbing paths* includes paths that correspond to intermediate states whose nodes would be in the real world intermediate states *during the absorbing phase.* These paths are of the form

$$[m] \xrightarrow{m_1/a} S_2 \xrightarrow{m_2/a} \cdots \xrightarrow{m_n/a} S_{n+1}\,.$$

We will abbreviate those as $[m] \xrightarrow{m_1\|\cdots\|m_n/a} S_{n+1}$. The set of *squeezing paths* include paths whose nodes would be in the real world intermediate states *during the squeezing phase.* They are of the form

$$[m] \xrightarrow{m_1/a} S_2 \xrightarrow{m_2/a} \cdots \xrightarrow{m_{n-1}/a} S_m \xrightarrow{m_n/d} Z_1 \longrightarrow Z_2 \longrightarrow \cdots \longrightarrow Z_\ell\,,$$

for $d \in \{p, f\}$. We will abbreviate this as $[m] \xrightarrow{m_1\|\cdots\|m_n,\ell} Z_\ell$. Note that the $/d$ symbol in the path is not included. This is because the message blocks are not padded in this graph representation, so that the length of $m_1\|\cdots\|m_n$ provides sufficient information to deduce whether the last call uses a full or partial block. The set of paths that are *valid* corresponds to the union of absorbing paths with squeezing paths. We make a clear distinction between them, since the simulator only needs to guarantee consistency for squeezing paths. Moreover, we define $\texttt{Rooted}(\texttt{Tab})$ as the set of rooted nodes $Z \in \mathbb{F}_p^b$ from which one can build a valid path $[m] \xrightarrow{M/a} Z$ or $[m] \xrightarrow{M,\ell} Z$.

8

Even when there are no collisions, there might be some overlap between squeezing paths and absorbing paths when π^p is the identity permutation. For instance, assume that $(IV \oplus m\|10^{c''}, S_1, \mathit{fwd}), (S_1, S_2, \mathit{fwd}) \in \texttt{TabP}$ for some $(IV, m) \in \texttt{ValidIV}()$, then this leads to two paths:

$$[m] \xrightarrow{m/p,2} S_2 \text{ and } [m] \xrightarrow{m\|1\|0^r/a} S_2\,.$$

Note that this overlap also happens with the plain sponge construction.

The Simulator. The simulator keeps track of the graph construction and uses it to output consistent answers. It is essentially the same sort of simulator as in the sponge construction [BDPV08], but it manages a more complex set of valid paths. Since Sponge-pi and Sponge-pi\$ differ only in the initial state, $\mathbf{S}_P$ and $\mathbf{S}\$_P$ (resp., $\mathbf{S}_{P^{-1}}$ and $\mathbf{S}\$_{P^{-1}}$) are the same. Algorithms 15 to 17 describe the simulators.

Algorithm 16 Simulator interfaces for Sponge-pi and Sponge-pi$, inverse permutation queries and H queries.

```
1: function $\mathbf{S}_{P^{-1}}(Y)$
2:     // The algorithm is the same for $\mathbf{S\$}_{P^{-1}}$
3:     if $\exists (X, Y) \in \mathbf{S}.\mathtt{TabP}$
4:         return $Y$
5:     $X \xleftarrow{\$} \mathbb{F}_p^b$
6:     $\mathbf{S}.\mathtt{TabP} \leftarrow \mathbf{S}.\mathtt{TabP} \cup \{(X, Y)\}$
7:     return $X$
```

```
1: function $\mathbf{S\$}_H(m)$
2:     if $\exists (m, h) \in \mathbf{S\$}.\mathtt{TabH}$
3:         return $h$
4:     $h \xleftarrow{\$} \mathbb{F}_p^{c''}$
5:     $\mathbf{S\$}.\mathtt{TabH} \leftarrow \mathbf{S\$}.\mathtt{TabH} \cup \{(m, h)\}$
6:     return $h$
```

Construction: $\mathcal{RO}$
Primitive: $(\mathbf{S}_P[\mathcal{RO}], \mathbf{S}_{P^{-1}})$

W_I (Sponge-pi)

Construction: $\mathcal{RO}$
Primitive: $(\mathbf{S}_P[\mathcal{RO}], \mathbf{S}_{P^{-1}}, \mathbf{S\$}_H)$

W_I (Sponge-pi$)

Construction: $\mathcal{C}^{\mathbf{S}_P[\mathcal{RO}]}$
Primitive: $(\mathbf{S}_P[\mathcal{RO}], \mathbf{S}_{P^{-1}})$

W_{IM} (Sponge-pi)

Construction: $\mathcal{C}^{\mathbf{S}[\mathcal{RO}]}$
Primitive: $(\mathbf{S}_P[\mathcal{RO}], \mathbf{S}_{P^{-1}}, \mathbf{S\$}_H)$

W_{IM} (Sponge-pi$)

Construction: $\mathcal{C}^{\mathcal{P}}$
Primitive: $(\mathcal{P}, \mathcal{P}^{-1})$

W_R (Sponge-pi)

Construction: $\mathcal{C}^{\mathcal{P},h}$
Primitive: $(\mathcal{P}, \mathcal{P}^{-1}, h)$

W_R (Sponge-pi$)

Figure 8.3: Worlds involved in the security proof.

World Splitting. Similarly to [NO14], we introduce an intermediate world W_{IM}, as illustrated in Figure 8.3. In this world, the construction oracle gives access to Sponge-pi or Sponge-pi$, based on the simulator, which is itself based on a random oracle hidden from the adversary. When dealing with Sponge-pi$, W_{IM} additionally includes the simulator $\mathbf{S\$}_H$. For the sake of the proof, we introduce a table $\widehat{\mathtt{TabP}}$, that comprises tuples of the form (X, Y, d, O), where $O = D$ means that the query was from the distinguisher, else $O = C$ means that the query comes from the construction. Similarly with Sponge-pi$, define $\widehat{\mathtt{TabH}}$ as an extension of $\mathtt{TabH}$, that comprises tuples of the form (I, h, O), where the image of I by h is h, and the query comes from either the distinguisher (i.e., $O = D$), or the construction (i.e., $O = C$). With these extended tables, we can differentiate the set of rooted nodes: $\mathtt{Rooted}_D(\widehat{\mathtt{Tab}})$ represents the rooted

Algorithm 17 Simulator interfaces for Sponge-pi and Sponge-pi\$, forward permutation queries.

```
function $\mathbf{S}_P(X)$
    // The algorithm is the same for $\mathbf{S\$}_P$
    if $\exists (X, Y) \in \mathbf{S}.\mathtt{TabP}$
        return $X$
    $Y \xleftarrow{\$} \mathbb{F}_p^b$
    // Start of squeezing phase with partial block
    if $\exists (IV, m) \in \mathbf{S}.\mathtt{ValidIV}()$, $S \in \mathbb{F}_p^b$, $M \in \mathbb{F}_p^* \setminus \{\epsilon\}$, and $m \in \mathbb{F}_p^{\leq r-1} \setminus \{\epsilon\}$
    such that $[m] \xrightarrow{M/a} S$ and $X = \pi^p\left(S \oplus m \| 10^*\right)$
        $Y_o \leftarrow \mathcal{RO}((m, M\|m), r')$
        $Y_i \xleftarrow{\$} \mathbb{F}_p^{c'}$
        $Y \leftarrow Y_o \,\|\, Y_i$
    // Start of squeezing phase with full block
    if $\exists (IV, m) \in \mathbf{S}.\mathtt{ValidIV}()$, $S \in \mathbb{F}_p^b$, $M \in \mathbb{F}_p^* \setminus \{\epsilon\}$, and $m \in \mathbb{F}_p^r$ such
    that $[m] \xrightarrow{M/a} S$ and $X = \pi^f\left(S \oplus m \| 0^c\right)$
        $Y_o \leftarrow \mathcal{RO}((m, M\|m), r')$
        $Y_i \xleftarrow{\$} \mathbb{F}_p^{c'}$
        $Y \leftarrow Y_o \,\|\, Y_i$
    // In the middle of a squeezing phase
    if $\exists (IV, m) \in \mathbf{S}.\mathtt{ValidIV}()$, $M \in \mathbb{F}_p^*$, and $\ell \geq 1$ such that $[m] \xrightarrow{M,\ell} X$
        $Y_o \leftarrow \mathcal{RO}((m, M), (\ell+1)r')[\ell r' : (\ell+1)r' - 1]$
        $Y_i \xleftarrow{\$} \mathbb{F}_p^{c'}$
        $Y \leftarrow Y_o \,\|\, Y_i$
    // Start of squeezing phase from IV with one partial block
    if $\exists (IV, m) \in \mathbf{S}.\mathtt{ValidIV}()$ and $m \in \mathbb{F}_p^{\leq r''-1}$ such that $X =$
    $\pi^p\left(IV \oplus m \| 10^*\right)$
        $Y_o := \mathcal{RO}((m, m), r')$
        $Y_i \xleftarrow{\$} \mathbb{F}_p^{c'}$
        $Y \leftarrow Y_o \,\|\, Y_i$
    // Start of squeezing phase from IV with one full block
    if $\exists (IV, m) \in \mathbf{S}.\mathtt{ValidIV}()$ and $m \in \mathbb{F}_p^{r''}$ such that $X =$
    $\pi^f\left(IV \oplus m \| 0^{c''}\right)$
        $Y_o := \mathcal{RO}((m, m), r')$
        $Y_i \xleftarrow{\$} \mathbb{F}_p^{c'}$
        $Y \leftarrow Y_o \,\|\, Y_i$
    $\mathbf{S}.\mathtt{TabP} \leftarrow \mathbf{S}.\mathtt{TabP} \cup \{(X, Y)\}$
    return $Y$
```

nodes reached solely through distinguisher queries.

We have

$$\begin{aligned}\mathbf{Adv}^{\text{indif}}_{\mathcal{C},\mathbf{S}}(\mathcal{D}) &\le \Delta_{\mathcal{D}}\left(W_R\,;\,W_{IM}\right) &(8.1)\\ &+ \Delta_{\mathcal{D}}\left(W_{IM}\,;\,W_I\right). &(8.2)\end{aligned}$$

We first bound (8.1), which boils down to evaluating the distance between the simulator and a random permutation. Remark that $\mathbf{S}\$_H$ is a random oracle that is independent from $\mathbf{S}_P$ and $\mathbf{S}_{P^{-1}}$. Then, since $\mathbf{S}_P$ makes random oracle calls with disjoint inputs for any fresh query, it behaves like a random function. This distance can thus be upper bounded by the PRP/PRF switching lemma, and we obtain

$$\Delta_{\mathcal{D}}\left(W_R\,;\,W_{IM}\right) \le \frac{\mathcal{N}(\mathcal{N}-1)}{2p^b}\,. \tag{8.3}$$

The remainder of the proof is dedicated to the analysis of (8.2).

Bad Events. Let Q denote the total number of queries made by the distinguisher, so that $Q = \mathcal{N}$ for Sponge-pi, and $Q = \mathcal{N} + q_H$ for Sponge-pi\$. For $i \in [\![1, Q]\!]$, let $\texttt{TabP}[i]$ and $\texttt{TabH}[i]$ denote the state of respectively TabP and TabH after the first i queries. We define a family of bad events, indexed by a query index $i \in [\![1, Q]\!]$, and defined over Tab. Sponge-pi and Sponge-pi\$ only diverge in the definition of bad events related to the initial values.

- CollP_i: $\exists(X, Y, d) \in \texttt{TabP}[i] \setminus \texttt{TabP}[i-1]$, $(X', Y', d') \in \texttt{TabP}[i-1]$ such that

$$\left\{\text{inner}_c(Z), \pi^p\left(\text{inner}_c(Z)\right), \pi^f\left(\text{inner}_c(Z)\right)\right\} \cap \left\{\text{inner}_c(Z'), \pi^p\left(\text{inner}_c(Z')\right), \pi^f\left(\text{inner}_c(Z')\right)\right\} \neq \emptyset\,,$$

  where

$$(Z, Z') = \begin{cases}(Y, Y') & \text{if } d = \mathit{fwd}\,,\\ (X, X') & \text{otherwise}\,;\end{cases}$$

- $\mathsf{ConnectP}_i$: $\exists(X, Y, d) \in \texttt{TabP}[i] \setminus \texttt{TabP}[i-1]$, $(X', Y', d') \in \texttt{TabP}[i-1]$ such that

$$\left\{\text{inner}_c(Z), \pi^p\left(\text{inner}_c(Z)\right), \pi^f\left(\text{inner}_c(Z)\right)\right\} \cap \left\{\text{inner}_c(\bar{Z}'), \pi^p\left(\text{inner}_c(\bar{Z}')\right), \pi^f\left(\text{inner}_c(\bar{Z}')\right)\right\} \neq \emptyset\,,$$

where

$$(Z, \bar{Z}') = \begin{cases} (Y, X') & \text{if } d = \mathit{fwd}\,, \\ (X, Y') & \text{otherwise}\,; \end{cases}$$

- Guess_i: $\exists (X, Y, d) \in \mathtt{TabP}[i] \setminus \mathtt{TabP}[i-1]$; such that $Y \notin \mathtt{Rooted}_D(\widehat{\mathtt{Tab}}[i])$, but $Y \in \mathtt{Rooted}(\widehat{\mathtt{Tab}}[i])$;
- $\mathsf{ConnectIV}_i$ (for Sponge-pi only): $\exists (X, Y, d) \in \mathtt{TabP}[i]$, $id \in [\![1, \mu]\!]$ such that

$$\left\{\text{inner}_{c''}(Z), \pi^p\left(\text{inner}_{c''}(Z)\right), \pi^f\left(\text{inner}_{c''}(Z)\right)\right\} \cap \left\{IV_{id}, \pi^p\left(IV_{id}\right), \pi^f\left(IV_{id}\right)\right\} \neq \emptyset\,,$$

where

$$Z = \begin{cases} Y & \text{if } d = \mathit{fwd}\,, \\ X & \text{otherwise}\,; \end{cases}$$

- CollH_i (for Sponge-pi\$ only): $\exists (m, h) \in \mathtt{TabH}[i] \setminus \mathtt{TabH}[i-1]$, $(m', h') \in \mathtt{TabH}[i-1]$ such that

$$\left\{h, \pi^p(h), \pi^f(h)\right\} \cap \left\{h', \pi^p(h'), \pi^f(h')\right\} \neq \emptyset\,;$$

- $\mathsf{ConnectPH}_i$ (for Sponge-pi\$ only):
  - either $\exists (X, Y, d) \in \mathtt{TabP}[i] \setminus \mathtt{TabP}[i-1]$, $(m, h) \in \mathtt{TabH}[i-1]$ such that

$$\left\{h, \pi^p(h), \pi^f(h)\right\} \cap \left\{\text{inner}_{c''}(Z), \pi^p\left(\text{inner}_{c''}(Z)\right), \pi^f\left(\text{inner}_{c''}(Z)\right)\right\} \neq \emptyset\,,$$

    where

$$Z = \begin{cases} Y & \text{if } d = \mathit{fwd}\,, \\ X & \text{otherwise}\,, \end{cases}$$

  - or $\exists (m, h) \in \mathtt{TabH}[i] \setminus \mathtt{TabH}[i-1]$, $(X, Y, d) \in \mathtt{TabP}[i-1]$ such that there exists $Z \in \{X, Y\}$ with

$$\left\{h, \pi^p(h), \pi^f(h)\right\} \cap \left\{\text{inner}_{c''}(Z), \pi^p\left(\text{inner}_{c''}(Z)\right), \pi^f\left(\text{inner}_{c''}(Z)\right)\right\} \neq \emptyset\,.$$

We define BadIV_i as follows:

$$\mathsf{BadIV}_i = \begin{cases} \mathsf{ConnectIV}_i & \text{for Sponge-pi}\,, \\ \mathsf{CollH}_i \lor \mathsf{ConnectPH}_i & \text{for Sponge-pi\$}\,, \end{cases}$$

and $\mathsf{Bad}_i := \mathsf{CollP}_i \lor \mathsf{ConnectP}_i \lor \mathsf{BadIV}_i \lor \mathsf{Guess}_i$. Moreover, for any of those bad events Event_i, let Event be

$$\bigvee_{i=1}^{Q} \mathsf{Event}_i\,.$$

The bad events $\mathsf{CollP} \lor \mathsf{ConnectP} \lor \mathsf{BadIV}$ ensure that the simulator responds consistently with respect to the random oracle. Intuitively, CollP addresses collisions in the inner part of the construction, modulo the application of the NCPs, ensuring that each simulator query is mapped to at most one squeezing path. The absence of $\mathsf{ConnectP}$ guarantees that an unrooted path cannot later become rooted, which is crucial for maintaining consistency in the simulator's responses. Finally, BadIV ensures that the rightmost c'' field elements of the initial values appear only at the roots of the tree construction and that the simulator does not connect to the IVs via inverse calls. Additionally, for Sponge-pi\$, this bad event ensures that a hash evaluation does not inadvertently create a new valid path and that the hash outputs adhere to constraint 2.

W_{IM} Versus W_I as Long as No Bad. We will show that, conditioned on $\neg\mathsf{Bad}$, the worlds W_I and W_{IM} have the same distribution. This requires two steps. The first step, detailed in Lemma 8.4.1, shows that as long as no bad event occurs, the simulator in W_{IM} provides consistent answers. However, this alone is not sufficient because the simulator in W_{IM} has more information than the one in W_I. To address this, we need to ensure that the additional knowledge available to the simulator in W_{IM} does not provide it with extra information. Intuitively, this is ensured by the absence of Guess. Lemma 8.4.2 makes explicit that W_{IM} and W_I behave identically until Bad occur.

8

Lemma 8.4.1. *Let $\mathcal{C}$ be either the* Sponge-pi *or* Sponge-pi\$ *construction, based on the simulator, which is itself based on a random oracle $\mathcal{RO}$. For any $M \in p^*$ such that there exists a valid path $[m] \xrightarrow{M,\ell} Z$ in* `Tab`*, where $(IV, m) \in$* `ValidIV()`*, we have*

$$\mathcal{C}(m, M, \ell r') = \mathcal{RO}((m, M), \ell r')\,.$$

Proof. By $\neg\mathsf{ConnectP} \wedge \neg\mathsf{BadIV}$, the valid paths are expanded from left to right, and only with forward queries. For instance, if we have a valid path $[m] \xrightarrow{m_1/a} S_1 \xrightarrow{m_2/a} S_2$ with $m_1 \in (\mathbb{F}_p)^{r''}$, $m_2 \in (\mathbb{F}_p)^r$, then the following queries were made in order: (i) (for the case of Sponge-pi\$) query to $\mathbf{S}\$_H$ with input m, and output IV_m; (ii) query to $\mathbf{S}_P$ with input $0^{r''} \| IV_{id} \oplus m_1 \| 0^{c''}$, and output S_1; (iii) query to $\mathbf{S}_P$ with input $S_1 \oplus m_2 \| 0^c$, and output S_2.

Keeping this in mind, we argue next that $\neg\mathsf{Bad}$ prevents from having two valid paths of the form $[m_1] \xrightarrow{P_1} S_1$ and $[m_2] \xrightarrow{P_2} S_2$ with $S_1 \neq S_2$, but such that a subsequent forward query with input X and output Y extends these two paths into two valid, squeezing paths that look like $[m_1] \xrightarrow{P_1'} Y$ and $[m_2] \xrightarrow{P_2'} Y$. The list of possible valid paths $[m_e] \xrightarrow{P_e} S_e$ for $e \in \{1, 2\}$ and possibilities for X are the following:

a. $\exists M_e \in \mathbb{F}_p^* \setminus \{\epsilon\}$, $m_e \in \mathbb{F}_p^{\leq r-1} \setminus \{\epsilon\}$, and $[m_e] \xrightarrow{M_e/a} S_e$ such that $X = \pi^p(S_e \oplus m \| 10^*)$;

b. $\exists M_e \in \mathbb{F}_p^* \setminus \{\epsilon\}$, $m \in \mathbb{F}_p^r$, and $[m_e] \xrightarrow{M_e/a} S_e$ such that $X = \pi^f(S_e \oplus m \| 0^c)$;

c. $\exists M_e \in \mathbb{F}_p^*, \ell \geq 1$, and $[m_e] \xrightarrow{M_e, \ell} S_e$ such that $X = S_e$;

d. $\exists m \in \mathbb{F}_p^{\leq r''-1}$, $(IV_e, m_e) \in \mathtt{ValidIV}()$ such that $X = \pi^p (IV_e \oplus m \| 10^*)$ (i.e., $S_e = IV_e$);

e. $\exists m \in \mathbb{F}_p^{r''}$, $(IV_e, m_e) \in \mathtt{ValidIV}()$ such that $X = \pi^f \left(IV_e \oplus m \| 0^{c''} \right)$ (i.e., $S_e = IV_e$).

The initial values IV_e are never present in the middle of a path thanks to $\neg\mathsf{BadIV}$. Moreover, thanks to constraint 2 for Sponge-pi and $\neg\mathsf{CollH}$ for Sponge-pi\$, it is not possible that two different initial values are bound to the same permutation call. Moreover, $\neg\mathsf{CollP}$ prevents that two forward permutation calls collide on their inner part modulo application of the permutations π^d. Considering all possible combinations, we conclude that having two valid, converging paths as described above is impossible.

Now, assume that we have two paths sharing the same node, i.e., there exists $S \in \mathbb{F}_p^b, (IV_1, m_1), (IV_2, m_2) \in \mathtt{ValidIV}()$ and two path labels P_1, P_2 with $(id_1, P_1) \neq (id_2, P_2)$ such that $[m_1] \xrightarrow{P_1} S$ and $[m_2] \xrightarrow{P_2} S$. Thanks to the previous paragraph, and $\neg\mathsf{BadIV}$, this means that the intermediate states of both paths are pairwise the same, but that the paths can be read differently. First of all, remark that with constraint 2 for Sponge-pi and $\neg\mathsf{CollH}$

for Sponge-pi$ we must have $m_1 = m_2$. Finally, thanks to constraint 1 and the absence of Bad, this overlapping does not involve confusing $\cdot/f$ with $\cdot/p$ paths and $\cdot/f$ with $\cdot/a$ paths. In the event where $\pi^p(x) = x$, this leaves us with the only possibility where path P_1 is still in the absorption phase, while path P_2 is in the squeezing phase. For such a scenario, in order to extend P_1 into a squeezing path (which would require consistent answers), either the permutation π^f, or a non-zero message block must be added/applied to the state, which will therefore yield to an extended path P_1^* that diverges from any valid extension of P_2. This is therefore not problematic for the consistency.

To summarize, as long as no Bad occurs, we only have to take care of the consistency of the answers of the forward simulator. This simulator, in Algorithms 16 and 17 contains 5 "if" conditions, which cover all possible scenarios where the query extends any path to a squeezing path. These "if" conditions are analyzed in the second paragraph of the current proof, and we showed as long as no Bad occurs there is at most one path that satisfies them. Moreover, such a path can only satisfy one of the "if" conditions. Therefore, the simulator is consistent with respect to the random oracle. □

We finally argue that W_I and W_{IM} are identically distributed as long as no bad occurs.

Lemma 8.4.2. *We have*

$$\mathbf{Pr}\left(\mathcal{D}^{W_I} = 1 \mid \neg\mathsf{Bad}\right) = \mathbf{Pr}\left(\mathcal{D}^{W_{IM}} = 1 \mid \neg\mathsf{Bad}\right) .$$

Proof. We already proved in Lemma 8.4.1 that the simulator in W_{IM} replies consistently with respect to the random oracle, but this is insufficient. Indeed, another difference between W_{IM} and W_I is that W_{IM} is indirectly aware of the construction queries made by the adversary. If we were in the setting of public indifferentiability [YMO09, DRS09, MPS12], this would not be a problem, as in that setting the simulator in the ideal world is aware of the construction queries. However, public indifferentiability is weaker than indifferentiability, and we need to further argue that ¬Bad makes them not distinguishable. The bad event Guess guarantees that the simulator cannot guess a rooted path that comes from construction queries. For instance, if there exists $(IV, m) \in \texttt{ValidIV}()$ and a valid path $[m] \xrightarrow{m_2/a} S_1 \xrightarrow{m_2/f} S_2$ that was generated only from construction queries, then ¬Guess guarantees that the adversary will never make an inverse query with input S_1 or S_2, nor a forward query with input $\pi^f(S_1 \oplus m_2\|0^c)$ *before* the forward query with input $IV \oplus m_2\|0^{c''}$ was made. In that case, the fact that the simulator has earlier access to the construction queries in W_{IM} does not change the distribution of the answers from he point of view of the adversary. Thus

$$\mathbf{Pr}\left(\mathcal{D}^{W_I} = 1 \mid \neg\mathsf{Bad}\right) = \mathbf{Pr}\left(\mathcal{D}^{W_{IM}} = 1 \mid \neg\mathsf{Bad}\right) . \qquad \square$$

Probability of Bad. We showed that W_{IM} and W_I behave identically as long as no bad event occurs. What remains is to bound the probability of the bad events, which we do in Lemma 8.4.3.

Lemma 8.4.3. *For* $\mathtt{X} \in \{\mathtt{S}, \mathtt{I}\}$*, with the construction* Sponge-pi*, we have*

$$\mathbf{Pr}\left(\mathcal{D}^{W_X} \text{ sets } \mathsf{Bad}\right) \leq \frac{7\mathcal{N}(\mathcal{N}-1)}{p^c} + \frac{7\mu\mathcal{N}}{2p^{c''}} + \frac{\mathrm{mucol}(\mathcal{N}, p^{r'-r})\mathcal{N}}{p^{c'}} .$$

For $\mathtt{X} \in \{\mathtt{S}, \mathtt{I}\}$*, with the construction* Sponge-pi\$*, we have*

$$\mathbf{Pr}\left(\mathcal{D}^{W_X} \text{ sets } \mathsf{Bad}\right) \leq \frac{7\mathcal{N}(\mathcal{N}-1)}{p^c} + \frac{7q_H(q_H-1)}{2p^{c''}} + \frac{22\mathcal{N}q_H}{p^{c''}} + \frac{\mathrm{mucol}(\mathcal{N}, p^{r'-r})\mathcal{N}}{p^{c'}} .$$

Proof. The bad events are defined over the simulator table `Tab`, and Guess can only be set in W_{IM}. Moreover, the simulator receives extra queries in W_{IM}, which only increases its success probability to set Bad. We will thus evaluate the probability of Bad in W_{IM}. A “query” here refers to a call to the simulator, regardless whether this comes from the adversary or the construction. We have

$$\begin{aligned}\mathbf{Pr}_{W_{IM}}(\mathsf{Bad}) &\leq \sum_{i=1}^{Q} \mathbf{Pr}\left(\mathsf{Bad}_i \mid \neg\mathsf{Bad}_{i-1}\right) \\ &\leq \sum_{i=1}^{Q} \mathbf{Pr}\left(\mathsf{CollP}_i \mid \neg\mathsf{Bad}_{i-1}\right) + \mathbf{Pr}\left(\mathsf{ConnectP}_i \mid \neg\mathsf{Bad}_{i-1}\right) \\ &\qquad + \mathbf{Pr}\left(\mathsf{BadIV}_i \mid \neg\mathsf{Bad}_{i-1}\right) + \mathbf{Pr}\left(\mathsf{Guess}_i \mid \neg\mathsf{Bad}_{i-1}\right) , \end{aligned} \tag{8.4}$$

where Bad_0 denotes an event that never holds. We will evaluate each probability individually. First of all, it is important to note that as long as $\neg\mathsf{Bad}$ holds, the simulator samples its answers uniformly at random, either with direct sampling, or samples the outer part with $\mathcal{RO}$ calls, and the inner part by itself. In the latter case, the same $\mathcal{RO}$ entry is never accessed for two different inputs.

Given $i \in [\![1, Q]\!]$, let i_P denote the number of queries to $\mathbf{S}_P$ or $\mathbf{S}_{P^{-1}}$ made before the i^{th} query, and i_H the number of queries to $\mathbf{S\$}_H$ made before the i^{th} query. Moreover, let $\mathbf{1}_P^i$ (resp., $\mathbf{1}_H^i$) denote the indicator function equal to 1 whenever the i^{th} query is a query to $\mathbf{S}_P$ or $\mathbf{S}_{P^{-1}}$ (resp., $\mathbf{S\$}_H$).

CollP_i**.** This bad event concerns collisions over c field elements. For each old query (X', Y'), the bad event covers 7 different collision scenarios (noting that $\pi^d(x) = \pi^d(x')$ and $x = x'$ are the same case). Therefore,

$$\mathbf{Pr}_{W_{IM}}\left(\mathsf{CollP}_i \mid \neg\mathsf{Bad}_{i-1}\right) \leq \mathbf{1}_P^i \frac{7 i_P}{p^c} . \tag{8.5}$$

$\mathsf{ConnectP}_i$**.** Similarly, we have

$$\mathbf{Pr}_{W_{IM}}\left(\mathsf{ConnectP}_i \mid \neg\mathsf{Bad}_{i-1}\right) \leq \mathbf{1}_P^i \frac{7 i_P}{p^c} . \tag{8.6}$$

BadIV_i **with** Sponge-pi**.** This bad event concerns colliding with one of the μ initial values. For each of these values, there are at most 7 possible collision scenarios, over c'' field elements. Therefore,

$$\mathbf{Pr}_{W_{IM}}\left(\mathsf{BadIV}_i \mid \neg\mathsf{Bad}_{i-1}\right) = \mathbf{Pr}_{W_{IM}}\left(\mathsf{ConnectIV}_i \mid \neg\mathsf{Bad}_{i-1}\right) \leq \mathbf{1}_P^i \frac{7\mu}{p^{c''}} . \tag{8.7}$$

BadIV_i **with** Sponge-pi\$**.** We have

$$\mathbf{Pr}\left(\mathsf{CollH}_i \mid \neg\mathsf{Bad}_{i-1}\right) \leq \mathbf{1}_H^i \frac{7 i_H}{p^{c''}} .$$

$\mathsf{ConnectPH}_i$ also concerns collisions on c'' field elements. Assuming $\neg\mathsf{Bad}_{i-1}$, we have two possibilities:

- The i^{th} query is a hash query. There are at most $14 i_P$ values from `TabP` to hit. The hash answer is uniformly random, thus the query sets $\mathsf{ConnectPH}$ with a probability of at most $\mathbf{1}_H^i \frac{14 i_P}{p^{c''}}$;

- The i^{th} query is a query to $\mathbf{S}_P$ or $\mathbf{S}_{P^{-1}}$. Then the uniformly random answer collides with a prior hash output with probability of at most $\mathbf{1}_P^i \frac{7 i_H}{p^{c''}}$.

Therefore:

$$\mathbf{Pr}_{W_{IM}}\left(\mathsf{BadIV}_i \mid \neg\mathsf{Bad}_{i-1}\right) \leq \mathbf{1}_H^i \frac{7 i_H}{p^{c''}} + \mathbf{1}_H^i \frac{14 i_P}{p^{c''}} + \mathbf{1}_P^i \frac{7 i_H}{p^{c''}} . \tag{8.8}$$

Guess_i**.** Setting this event is similar to a guessing game: in order to win, the adversary must be able to guess a node in a valid path without having constructed the path from left to right with its primitive queries. First of all, with Sponge-pi\$, this case can occur if the adversary makes a forward query with input X, where there exists $(I, h, C) \in \widehat{\mathtt{TabH}}[i-1]$, but there exists no $(I, h, m) \in \widehat{\mathtt{TabH}}[i-1]$ such that $\text{inner}_{c''}(X) = h$. The simulator is a random oracle, thus the values h are generated uniformly at random, and hidden from the adversary as long as it does not make the queries. Therefore, we obtain a probability of at most

$$\mathbf{1}_P^i \frac{i_H}{p^{c''}} .$$

The remaining cases consist of guessing nodes that were generated by queries to $\mathbf{S}_P$ from the construction Sponge-pi or Sponge-pi\$. Thanks to the construction oracle, $\mathcal{D}$ has access to the r' upper field elements of at most $\mathcal{N}$ different nodes. Given $u \in (\mathbb{F}_p)^{r'}$, we define the random variable F_u as follows:

$$F_u := \left| \{(x, y, fwd, C) \in \widehat{\mathtt{TabP}} \mid \text{outer}_{r'}(y) = u\} \right| ,$$

i.e., F_u is the number of construction queries which outer part hit u. Now, given a query $v \| w$ with $v \in (\mathbb{F}_p)^{r'}$ and $w \in (\mathbb{F}_p)^{c'}$, the probability that Guess_i is set, conditioned on the query history of the $i-1$ previous queries $\mathtt{TabP}$, is upper bounded by

$$\frac{\max_{u \in (\mathbb{F}_p)^{r'}} F_u}{p^{c'}} .$$

Therefore, by summing over all possible $\mathtt{TabP}$ we obtain

$$\mathbf{Pr}_{W_{IM}}\left(\mathsf{Guess}_i \mid \neg \mathsf{Bad}_{i-1}\right) \leq \frac{\mathsf{E}\left(\max_{u \in (\mathbb{F}_p)^{r'}} F_u\right)}{p^{c'}} .$$

In fact, the distribution of the random variables $(F_u)_u$ is the same as the bins-and-balls experiment described in Lemma 2.5.2. Since the adversary is allowed to overwrite the outer r field elements of the states, the worst-case scenario corresponds to throwing $\mathcal{N}$ balls into $p^{r'-r}$ bins. Therefore,

$$\mathsf{E}\left(\max_{a \in (\mathbb{F}_p)^{r'}} F_u\right) \leq \text{mucol}(\mathcal{N}, p^{r'-r}) .$$

so that

$$\mathbf{Pr}_{W_{IM}}\left(\mathsf{Guess}_i \mid \neg\mathsf{Bad}_{i-1}\right) \\ \leq \begin{cases} \mathbf{1}_P^i \frac{\mathrm{mucol}(\mathcal{N},p^{r'-r})}{p^{c'}} \text{ for Sponge-pi}\,, \\ \mathbf{1}_P^i \frac{\mathrm{mucol}(\mathcal{N},p^{r'-r})}{p^{c'}} + \mathbf{1}_P^i \frac{i_H}{p^{c''}} \text{ for Sponge-pi\$}\,. \end{cases} \tag{8.9}$$

Conclusion. For Sponge-pi, plugging (8.5) to (8.7) and (8.9) into (8.4) gives

$$\mathbf{Pr}_{W_{IM}}\left(\mathsf{Bad}\right) \leq \frac{7\mathcal{N}(\mathcal{N}-1)}{p^c} + \frac{7\mu\mathcal{N}}{2p^{c''}} + \frac{\mathrm{mucol}(\mathcal{N},p^{r'-r})\mathcal{N}}{p^{c'}}\,.$$

For Sponge-pi\$, by plugging (8.5), (8.6), (8.8) and (8.9) into (8.4), we obtain

$$\mathbf{Pr}_{W_{IM}}\left(\mathsf{Bad}\right) \leq \frac{7\mathcal{N}(\mathcal{N}-1)}{p^c} + \frac{7q_H(q_H-1)}{2p^{c''}} + \frac{22\mathcal{N}q_H}{p^{c''}} + \frac{\mathrm{mucol}(\mathcal{N},p^{r'-r})\mathcal{N}}{p^{c'}}\,,$$

hence the lemma. □

Conclusion. From Lemma 8.4.2, and using the fundamental lemma of game-playing [BR06],

$$\Delta_{\mathcal{D}}\left(W_{IM}\,;\,W_I\right) \leq \mathbf{Pr}_{W_{IM}}\left(\mathsf{Bad}\right)\,. \tag{8.10}$$

We can use Lemma 8.4.3 to upper bound this term. We plug the obtained bound into (8.1), along with (8.3) into (8.2), and this concludes the theorem.

8.4.2 Proofs of Theorems 8.3.1 and 8.3.2

Although it is not possible to reduce the security of Duplex-pi and Duplex-pi\$ to the one of Sponge-pi and Sponge-pi\$, the underlying proofs share many similarities. The graphical representation of the simulator's query history slightly differs, as each duplex call squeezes data. As a consequence, there are no absorbing paths, and no overlap between different paths is possible unless a bad event occurs. Moreover, due to the presence of multiple NCPs, the number of path types is multiplied by a factor of ζ. This will increase the number collision combinations in the bad events, hence the quadratical factor of the form ζ^2 in the bounds. Another distinction is that, unlike in the Sponge-pi and Sponge-pi\$ settings, the idealized object here is *stateful*. However, due to the way the distinguisher resources are measured, we can resort to a step akin to Bertoni et al. [BDPV11b] to serialize the `ORO` calls, and obtain direct access to the underlying $\mathcal{RO}$ at no cost.

Algorithm 18 Setup of the simulators **S** and **S**\$ for Duplex-pi and Duplex-pi\$.

```
1: function S.Initialize()
2:     S.TabP ← $\emptyset$
3:     S.$G$ ← $\emptyset$
4:     S.IV ← $\{(IV_m, m)\}_{m \in [\![1,\mu]\!]}$

5: function S.ValidIV()
6:     $S \leftarrow \emptyset$
7:     for all $(IV_{id}, id) \in$ S.IV do
8:         $S \leftarrow S \cup \{(0^{r''} \| IV_m, m)\}$
9:     return $S$
```

```
1: function S$.Initialize()
2:     S$.TabP ← $\emptyset$
3:     S.$G$ ← $\emptyset$
4:     S$.TabH ← $\emptyset$

5: function S$.ValidIV()
6:     $S \leftarrow \emptyset$
7:     for all $(m, h) \in$ S$.TabH do
8:         $S \leftarrow S \cup \{(0^{r''} \| h, m)\}$
9:     return $S$
```

Setup. The setup is defined very similarly as the one of Section 8.4.1, except that the simulator maintains an additional structure named G that will be useful later. The setup algorithms are specified in Algorithm 18, where **S** and **S**\$ denote respectively the simulator for Duplex-pi and Duplex-pi\$.

Graph Notation. From its table `Tab`, the simulator derives a tree construction, similar to the one in the proof of Theorems 8.2.1 and 8.2.2. However, this time every *valid* path corresponds to a *squeezing* path, and the paths are overloaded with the domain separator indexes. Given $X, Y \in \mathbb{F}_p^b$, $m_p \in (\mathbb{F}_p)^{\leq r-1}$, $m_f \in (\mathbb{F}_p)^r$, and $i \in [\![1, \zeta]\!]$, we define two kinds of edges:

- $X \xrightarrow{m_f/i} Y$ denotes that $(\pi_i^f (X \oplus (m \| 0^c)), Y, d) \in$ `TabP`;
- $X \xrightarrow{m_p/i} Y$ denotes that $(\pi_i^p (X \oplus (m \| 10^*)), Y, d) \in$ `TabP`.

Similarly, given $Y \in \mathbb{F}_p^b$, $(IV, m) \in$ `ValidIV()`, $m_p \in (\mathbb{F}_p)^{\leq r''-1}$, and $m_f \in (\mathbb{F}_p)^{r''}$, we define two kinds of edges between a root node $[m]$ and Y as follows:

- $[m] \xrightarrow{m_f/i} Y$ denotes that $(\pi_i^f \left(IV \oplus (m \| 0^{c''})\right), Y, d) \in$ `TabP`;
- $[m] \xrightarrow{m_p/i} Y$ denotes that $(\pi_i^p (IV \oplus (m \| 10^*)), Y, d) \in$ `TabP`.

The set of *valid paths* is the set of paths that correspond to intermediate states that could be derived from a Duplex-pi (or Duplex-pi\$) call. They are of the form

$$[m] \xrightarrow{m_1/i_1} S_2 \xrightarrow{m_2/i_2} \cdots \xrightarrow{m_n/i_n} S_{n+1} .$$

Algorithm 19 Simulator interfaces for Duplex-pi and Duplex-pi\$, inverse queries, hash queries, and auxiliary function.

```
1: function $\mathbf{S\$}_H(m)$
2:     if $\exists(m,h) \in \mathbf{S\$}.\mathtt{TabH}$
3:         return $h$
4:     $h \xleftarrow{\$} \mathbb{F}_p^{c''}$
5:     $\mathbf{S\$}.\mathtt{TabH} \leftarrow \mathbf{S\$}.\mathtt{TabH} \cup \{(m,h)\}$
6:     return $h$
```

```
1: function $\mathbf{S}_{P^{-1}}(Y)$
2:     // The algorithm is the same for $\mathbf{S\$}_{P^{-1}}$
3:     if $\exists(X,Y) \in \mathbf{S}.\mathtt{TabP}$
4:         return $Y$
5:     $X \xleftarrow{\$} \mathbb{F}_p^b$
6:     $\mathbf{S}.\mathtt{TabP} \leftarrow \mathbf{S}.\mathtt{TabP} \cup \{(X,Y)\}$
7:     return $X$
```

```
1: function Serialize$(m,P)$
2:     Parse $P$ as $(M_1,i_1),\dots,(M_\ell,i_\ell)$
3:     if $\ell > 1$ and $(M_1,i_1),\dots,(M_{\ell-1},i_{\ell-1}) \notin \mathbf{S}.G$
4:         return $\perp$
5:     $(Z_1,u) \leftarrow$ ORO.initialize$(m,M_1,i_1)$
6:     for $l = 2,\dots,\ell$ do
7:         $Z_l \leftarrow$ ORO.duplexing$(M_l,i_l)$
8:     $\mathbf{S}.G \leftarrow \mathbf{S}.G$.append$((m,P))$
9:     return $Z_\ell$
```

We will abbreviate those as $[m] \xrightarrow{(m_1,i_1),(m_2,i_2),\dots,(m_n,i_n)} S_{n+1}$. Moreover, we define Rooted(Tab) as the set of rooted nodes $Z \in \mathbb{F}_p^b$, from which one can build a valid path $[m] \xrightarrow{P} Z$.

Contrarily to the setting in Theorems 8.2.1 and 8.2.2, there cannot be an overlap between two distinct paths as long as no bad event occurs. In other words, one cannot have $[m_1] \xrightarrow{P_1} S$ and $[m_2] \xrightarrow{P_2} S$ where $(m_1,P_1) \neq (m_2,P_2)$.

Serializing the ORO and Simulator Definition. As we explained in Section 8.3.2, we count the number of distinguisher queries without doubly counting repeating paths. Thanks to this metric, the simulator can serialize the calls to the ORO with the function named Serialize() from Algorithm 19. This function takes as input a path $P := (M_1,i_1),\dots,(M_\ell,i_\ell)$, and if $\ell > 1$, checks whether a $\mathcal{RO}$ call with input $(M_1,i_1),\dots,(M_{\ell-1},i_{\ell-1})$ has already been made (this is tracked explicitly thanks to the structure G), and if yes, it returns $\mathcal{RO}(P,r')$. In other words, this function allows to directly query the stateless object $\mathcal{RO}$, without any overhead. This makes the proof significantly easier, and close to the one of Theorems 8.2.1 and 8.2.2 at no cost. The simulators for Duplex-pi and Duplex-pi\$ are described in detail in Algorithms 19 and 20.

Algorithm 20 Simulator interfaces for Duplex-pi and Duplex-pi$, forward queries.

```
function $\mathbf{S}_P(X)$
    // The algorithm is the same for $\mathbf{S}\$_P$
    if $\exists (X, Y) \in \mathbf{S}.\texttt{TabP}$
        return $X$
    $Y \xleftarrow{\$} \mathbb{F}_p^b$
    // In the middle of a valid path with full absorption
    if $\exists\,(IV, m) \in \mathbf{S}.\texttt{ValidIV}()$, $S \in \mathbb{F}_p^b, (m, P) \in \mathbf{S}.G$, $i \in [\![1, \zeta]\!]$, and
    $m \in \mathbb{F}_p^r$ such that $[m] \xrightarrow{P} S$ and $X = \pi_i^f\,(S \oplus m\|0^c)$
        $P \leftarrow P.\texttt{append}(m, i)$
        $Y_o \leftarrow \texttt{Serialize}\,(m, P)$
        $Y_i \xleftarrow{\$} \mathbb{F}_p^{c'}$
        $Y \leftarrow Y_o \,\|\, Y_i$
    // In the middle of a valid path with partial absorption
    if $\exists\,(IV, m) \in \mathbf{S}.\texttt{ValidIV}()$, $S \in \mathbb{F}_p^b, (m, P) \in \mathbf{S}.G$, $i \in [\![1, \zeta]\!]$, and
    $m \in \mathbb{F}_p^{\leq r-1}$ such that $[m] \xrightarrow{P} S$ and $X = \pi_i^p\,(S \oplus m\|10^*)$
        $P \leftarrow P.\texttt{append}(m, i)$
        $Y_o \leftarrow \texttt{Serialize}\,(m, P)$
        $Y_i \xleftarrow{\$} \mathbb{F}_p^{c'}$
        $Y \leftarrow Y_o \,\|\, Y_i$
    // Start of squeezing phase from IV with one partial block
    if $\exists\,(IV, m) \in \mathbf{S}.\texttt{ValidIV}()$, $i \in [\![1, \zeta]\!]$, and $m \in \mathbb{F}_p^{\leq r''-1}$ such that
    $X = \pi_i^p\,(IV \oplus m\|10^*)$
        $Y_o := \texttt{Serialize}\,(m, (m, i))$
        $Y_i \xleftarrow{\$} \mathbb{F}_p^{c'}$
        $Y \leftarrow Y_o \,\|\, Y_i$
    // Start of squeezing phase from IV with one full block
    if $\exists\,(IV, m) \in \mathbf{S}.\texttt{ValidIV}()$, $i \in [\![1, \zeta]\!]$, and $m \in \mathbb{F}_p^{r''}$ such that $X =$
    $\pi_i^f\left(IV \oplus m\|0^{c''}\right)$
        $Y_o \leftarrow \texttt{Serialize}\,(m, (m, i))$
        $Y_i \xleftarrow{\$} \mathbb{F}_p^{c'}$
        $Y \leftarrow Y_o \,\|\, Y_i$
    $\mathbf{S}.\texttt{TabP} \leftarrow \mathbf{S}.\texttt{TabP} \cup \{(X, Y)\}$
    return $Y$
```

World Splitting. We again introduce an intermediate world W_{IM}, where construction queries give access to the duplex algorithm from Figure 8.2, but with the permutation $\mathcal{P}$ replaced with the simulator $\mathbf{S}$ or $\mathbf{S}\$$, itself based on an instance of an `ORO` hidden from the adversary. Moreover, primitive queries give also access to the same simulator. We have

$$\begin{aligned}\mathbf{Adv}^{\text{indif}}_{\mathcal{C},\mathbf{S}}(\mathcal{D}) \leq \Delta_{\mathcal{D}}(W_R\,;\,W_{IM}) && (8.11)\\ +\ \Delta_{\mathcal{D}}(W_{IM}\,;\,W_I)\ . && (8.12)\end{aligned}$$

As before, we have

$$\Delta_{\mathcal{D}}(W_R\,;\,W_{IM}) \leq \frac{\mathcal{N}(\mathcal{N}-1)}{2p^b}\,. \tag{8.13}$$

The remainder of the proof is dedicated to the analysis of (8.12).

Bad Events. The bad events are almost identical to the ones defined in the proof of Theorems 8.2.1 and 8.2.2, with the difference that there are more NCPs. Let Q denote the total number of queries made by the distinguisher, so that $Q = \mathcal{N}$ for Duplex-pi, and $Q = \mathcal{N} + q_H$ for Duplex-pi\$. For $i \in [\![1, Q]\!]$, let $\texttt{TabP}[i]$ and $\texttt{TabH}[i]$ denote the state of respectively `TabP` and `TabH` after the first i queries. The family of bad events is indexed by a query index $i \in [\![1, Q]\!]$, and over the tables of the simulator.

- CollP_i: $\exists (X, Y, d) \in \texttt{TabP}[i] \setminus \texttt{TabP}[i-1]$, $(X', Y', d') \in \texttt{TabP}[i-1]$, and $(i_1, d_1), (i_2, d_2) \in [\![1, \zeta]\!] \times \{p, f\}$ such that

  $$\pi^{d_1}_{i_1}(\text{inner}_c(Z)) = \pi^{d_2}_{i_2}(\text{inner}_c(Z'))\ ,$$

  where

  $$(Z, Z') = \begin{cases}(Y, Y') & \text{if } d = \mathit{fwd}\,,\\ (X, X') & \text{otherwise}\,;\end{cases}$$

- $\mathsf{ConnectP}_i$: $\exists (X, Y, d) \in \texttt{TabP}[i] \setminus \texttt{TabP}[i-1]$, $(X', Y', d') \in \texttt{TabP}[i-1]$, and $(i_1, d_1), (i_2, d_2) \in [\![1, \zeta]\!] \times \{p, f\}$ such that

  $$\pi^{d_1}_{i_1}(\text{inner}_c(Z)) = \pi^{d_2}_{i_2}\left(\text{inner}_c(\bar{Z}')\right)\ ,$$

  where

  $$(Z, \bar{Z}') = \begin{cases}(Y, X') & \text{if } d = \mathit{fwd}\,,\\ (X, Y') & \text{otherwise}\,;\end{cases}$$

- Guess_i: $\exists (X, Y, d) \in \mathtt{TabP}[i] \setminus \mathtt{TabP}[i-1]$ such that $Y \notin \mathtt{Rooted}_D(\widehat{\mathtt{TabP}}[i])$, but $Y \in \mathtt{Rooted}(\widehat{\mathtt{TabP}}[i])$;
- $\mathsf{ConnectIV}_i$ (for Duplex-pi only): $\exists (X, Y, d) \in \mathtt{TabP}[i]$, $id \in [\![1, \mu]\!]$, and $(i_1, d_1), (i_2, d_2) \in [\![1, \zeta]\!] \times \{p, f\}$ such that
$$\pi_{i_1}^{d_1}\left(\mathrm{inner}_{c''}(Z)\right) = \pi_{i_2}^{d_2}\left(IV_{id}\right) ,$$
where
$$Z = \begin{cases} X & \text{if } d = inv\,, \\ Y & \text{otherwise}\,; \end{cases}$$
- CollH_i (for Duplex-pi\$ only): $\exists (m, h) \in \mathtt{TabH}[i] \setminus \mathtt{TabH}[i-1]$, $(m', h') \in \mathtt{TabH}[i-1]$, and $(i_1, d_1), (i_2, d_2) \in [\![1, \zeta]\!] \times \{p, f\}$ such that
$$\pi_{i_1}^{d_1}(h) = \pi_{i_2}^{d_2}(h') \;;$$
- $\mathsf{ConnectPH}_i$ (for Duplex-pi\$ only):
  - either $\exists (X, Y, d) \in \mathtt{TabP}[i] \setminus \mathtt{TabP}[i-1]$, $(m, h) \in \mathtt{TabH}[i-1]$, and $(i_1, d_1), (i_2, d_2) \in [\![1, \zeta]\!] \times \{p, f\}$ such that
  $$\pi_{i_1}^{d_1}(h) = \pi_{i_2}^{d_2}\left(\mathrm{inner}_{c''}(Z)\right) ,$$

    where
  $$Z = \begin{cases} X & \text{if } d = inv\,, \\ Y & \text{otherwise}\,, \end{cases}$$
  - or $\exists (m, h) \in \mathtt{TabH}[i] \setminus \mathtt{TabH}[i-1]$, $(X, Y, d) \in \mathtt{TabP}[i-1]$ such that there exists $Z \in \{X, Y\}$ with
  $$\pi_{i_1}^{d_1}(h) = \pi_{i_2}^{d_2}\left(\mathrm{inner}_{c''}(Z)\right) .$$

We define BadIV_i as follows:
$$\mathsf{BadIV}_i \begin{cases} \mathsf{ConnectIV}_i & \text{for Duplex-pi}\,, \\ \mathsf{CollH}_i \vee \mathsf{ConnectPH}_i & \text{for Duplex-pi\$}\,, \end{cases}$$
and $\mathsf{Bad}_i := \mathsf{CollP}_i \vee \mathsf{ConnectP}_i \vee \mathsf{BadIV}_i \vee \mathsf{Guess}_i$. Moreover, for any of those bad events Event_i, let Event be
$$\bigvee_{i=1}^{Q} \mathsf{Event}_i \,.$$

W_{IM} versus W_I, as Long as No Bad. The reasoning from Theorems 8.2.1 and 8.2.2 carries over as we show in Lemmas 8.4.4 and 8.4.6.

Lemma 8.4.4. *Let $\mathcal{C}$ be either the* Duplex-pi *or* Duplex-pi$ *construction, based on the simulator* **S** *or* **S**$, *which is itself based on an* `ORO`. *For any $(IV, m) \in \texttt{ValidIV}()$ and valid path $[m] \xrightarrow{P} Z$ derived from the simulator table, it holds that the output of the sequence of calls $\mathcal{C}$.`initialize` and $\mathcal{C}$.`duplexing` made with the path (m, P) matches the output of the corresponding sequence of calls* `ORO.initialize` *and* `ORO.duplexing` *made with the path (m, P).*

Proof. Compared to Sponge-pi/Sponge-pi$ (Lemma 8.4.1), there is no distinction between absorb and squeeze paths, since every valid path leads to squeezing values. This is compensated by constraint 3 (and constraint 4 for Duplex-pi), which enforces that no two NCPs can coincide. Therefore, as long as no bad event occurs, there cannot be an overlap between two different paths, and the reasoning from Lemma 8.4.1 carries over. □

We finally argue that W_I and W_{IM} are identically distributed as long as Bad does not occur.

Lemma 8.4.5. *We have*

$$\mathbf{Pr}\left(\mathcal{D}^{W_I} \Rightarrow 1 \mid \neg\mathsf{Bad}\right) = \mathbf{Pr}\left(\mathcal{D}^{W_{IM}} \Rightarrow 1 \mid \neg\mathsf{Bad}\right)$$

Proof. Recall that the proof of Lemma 8.4.2 shows that this guarantee follows from $\neg\mathsf{Guess}$ combined with the consistency of the answers established in Lemma 8.4.4. The same reasoning extends directly to the current case. □

Probability of Bad.

Lemma 8.4.6. *For $\mathtt{X} \in \{\mathtt{S}, \mathtt{I}\}$, with the construction* Duplex-pi, *we have*

$$\mathbf{Pr}\left(\mathcal{D}^{W_X} \textit{ sets } \mathsf{Bad}\right) \leq \frac{(4\zeta^2 - 2\zeta + 1)\mathcal{N}(\mathcal{N}-1)}{p^c} + \frac{(4\zeta^2 - 2\zeta + 1)\mu\mathcal{N}}{2p^{c''}} + \frac{\mathrm{mucol}(\mathcal{N}, p^{r'-r})\mathcal{N}}{p^{c'}} .$$

For $\mathtt{X} \in \{\mathtt{S}, \mathtt{I}\}$, with the construction Duplex-pi$, *we have*

$$\mathbf{Pr}\left(\mathcal{D}^{W_X} \textit{ sets } \mathsf{Bad}\right) \leq \frac{(4\zeta^2 - 2\zeta + 1)\mathcal{N}(\mathcal{N}-1)}{p^c} + \frac{(4\zeta^2 - 2\zeta + 1)q_H(q_H-1)}{2p^{c''}} + \frac{3(4\zeta^2 - 2\zeta + 1)\mathcal{N}q_H}{p^{c''}} + \frac{\mathrm{mucol}(\mathcal{N}, p^{r'-r})\mathcal{N}}{p^{c'}} .$$

Proof. The proof is identical as that of Lemma 8.4.3, except that the number of collision cases appearing in the bad events CollP,$\mathsf{ConnectP}$, $\mathsf{ConnectIV}$, CollH, $\mathsf{ConnectPH}$ grows quadratically in the number of domain separators. In more detail, there are at most $4\zeta^2$ different combinations of collisions of form $\pi_{i_1}^{d_1}(X) = \pi_{i_2}^{d_2}(X')$. However, $\pi_i^d(X) = \pi_i^d(X')$ is equivalent to $X = X'$, hence $2\zeta - 1$ duplicate cases can be removed from the count. □

Conclusion. From Lemma 8.4.2, and using the fundamental lemma of game-playing [BR06], we deduce that

$$\Delta_{\mathcal{D}}\left(W_{IM}\,;\,W_I\right) \le \mathbf{Pr}_{W_{IM}}\left(\mathsf{Bad}\right) .$$

This term can be upper bounded using Lemma 8.4.6. Further plugging (8.13) into (8.12) completes the proofs of the theorems.

8.5 Applications

Applications of Sponge-pi **and** Sponge-pi$. The Sponge-pi variant is basically the plain sponge construction, but with an application of a NCP on the inner part to account for domain separation between full or partial data. This idea has appeared before in a bit-oriented fashion. In particular, the ESCH hash function of the NIST lightweight cryptography competition SPARKLE [BBC$^+$19] can be seen as a special variant of Sponge-pi: it injectively pads the message only if its length is not a multiple of the rate. Then, right before squeezing, it adds a constant to the full state, which differs depending on whether the data was full or partial. As such, security of ESCH follows as an immediate corollary of Theorem 8.2.1, but for $p = 2$, the 0-string as initial value, and either $(b, c) = (384, 256)$ or $(b, c) = (512, 384)$ (ESCH operates with equal initial, absorbing, and squeezing rate).

Corollary 8.5.1. *Let $b, c \in \mathbb{N}^*$. Let $\mathcal{N} \in \mathbb{N}^*$. Let $\mathcal{C}_E$ denote the construction underlying the* ESCH *hash function based on a b-bit random permutation $\mathcal{P}$, with c-bit capacity. There exists a simulator* $\mathbf{S}$ *with complexity $\tilde{\mathcal{O}}(\mathcal{N})$ queries such that, for any distinguisher $\mathcal{D}$ making at most $\mathcal{N}$ permutation evaluations,*

$$\mathbf{Adv}^{\mathrm{indif}}_{\mathcal{C}_E, \mathbf{S}}(\mathcal{D}) \le \frac{9\mathcal{N}^2}{2^c} + \frac{\mathcal{N}(\mathcal{N}-1)}{2^{b-1}} .$$

That said, as already mentioned in Section 8.1, the gains of our Sponge-pi construction over the sponge become more pronounced when considering larger

Algorithm 21 Sponge-pi using Duplex-pi.

```
1: function Sponge-pi($m, M, \nu$)
2:     $(M_1, \ldots, M_\ell) \leftarrow cutBlock_{r'',r}(M)$
3:     if $\ell = 0$
4:         $Z \leftarrow$ Duplex-pi.initialize($\epsilon, 2$)
5:     else if $\ell = 1$
6:         $Z \leftarrow$ Duplex-pi.initialize($M_1, 2$)
7:     else
8:         Duplex-pi.initialize($IV, M_1, 1$) // discard output
9:         for $2 \leq i \leq \ell - 1$ do
10:            Duplex-pi.duplexing($M_i, 1$) // discard output
11:        $Z \leftarrow$ Duplex-pi.duplexing($M_\ell, 2$)
12:    for $2 \leq i \leq \lceil \frac{\nu}{r} \rceil$ do
13:        $Z \leftarrow Z \| $Duplex-pi.duplexing($0^r, 1$)
14:    return $Z[1 : \nu]$
```

fields, such as fields with $p \approx 2^{256}$ as for Reinforced Concrete [GKL+22]. The Sponge-pi\$ variant gives additional freedom over Sponge-pi in that it allows to hash additional data into the initial inner part. This can be useful if both bits and field elements have to be hashed, or if a random-looking initialization value is needed for cross-protocol security, akin to SAFE [AKMQ23, KBM23].

Applications of Duplex-pi **and** Duplex-pi\$**.** The duplex variants Duplex-pi and Duplex-pi\$ significantly generalize these applications. That said, these constructions are described for a general amount of 2ζ NCPs, but the actual amount needed highly depends on the actual use case of the duplex. For example, one can describe Sponge-pi (Algorithm 12) when π^p has no fixed points in terms of Duplex-pi, as we demonstrate in Algorithm 21. In this case, the Duplex-pi is always called with second input 1 for a full data block, and with second input 2 for either full or partial data block, which means that it invokes 3 NCPs at most. A comparable reduction from Duplex-pi\$ to Sponge-pi\$ applies.

One can also describe a SpongeWrap [BDPV11b] style authenticated encryption scheme. At a high level, in authenticated encryption, we get as input a key $K \in \mathbb{F}_p^\kappa$, nonce $N \in \mathbb{F}_p^n$, associated data $A \in \mathbb{F}_p^*$, and plaintext $M \in \mathbb{F}_p^*$, which get turned into a ciphertext $C \in \mathbb{F}_p^{|M|}$ and tag $T \in \mathbb{F}_p^\tau$. Such a function can be naively[1] implemented using Duplex-pi, as we demonstrate in Algo-

[1]More clever implementations may exist, for instance ones that make use of the freedom

Algorithm 22 SpongeWrap-pi.

```
1:  function SpongeWrap-pi($K, N, A, M$)
        input: $(K, N, A, M) \in \mathbb{F}_p^\kappa \times \mathbb{F}_p^n \times \mathbb{F}_p^* \times \mathbb{F}_p^*$
        output: $(C, T) \in \mathbb{F}_p^{|M|} \times \mathbb{F}_p^\tau$
2:      $B \leftarrow K \| N \| A$
3:      $(B_1, \ldots, B_\ell) \leftarrow \mathit{cutBlock}_{r'',r}(B)$
4:      if $\ell = 1$
5:          $Z \leftarrow$ Duplex-pi.initialize($B_1, 2$)
6:      else
7:          Duplex-pi.initialize($IV, B_1, 1$) // discard output
8:          for $2 \le i \le \ell - 1$ do
9:              Duplex-pi.duplexing($B_i, 1$) // discard output
10:         $Z \leftarrow$ Duplex-pi.duplexing($B_\ell, 2$)
11:     $(M_1, \ldots, M_\ell), \leftarrow \mathit{cutBlock}_r(M)$
12:     if $\ell = 0$
13:         $Z \leftarrow Z \|$Duplex-pi.duplexing($\epsilon, 2$)
14:     else
15:         for $1 \le i \le \ell - 1$ do
16:             $Z \leftarrow Z \|$Duplex-pi.duplexing($M_i, 1$)
17:         $Z \leftarrow Z \|$Duplex-pi.duplexing($M_\ell, 2$)
18:     for $2 \le i \le \lceil \tau / r \rceil$ do
19:         $Z \leftarrow Z \|$Duplex-pi.duplexing($0^r, 1$)
20:     return $Z[1 : |M| + \tau] \oplus (M \| 0^\tau)$
```

rithm 22. The decryption functionality goes the straightforward way, with the main difference that ciphertexts are now overwritten into the state, which is addressed by our indifferentiability proof. In this case, one requires $2\zeta = 4$ NCPs. Security of the function immediately follows from the indifferentiability of the Duplex-pi construction. The SpongeWrap-pi scheme can also be generalized to the SCHWAEMM authenticated encryption scheme of above-mentioned SPARKLE submission [BBC+19], noting that it (just like ESCH) adds a constant to the inner part to separate between full and partial data. (We note that SCHWAEMM absorbs data in a Beetle-style fashion [CDNY18] but this is captured by our analysis because we basically prove blockwise adaptive security.)

Again, the true power of Duplex-pi and Duplex-pi$ becomes only more apparent if arbitrary fields are considered. Even beyond that, the flexibil-

one has in the index m for the initial value.

ity of these duplexes allows them to be applicable in many more protocols. To be more precise, as these functionalities extend the power of the SAFE API [AKMQ23, KBM23] in the sense that no input-output pattern encoding has to be hashed prior to the evaluation, all applications that SAFE API has (commitment schemes, multi-round interactive protocols, Fiat-Shamir, etc.) would fare well with Duplex-pi and Duplex-pi$, too.

8.6 Conclusion

In this work, we introduced arithmetization-oriented permutation-based sponges and duplexes, both with length-preserving padding using NCPs, without sacrificing security. In a bit more detail, in a bit-oriented setting, the security degradation is similar to padding into the inner part of the state (basically reducing the capacity by 1), but the gains of our constructions become more pronounced when they are used on large finite fields. In this case, depending on the size of the field, and the choice of permutation size, capacity, and rate, the existing security proofs may be void but our bounds still yield good security.

We also demonstrated the potential of our Duplex-pi, as it can be used to describe plain hashing (notably, Sponge-pi) as well as authenticated encryption (SpongeWrap-pi). It can also be used in many other applications such as interactive protocols. That said, we wish to stress that the resulting bounds will always be birthday bound in the inner part. This is because we proved Duplex-pi and Duplex-pi$ in the indifferentiability framework (as in the original analysis [BDPV11b] and that of Degabriele et al. [DFG23]). It has been observed before that a keyed version of the duplex typically yields better security as the offline and online complexity can be separated [ADMV15, MRV15, DMV17] However, integrating the technique of using NCPs into keyed duplexes seems to require a non-trivial analysis.

In the paper on which this chapter is based [LBM25], we also introduce an arithmetization-oriented permutation-based PRF that employs NCPs. The construction obtained is a variant of the full-state keyed sponge (cf., Section 3.2.2). Notably, the use of NCPs there enables domain separation between the absorption and squeezing phases, akin to Ascon-PRF, and has a security bound comparable to that of Ascon-PRF (see Section 9.9).

Part IV

Design and Analysis of Sponge-Based Pseudorandom Functions and Authenticated Encryption Constructions

Chapter 9

Security of the Ascon Modes

9.1 Introduction

Over the past decades, lightweight cryptography has been a dominant research domain in the field of symmetric cryptography, with designs tailored for constrained environments, as we detailed in Section 1.2. Among the most notable advancements is the adoption of permutation-based designs. In particular, the sponge and duplex, are, as discussed in Section 1.3, extremely well-suited for lightweight cryptography.

This chapter focuses on the Ascon suite, a prominent family of sponge-based lightweight cryptographic schemes. Ascon [DEMS21b] was first selected [DEMS14] as the winner in the “lightweight” category of the CAESAR [CAE14] competition, and was in 2023 selected [SMC$^+$25] as the overall winner of the NIST lightweight cryptography competition [Nat19].

In a bit more detail, Ascon typically refers to the authenticated encryption scheme, dubbed Ascon-AE in this chapter to avoid ambiguity. (Looking ahead, the NIST standard also includes a hash function standard Ascon-Hash, a XOF Ascon-XOF, and a customized XOF Ascon-CXOF, and another relevant scheme is a PRF called Ascon-PRF [DEMS24], but we will come back to these later.) Ascon-AE is an authenticated encryption scheme inspired by the duplex construction [BDPV11b, MRV15, DMV17] (cf., Section 3.1.4), but with some subtle differences. In a nutshell, Ascon-AE operates on two ($b = 320$)-bit permutations, an outer permutation $\mathcal{P}_\mathrm{o}$ and inner permutation $\mathcal{P}_\mathrm{i}$, which differ in the number of rounds and round constants. For a new authenticated encryption operation, it initializes a 320-bit state with a 64-bit initialization vector (encoding the specific instance), a 128-bit key, and a 128-bit nonce. Then, it permutes the state using the outer permutation $\mathcal{P}_{\mathrm{o}}$, and compresses the key again into the state. Then, the scheme processes the associated data by absorbing it block-by-block into the state, interleaved with evaluations of $\mathcal{P}_{\mathrm{i}}$, and encrypts the plaintext block-by-block by using part of the state as keystream and subsequently absorbing the plaintext into the state, again interleaved with evaluations of $\mathcal{P}_{\mathrm{i}}$. Finally, the state is blinded once again with the key, a last evaluation of $\mathcal{P}_{\mathrm{o}}$ is made and a t-bit chunk of the state is blinded a final time with the key before it is output as tag. We refer to Section 9.2 for a detailed description of the Ascon-AE mode, including a visualization in Figure 9.2a. Sometimes, when looking at the construction generically, we discard the difference between the two permutations and assume a single permutation $\mathcal{P}$.

As mentioned, the NIST standard [SMC+25] also includes a hash function standard Ascon-Hash, a XOF Ascon-XOF, and a customized XOF Ascon-CXOF. They are fairly direct instantiations of the sponge, operating on the 320-bit permutation $\mathcal{P}$. The difference between the hash function and the XOFs is that Ascon-Hash only outputs fixed-length digests but Ascon-XOF/Ascon-CXOF accommodate for variable-length digests. The difference between the two XOFs is that the customized XOF allows for a customization string to be prepended to the plaintext. Refer to Section 9.8 for a specification of Ascon-Hash, Ascon-XOF, and Ascon-CXOF. Another scheme worth mentioning is a PRF called Ascon-PRF [DEMS24], of which the main goal is to authenticate the plaintext. Ascon-PRF is basically a keyed sponge construction. It can be seen to operate the Ascon-XOF function but initialized with an initialization vector and a 128-bit key, and right before squeezing the tag, domain separation is applied by flipping a bit in the inner part. A description of Ascon-PRF is given in Section 9.9.

9.1.1 Generic Security of Sponges and Duplexes

As detailed in Chapter 3, the sponge/duplex paradigms come with a decent security foundation: the sponge achieves tight $c/2$-bit security in the indifferentiability framework [MRH04,CDMP05], and has tight security bounds for collision, second preimage and preimage resistance (see Sections 3.1.2 and 3.1.3 and Chapter 4). These results directly apply to the generic security of Ascon-Hash, Ascon-XOF, and Ascon-CXOF (as we also outline in Section 9.8).

The indifferentiability result implies that one can also use the sponge in keyed applications, but dedicated analyses were performed to obtain more fine-grained bounds. The duplex is a general construction that enables these fine-grained analyses, and it can be used to realize various functionalities, such as authenticated encryption (see Section 3.3) or pseudorandom functions (see Section 3.2). An excellent systematization of knowledge on the security of the duplex and its implications is given by Mennink [Men23], and it also contains a proof of the security of Ascon-PRF (we will elaborate on this in Section 9.9).

9.1.2 Generic Security of Ascon-AE Constructions

Given this state of affairs, it is tempting to state that the security of the Ascon-AE mode immediately follows. Indeed, the Ascon-AE mode resembles SpongeWrap [BDPV11b] (cf., Section 3.3.1.1) or MonkeySpongeWrap [Men23] (cf., Section 3.3.1.3), and those security results give *some* certainty that the Ascon-AE mode is sound. Likewise, along with their security proof of NORX, Jovanovic et al. [JLM14, JLM$^+$19] mentioned that their proof can be generalized to the Ascon-AE mode, though without proof. The main difference between those analyses and the mode of Ascon-AE is the presence of the additional key blindings.

Thus, a dedicated analysis of the Ascon-AE mode, and particular the impact of these key blindings, turned out to be necessary and relevant, and this has led to several works considering the security of the Ascon-AE mode. Chakraborty et al. [CDN23b] performed a single-user security analysis in the nonce-respecting setting [BN00], and later [CDN24] extended their proof to multi-user security and to the nonce-misuse setting [RS06].[1] On top of that, there is a "proof sketch" of Guo et al. [GPPS19b] (full version of [GPPS20]) in the nonce-misuse resilience setting that guarantees security for fresh nonces only [ADL17] and in the leakage resilience setting where the inner permutations may leak side-channel information [DP08, PSV15]. It should be noted that all these results are in the random permutation model, where the permutation $\mathcal{P}$ is assumed to be a random permutation, or (in some of these results) the outer permutation and inner permutation are both random and assumed to be independent (see Remark 9.2.1).

9.1.3 Main Contributions

From this overall state-of-the-art discussion, it can be concluded that the security analyzes of the Ascon-AE mode has lead to several different results,

[1]To clarify the timeline, in-between the two works of Chakraborty et al., we [LM24a] delivered a multi-user security analysis in the nonce-respecting, nonce-misuse, and state-recovery setting. These results are detailed in the current chapter.

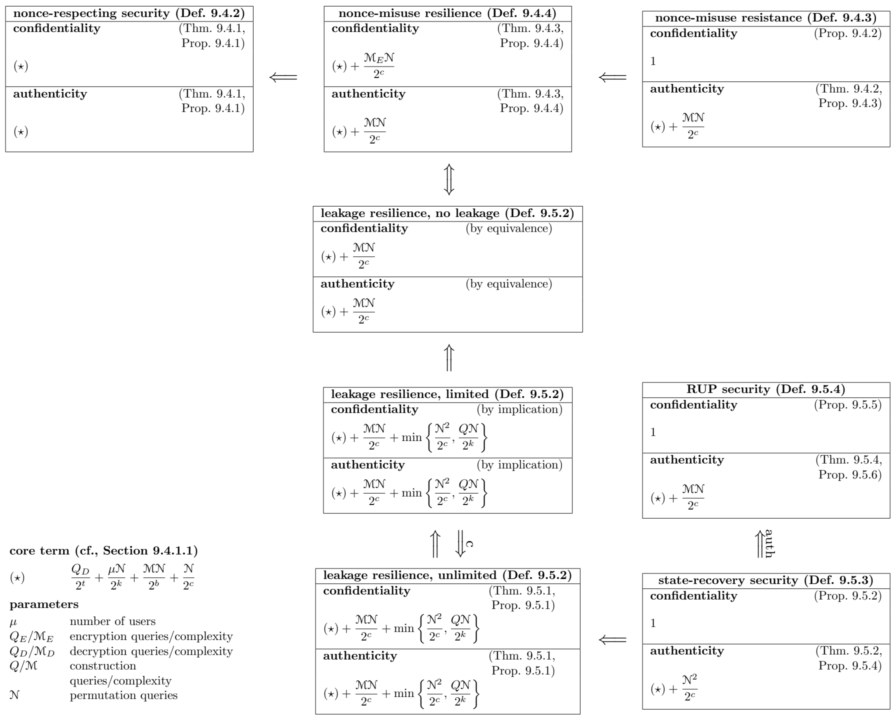

nonce-respecting security (Def. 9.4.2)
confidentiality (Thm. 9.4.1, Prop. 9.4.1)
$(\star)$
authenticity (Thm. 9.4.1, Prop. 9.4.1)
$(\star)$
nonce-misuse resilience (Def. 9.4.4)
confidentiality (Thm. 9.4.3, Prop. 9.4.4)
$(\star) + \frac{\mathcal{M}_E \mathcal{N}}{2^c}$
authenticity (Thm. 9.4.3, Prop. 9.4.4)
$(\star) + \frac{\mathcal{M}\mathcal{N}}{2^c}$
nonce-misuse resistance (Def. 9.4.3)
confidentiality (Prop. 9.4.2)
1
authenticity (Thm. 9.4.2, Prop. 9.4.3)
$(\star) + \frac{\mathcal{M}\mathcal{N}}{2^c}$
leakage resilience, no leakage (Def. 9.5.2)
confidentiality (by equivalence)
$(\star) + \frac{\mathcal{M}\mathcal{N}}{2^c}$
authenticity (by equivalence)
$(\star) + \frac{\mathcal{M}\mathcal{N}}{2^c}$
leakage resilience, limited (Def. 9.5.2)
confidentiality (by implication)
$(\star) + \frac{\mathcal{M}\mathcal{N}}{2^c} + \min\left\{\frac{\mathcal{N}^2}{2^c}, \frac{Q\mathcal{N}}{2^k}\right\}$
authenticity (by implication)
$(\star) + \frac{\mathcal{M}\mathcal{N}}{2^c} + \min\left\{\frac{\mathcal{N}^2}{2^c}, \frac{Q\mathcal{N}}{2^k}\right\}$
RUP security (Def. 9.5.4)
confidentiality (Prop. 9.5.5)
1
authenticity (Thm. 9.5.4, Prop. 9.5.6)
$(\star) + \frac{\mathcal{M}\mathcal{N}}{2^c}$
c
auth
leakage resilience, unlimited (Def. 9.5.2)
confidentiality (Thm. 9.5.1, Prop. 9.5.1)
$(\star) + \frac{\mathcal{M}\mathcal{N}}{2^c} + \min\left\{\frac{\mathcal{N}^2}{2^c}, \frac{Q\mathcal{N}}{2^k}\right\}$
authenticity (Thm. 9.5.1, Prop. 9.5.1)
$(\star) + \frac{\mathcal{M}\mathcal{N}}{2^c} + \min\left\{\frac{\mathcal{N}^2}{2^c}, \frac{Q\mathcal{N}}{2^k}\right\}$
state-recovery security (Def. 9.5.3)
confidentiality (Prop. 9.5.2)
1
authenticity (Thm. 9.5.2, Prop. 9.5.4)
$(\star) + \frac{\mathcal{N}^2}{2^c}$
core term (cf., Section 9.4.1.1)
$(\star)$ $\frac{Q_D}{2^t} + \frac{\mu\mathcal{N}}{2^k} + \frac{\mathcal{M}\mathcal{N}}{2^b} + \frac{\mathcal{N}}{2^c}$
parameters
μ number of users
$Q_E/\mathcal{M}_E$ encryption queries/complexity
$Q_D/\mathcal{M}_D$ decryption queries/complexity
$Q/\mathcal{M}$ construction queries/complexity
$\mathcal{N}$ permutation queries

Figure 9.1: High-level overview of the considered security models and the corresponding results. Intuitively, horizontal orientation represents the amount of nonce-misuse power whereas vertical orientation represents the amount of additional leakage. All bounds are simplified, they are expressed in big-O notation, and are tight in the simplified setting of Section 9.3.2. The conditional implication $\overset{\text{c}}{\Longrightarrow}$ depends on the set of allowed leakage functions but applies in our case (cf., Section 9.5.1). The implication $\overset{\text{auth}}{\Longleftarrow}$ only holds for authenticity (cf., Lemma 9.5.3).

all in different security models, different attack settings, different proof techniques, and in fact also with different levels of accuracy. On top of that, most of these bounds are not matched with tight attacks, which means that we do not know if all the bounds are tight and can be improved. This is a particularly relevant question in the area of lightweight cryptography, where schemes are minimized and a too loose bound would give a false sense of insecurity (as also already mentioned in myriad earlier works [DM20, LNS18, JN20, DDNT23, LMP17, BM24]).

In this chapter, we give a complete and comprehensive overview of the levels of security of the Ascon-AE mode in various security settings. We cover three flavors of conventional security (in Section 9.4): (i) nonce-respecting security [BN00], (ii) nonce-misuse resistance [RS06], and (iii) nonce-misuse resilience [ADL17]. We subsequently cover three flavors of leaky security (in Section 9.5): (iv) bounded leakage resilience in a leveled implementation setup [DP08, PSV15], (v) state-recovery security, and (vi) security under release of unverified plaintext [ABL+14].

The results (iv) and (v) show that the key blindings play a crucial role in guaranteeing security when (part of) the internal state leaks. This matches the claims of the designers: they have chosen to include the key blindings in their scheme for security reasons, and claim that these extra key blindings allow the scheme to achieve authenticity even under state recovery. This attack setting, for example, applies to the case of leveled implementations, where the outer permutations get stronger protection than the (round-reduced) inner permutation. In this case, attackers may get a higher chance at recovering the state of an inner permutation call. Notably, the fifth result (v) is in a setting *that was never formally explored* in the first place, and captures authenticity even if the adversary learns *all* internal states.[2]

For each of these security models, we categorize the existing security lower and upper bounds, we point out multiple flaws and issues in existing analyses, and we derive new security bounds and generic attacks to complete the overview. A high-level overview is given in Figure 9.1.

Most notably, besides classifying existing results, this chapter makes the following contributions:

[2]The only earlier work in this direction is the aforementioned "proof sketch" of Guo et al. [GPPS19b, Theorem 4]. However, a closer look at the reasoning reveals that it contains several incomplete and incorrect steps. In fact, we derived our result independently, and even discarded the idea behind the reasoning of [GPPS19b] as it would not allow us to obtain a tight security bound. We elaborate on this in Section 9.4.3.2.

- We formalize the model that captures the idea that Ascon-AE still achieves authenticity even when the adversary learns internal states, named *state recovery*;

- We develop new security proofs for nonce-misuse resistance, nonce-misuse resilience, leakage resilience, state-recovery security, and release of unverified plaintext, as these were lacking;

- We point out flaws in the analyses of Chakraborty et al. [CDN24, Theorem 2] and Guo et al. [GPPS19b, Theorem 4];

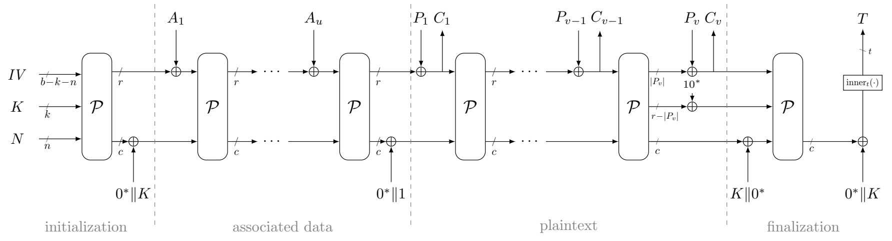
- We give matching attacks for all security proofs.

All new security proofs are gathered together in Section 9.6, and all elaborate generic attacks in Section 9.7.

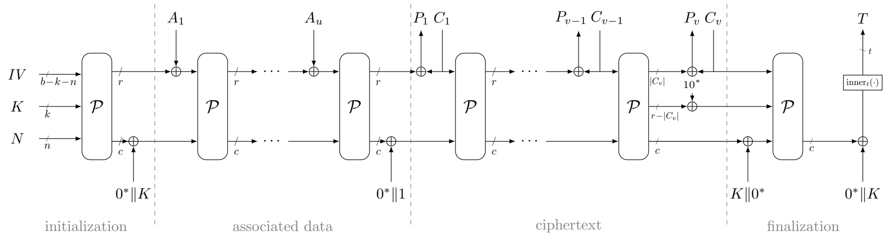
As a bonus, we also comprehensibly discuss how existing literature covers the generic security of Ascon-Hash and Ascon-PRF in Section 9.8 and Section 9.9, respectively. These sections are not so surprising: the generic security of Ascon-Hash follows from the results discussed in Section 3.1.2, and Mennink [Men23] gave a security proof of Ascon-PRF. Finally, we conclude the work in Section 9.10, where we highlight models or settings that we do not cover and give a further final discussion.

9.1.4 Outline

The Ascon authenticated encryption (Ascon-AE) mode is described in detail in Section 9.2. We describe the general attack model, including a description of the adversarial resources and some notational conventions in Section 9.3. Then, in Section 9.4, we discuss the conventional security models of nonce-respecting security (Section 9.4.1), nonce-misuse resistance (Section 9.4.2), and nonce-misuse resilience (Section 9.4.3). Then, we extend the analysis to security in leaky settings in Section 9.5, covering leakage resilience (Section 9.5.1), state-recovery security (Section 9.5.2), and release of unverified plaintext (Section 9.5.3). The security proofs are all gathered in Section 9.6 and the generic attacks in Section 9.7. We then extend our discussion to Ascon-Hash/Ascon-(C)XOF in Section 9.8 and to Ascon-PRF in Section 9.9. We conclude the work in Section 9.10.



(a)



(b)

Figure 9.2: The Ascon-AE mode of operation in case of non-empty associated data: (a) encryption **Enc** and (b) decryption **Dec**. Here, A is injectively padded as $(A_1, \ldots, A_u) \leftarrow pad_r^{10^*}(A)$. For encryption, the plaintext $P \in \{0,1\}^*$ is padded as $(P_1, \ldots, P_v) \leftarrow cut_r(P)$, and for decryption the ciphertext $C \in \{0,1\}^*$ is padded as $(C_1, \ldots, C_v) \leftarrow cut_r(C)$, noting that we put the 10^*-padding explicit in the picture. cut_s denotes the function that gets as input a bit string $X \in \{0,1\}^*$, and that splits it into s-bit blocks, where the last block is of size between 0 and $s-1$ bits.

9.2 Ascon-AE Mode

The Ascon-AE mode [DEMS21b,DEMS14,DEMS21a] is a variant of SpongeWrap [BDPV11b], with additional key blinding during the initialization and the finalization phases. Let $b, c, r, k, n, t \in \mathbb{N}^*$ such that $b = r+c$, $k+n \leq b$, $t \leq k$, $k \leq c$, and let $\mathcal{P}$ be a cryptographic permutation over b bits. The Ascon-AE mode is an ensemble of two algorithms, encryption $\mathbf{Enc}^{\mathcal{P}}$ and decryption $\mathbf{Dec}^{\mathcal{P}}$. The encryption algorithm $\mathbf{Enc}^{\mathcal{P}}$ takes as input a key $K \in \{0,1\}^k$, a nonce $N \in \{0,1\}^n$, associated data $A \in \{0,1\}^*$, and a plaintext $P \in \{0,1\}^*$. It returns a ciphertext $C \in \{0,1\}^*$ with $|C| = |P|$, and a tag $T \in \{0,1\}^t$. For simplicity of notation, we will put the key as a subscript to $\mathbf{Enc}^{\mathcal{P}}$. Therefore, the encryption algorithm based on the key K and permutation $\mathcal{P}$ is denoted as:

$$\begin{aligned}\mathbf{Enc}_K^{\mathcal{P}} : \{0,1\}^n \times \{0,1\}^* \times \{0,1\}^* &\longrightarrow \{0,1\}^* \times \{0,1\}^t\,, \\ (N, A, P) &\longrightarrow (C \in \{0,1\}^{|P|}, T)\,.\end{aligned}$$

The decryption algorithm $\mathbf{Dec}^{\mathcal{P}}$ takes as input a key $K \in \{0,1\}^k$ (again put as a subscript), a nonce $N \in \{0,1\}^n$, associated data $A \in \{0,1\}^*$, a ciphertext $C \in \{0,1\}^*$, and a tag $T \in \{0,1\}^t$. It returns either the corresponding plaintext $P \in \{0,1\}^*$ with $|P| = |C|$ if authentication with the tag succeeds, or a failure symbol $\perp$. Therefore, the decryption algorithm based on the key K and permutation $\mathcal{P}$ is denoted as:

$$\begin{aligned}\mathbf{Dec}_K^{\mathcal{P}} : \{0,1\}^n \times \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^t &\longrightarrow \{0,1\}^* \cup \{\perp\}\,, \\ (N, A, C, T) &\longrightarrow P \in \{0,1\}^{|C|} \text{ or } \perp.\end{aligned}$$

The encryption and decryption algorithms of Ascon-AE are described in Algorithm 23 and illustrated in Figure 9.2. Here, $IV \in \{0,1\}^{b-k-n}$ is a fixed initialization value encoding the specific instance of Ascon-AE. The Ascon specification [DEMS21a] specifies Ascon-AE to always operate with nonce size n and tag size t equal to 128 bits. The basic variant, Ascon-128, has a capacity $c = 256$ and rate $r = 64$, while the accelerated variant, Ascon-128a, has a capacity $c = 192$ and rate $r = 128$, and both use a key size $k = 128$. The variant Ascon-80pq differs from the basic variant Ascon-128 in the fact that the key is increased to size $k = 160$. The NIST standard [SMC+25] specifies a single instance, namely Ascon-AEAD128, which is based on Ascon-128a and shares the same parameters sizes (i.e., n, t, k, r, and c). On top of that, Ascon-AEAD128 supports two implementation options: (i) truncating the tag to a size of up to $t = 64$ bits, and (ii) masking the nonce with an additional 128-bit key.

Remark 9.2.1. *We would like to remark that the* Ascon-AE *specification, in fact, operates on two different permutations: an outer permutation $\mathcal{P}_o$ for the initialization and finalization and an inner permutation $\mathcal{P}_i$ for the inner evaluations. These permutations differ only in the number of rounds and the round constants, which were carefully chosen by the designers to balance efficiency and security. On the other hand, this work focuses on generic security in the ideal permutation model. Current techniques, however, cannot model the dependency between $\mathcal{P}_o$ and $\mathcal{P}_i$. This leaves us with two options to model $\mathcal{P}_o$ and*

Algorithm 23 Ascon-AE mode.

```
Encryption algorithm Enc
input: $K \in \{0,1\}^k$, $N \in \{0,1\}^n$,
$A \in \{0,1\}^*$, $P \in \{0,1\}^*$
output: $C \in \{0,1\}^{|P|}$,
$T \in \{0,1\}^t$

// Initialization
$S \leftarrow \mathcal{P}(IV \| K \| N) \oplus (0^{b-k} \| K)$
// Absorb $A$
if $|A| \geq 1$
    $(A_1, \ldots, A_u) \leftarrow pad_r^{10^*}(A)$
    for $i = 1, \ldots, u$ do
        $S \leftarrow \mathcal{P}(S \oplus (A_i \| 0^c))$
// Domain Separation
$S \leftarrow S \oplus (0^{b-1} \| 1)$
// Absorb $P$, Extract $C$
$(P_1, \ldots, P_v) \leftarrow pad_r^{10^*}(P)$
for $i = 1, \ldots, v-1$ do
    $S \leftarrow S \oplus (P_i \| 0^c)$
    $C_i \leftarrow \mathrm{outer}_r(S)$
    $S \leftarrow \mathcal{P}(S)$
$S \leftarrow S \oplus (P_v \| 0^c)$
$C_v \leftarrow \mathrm{outer}_{|P| \bmod r}(S)$
// Finalization
$S \leftarrow \mathcal{P}(S \oplus (0^r \| K \| 0^{c-k})) \oplus (0^{b-k} \| K)$
$T \leftarrow \mathrm{inner}_t(S)$
return $(C_1 \| \cdots \| C_v, T)$
```

```
Decryption algorithm Dec
input: $K \in \{0,1\}^k$, $N \in \{0,1\}^n$,
$A \in \{0,1\}^*$, $C \in \{0,1\}^*$,
$T \in \{0,1\}^t$
output: Either $P \in \{0,1\}^{|C|}$ or $\perp$

// Initialization
$S \leftarrow \mathcal{P}(IV \| K \| N) \oplus (0^{b-k} \| K)$
// Absorb $A$
if $|A| \geq 1$
    $(A_1, \ldots, A_u) \leftarrow pad_r^{10^*}(A)$
    for $i = 1, \ldots, u$ do
        $S \leftarrow \mathcal{P}(S \oplus (A_i \| 0^c))$
// Domain Separation
$S \leftarrow S \oplus (0^{b-1} \| 1)$
// Absorb $C$, Extract $P$
$(C_1, \ldots, C_v) \leftarrow pad_r^{10^*}(C)$
for $i = 1, \ldots, v-1$ do
    $P_i \leftarrow \mathrm{outer}_r(S) \oplus C_i$
    $S \leftarrow C_i \| \mathrm{inner}_c(S)$
    $S \leftarrow \mathcal{P}(S)$
$P_v \leftarrow \mathrm{outer}_{|C| \bmod r}(\mathrm{outer}_r(S) \oplus C_v)$
$S \leftarrow S \oplus (P_v \| 10^{b-1-|P_v|})$
// Finalization
$S \leftarrow \mathcal{P}(S \oplus (0^r \| K \| 0^{c-k})) \oplus (0^{b-k} \| K)$
$T^* \leftarrow \mathrm{inner}_t(S)$
if $T^* = T$ return $P_1 \| \cdots \| P_v$
else return $\perp$
```

$\mathcal{P}_i$: either model them as being independent, as Guo et al. [GPPS19b] and we did [LM24a], or model them as being the same, as Chakraborty et al. [CDN23b, CDN24] did. Security proofs with independent primitives are often stronger than with identical permutations (we have seen this, for example, for the sum of two secret permutations [BKR98, Luc00, Pat08a, DHT17, Din24]), and this seems to be the case for the Ascon mode, too. For this reason, we chose to model the two permutations as being identical.

9.3 Adversarial Setup

We will consider the security of Ascon-AE in various attack models, but to describe these models and the levels of security appropriately, we first have to define certain conventions in notation in Section 9.3.1. We then describe how we quantify adversaries in Section 9.3.2.

9.3.1 Notational Conventions

In this chapter, we denote by $\mathcal{A}[\mathcal{O}]$ the fact that the adversary $\mathcal{A}$ is given access to a collection of oracles $\mathcal{O}$. The collections of oracles in this chapter will always be composed of Ascon-AE algorithms, i.e., **Enc** and **Dec** of Section 9.2 or their random equivalents (and the precise definition of the oracles highly depends on the specific security model). As a matter of fact, we will consider security of Ascon-AE in the multi-user setting, where the adversary has access to μ instances of the construction. Finally, as we will only consider security in the random permutation model, $\mathcal{P} \xleftarrow{\$} \texttt{Perm}(b)$ will always be one of the oracles, to which $\mathcal{A}$ even has forward and inverse access, which we denote by $\mathcal{P}^{\pm}$. This means that, typically, $\mathcal{O}$ would be of the following form (in our security games, the number of construction oracles ranges from 1 to 4):

$$\mathcal{O} = \left((\mathcal{O}_{1,m}, \mathcal{O}_{2,m}, \mathcal{O}_{3,m}, \mathcal{O}_{4,m})_{m=1}^{\mu}, \mathcal{P}^{\pm}\right) . \tag{9.1}$$

Given that the oracles to which $\mathcal{A}$ has access are encryption and decryption functionalities of Ascon-AE, or their ideal equivalent, we will need to impose restrictions on nonce-repetition or even query-repetition to avoid trivial attacks. These restrictions depend on the actual security model, but we will define shortcut notation for this, for any typical oracle of the form (9.1):

- $\mathcal{O}_{i,m} \overset{N}{\not\hookrightarrow} \mathcal{O}_{j,m}$ means that if $\mathcal{A}$ queries oracle $\mathcal{O}_{i,m}$ with a certain nonce N, then it cannot later query $\mathcal{O}_{j,m}$ with the same nonce N;

- $\mathcal{O}_{i,m} \overset{N}{\not\rightsquigarrow} \mathcal{O}_{j,m}$ means that if $\mathcal{A}$ makes *two* distinct queries to $\mathcal{O}_{i,m}$ with the same nonce N, then it cannot later query $\mathcal{O}_{j,m}$ with the same nonce N;

- $\mathcal{O}_{i,m} \overset{*}{\not\rightsquigarrow} \mathcal{O}_{j,m}$ means that a query to $\mathcal{O}_{i,m}$ cannot later be repeated to $\mathcal{O}_{j,m}$. The definition has slightly different meanings whenever the oracles are encryption or decryption oracles: if $\mathcal{O}_{i,m}$ and $\mathcal{O}_{j,m}$ are both encryption or both decryption oracles, the definition means that $\mathcal{A}$ cannot make the same query to both oracles; if one of them is an encryption and one a decryption oracle, the definition means that $\mathcal{A}$ cannot use the response of oracle $\mathcal{O}_{i,m}$ as input to oracle $\mathcal{O}_{j,m}$.

We remark that we *never* impose anything on nonce-repetition or query-repetition among *different* users, i.e., from user m to m'. We do make one general restriction, though, namely that $\mathcal{A}$ never repeats the exact same query to an oracle. For the primitive access $\mathcal{P}^{\pm}$ this additionally means that $\mathcal{A}$ never makes an inverse query for an earlier forward query, or vice versa.

9.3.2 Adversarial Resources

We always consider an information-theoretic distinguisher $\mathcal{A}$. In terms of a collection of oracles of the form (9.1), its resources are quantified as follows:

- The total number of queries to $(\mathcal{O}_{1,m}, \mathcal{O}_{2,m}, \mathcal{O}_{3,m}, \mathcal{O}_{4,m})_{m=1}^{\mu}$ is denoted by Q, and the total *online complexity* is denoted by $\mathcal{M}$, which counts the number of "blocks", i.e., the minimal number of permutation evaluations that would be required by the Ascon-AE mode to process the query (cf., Sections 3.2.1.2 and 3.3.1.2). The quantities $(Q, \mathcal{M})$ may be refined into encryption complexities $(Q_E, \mathcal{M}_E)$, counting only encryption queries, and decryption complexities $(Q_D, \mathcal{M}_D)$, counting only decryption queries;[3]

- The total number of queries to $\mathcal{P}^{\pm}$ is counted by the *offline complexity* $\mathcal{N}$.

Without loss of generality, we assume that $\mu \leq Q_E$, since an oracle that the adversary cannot query is of no use. As a rule of thumb, $\mathcal{M} \ll \mathcal{N}$, as $\mathcal{M}$ is

[3]Looking ahead, there is one exception to this, namely in security under release of unverified plaintext (Section 9.5.3), where the decryption functionality is split into an unverified decryption function and a verification function. The notation will be refined there in an ad-hoc way.

limited by the use case in which Ascon-AE is employed whereas $\mathcal{N}$ is limited by the wealth of the adversary (i.e., its computing power).

When investigating tightness of bounds, we ignore constant factors and logarithmic factors, and furthermore assume that $\mathcal{M}_D \leq \mathcal{M}_E$ and $Q_D \leq Q_E$. The reason behind the latter assumption is that, in real-world protocols [GTW24], connections may be broken once too many failed forgery attempts have been mounted. Of course, in the multi-user setting, this assumption is a bit more debatable, as an adversary may apply a forgery attempt on multiple users, but we only use it in the simplified bounding.

9.4 Conventional Security of Ascon-AE

We start with the more conventional security notions for authenticated encryption and what level of security the Ascon-AE mode achieves in these models: nonce-respecting security in Section 9.4.1, nonce-misuse resistance in Section 9.4.2, and nonce-misuse resilience in Section 9.4.3.

9.4.1 Nonce-Respecting Security

The most conventional security model for nonce-based authenticated encryption is security in the nonce-respecting setting. A discussion of that notion can be found in Section 2.4.3.1, where it was formally defined in Definition 2.4.7 under the name "AEAD security". Here we adapt it to Ascon in the multi-user setting, working in the random permutation model where $\mathcal{P} \xleftarrow{\$} \texttt{Perm}(b)$.

Definition 9.4.1. *Consider the* Ascon-AE *mode of Section 9.2. Let* $(\$_m)_{m=1}^{\mu}$ *be a family of* μ *independent random functions,* $\mathcal{P} \xleftarrow{\$} \texttt{Perm}(b)$*, and* $K_1, \ldots, K_\mu \xleftarrow{\$} \{0,1\}^k$*. The AEAD security of* Ascon-AE *against an adversary* $\mathcal{A}$ *is defined as*

$$\mathbf{Adv}^{\mu\text{-ae}}_{\text{Ascon-AE}}(\mathcal{A}) = \Delta_{\mathcal{A}}\left(\left(\mathbf{Enc}^{\mathcal{P}}_{K_m}, \mathbf{Dec}^{\mathcal{P}}_{K_m}\right)_{m=1}^{\mu}, \mathcal{P}^{\pm} \; ; \; (\$_m, \perp)_{m=1}^{\mu}, \mathcal{P}^{\pm}\right),$$

where $\mathcal{A}$ *is restricted as follows:* $\mathcal{O}_{1,m} \overset{N}{\not\hookrightarrow} \mathcal{O}_{1,m}$ *and* $\mathcal{O}_{1,m} \overset{*}{\not\hookrightarrow} \mathcal{O}_{2,m}$*.*

The notion of AEAD security is related to plain confidentiality and authenticity of the mode, as demonstrated by Bellare and Namprempre [BN00, BN08] (cf., Section 2.4.3.2). In fact, security proofs for authenticated encryption schemes *can* be derived directly under the AEAD security model, as Chakraborty et al. [CDN23b, CDN24] did for Ascon-AE, for instance. However, looking ahead, Ascon-AE achieves authenticity but not confidentiality in some of the models discussed in this chapter. Therefore, we consider confidentiality and authenticity separately whenever possible.

Definition 9.4.2. *Consider the* Ascon-AE *mode of Section 9.2. Let* $(\$_m)_{m=1}^{\mu}$ *be a family of* μ *independent random functions,* $\mathcal{P} \xleftarrow{\$} \texttt{Perm}(b)$*, and* $K_1, \ldots, K_\mu \xleftarrow{\$} \{0,1\}^k$*.*

- *The nonce-respecting confidentiality of* Ascon-AE *against an adversary* $\mathcal{A}$ *is defined as*

$$\mathbf{Adv}^{\mu\text{-conf}}_{\text{Ascon-AE}}(\mathcal{A}) = \Delta_{\mathcal{A}}\left(\left(\mathbf{Enc}^{\mathcal{P}}_{K_m}\right)_{m=1}^{\mu}, \mathcal{P}^{\pm} \;;\; (\$_m)_{m=1}^{\mu}, \mathcal{P}^{\pm}\right),$$

  where $\mathcal{A}$ *is restricted as follows:* $\mathcal{O}_{1,m} \overset{N}{\not\hookrightarrow} \mathcal{O}_{1,m}$*;*

- *The nonce-respecting authenticity of* Ascon-AE *against an adversary* $\mathcal{A}$ *is defined as*

$$\mathbf{Adv}^{\mu\text{-auth}}_{\text{Ascon-AE}}(\mathcal{A}) = \mathbf{Pr}\left(\mathcal{A}\left[\left(\mathbf{Enc}^{\mathcal{P}}_{K_m}, \mathbf{Dec}^{\mathcal{P}}_{K_m}\right)_{m=1}^{\mu}, \mathcal{P}^{\pm}\right] \textit{ forges}\right),$$

  where $\mathcal{A}$ *is restricted as follows:* $\mathcal{O}_{1,m} \overset{N}{\not\hookrightarrow} \mathcal{O}_{1,m}$ *and* $\mathcal{O}_{1,m} \overset{*}{\not\hookrightarrow} \mathcal{O}_{2,m}$*. Here, "forges" denotes the event that* $\mathcal{A}$ *makes a query to one of the oracles* $\mathbf{Dec}^{\mathcal{P}}_{K_m}$ *that does not return* $\perp$*.*

In (2.4), we saw that

$$\mathbf{Adv}^{\mu\text{-ae}}_{\text{Ascon-AE}}(\mathcal{A}) \leq \mathbf{Adv}^{\mu\text{-auth}}_{\text{Ascon-AE}}(\mathcal{A}') + \mathbf{Adv}^{\mu\text{-conf}}_{\text{Ascon-AE}}(\mathcal{A}''),$$

for some adversaries $\mathcal{A}'$ and $\mathcal{A}''$ with the same query complexities as $\mathcal{A}$.

9.4.1.1 Overview

In 2014, Jovanovic et al. [JLM14,JLM+19] analyzed the security of the duplex-based mode NORX [AJN14b], providing bounds for both confidentiality and authenticity in the nonce-respecting setting. They mentioned that their analysis can be generalized to Ascon-AE, but they did not provide a proof. We [LM24a] gave a dedicated analysis and recovered the bounds claimed by Jovanovic et al., i.e., we proved that (assuming that $\mu \leq \mathcal{M} \ll \mathcal{N}$),

$$\mathbf{Adv}^{\mu\text{-ae}}_{\text{Ascon-AE}}(\mathcal{A}) = \tilde{\mathcal{O}}\left(\frac{Q_D}{2^t} + \frac{\mu\mathcal{N}}{2^k} + \frac{\mathcal{N}^2}{2^b} + \frac{\mathcal{M}_D\mathcal{N}}{2^c}\right).$$

In an independent work, Chakraborty et al. [CDN23b] derived a tighter bound in the single user setting, and later [CDN24] extended it to the multi-user setting. We present their result in Theorem 9.4.1, leaving the multicollision terms in an abstract form (cf., Section 2.5.2).

Theorem 9.4.1 ([CDN23b, CDN24]). *Let $b, c, r, k, n, t \in \mathbb{N}^*$ with $b = r + c$, $k + n \leq b$, $t \leq k$, and $k \leq c$. Let $\mu, \mathcal{N}, \mathcal{M}_E, \mathcal{M}_D, Q_E, Q_D \in \mathbb{N}$. Consider the* Ascon-AE *mode of Section 9.2 with parameters b, c, r, k, n, t. Let $\mathcal{A}$ be an adversary with complexity $(\mathcal{N}, Q_E, \mathcal{M}_E, Q_D, \mathcal{M}_D)$ (see Section 9.3.2 for a detailed definition of complexity). We have*

$$\begin{aligned}\mathbf{Adv}^{\mu\text{-ae}}_{\text{Ascon-AE}}(\mathcal{A}) \leq{} & \frac{\mu^2}{2^k} + \frac{2Q_D}{2^t} + \frac{\mathcal{M}_E^2}{2^b} + \frac{\mathcal{M}_D(\mathcal{N} + \mathcal{M}_D)}{2^b} + \frac{Q_D(\mathcal{M} + \mathcal{N})}{2^{c+t}} \\ & + \frac{\text{mucol}(\mathcal{M}_E, 2^r)(\mathcal{M}_D + \mathcal{N})}{2^c} + \frac{\text{mucol}(\mathcal{M} + \mathcal{N}, 2^t)Q_D}{2^k} \\ & + \frac{Q_E^2 + Q_D^2 + Q_E Q_D + (2Q_E + Q_D)(\mathcal{M} + \mathcal{N})}{2^b} + \frac{\mu(\mathcal{N} + \mathcal{M})}{2^k} \\ & + \frac{\text{mucol}(Q_E, 2^{b-k})(\mathcal{M} + \mathcal{N})}{2^k} + \frac{\text{mucol}(Q_E, 2^t)Q_D}{2^c}\,.\end{aligned}$$

Ignoring constant and logarithmic factors, and using that $\text{mucol}(q, R) = \tilde{\mathcal{O}}\left(1 + \frac{q}{R}\right)$ and $\mu \leq \mathcal{M} \ll \mathcal{N}$ (cf., Section 9.3.2), we obtain

$$\mathbf{Adv}^{\mu\text{-ae}}_{\text{Ascon-AE}}(\mathcal{A}) = \tilde{\mathcal{O}}\left(\frac{Q_D}{2^t} + \frac{\mu\mathcal{N}}{2^k} + \frac{\mathcal{M}\mathcal{N}}{2^b} + \frac{\mathcal{N}}{2^c}\right). \tag{$\star$}$$

Throughout this work, we will refer to this term as the *core term*, noting that it contributes to many of the bounds that will follow in the rest of this work. A notable observation is that the term $\tilde{\mathcal{O}}\left(\frac{\mathcal{M}_D\mathcal{N}}{2^c}\right)$ is absent in ($\star$). Indeed, in typical duplex-based authenticated encryption schemes, where keys are only added to the initial state (such as in MonkeySpongeWrap [Men23]), this term appears in the nonce-respecting bounds (cf., (3.11)), along with a tight attack for certain parameter sets: Gilbert et al. [GBKR23] described an attack tailored to precise values of $\mathcal{N}$ and $\mathcal{M}_D$ such that $\mathcal{M}_D\mathcal{N}$ is above 2^c, and in the last paragraph of Section 3.3.1.3, we remarked that taking $\mathcal{N} \approx \mathcal{M}_D \approx 2^{c/2}$ allows their attack to succeed with high probability. For Ascon-AE, and in particular the bound of Theorem 9.4.1, this term is absent due to the additional key blindings, that turn out to enhance conventional nonce-respecting security.

The core bound ($\star$) is tight in the simplified setting of Section 9.3.2, as we show in Proposition 9.4.1.

Proposition 9.4.1. *Let $b, c, r, k, n, t \in \mathbb{N}^*$ with $b = r + c$, $k + n \leq b$, $t \leq k$, and $k \leq c$. Consider the* Ascon-AE *mode of Section 9.2 with parameters b, c, r, k, n, t. There exist adversaries $\mathcal{A}_1$ with $Q_D \approx 2^t$, $\mathcal{A}_2$ with $\mathcal{N} \approx 2^k/\mu$, and $\mathcal{A}_3$ with $(\mathcal{M}_E + 2^r)\mathcal{N} \approx 2^b$, such that*

$$\mathbf{Adv}^{\mu\text{-ae}}_{\text{Ascon-AE}}(\mathcal{A}_1)\,,\ \mathbf{Adv}^{\mu\text{-ae}}_{\text{Ascon-AE}}(\mathcal{A}_2)\,,\ \mathbf{Adv}^{\mu\text{-ae}}_{\text{Ascon-AE}}(\mathcal{A}_3) \approx 1\,.$$

The proof of Proposition 9.4.1 is given in Section 9.7.1.

9.4.2 Nonce-Misuse Resistance

9.4.2.1 Security Model

The security model of Section 9.4.1 restricts the adversary to only use fresh nonces for encryption queries. Rogaway and Shrimpton [RS06] proposed the notion of nonce-misuse resistance, to capture settings where the adversary may have the power to reuse nonces. As, throughout this work, we do not focus on a unified AEAD security definition but rather on the separated notions of confidentiality and authenticity, we only extend Definition 9.4.2 to the nonce-misuse resistance setting. The generalization is straightforward: it mainly consists of dropping the nonce-misuse restrictions.

Definition 9.4.3. *Consider the* Ascon-AE *mode of Section 9.2. Let $(\$_m)_{m=1}^{\mu}$ be a family of μ independent random functions, $\mathcal{P} \xleftarrow{\$} \texttt{Perm}(b)$, and $K_1, \ldots, K_\mu \xleftarrow{\$} \{0,1\}^k$.*

- *The nonce-misuse resistance confidentiality of* Ascon-AE *against an adversary $\mathcal{A}$ is defined as*

$$\mathbf{Adv}^{\mu\text{-m-conf}}_{\text{Ascon-AE}}(\mathcal{A}) = \Delta_{\mathcal{A}}\left(\left(\mathbf{Enc}^{\mathcal{P}}_{K_m}\right)_{m=1}^{\mu}, \mathcal{P}^{\pm}\ ;\ (\$_m)_{m=1}^{\mu}, \mathcal{P}^{\pm}\right)\ ;$$

- *The nonce-misuse resistance authenticity of* Ascon-AE *against an adversary $\mathcal{A}$ is defined as*

$$\mathbf{Adv}^{\mu\text{-m-auth}}_{\text{Ascon-AE}}(\mathcal{A}) = \mathbf{Pr}\left(\mathcal{A}\left[\left(\mathbf{Enc}^{\mathcal{P}}_{K_m}, \mathbf{Dec}^{\mathcal{P}}_{K_m}\right)_{m=1}^{\mu}, \mathcal{P}^{\pm}\right] \text{forges}\right)\ ,$$

where $\mathcal{A}$ is restricted as follows: $\mathcal{O}_{1,m} \overset{*}{\not\hookrightarrow} \mathcal{O}_{2,m}$. Here, "forges" denotes the event that $\mathcal{A}$ makes a query to one of the oracles $\mathbf{Dec}^{\mathcal{P}}_{K_m}$ that does not return $\perp$.

This is the strongest possible attack setting with respect to nonce-reuse, and it is trivial to observe that nonce-misuse security implies nonce-respecting security. We wish to remark that one-pass schemes, i.e., authenticated encryption functions that make only one pass over the data, are known to impossibly achieve nonce-misuse confidentiality in terms of Definition 9.4.3, and this particularly applies to Ascon-AE as we demonstrate in Proposition 9.4.2 below.

Proposition 9.4.2. *Consider the* Ascon-AE *mode of Section 9.2 with parameters b, c, r, k, n, t. There exists an adversary $\mathcal{A}$ with $Q_E = 2$, such that*

$$\mathbf{Adv}^{\mu\text{-m-conf}}_{\text{Ascon-AE}}(\mathcal{A}) \approx 1\,.$$

Proof. Let $P_1, P_2 \in \{0,1\}^r$, $N \in \{0,1\}^n$, $m \in [\![1, \mu]\!]$. Consider the following attack:

1. Make an encryption query with user m with input (N, ϵ, P_1), denote the ciphertext by $C_1 \in \{0,1\}^r$;
2. Make an encryption query with user m with input $(N, \epsilon, P_1 \| P_2)$, denote the ciphertext by $C_1' \| C_2'$ where $C_1', C_2' \in \{0,1\}^r$;
3. If $C_1 = C_1'$ return 0, else 1.

In the real world, a repeated block will always output the same ciphertext block, while in the ideal world, this happens with probability $\frac{1}{2^r}$. This term can be reduced further by repeating the attack or by mounting the attack for longer encryption queries. □

9.4.2.2 Overview

We now focus on authenticity. We [LM24a] derived a nonce-misuse authenticity bound of the order

$$\tilde{\mathcal{O}}\left((\star) + \frac{\mathcal{M}\mathcal{N}}{2^c}\right)\,. \tag{9.2}$$

Chakraborty et al. [CDN24] derived a nonce-misuse authenticity bound of the order

$$\tilde{\mathcal{O}}\left((\star) + \frac{\mathcal{M}_E^2}{2^c}\right)\,.$$

However, we identify a flaw in Chakraborty et al.'s analysis. In Proposition 9.4.3, we show that there exists a forgery attack with a success probability of $\approx \frac{\mathcal{M}_E \mathcal{N}}{2^c}$, which contradicts their bound. The cause of this flaw is that the nonce-misuse resistance analysis of Chakraborty et al. [CDN24] is a fairly direct extension of their nonce-respecting setting analysis [CDN23b]. However, in this generalization, some unique aspects of nonce-misuse seem to have been overlooked. In detail, in their proof, the bad event $\mathbf{bad}_5$ is used to compute the probability that there exists a collision between permutation evaluations from encryption queries and permutation evaluations from decryption or permutation queries. In the nonce-misuse setting, the adversary can set the outer part of inner states during encryption queries to a value of its choice, thus this event happens with a probability of form $\tilde{\mathcal{O}}\left(\frac{\mathcal{M}_E\mathcal{N}}{2^c}\right)$, and not $\tilde{\mathcal{O}}\left(\frac{\mathcal{M}_D+\mathcal{N}}{2^c}+\frac{\mathcal{M}_E(\mathcal{M}_D+\mathcal{N})}{2^b}\right)$ as claimed in [CDN24].

We therefore consider our result (9.2). However, it holds in a model where the outer and inner permutations are assumed to be independent, making it incompatible with the model that we consider in this chapter (cf., Remark 9.2.1). Therefore, we establish a new nonce-misuse authenticity bound in Theorem 9.4.2.

Theorem 9.4.2. *Let $b, c, r, k, n, t \in \mathbb{N}^*$ with $b = r + c$, $k + n \leq b$, $t \leq k$, and $k \leq c$. Let $\mu, \mathcal{N}, \mathcal{M}_E, \mathcal{M}_D, Q_E, Q_D \in \mathbb{N}$. Consider the* Ascon-AE *mode of Section 9.2 with parameters b, c, r, k, n, t. Let $\mathcal{A}$ be an adversary with complexity $(\mathcal{N}, Q_E, \mathcal{M}_E, Q_D, \mathcal{M}_D)$ (see Section 9.3.2 for a detailed definition of complexity). We have*

$$\mathbf{Adv}^{\mu\text{-m-auth}}_{\text{Ascon-AE}}(\mathcal{A}) \leq \frac{\mu(\mu-1)}{2^{k+1}} + \frac{2\mu\,(\mathcal{M}+\mathcal{N})}{2^k} + \frac{18\mathcal{M}\,(\mathcal{M}+\mathcal{N})}{2^c} + \frac{2Q_D}{2^t}\,.$$

The proof of Theorem 9.4.2 is given in Section 9.6.1.

Using that $\mu \leq \mathcal{M}_E \ll \mathcal{N}$ (cf., Section 9.3.2), we obtain a bound of the order

$$\mathbf{Adv}^{\mu\text{-m-auth}}_{\text{Ascon-AE}}(\mathcal{A}) = \tilde{\mathcal{O}}\left((\star) + \frac{\mathcal{M}\mathcal{N}}{2^c}\right)\,. \tag{9.3}$$

The bound (9.3) is tight in the simplified setting of Section 9.3.2. The attacks from Proposition 9.4.1 targeting $(\star)$ also apply here, and below Proposition 9.4.3 matches the term $\frac{\mathcal{M}\mathcal{N}}{2^c}$.

Proposition 9.4.3. *Let $b, c, r, k, n, t \in \mathbb{N}^*$ with $b = r + c$, $k + n \leq b$, $t \leq k$, and $k \leq c$. Consider the* Ascon-AE *mode of Section 9.2 with parameters b, c, r, k, n, t. There exists an adversary $\mathcal{A}$ with $\mathcal{M}_E\mathcal{N} \approx 2^c$, such that*

$$\mathbf{Adv}^{\mu\text{-m-auth}}_{\text{Ascon-AE}}(\mathcal{A}) \approx 1\,.$$

The proof of Proposition 9.4.3 is given in Section 9.7.2. Notably, this generic attack invalidates the bound of Chakraborty et al. [CDN24].

9.4.3 Nonce-Misuse Resilience

9.4.3.1 Security Model

In Section 9.4.2, we already mentioned that nonce-misuse resistance is the strongest possible attack setting with respect to nonce-misuse, and one-pass schemes can never achieve confidentiality in this setting. However, one may argue that such one-pass modes still achieve *some* level of confidentiality, namely confidentiality up to common prefix. This idea was formalized under the notion of online authenticated encryption by Fleischmann et al. [FFL12], and also used by notable authenticated encryption schemes like COPA [ABL$^+$13]. However, this notion has been heavily debated because nonce-misuse may still be devastating through a trivial attack (and also because of different conceptual reasons) [HRRV15]. Although Hoang et al. [HRRV15] do amend their criticism with an alternative notion for online authenticated encryption security, we will not adopt this notion.

Instead, to define a middle-ground between nonce-respecting security (of Section 9.4.1) and nonce-misuse resistance (of Section 9.4.2), we resort to the notion of nonce-misuse resilience of Ashur et al. [ADL17]. At a high level, this notion covers that nonce-misuse only affects encryptions under that nonce, and for new nonces, security is still guaranteed. In a bit more detail, for confidentiality, the adversary gets encryption access in two ways: through *challenge* and *non-challenge* queries, where challenge queries should always be for new nonces and non-challenge queries may be for repeated nonces. For authenticity, the adversary may repeat nonces for encryption, but if a nonce is reused, it may not be used in a forgery attempt anymore.

Definition 9.4.4. *Consider the* Ascon-AE *mode of Section 9.2. Let* $(\$_m)_{m=1}^{\mu}$ *be a family of* μ *independent random functions,* $\mathcal{P} \xleftarrow{\$} \texttt{Perm}(b)$*, and* $K_1, \ldots, K_\mu \xleftarrow{\$} \{0,1\}^k$.

- *The nonce-misuse resilience confidentiality of* Ascon-AE *against an adversary* $\mathcal{A}$ *is defined as*

$$\mathbf{Adv}^{\mu\text{-mr-conf}}_{\text{Ascon-AE}}(\mathcal{A}) = \Delta_{\mathcal{A}}\left(\left(\mathbf{Enc}^{\mathcal{P}}_{K_m}, \mathbf{Enc}^{\mathcal{P}}_{K_m}\right)_{m=1}^{\mu}, \mathcal{P}^{\pm} \; ; \; \left(\mathbf{Enc}^{\mathcal{P}}_{K_m}, \$_m\right)_{m=1}^{\mu}, \mathcal{P}^{\pm}\right),$$

  where $\mathcal{A}$ *is restricted as follows:* $\mathcal{O}_{1,m}/\mathcal{O}_{2,m} \overset{N}{\not\hookrightarrow} \mathcal{O}_{2,m}$*, and* $\mathcal{O}_{2,m} \overset{N}{\not\hookrightarrow} \mathcal{O}_{1,m}$*;*

- *The nonce-misuse resilience authenticity of* Ascon-AE *against an adversary*

$\mathcal{A}$ is defined as

$$\mathbf{Adv}^{\mu\text{-mr-auth}}_{\text{Ascon-AE}}(\mathcal{A}) = \mathbf{Pr}\left(\mathcal{A}\left[\left(\mathbf{Enc}^{\mathcal{P}}_{K_m}, \mathbf{Dec}^{\mathcal{P}}_{K_m}\right)^{\mu}_{m=1}, \mathcal{P}^{\pm}\right] \text{ forges}\right) ,$$

where $\mathcal{A}$ is restricted as follows: $\mathcal{O}_{1,m} \overset{N}{\not\hookrightarrow} \mathcal{O}_{2,m}$ and $\mathcal{O}_{1,m} \overset{*}{\not\hookrightarrow} \mathcal{O}_{2,m}$. Here, "forges" denotes the event that $\mathcal{A}$ makes a query to one of the oracles $\mathbf{Dec}^{\mathcal{P}}_{K_m}$ that does not return $\perp$.

It is worth noting that nonce-misuse resilience is indeed situated in-between nonce-respecting security and nonce-misuse resistance, or more technically, it implies nonce-respecting security and it is itself implied by nonce-misuse resistance. This observation will also be used below to argue nonce-misuse resilience of Ascon-AE.

9.4.3.2 Overview

Guo et al. [GPPS19b] considered nonce-misuse resilience of Ascon-AE under two models, named muCCAmL1 (multi-user Chosen-Ciphertext Attack security with misuse-resilience and Leakage), and muCIML2 (multi-user Ciphertext Integrity with Misuse-resistance and Leakage). The bounds are in a leaky setting, therefore yielding lossy bounds if we are only interested in nonce-misuse resilience.

More importantly, as the authors admit, their results are merely proof sketches and only stated in big-O terms. A more detailed look at their informal reasoning reveals various unclear and unconvincing steps. Most importantly, the core step of their reasoning is to replace KDF_{K_i} (basically, in our terminology, the first permutation call including the key blindings) and TGF_{K_i} (the last permutation call including the key blindings) by secret random functions. This step is made at the cost of the PRF security of the multi-instance partial-key Even-Mansour cipher [EM91, EM97], which the authors claim to be (in our terminology) $\tilde{\mathcal{O}}\left(\frac{(\mathcal{N}+\mathcal{M})^2}{2^c}\right)$. As $\mathcal{N}+\mathcal{M}$ is de facto the total complexity, this bound seems lossy when purely focusing on the initialization and finalization. Of course, this is a minor issue as a tighter term for the multi-instance partial-key Even-Mansour cipher was already derived by Andreeva et al. [ADMV15, Theorem 4] and as the claimed term also appears in the security of the hash portion between key derivation and tag generation. What is more worrisome is that the reasoning is incorrect. To be precise: the security of the multi-instance partial-key Even-Mansour cipher is relative to the key size and not to the capacity. With the parametrization of Ascon-AE, the key is of size $k \leq c$, and the bound of Andreeva et al. [ADMV15, Theorem 4] would carry over with c replaced by k to match our context. This substitution leads to a term of the order $\frac{\mathcal{N}\cdot\text{multiplicity}+Q^2}{2^k}$. We stress that this gives a dramatic security loss, noting that with the parametrization of Ascon-128a, $k = 128$ and $c = 192$, and additionally, the multiplicity term of the partial-key Even-Mansour cipher replacing KDF_{K_i} can be as large as multiplicity $\approx \min\{Q, 2^n\}$. As a matter of fact, in earlier proof attempts, we independently followed the same reasoning as Guo et al., but quickly departed from it as the security loss described here turned out to be unavoidable.

Therefore, we derive our own nonce-misuse resilience security analysis, in Theorem 9.4.3.

Theorem 9.4.3. *Let $b, c, r, k, n, t \in \mathbb{N}^*$ with $b = r + c$, $k + n \leq b$, $t \leq k$, and $k \leq c$. Let $\mu, \mathcal{N}, \mathcal{M}_E, \mathcal{M}_D, Q_E, Q_D \in \mathbb{N}$. Consider the* Ascon-AE *mode of Section 9.2 with parameters b, c, r, k, n, t. Let $\mathcal{A}^{\text{conf}}$ be an adversary with complexity $(\mathcal{N}, Q_E, \mathcal{M}_E)$, and $\mathcal{A}^{\text{auth}}$ with complexity $(\mathcal{N}, Q_E, \mathcal{M}_E, Q_D, \mathcal{M}_D)$ (see Section 9.3.2 for a detailed definition of complexity). We have*

$$\mathbf{Adv}^{\mu\text{-mr-conf}}_{\text{Ascon-AE}}\left(\mathcal{A}^{\text{conf}}\right) \leq \frac{\mu(\mu-1)}{2^{k+1}} + \frac{2\mu\left(\mathcal{M}_E + \mathcal{N}\right)}{2^k} + \frac{18\mathcal{M}_E\left(\mathcal{M}_E + \mathcal{N}\right)}{2^c},$$
$$\mathbf{Adv}^{\mu\text{-mr-auth}}_{\text{Ascon-AE}}\left(\mathcal{A}^{\text{auth}}\right) \leq \frac{\mu(\mu-1)}{2^{k+1}} + \frac{2\mu\left(\mathcal{M} + \mathcal{N}\right)}{2^k} + \frac{18\mathcal{M}\left(\mathcal{M} + \mathcal{N}\right)}{2^c} + \frac{2Q_D}{2^t}.$$

Here, the authenticity bound follows from our nonce-misuse resistance proof of Theorem 9.4.2. The confidentiality proof of Theorem 9.4.3 is new and is given in Section 9.6.2.

Using that $\mu \leq \mathcal{M}_E \ll \mathcal{N}$ (cf., Section 9.3.2), we obtain bounds of the order

$$\mathbf{Adv}^{\mu\text{-mr-conf}}_{\text{Ascon-AE}}\left(\mathcal{A}^{\text{conf}}\right) = \tilde{\mathcal{O}}\left((\star) + \frac{\mathcal{M}_E\mathcal{N}}{2^c}\right), \tag{9.4}$$
$$\mathbf{Adv}^{\mu\text{-mr-auth}}_{\text{Ascon-AE}}\left(\mathcal{A}^{\text{auth}}\right) = \tilde{\mathcal{O}}\left((\star) + \frac{\mathcal{M}\mathcal{N}}{2^c}\right). \tag{9.5}$$

In particular, nonce-misuse resilience confidentiality and authenticity have a bound of the same order as authenticity in the nonce-misuse resistance setting.

The attacks from Proposition 9.4.1 targeting the core term $(\star)$ also apply here, and Proposition 9.4.4 matches the term $\frac{\mathcal{M}_E\mathcal{N}}{2^c}$. However, the parametrization of $\mathcal{N}$ and $\mathcal{M}_E$ is not completely free here, as the attack requires $\mathcal{N} \geq 2^{k-t}$. Considering the parameter sets of the instances Ascon-128 and Ascon-128a (see Section 9.2), the dominating term in the bound is $\frac{\mu\mathcal{N}}{2^k}$ anyway. On the other hand, for the parameter sets of Ascon-80pq, the constraint translates to

$\mathcal{N} \geq 2^{32}$, which is more than reasonable. Therefore, in the simplified setting of Section 9.3.2, the bounds (9.4) and (9.5) are tight for meaningful parameter sets.

Proposition 9.4.4. *Let $b, c, r, k, n, t \in \mathbb{N}^*$ with $b = r + c$, $k + n \leq b$, $t \leq k$, and $k \leq c$. Consider the* Ascon-AE *mode of Section 9.2 with parameters b, c, r, k, n, t. There exist two adversaries $\mathcal{A}^{\text{conf}}$ and $\mathcal{A}^{\text{auth}}$ with $\mathcal{M}_E\mathcal{N} \approx 2^c$ and $\mathcal{N} \geq 2^{k-t}$, such that*

$$\mathbf{Adv}^{\mu\text{-mr-conf}}_{\text{Ascon-AE}}\left(\mathcal{A}^{\text{conf}}\right), \mathbf{Adv}^{\mu\text{-mr-auth}}_{\text{Ascon-AE}}\left(\mathcal{A}^{\text{auth}}\right) \approx 1\,.$$

A proof of Proposition 9.4.4 can be found in Section 9.7.3.

9.5 Leakage Security of Ascon-AE

We will consider the security of the Ascon-AE mode in leaky settings, where the adversary may learn some internal information during executions through some implementation mistake or through side-channels. We start with leakage resilience in Section 9.5.1, followed by state-recovery security in Section 9.5.2, and release of unverified plaintext in Section 9.5.3.

9.5.1 Leakage Resilience

9.5.1.1 Security Model

Authenticated encryption schemes are implemented on a wide variety of platforms, and particularly lightweight authenticated encryption schemes may be implemented in constrained environments which may leak information. It thus makes sense to analyze the leakage resilience of Ascon-AE. There exist various different models on how to model leakage and how to model oracle access. In this work, we restrict our focus to leakage resilience in the bounded leakage model, as set forth by Dziembowski and Pietrzak [DP08] and adopted in various later works [Pie09, YSPY10, FPS12, SPY$^+$10, DP10]. In this model, the adversary gets access to a leak-free version of the construction which it has to distinguish from random, exactly as in the models of Section 9.4, but *in addition* it gets access to a leaky version of the scheme, which it may use to gather information. In this leaky version, we assume a priori that every permutation evaluation $\mathcal{P} : X \mapsto Y$ leaks certain information. This leakage is captured through a leakage function $L : \{0,1\}^b \times \{0,1\}^b \to \{0,1\}^\lambda$ for some small value λ. For a function $\mathbf{F}^{\mathcal{P}}$ on top of permutation $\mathcal{P}$, we define $\left[\mathbf{F}^{\mathcal{P}}\right]_L$ to be an evaluation of $\mathbf{F}^{\mathcal{P}}$ that additionally leaks $L(X, Y)$ for each evaluation $\mathcal{P} : X \mapsto Y$.

We assume non-adaptive leakage, where the leakage function is defined prior to the experiment and stays fixed throughout [FPS12]. In detail, we define some set of permitted leakage functions $\mathcal{L} = \{L : \{0,1\}^b \times \{0,1\}^b \to \{0,1\}^\lambda\}$ and expect security for any $L \in \mathcal{L}$.

We finally remark that, in its natural form, Ascon-AE cannot achieve confidentiality and integrity under any set of leakage functions. The reason is that the leakage function L can be chosen in such a way that, from the first evaluation of $\mathcal{P}$, different portions of the key are leaked for different choices of N (a similar attack is described in a bit more detail in Section 9.5.1.2). To salvage this, we adopt the notion of leveled implementations as put forward by Pereira et al. [PSV15], and that was also adopted in leakage resilience analyzes of authenticated encryption schemes [GPPS19a,BGP⁺20,GPPS20,GPPS19b]. In the concept of leveled implementations, applied to our context, the outer permutations are strongly protected and do not leak any information, whereas the inner permutations may leak.

In detail, we refine $\left[\mathbf{F}^{\mathcal{P}}\right]_L$ to be an evaluation of $\mathbf{F}^{\mathcal{P}}$ that additionally leaks $L(X,Y)$ for each *inner* evaluation $\mathcal{P} : X \mapsto Y$. This, finally, leads us to the following model, which is based on Barwell et al. [BMOS17] but with two differences: (i) we translate the model to the random permutation model, and (ii) we do not generalize from conventional nonce-respecting security (Section 9.4.1) but rather from nonce-misuse resilience (Section 9.4.3).

Definition 9.5.1. *Consider the* Ascon-AE *mode of Section 9.2. Let* $(\$_m)_{m=1}^{\mu}$ *be a family of* μ *independent random functions,* $\mathcal{P} \xleftarrow{\$} \mathtt{Perm}(b)$*, and* $K_1, \dots, K_\mu \xleftarrow{\$} \{0,1\}^k$*. Let* $\mathcal{L}$ *be a set of leakage functions. The leakage resilience AEAD-security of* Ascon-AE *against an adversary* $\mathcal{A}$ *with respect to* $\mathcal{L}$ *is defined as*

$$\mathbf{Adv}^{\mu\text{-lr-ae}}_{\text{Ascon-AE},\mathcal{L}}(\mathcal{A}) = \max_{L \in \mathcal{L}} \Delta_{\mathcal{A}}\Bigg(\left(\left[\mathbf{Enc}^{\mathcal{P}}_{K_m}\right]_L, \mathbf{Enc}^{\mathcal{P}}_{K_m}, \left[\mathbf{Dec}^{\mathcal{P}}_{K_m}\right]_L, \mathbf{Dec}^{\mathcal{P}}_{K_m} \right)_{m=1}^{\mu}, \mathcal{P}^{\pm} ; \left(\left[\mathbf{Enc}^{\mathcal{P}}_{K_m}\right]_L, \$_m, \left[\mathbf{Dec}^{\mathcal{P}}_{K_m}\right]_L, \bot \right)_{m=1}^{\mu}, \mathcal{P}^{\pm} \Bigg),$$

where $\mathcal{A}$ *is restricted as follows:* $\mathcal{O}_{2,m} \overset{N}{\not\hookrightarrow} \mathcal{O}_{1,m}$*,* $\mathcal{O}_{1,m}/\mathcal{O}_{2,m}/\mathcal{O}_{3,m} \overset{N}{\not\hookrightarrow} \mathcal{O}_{2,m}$*,* $\mathcal{O}_{2,m} \overset{N}{\not\hookrightarrow} \mathcal{O}_{3,m}$*,* $\mathcal{O}_{1,m}/\mathcal{O}_{3,m} \overset{N}{\not\hookrightarrow} \mathcal{O}_{4,m}$*, and* $\mathcal{O}_{2,m} \overset{*}{\not\hookrightarrow} \mathcal{O}_{4,m}$*.*

Just like in Section 9.4.1, it is more convenient to consider confidentiality and authenticity separately.

Definition 9.5.2. *Consider the* Ascon-AE *mode of Section 9.2. Let* $(\$_m)_{m=1}^{\mu}$ *be a family of* μ *independent random functions,* $\mathcal{P} \xleftarrow{\$} \texttt{Perm}(b)$*, and* $K_1, \ldots, K_\mu \xleftarrow{\$} \{0,1\}^k$*. Let* $\mathcal{L}$ *be a set of leakage functions.*

- *The leakage resilience confidentiality of* Ascon-AE *against an adversary* $\mathcal{A}$ *with respect to* $\mathcal{L}$ *is defined as*

$$\begin{aligned}\mathbf{Adv}^{\mu\text{-lr-conf}}_{\text{Ascon-AE},\mathcal{L}}(\mathcal{A}) = \max_{L\in\mathcal{L}} \Delta_{\mathcal{A}}\Big(&\Big(\Big[\mathbf{Enc}^{\mathcal{P}}_{K_m}\Big]_L, \mathbf{Enc}^{\mathcal{P}}_{K_m}, \Big[\mathbf{Dec}^{\mathcal{P}}_{K_m}\Big]_L\Big)_{m=1}^{\mu}, \mathcal{P}^{\pm} ;\\ &\Big(\Big[\mathbf{Enc}^{\mathcal{P}}_{K_m}\Big]_L, \$_m, \Big[\mathbf{Dec}^{\mathcal{P}}_{K_m}\Big]_L\Big)_{m=1}^{\mu}, \mathcal{P}^{\pm}\Big),\end{aligned}$$

where $\mathcal{A}$ *is restricted as follows:* $\mathcal{O}_{2,m} \overset{N}{\not\to} \mathcal{O}_{1,m}$, $\mathcal{O}_{1,m}/\mathcal{O}_{2,m}/\mathcal{O}_{3,m} \overset{N}{\not\to} \mathcal{O}_{2,m}$, *and* $\mathcal{O}_{2,m} \overset{N}{\not\to} \mathcal{O}_{3,m}$*;*

- *The leakage resilience authenticity of* Ascon-AE *against an adversary* $\mathcal{A}$ *with respect to* $\mathcal{L}$ *is defined as*

$$\begin{aligned}&\mathbf{Adv}^{\mu\text{-lr-auth}}_{\text{Ascon-AE},\mathcal{L}}(\mathcal{A}) =\\ &\max_{L\in\mathcal{L}} \mathbf{Pr}\left(\mathcal{A}\left[\Big(\Big[\mathbf{Enc}^{\mathcal{P}}_{K_m}\Big]_L, \mathbf{Enc}^{\mathcal{P}}_{K_m}, \Big[\mathbf{Dec}^{\mathcal{P}}_{K_m}\Big]_L, \mathbf{Dec}^{\mathcal{P}}_{K_m}\Big)_{m=1}^{\mu}, \mathcal{P}^{\pm}\right] \textit{ forges}\right),\end{aligned}$$

where $\mathcal{A}$ *is restricted as follows:* $\mathcal{O}_{2,m} \overset{N}{\not\to} \mathcal{O}_{1,m}$, $\mathcal{O}_{1,m}/\mathcal{O}_{2,m}/\mathcal{O}_{3,m} \overset{N}{\not\to} \mathcal{O}_{2,m}$, $\mathcal{O}_{2,m} \overset{N}{\not\to} \mathcal{O}_{3,m}$, $\mathcal{O}_{1,m}/\mathcal{O}_{3,m} \overset{N}{\not\to} \mathcal{O}_{4,m}$, *and* $\mathcal{O}_{2,m} \overset{*}{\not\to} \mathcal{O}_{4,m}$*.*

We can again demonstrate that the notions are equivalent, using a similar reasoning as in Section 9.4.1 (or, to be precise, using a similar reasoning as Ashur et al. [ADL17, Section 4.4]). In detail, let $\mathcal{A}$ be any adversary against the AEAD security of Ascon-AE. Then,

$$\begin{aligned}
&\mathbf{Adv}^{\mu\text{-lr-ae}}_{\text{Ascon-AE},\mathcal{L}}(\mathcal{A}) \\
&\quad = \max_{L\in\mathcal{L}} \Delta_{\mathcal{A}}\Bigg(\left(\left[\mathbf{Enc}^{\mathcal{P}}_{K_m}\right]_L, \mathbf{Enc}^{\mathcal{P}}_{K_m}, \left[\mathbf{Dec}^{\mathcal{P}}_{K_m}\right]_L, \mathbf{Dec}^{\mathcal{P}}_{K_m}\right)_{m=1}^{\mu}, \mathcal{P}^{\pm} \,; \\
&\qquad\qquad \left(\left[\mathbf{Enc}^{\mathcal{P}}_{K_m}\right]_L, \$_m, \left[\mathbf{Dec}^{\mathcal{P}}_{K_m}\right]_L, \perp\right)_{m=1}^{\mu}, \mathcal{P}^{\pm}\Bigg) \\
&\quad \le \max_{L\in\mathcal{L}} \Delta_{\mathcal{A}}\Bigg(\left(\left[\mathbf{Enc}^{\mathcal{P}}_{K_m}\right]_L, \mathbf{Enc}^{\mathcal{P}}_{K_m}, \left[\mathbf{Dec}^{\mathcal{P}}_{K_m}\right]_L, \mathbf{Dec}^{\mathcal{P}}_{K_m}\right)_{m=1}^{\mu}, \mathcal{P}^{\pm} \,; \\
&\qquad\qquad \left(\left[\mathbf{Enc}^{\mathcal{P}}_{K_m}\right]_L, \mathbf{Enc}^{\mathcal{P}}_{K_m}, \left[\mathbf{Dec}^{\mathcal{P}}_{K_m}\right]_L, \perp\right)_{m=1}^{\mu}, \mathcal{P}^{\pm}\Bigg) \\
&\qquad + \max_{L\in\mathcal{L}} \Delta_{\mathcal{A}}\Bigg(\left(\left[\mathbf{Enc}^{\mathcal{P}}_{K_m}\right]_L, \mathbf{Enc}^{\mathcal{P}}_{K_m}, \left[\mathbf{Dec}^{\mathcal{P}}_{K_m}\right]_L, \perp\right)_{m=1}^{\mu}, \mathcal{P}^{\pm} \,; \\
&\qquad\qquad \left(\left[\mathbf{Enc}^{\mathcal{P}}_{K_m}\right]_L, \$_m, \left[\mathbf{Dec}^{\mathcal{P}}_{K_m}\right]_L, \perp\right)_{m=1}^{\mu}, \mathcal{P}^{\pm}\Bigg) \\
&\quad \le \mathbf{Adv}^{\mu\text{-lr-auth}}_{\text{Ascon-AE},\mathcal{L}}(\mathcal{A}') + \mathbf{Adv}^{\mu\text{-lr-conf}}_{\text{Ascon-AE},\mathcal{L}}(\mathcal{A}'') \,,
\end{aligned}$$

for some adversaries $\mathcal{A}'$ and $\mathcal{A}''$ with the same query complexities as $\mathcal{A}$.

Note that if we take the model of Definition 9.5.2 with no leakage, hence with $\lambda = 0$, the definition is equivalent to nonce-misuse resilience, which was the starting point of our model. Looking ahead, we may then consider the notion for arbitrary limited leakage (for any λ) or the notion for unlimited leakage where $\lambda = b$. Clearly, leakage resilience with no leakage is implied by leakage resilience with limited leakage, which is in turn implied by leakage resilience with unlimited leakage.

Remark 9.5.1. *In the bounded leakage model,*[4] *that we adopt, each evaluation of $\mathcal{P}$ leaks λ bits non-adaptively. This could be λ bits of the secret state. The intuition behind this modeling is that it upper bounds the total amount of knowledge that an adversary can obtain after repeated evaluations of that permutation. Showing that this assumption is reasonable, is hard, and likely impossible as it is a rather loose bound [DMP22]. A model that does slightly*

[4]The notion "bounded" is standard terminology and refers to the fact that there *is* a fixed λ. This is subtly different from the terminology "limited" that we adopt in our analysis and that specifies whether the bound λ is smaller or equal to b.

better would be hard-to-invert leakage, which requires that the leakage has the property that, even under knowledge of the leakage, the secret state is hard to guess [DKL09, FH15], but it is a bit harder to work with. A recent approach that got closer to reality was simulatable leakage, where the adversary gets knowledge of either actual leakage or simulated leakage [SPY13], but the instantiation of this approach was demonstrated to be problematic [LMO+14]. Finally, one can opt to not bound leakage after all, and leave a yet-to-be-determined leakage term in the bound [DMP22]. This term is a function of all knowledge that is gained by the adversary, and actual side-channel analysis is needed to accurately bound this term.[5]

Also for oracle modeling, different approaches exist. Our approach consists of giving the adversary access to a leaky oracle and a leak-free challenge oracle, which it has to distinguish from random. Intuitively, this model captures the idea that, even though the adversary has obtained leakage in the past, new *evaluations are still secure. This idea somewhat aligns with the idea of nonce-misuse resilience, which is that even though the adversary has misused nonces in the past, evaluations for* new *nonces are still secure. (It is for this reason, that Definition 9.5.2 adopts the nonce restrictions from nonce-misuse resilience of Definition 9.4.4.) A notable alternative approach is of the work of Guo et al. [GPPS19a], that was also used to argue leakage resilience of TEDT [BGP+20] and the closely related work TETSponge [GPPS20, GPPS19b] (they also claim results on* Ascon-AE *in this model). This model structurally differs, at a high level, in the fact that also challenge queries leak, but the security games are not described in a real-or-random setting (as this, presumably, would require the disputable notion of simulatable leakage) but rather in the left-or-right setting where the adversary receives the encryption of either M_0 or M_1. In our impression, their model is incomparable to the model that we adopt above, and one model may better capture certain use cases than another model.*

9.5.1.2 Overview

The only analysis of Ascon-AE against leakage is by Guo et al. [GPPS20, GPPS19b]. However, their result is in a different, incomparable, model, as we explained in Remark 9.5.1. Also, their proof lacks a certain level of accuracy as pointed out in Section 9.4.3. We thus set out to derive new leakage resilience evidence for Ascon-AE, in the model of Definition 9.5.2. However, in doing to, we remark that, even though the adversary cannot reuse nonces for challenge

[5]Refer to Kalai and Reyzin [KR19] for a discussion on leakage models.

queries, it can repeatedly use nonces for non-challenge queries. Because of this, *depending on the set of permitted leakage functions* $\mathcal{L}$, Ascon-AE achieves the same level of leakage resilience under limited leakage as under unlimited leakage. Indeed, if leakage is limited to only $\lambda = 1$ bit but any leakage function is allowed, the leakage function can be defined to consider the first $\log_2(b)$ bits and use that bit string as encoding of the bit it will leak, and by making sufficient queries for the same nonce, the whole state gets eventually leaked. We admit that this is a rather liberal and non-realistic leakage function, but will still consider it for the sake of generality. We remark that specifically considering more realistic leakage functions, for example those that leak the Hamming weight of the first byte of the state, significantly adds to the complexity and would be a research problem on its own [BM24, DMMS21].

Because of this, we restrict our focus to leakage resilience under unlimited leakage, and derive nonce-misuse resilience authenticity and confidentiality in Theorem 9.5.1.

Theorem 9.5.1 (unlimited leakage)**.** *Let* $b, c, r, k, n, t \in \mathbb{N}^*$ *with* $b = r + c$, $k + n \leq b$, $t \leq k$, *and* $k \leq c$. *Let* $\mu, \mathcal{N}, \mathcal{M}_E, \mathcal{M}_D, Q_E, Q_D \in \mathbb{N}$. *Consider the* Ascon-AE *mode of Section 9.2 with parameters* b, c, r, k, n, t. *Let* $\mathcal{L}$ *be the set of all leakage functions over* Ascon-AE *that do not leak any information about the two outer permutation calls during the initialization and finalization phases. Let* $\mathcal{A}^{\mathrm{conf}}$ *be an adversary with complexity* $(\mathcal{N}, Q_E, \mathcal{M}_E)$, *and* $\mathcal{A}^{\mathrm{auth}}$ *with complexity* $(\mathcal{N}, Q_E, \mathcal{M}_E, Q_D, \mathcal{M}_D)$ *(see Section 9.3.2 for a detailed definition of complexity). We have*

$$\begin{aligned}
\mathbf{Adv}^{\mu\text{-lr-conf}}_{\text{Ascon-AE},\mathcal{L}}\left(\mathcal{A}^{\mathrm{conf}}\right) \leq{}& \frac{\mu(\mu-1)}{2^{k+1}} + \frac{2\mu\left(\mathcal{N}+\mathcal{M}\right)}{2^k} + \frac{18\mathcal{M}\left(\mathcal{M}+\mathcal{N}\right)}{2^c} \\
&+ \min\left\{\frac{14Q\left(\mathcal{N}+\mathcal{M}\right)}{2^k}, \frac{4\left(\mathcal{M}+\mathcal{N}\right)^2}{2^c} + \frac{6(\mathcal{N}+\mathcal{M})\cdot \mathrm{mucol}(\mathcal{M}+\mathcal{N}, 2^{c-k})}{2^k}\right\}, \\
\mathbf{Adv}^{\mu\text{-lr-auth}}_{\text{Ascon-AE},\mathcal{L}}\left(\mathcal{A}^{\mathrm{auth}}\right) \leq{}& \frac{\mu(\mu-1)}{2^{k+1}} + \frac{2\mu\left(\mathcal{N}+\mathcal{M}\right)}{2^k} + \frac{18\mathcal{M}\left(\mathcal{M}+\mathcal{N}\right)}{2^c} + \frac{2Q_D}{2^t} \\
&+ \min\left\{\frac{14Q\left(\mathcal{N}+\mathcal{M}\right)}{2^k}, \frac{4\left(\mathcal{M}+\mathcal{N}\right)^2}{2^c} + \frac{6(\mathcal{N}+\mathcal{M})\cdot \mathrm{mucol}(\mathcal{M}+\mathcal{N}, 2^{c-k})}{2^k}\right\}.
\end{aligned}$$

The proof of Theorem 9.5.1 is given in Section 9.6.3.

Using that $\mu \leq \mathcal{M}_E \ll \mathcal{N}$ (cf., Section 9.3.2), we obtain bounds of the order

$$\mathbf{Adv}^{\mu\text{-lr-conf}}_{\text{Ascon-AE},\mathcal{L}}\left(\mathcal{A}^{\text{conf}}\right), = \tilde{\mathcal{O}}\left((\star) + \frac{\mathcal{M}\mathcal{N}}{2^c} + \min\left\{\frac{\mathcal{N}^2}{2^c}, \frac{Q\mathcal{N}}{2^k}\right\}\right), \tag{9.6}$$

$$\mathbf{Adv}^{\mu\text{-lr-auth}}_{\text{Ascon-AE},\mathcal{L}}\left(\mathcal{A}^{\text{auth}}\right) = \tilde{\mathcal{O}}\left((\star) + \frac{\mathcal{M}\mathcal{N}}{2^c} + \min\left\{\frac{\mathcal{N}^2}{2^c}, \frac{Q\mathcal{N}}{2^k}\right\}\right). \tag{9.7}$$

The attacks from Propositions 9.4.1 and 9.4.4 targeting respectively the core term $(\star)$ and the term $\frac{\mathcal{M}_E\mathcal{N}}{2^c}$ also apply here, and Proposition 9.5.1 below matches the term $\min\left\{\frac{\mathcal{N}^2}{2^c}, \frac{Q_E\mathcal{N}}{2^k}\right\}$, under the constraint $\mathcal{N} \geq 2^{k-t}$, as in Proposition 9.4.4. Therefore, in the simplified setting of Section 9.3.2, the bounds (9.6) and (9.7) are tight for meaningful parameter sets.

Proposition 9.5.1. *Let $b, c, r, k, n, t \in \mathbb{N}^*$ with $b = r + c$, $k + n \leq b$, $t \leq k$, and $k \leq c$. Let $\mu, \mathcal{N}, \mathcal{M}_E, \mathcal{M}_D, Q_E, Q_D \in \mathbb{N}$. Consider the* Ascon-AE *mode of Section 9.2 with parameters b, c, r, k, n, t. Let $\mathcal{L}$ be the set of all leakage functions that do not leak any information about the two outer permutation calls. There exist two adversaries $\mathcal{A}^{\text{conf}}$ and $\mathcal{A}^{\text{auth}}$ with $\mathcal{N} = \max\left\{2^{c/2}, 2^k/Q_E, 2^{k-t}\right\}$ such that*

$$\mathbf{Adv}^{\mu\text{-lr-conf}}_{\text{Ascon-AE},\mathcal{L}}\left(\mathcal{A}^{\text{conf}}\right), \mathbf{Adv}^{\mu\text{-lr-auth}}_{\text{Ascon-AE},\mathcal{L}}\left(\mathcal{A}^{\text{auth}}\right) \approx 1.$$

Here, the adversaries $\mathcal{A}^{\text{conf}}$ and $\mathcal{A}^{\text{auth}}$ make $\min\left\{2^{k-c/2}, Q_E\right\}$ encryption queries.

The proof of Proposition 9.5.1 is given in Section 9.7.4.

9.5.2 State-Recovery Security

We define state-recovery security in Section 9.5.2.1. In Section 9.5.2.2 we consider a variant of the Ascon-AE mode without the key blindings, named BadAsscon-AE, and demonstrate that this construction fails to achieve authenticity under state recovery. Finally, Section 9.5.2.3 derives lower and upper bounds for state-recovery security of Ascon-AE.

9.5.2.1 Security Model

One property claimed by the designers of Ascon is that, if an inner state of Ascon is leaked, mounting forgeries or recovering the keys is hard. Intuitively, the idea is that whenever an inner state is leaked to the adversary $\mathcal{A}$, it may evaluate itself in forward/inverse direction using evaluations of $\mathcal{P}^{\pm}$, but $\mathcal{A}$

cannot get across the outer permutation evaluations due to the key blindings. After all, the secret target values outside these permutation evaluations are the key and the tag; hence the claim of the designers.

To formalize this security property, we take inspiration from the work on permutation-based leakage resilience authenticity [DM19a, DM19b, GPPS19b, GPPS20] (basically, authenticity of Definition 9.5.2), but stronger in the sense that we (i) consider unlimited leakage by default and (ii) do not put any restrictions on nonce-misuse. Point (i) is mostly out of generosity: the attacker learns *all* internal secret states, rather than just a selected amount of them. This is without loss of generality as the adversary can proceed itself from any leaked secret state using $\mathcal{P}^{\pm}$ anyway. As a consequence of adjustment (ii), the security game does not have to distinguish between leaky and non-leaky oracles, and the non-leaky ones can be dropped. We formally define state-recovery authenticity, below. For completeness, we also include the logical state-recovery confidentiality variant as a direct generalization of Definition 9.5.2 (but with the decryption oracle dropped as encryption and decryption leakages give the same information in this case), which admittedly is moot as Ascon-AE is insecure in this model.

Definition 9.5.3. *Consider the* Ascon-AE *mode of Section 9.2. Let* $(\$_m)_{m=1}^{\mu}$ *be a family of* μ *independent random functions,* $\mathcal{P} \xleftarrow{\$} \mathtt{Perm}(b)$*, and* $K_1, \ldots, K_\mu \xleftarrow{\$} \{0,1\}^k$*. Let* L *be the leakage function that leaks all inner permutation calls, excluding the ones during the initialization and finalization phase.*

- *The state-recovery confidentiality of* Ascon-AE *against an adversary* $\mathcal{A}$ *is defined as*

$$\mathbf{Adv}^{\mu\text{-sr-conf}}_{\text{Ascon-AE}}(\mathcal{A}) = \Delta_{\mathcal{A}}\left(\left(\left[\mathbf{Enc}^{\mathcal{P}}_{K_m}\right]_L, \mathbf{Enc}^{\mathcal{P}}_{K_m}\right)_{m=1}^{\mu}, \mathcal{P}^{\pm}\ ;\ \left(\left[\mathbf{Enc}^{\mathcal{P}}_{K_m}\right]_L, \$_m\right)_{m=1}^{\mu}, \mathcal{P}^{\pm}\right),$$

  where $\mathcal{A}$ *is restricted as follows:* $\mathcal{O}_{2,m} \overset{*}{\not\hookrightarrow} \mathcal{O}_{1,m}$*, and* $\mathcal{O}_{1,m} \overset{*}{\not\hookrightarrow} \mathcal{O}_{2,m}$*;*

- *The state-recovery authenticity of* Ascon-AE *against an adversary* $\mathcal{A}$ *is defined as*

$$\mathbf{Adv}^{\mu\text{-sr-auth}}_{\text{Ascon-AE}}(\mathcal{A}) = \mathbf{Pr}\left(\mathcal{A}\left[\left(\left[\mathbf{Enc}^{\mathcal{P}}_{K_m}\right]_L, \left[\mathbf{Dec}^{\mathcal{P}}_{K_m}\right]_L\right)_{m=1}^{\mu}, \mathcal{P}^{\pm}\right] \textit{ forges}\right),$$

  where $\mathcal{A}$ *is restricted as follows:* $\mathcal{O}_{1,m} \overset{*}{\not\hookrightarrow} \mathcal{O}_{2,m}$*. Here, "forges" denotes the event that* $\mathcal{A}$ *makes a query to one of the oracles* $\mathbf{Dec}^{\mathcal{P}}_{K_m}$ *that does not return* $\perp$*.*

To see that state-recovery authenticity implies leakage resilience authenticity with unlimited leakage, observe that any state-recovery adversary $\mathcal{A}'$ can easily simulate the oracles of a leakage resilience adversary $\mathcal{A}$ because $\mathcal{A}'$ is free from any nonce repetition restrictions. A similar observation applies to state-recovery confidentiality compared to leakage resilience confidentiality, with the reminder that state-recovery confidentiality is not achieved for Ascon-AE as we demonstrate in Proposition 9.5.2 below.

Proposition 9.5.2. *Let $b, c, r, k, n, t \in \mathbb{N}^*$ with $b = r + c$, $k + n \leq b$, $t \leq k$, and $k \leq c$. Consider the* Ascon-AE *mode of Section 9.2 with parameters b, c, r, k, n, t. There exists an adversary $\mathcal{A}$ with $Q_E = 2$, such that*

$$\mathbf{Adv}^{\mu\text{-sr-conf}}_{\text{Ascon-AE}}(\mathcal{A}) \approx 1\,.$$

Proof. As in the nonce-misuse confidentiality setting (i.e., Proposition 9.4.2), the adversary is allowed to repeat nonces between the challenge encryption oracle and the non-challenge encryption oracle, and this breaks security. Let $P_1, P_2 \in \{0,1\}^r$, $N \in \{0,1\}^n$, $m \in [\![1, \mu]\!]$. Consider the following attack:

1. Make an encryption query to oracle $\mathcal{O}_{1,m}$ with input (N, ϵ, P_1), and obtain the ciphertext, denoted by $C \in \{0,1\}^r$;
2. Make an encryption query to oracle $\mathcal{O}_{2,m}$ with input $(N, \epsilon, P_1 \| P_2)$, denote the leftmost r bits of the ciphertext by $C' \in \{0,1\}^r$;
3. If $C = C'$ return 0, else 1.

In the real world, C will always be equal to C' while in the ideal world, this happens with probability $\frac{1}{2^r}$. This term can be reduced further by repeating the attack or by mounting the attack for longer encryption queries. □

9.5.2.2 BadAsscon-AE

Before proceeding, we first consider BadAsscon-AE. This mode is equal to the mode of Figure 9.2 but *omits all key additions except the first one.* We demonstrate that BadAsscon-AE admits a trivial authenticity attack under state recovery.

Proposition 9.5.3. *Consider the mode* BadAsscon-AE *that equals* Ascon-AE *of Section 9.2 but with all key additions except the first one omitted, with parameters b, c, r, k, n, t. There exists an adversary $\mathcal{A}$ with $Q_E = 1$, $Q_D = 1$, $\mathcal{M} = 4$, $\mathcal{N} = 3$, such that*

$$\mathbf{Adv}^{\mu\text{-sr-auth}}_{\text{BadAsscon-AE}}(\mathcal{A}) = 1\,.$$

The idea of the attack is very simple: from a single leaky query one can recover the key, and once the key is recovered a forgery can be mounted. We include the attack for completeness.

Proof of Proposition 9.5.3. Let $P, P' \in \{0,1\}^{r-1}$, $N, N' \in \{0,1\}^n$, with $(N, P) \neq (N', P')$ and $m \in [\![1, \mu]\!]$. We consider an adversary that first recovers the key K and uses it to forge a tag under this key. Consider the following attack:

1. Make an encryption query with empty associated data and with a plaintext P: $\left[\mathbf{Enc}^{\mathcal{P}}_{K_m}\right]_L(N, \epsilon, P)$. It learns (C, T) as well as the state S right after absorption of P and before application of permutation $\mathcal{P}$;
2. Make a permutation query $\mathcal{P}^{-1}(S \oplus (P\|1\|0^{c-1}\|1))$ to obtain $IV\|K\|N$ and extract K from it;
3. Compute $\mathbf{Enc}^{\mathcal{P}}_K(N', P') = (C', T')$ offline with two calls to $\mathcal{P}$;
4. Output forgery (N', C', T').

The forgery obviously succeeds with probability 1. □

Remark 9.5.2. *The same attack applies to the setting of leakage resilience authenticity with unlimited leakage. Indeed, when the adversary learns the full internal state, the absence of key blindings leads directly to key recovery.*

9.5.2.3 Security Bounds

We prove a tight bound of security under state recovery in Theorem 9.5.2. The core idea why Ascon-AE achieves authenticity under state recovery (and, more generally, authenticity under leakage) whereas BadAsscon-AE of Section 9.5.2.2 did not is that the calls to the outer permutation are masked by the key at both sides. This means that, even if the adversary learns all intermediate states, it cannot "pass through" the permutations. That said, the security setting also gives rise to other potential attacks, most notably as in case of state leakage the inner portion of Ascon-AE behaves as a mere hash function of which the adversary knows all states, and for which it could potentially find inner collisions. This complicates the analysis of Theorem 9.5.2.

Theorem 9.5.2. *Let $b, c, r, k, n, t \in \mathbb{N}^*$ with $b = r + c$, $k + n \leq b$, $t \leq k$, and $k \leq c$. Let $\mu, \mathcal{N}, \mathcal{M}_E, \mathcal{M}_D, Q_E, Q_D \in \mathbb{N}$. Consider the* Ascon-AE *mode* *of Section 9.2 with parameters b, c, r, k, n, t. Let $\mathcal{A}$ be an adversary with complexity $(\mathcal{N}, Q_E, \mathcal{M}_E, Q_D, \mathcal{M}_D)$ (see Section 9.3.2 for a detailed definition of complexity). We have*

$$\mathbf{Adv}^{\mu\text{-sr-auth}}_{\text{Ascon-AE}}(\mathcal{A}) \leq \frac{\mu(\mu-1)}{2^{k+1}} + \frac{2\mu(\mathcal{N}+\mathcal{M})}{2^k} + \frac{2Q_D}{2^t} + \frac{18\mathcal{M}(\mathcal{M}+\mathcal{N})}{2^c} + \frac{4(\mathcal{M}+\mathcal{N})^2}{2^c} + \frac{6(\mathcal{N}+\mathcal{M})\cdot \text{mucol}(\mathcal{N}+\mathcal{M}, 2^{c-k})}{2^k}\,.$$

The proof of Theorem 9.5.2 is given in Section 9.6.4.

Using that $\mu \leq \mathcal{M}_E \ll \mathcal{N}$ (cf., Section 9.3.2), we obtain a bound of the order

$$\mathbf{Adv}^{\mu\text{-sr-auth}}_{\text{Ascon-AE}}\left(\mathcal{A}^{\text{auth}}\right) = \tilde{\mathcal{O}}\left((\star) + \frac{\mathcal{N}^2}{2^c}\right). \tag{9.8}$$

The attacks from Proposition 9.4.1 targeting the core term $(\star)$ also apply here, and Proposition 9.5.4 below matches the term $\frac{\mathcal{N}^2}{2^c}$. Therefore, the bound (9.8) is tight.

Proposition 9.5.4. *Let $b, c, r, k, n, t \in \mathbb{N}^*$ with $b = r + c$, $k + n \leq b$, $t \leq k$, and $k \leq c$. Consider the* Ascon-AE *mode of Section 9.2 with parameters b, c, r, k, n, t. There exists an adversary $\mathcal{A}$ with $\mathcal{N} \approx 2^{c/2}$ such that*

$$\mathbf{Adv}^{\mu\text{-sr-auth}}_{\text{Ascon-AE}}(\mathcal{A}) \approx 1\,.$$

The proof of Proposition 9.5.4 is given in Section 9.7.5.

9.5.3 Release of Unverified Plaintext Security

9.5.3.1 Security Model

Another weakness in typical use cases of authenticated encryption is in applications that (accidentally) release plaintext before the tag is verified. This may happen, for example, in use cases where there is insufficient secure memory to store the message or mistakes/incompletenesses in implementation occur (cf., Efail [PDM+18]). Andreeva et al. [ABL+14] formalized the idea of security under release of unverified plaintext (RUP). In this formalization, the authenticated decryption functionality **Dec** is separated into a pure decryption functionality **D** that outputs the plaintext (without verification) and a verification functionality **V** that verifies the authentication:

$$\begin{aligned}
\mathbf{D}_K^{\mathcal{P}} : \{0,1\}^n \times \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^t &\longrightarrow \{0,1\}^* \,, \\
(N, A, C, T) &\longrightarrow P \in \{0,1\}^{|C|} \,, \\
\mathbf{V}_K^{\mathcal{P}} : \{0,1\}^n \times \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^t &\longrightarrow \{\top, \bot\} \,, \\
(N, A, C, T) &\longrightarrow \top \text{ or } \bot \,.
\end{aligned}$$

RUP confidentiality is covered by *plaintext awareness*, that considers a distinguisher that has access to the encryption functionality, and either the (unverified) decryption or an extractor **Ext** that has knowledge of earlier encryption queries and aims to simulate the **D** functionality.[6] Authenticity is covered by an adversary that gets access to encryption, (unverified) decryption, and the verification function, and wins if it forges.

Definition 9.5.4. *Consider the* Ascon-AE *mode of Section 9.2. Let* $(\$_m)_{m=1}^{\mu}$ *be a family of* μ *independent random functions,* $\mathcal{P} \xleftarrow{\$} \texttt{Perm}(b)$*, and* $K_1, \ldots, K_\mu \xleftarrow{\$} \{0,1\}^k$*.*

- *Let* $\mathbf{Ext} = (\mathbf{Ext}_m)_{m=1}^{\mu}$ *be a family of stateful algorithms. The RUP confidentiality (or, plaintext awareness) of* Ascon-AE *against an adversary* $\mathcal{A}$ *with respect to* **Ext** *is defined as*

$$\mathbf{Adv}_{\text{Ascon-AE},\mathbf{Ext}}^{\mu\text{-rup-conf}}(\mathcal{A}) = \\ \Delta_{\mathcal{A}}\left(\left(\mathbf{Enc}_{K_m}^{\mathcal{P}}, \mathbf{D}_{K_m}^{\mathcal{P}}\right)_{m=1}^{\mu}, \mathcal{P}^{\pm} \,;\, \left(\mathbf{Enc}_{K_m}^{\mathcal{P}}, \mathbf{Ext}_m\right)_{m=1}^{\mu}, \mathcal{P}^{\pm}\right),$$

where $\mathbf{Ext}_m$ *has access to the query history made by* $\mathcal{A}$ *to* $\mathcal{O}_{1,m}$*;*

- *The RUP authenticity of* Ascon-AE *against an adversary* $\mathcal{A}$ *is defined as*

$$\mathbf{Adv}_{\text{Ascon-AE}}^{\mu\text{-rup-auth}}(\mathcal{A}) = \mathbf{Pr}\left(\mathcal{A}\left[\left(\mathbf{Enc}_{K_m}^{\mathcal{P}}, \mathbf{D}_{K_m}^{\mathcal{P}}, \mathbf{V}_{K_m}^{\mathcal{P}}\right)_{m=1}^{\mu}, \mathcal{P}^{\pm}\right] \text{ forges}\right),$$

where $\mathcal{A}$ *is restricted as follows:* $\mathcal{O}_{1,m} \overset{*}{\not\to} \mathcal{O}_{3,m}$*. Here, "forges" denotes the event that* $\mathcal{A}$ *makes a query to one of the oracles* $\mathbf{V}_{K_m}^{\mathcal{P}}$ *that does not return* $\bot$*.*

[6]This is called plaintext awareness 1 (PA1). Andreeva et al. [ABL+14] described the stronger setting of PA2 where **Ext** has no knowledge of earlier encryption queries, but a scheme that is not PA1 secure is also not PA2 secure.

In the resources $(\mathcal{N}, Q_E, \mathcal{M}_E, Q_D, \mathcal{M}_D)$ of Section 9.3.2, the terms Q_D and $\mathcal{M}_D$ now additionally account for the queries made to the verification oracle $\mathbf{V}$. We remark that there have been follow-up works of Ashur et al. [ADL17] and Chang et al. [CDD$^+$19] who presented RUPAE and AERUP, respectively, with the aim to unify RUP security into one definition. In our work, however, we restrict to considering separate confidentiality and authenticity notions.

We remark that RUP authenticity is implied by state-recovery authenticity (cf., Definition 9.5.3). This implication, however, is not immediately clear, so we write it out in detail. (In fact, this applies to any authenticated encryption scheme, but we write it out for Ascon-AE as this is the scope of the work.)

Lemma 9.5.3. *Consider the* Ascon-AE *mode of Section 9.2. Let* $\mathcal{P} \xleftarrow{\$} \texttt{Perm}(b)$ *and* $K_1, \ldots, K_\mu \xleftarrow{\$} \{0,1\}^k$. *Let* $\mathcal{A}$ *be a RUP authenticity adversary with complexity* $(\mathcal{N}, Q_E, \mathcal{M}_E, Q_D, \mathcal{M}_D)$. *There exists a state-recovery authenticity adversary* $\mathcal{A}'$ *with complexity* $(\mathcal{N}, Q_E, \mathcal{M}_E, Q_D, \mathcal{M}_D)$, *such that*

$$\mathbf{Adv}^{\mu\text{-rup-auth}}_{\text{Ascon-AE}}(\mathcal{A}) \leq \mathbf{Adv}^{\mu\text{-sr-auth}}_{\text{Ascon-AE}}(\mathcal{A}') \,.$$

Proof. Consider RUP authenticity adversary $\mathcal{A}$ that gets access to the following oracles:

$$\left(\left(\mathbf{Enc}^{\mathcal{P}}_{K_m}, \mathbf{D}^{\mathcal{P}}_{K_m}, \mathbf{V}^{\mathcal{P}}_{K_m}\right)_{m=1}^{\mu}, \mathcal{P}^{\pm}\right) .$$

It is restricted to $\mathcal{O}_{1,m} \overset{*}{\not\to} \mathcal{O}_{3,m}$. We construct state-recovery adversary $\mathcal{A}'$ that gets access to the following oracles:

$$\left(\left(\left[\mathbf{Enc}^{\mathcal{P}}_{K_m}\right]_L, \left[\mathbf{Dec}^{\mathcal{P}}_{K_m}\right]_L\right)_{m=1}^{\mu}, \mathcal{P}^{\pm}\right) ,$$

that is restricted to $\mathcal{O}_{1,m} \overset{*}{\not\to} \mathcal{O}_{2,m}$, and that will use its oracles to simulate the oracles of $\mathcal{A}$. Note that $\mathcal{A}$ may repeat certain queries whereas $\mathcal{A}'$ may not. To solve this, $\mathcal{A}'$ will maintain a database of its queries to the previous oracles. If $\mathcal{A}'$ is about to repeat a query, it will instead retrieve the result from the database. For the rest, adversary $\mathcal{A}'$ operates as follows:

- **$\boldsymbol{\mathcal{A}}$ makes an oracle query $\mathbf{Enc}^{\boldsymbol{\mathcal{P}}}_{\boldsymbol{K_m}}(\boldsymbol{N}, \boldsymbol{A}, \boldsymbol{P})$:** Adversary $\mathcal{A}'$ relays the query to $\left[\mathbf{Enc}^{\mathcal{P}}_{K_m}\right]_L$ to obtain (C, T) and all intermediate states. It discards the intermediate states and relays (C, T) to $\mathcal{A}$. It stores (N, A, P, C, T) in a database;
- **$\boldsymbol{\mathcal{A}}$ makes an oracle query $\mathbf{D}^{\boldsymbol{\mathcal{P}}}_{\boldsymbol{K_m}}(\boldsymbol{N}, \boldsymbol{A}, \boldsymbol{C}, \boldsymbol{T})$:**

- If (N, A, C, T) corresponds to an earlier encryption query (note that $\mathcal{A}$ may relay from its first to second oracle but $\mathcal{A}'$ may not), there must be a unique P such that (N, A, P, C, T) in $\mathcal{A}'$s database. $\mathcal{A}'$ replies with that plaintext P;
- If (N, A, C, T) does not correspond to an earlier encryption query, $\mathcal{A}'$ queries $\left[\mathbf{Dec}^{\mathcal{P}}_{K_m}\right]_L$ on the same inputs, it reconstructs P from the state leakages, and replies with P;

- **$\mathcal{A}$ makes an oracle query $\mathbf{V}^{\mathcal{P}}_{K_m}(N, A, C, T)$:** Adversary $\mathcal{A}'$ queries $\left[\mathbf{Dec}^{\mathcal{P}}_{K_m}\right]_L$ on the same inputs. If the response is a valid plaintext, $\mathcal{A}'$ replies with $\top$, otherwise it replies with $\perp$;
- **$\mathcal{A}$ makes an oracle query $\mathcal{P}^{\pm}$:** Adversary $\mathcal{A}'$ simply relays the query to its own permutation oracle $\mathcal{P}^{\pm}$ and relays the response back.

If $\mathcal{A}$ mounts a forgery in any of its queries to $\mathbf{V}^{\mathcal{P}}_{K_m}$, then $\mathcal{A}'$ also mounts a valid forgery. This proves the claim. $\square$

It does not seem possible to reduce RUP confidentiality to state-recovery confidentiality. There is some similarity, though: RUP confidentiality gives the adversary access to an encryption oracle and a decryption oracle, whereas the state-recovery adversary gets access to a leaky encryption oracle (which it can use to simulate the RUP encryption oracle) and a challenge encryption oracle (which it can use to simulate the RUP decryption oracle, noting that it can reuse nonces). However, the RUP adversary is allowed to freely relay queries whereas the state-recovery adversary is not, and in fact upcoming RUP confidentiality attack of Proposition 9.5.5 exploits this, and it cannot be turned into a state-recovery confidentiality attack.

9

9.5.3.2 Overview

In their formalism of release of unverified plaintext, Andreeva et al. [ABL+14] also demonstrated that nonce-based length-preserving (i.e., $|C| = |P|$) authenticated encryption schemes cannot achieve PA1 security. We repeat their result in the context of Ascon-AE.

Proposition 9.5.5. *Let $b, c, r, k, n, t \in \mathbb{N}^*$ with $b = r + c$, $k + n \leq b$, $t \leq k$, and $k \leq c$. Consider the* Ascon-AE *mode of Section 9.2 with parameters b, c, r, k, n, t. There exists an adversary $\mathcal{A}$ with $Q_E = 1$ and $Q_D = 1$, such that*

$$\mathbf{Adv}^{\mu\text{-rup-conf}}_{\text{Ascon-AE}}(\mathcal{A}) \approx 1\,.$$

Proof. Let $C \in \{0,1\}^r$, $N \in \{0,1\}^n$, and $m \in [\![1,\mu]\!]$. Consider the following attack:

1. Make a decryption query with user m with input (N, ϵ, C), denote the plaintext by $P \in \{0,1\}^r$;
2. Make an encryption query with user m with input (N, ϵ, P), denote the ciphertext by $C' \in \{0,1\}^r$;
3. If $C = C'$ return 0, else 1.

In the real world, these are identical evaluations of Ascon-AE and $C = C'$, whereas in the ideal world, this only holds if $\mathbf{Ext}_m$ output the right plaintext P and this happens with probability $\frac{1}{2^r}$. This term can be reduced further by repeating the attack or by mounting the attack for a longer decryption query. $\square$

That said, Ascon-AE achieves authenticity under release of unverified plaintext. This already follows from Theorem 9.5.2 and Lemma 9.5.3, but this bound is not tight and we derive a better bound below.

Theorem 9.5.4. *Let $b, c, r, k, n, t \in \mathbb{N}^*$ with $b = r + c$, $k + n \leq b$, $t \leq k$, and $k \leq c$. Let $\mu, \mathcal{N}, \mathcal{M}_E, \mathcal{M}_D, Q_E, Q_D \in \mathbb{N}$. Consider the* Ascon-AE *mode of Section 9.2 with parameters b, c, r, k, n, t. Let $\mathcal{A}$ be an adversary with complexity $(\mathcal{N}, Q_E, \mathcal{M}_E, Q_D, \mathcal{M}_D)$ (see Section 9.3.2 for a detailed definition of complexity). We have*

$$\mathbf{Adv}^{\mu\text{-rup-auth}}_{\text{Ascon-AE}}(\mathcal{A}) \leq \frac{\mu(\mu-1)}{2^{k+1}} + \frac{2\mu(\mathcal{M}+\mathcal{N})}{2^k} + \frac{18\mathcal{M}(\mathcal{M}+\mathcal{N})}{2^c} + \frac{2Q_D}{2^t}\,.$$

The proof of Theorem 9.5.4 is given in Section 9.6.5.

Using that $\mu \leq \mathcal{M}_E \ll \mathcal{N}$ (cf., Section 9.3.2), we obtain a bound of the order

$$\mathbf{Adv}^{\mu\text{-rup-auth}}_{\text{Ascon-AE}}(\mathcal{A}) = \tilde{\mathcal{O}}\left((\star) + \frac{\mathcal{M}\mathcal{N}}{2^c}\right). \tag{9.9}$$

In particular, RUP authenticity has a bound of the same order as that of nonce-misuse resistance authenticity.

The adversary is more powerful in the RUP setting than in the nonce-misuse resistance setting: the attack described in Proposition 9.4.3 applies, where encryption queries can be substituted for decryption queries. Therefore, the bound (9.9) is tight, as we make explicit in Proposition 9.5.6.

Proposition 9.5.6. *Let $b, c, r, k, n, t \in \mathbb{N}^*$ with $b = r + c$, $k + n \leq b$, $t \leq k$, and $k \leq c$. Consider the* Ascon-AE *mode of Section 9.2 with parameters b, c, r, k, n, t. There exists an adversary $\mathcal{A}$ with $\mathcal{M}\mathcal{N} \approx 2^c$, such that*

$$\mathbf{Adv}^{\mu\text{-rup-auth}}_{\text{Ascon-AE}}(\mathcal{A}) \approx 1 \,.$$

9.6 Security Proofs for Ascon-AE

We include the security proofs of Theorems 9.4.2–9.5.4 here, in Sections 9.6.1–9.6.5, respectively. The first proof, that of Section 9.6.1, is worked out in full detail as it lays the foundation for the subsequent proofs.

9.6.1 Proof of Theorem 9.4.2

Let $\mathcal{A}$ be a nonce-misuse adversary that makes at most $\mathcal{N}$ permutation queries, Q_E encryption queries of at most $\mathcal{M}_E$ blocks, and Q_D decryption queries of at most $\mathcal{M}_D$ blocks, as in the theorem statement. Our goal is to upper bound $\mathbf{Adv}^{\mu\text{-m-auth}}_{\text{Ascon}}(\mathcal{A})$. We will adopt a distinguishing game approach, i.e., consider the distinguishing game version of authenticity, where the challenge decryption oracle is replaced by $\perp$. Therefore, the adversary interacts either with the *real world* W_R, which gives access to $\left[\left(\mathbf{Enc}^{\mathcal{P}}_{K_m}, \mathbf{Dec}^{\mathcal{P}}_{K_m}\right)^{\mu}_{m=1}, \mathcal{P}^{\pm}\right]$, or with the *ideal world* W_I, which gives access to $\left[\left(\mathbf{Enc}^{\mathcal{P}}_{K_m}, \perp\right)^{\mu}_{m=1}, \mathcal{P}^{\pm}\right]$. Without loss of generality, we assume in all subsequent proofs that the associated data and plaintext provided as input to the construction oracles are already padded, and if not, the construction oracles return $\perp$.

Transcript Notation. We define notation for the transcript that can be obtained from the adversarial interaction with the different oracles. The transcript, named τ, is an ordered list of tuples. Each tuple registers a query made to an oracle, and its structure depends on the type of query:

- A forward (resp., inverse) permutation query with input X and output Y generates the transcript element (X, Y, fwd) (resp., (X, Y, inv));
- An encryption query with user m, input (N, A, P), and output (C, T) generates the transcript element (E, m, N, A, P, C, T);
- A decryption query with user m, input (N, A, C, T), and output $\tilde{P}$ generates the transcript element $(D, m, N, A, C, T, \tilde{P})$.

Paths Notation. In both worlds, the encryption queries generate intermediate states through permutation evaluations; in the real world, decryption queries also produce intermediate states. In order to label these states properly, let us define some notation. Any element $(O, m, N, A, B, -, -) \in \tau$ is associated to a *path*, denoted by $\texttt{path} = (O, m, N, A, B, 1)$. Here B is equal to the plaintext blocks if $O = E$, or the ciphertext blocks if $O = D$. The last element is a bit; it equals 1 if the path is *final*, so that the final key blinding has been applied to the last permutation call. From $\texttt{path}$, we define inductively subpaths. We say that $\texttt{path}' = (O', m', N', A', B', f')$ is a *parent* of $\texttt{path} = (O, m, N, A, B, f)$ whenever $(O', m', N') = (O, m, N)$, $f' = 0$, and

- either $A' = A$, $|B'| = |B| - 1$, and $B' = B\,[1 : |B'|]$,
- or $B' = B = \epsilon$, $|A'| = |A| - 1$, and $A' = A\,[1 : |A'|]$.

The only paths without parents are of the form $(O, m, N, \epsilon, \epsilon, 0)$, while all other paths have a unique parent. Note that either $\texttt{path}$ and $\texttt{path}'$ are both encryption paths (i.e., their first element is E), or both decryption paths (i.e., their first element is D). For convenience, given a construction query, the set defined by the path $\texttt{path}$ associated with this query, along with all ancestors of $\texttt{path}$, is referred to as the set of paths generated by that query. Denote by $\texttt{P}$ the set of all paths which are generated by all queries. According to the inductive definition above, the number of encryption paths is equal to $\mathcal{M}_E$, the number of decryption paths is equal to $\mathcal{M}_D$, and $|\texttt{P}| = \mathcal{M}_E + \mathcal{M}_D$. To illustrate this, consider the example encryption query made in Figure 9.3, with input $(m, N, (A_1, \dots, A_u), (P_1, \dots, P_v))$. The final path associated to this encryption query is called $\texttt{path}_4$, which is a child of $\texttt{path}_3$, which is itself a grand$^{\mathrm{X}}$-child of $\texttt{path}_2$ for some $X \geq 0$, which is itself a grand$^{\mathrm{X'}}$-child of $\texttt{path}_1$ for some $X' \geq 0$, which has itself no parent.

Intermediate States. We can now effectively label the intermediate states generated by permutation evaluations from encryption queries, and in the real world additionally from decryption queries. We define two dictionaries $\texttt{S}_{\text{in}}$ and $\texttt{S}_{\text{out}}$, with labels either in $\texttt{P}$ in the real world, or in the set of encryption paths in the ideal world. Given $\texttt{path} = (O, m, N, A, B, f)$, $\texttt{S}_{\text{in}}[\texttt{path}]$ (resp., $\texttt{S}_{\text{out}}[\texttt{path}]$) corresponds to the input (resp., output) of the permutation evaluation made when doing a construction query with user m, nonce N, after having absorbed the associated data blocks A, and having processed the blocks B (to clarify, if $\texttt{path}$ is an encryption path, those blocks are considered as being added to the outer part of the state, while if $\texttt{path}$ is a decryption path, the blocks overwrite the outer parts of the state). In particular, if $\texttt{path} = (O, m, N, A, P, 1)$, then

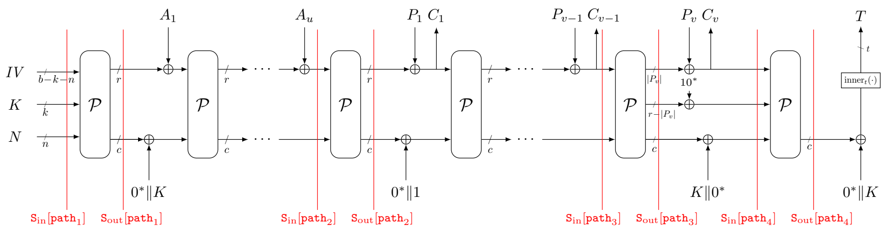


Figure 9.3: Illustration of intermediate states in the real world of an encryption query. Assuming that the key is the one of user number m, in our example we have $\mathtt{path}_1 = (E, m, N, \epsilon, \epsilon, 0)$, $\mathtt{path}_2 = (E, m, N, (A_1, \ldots, A_u), \epsilon, 0)$, $\mathtt{path}_3 = (E, m, N, (A_1, \ldots, A_u), (P_1, \ldots, P_{v-1}), 0)$, and $\mathtt{path}_4 = (E, m, N, (A_1, \ldots, A_u), (P_1, \ldots, P_v), 1)$.

$\mathtt{S}_{\text{in}}[\mathtt{path}]$ must include the key addition, and if $\mathtt{path} = (O, m, N, \epsilon, \epsilon, 0)$, then $\mathtt{S}_{\text{in}}[\mathtt{path}] = IV \| K_m \| N$. Figure 9.3 illustrates $\mathtt{S}_{\text{in}}[\mathtt{path}]$ and $\mathtt{S}_{\text{out}}[\mathtt{path}]$ given our example encryption query.

We will specify a procedure to generate mock intermediate states in the ideal world, but before that we need to introduce further notation to pinpoint the existing relationships between two intermediate states in the real world. First, we say that a decryption path $\mathtt{path} = (D, m, N, A, C, f)$ is *superseded* by an encryption path $\mathtt{path}' = (E, m, N, A, P, f)$ associated to an encryption query $(E, m, N, A', P', C', T) \in \tau$ if $|P| = |C|$ and $C = C'[1 : |C|]$. Intuitively, this means that $\mathtt{path}$ gets the exact same user, nonce, and ciphertexts as an encryption path $\mathtt{path}'$, so that their intermediate states $\mathtt{S}_{\text{in}}[\mathtt{path}]$ and $\mathtt{S}_{\text{in}}[\mathtt{path}']$ (resp., $\mathtt{S}_{\text{out}}[\mathtt{path}]$ and $\mathtt{S}_{\text{out}}[\mathtt{path}']$) must be the same. Let $\mathtt{P}_{\mathtt{S}}$ be the set of decryption paths that are superseded.

Moreover, given two paths $\mathtt{pathP}$ and $\mathtt{pathC}$, $\mathtt{S}_{\text{out}}[\mathtt{pathP}]$ and $\mathtt{S}_{\text{in}}[\mathtt{pathC}]$ must be related whenever $\mathtt{pathP}$ is a parent of $\mathtt{pathC}$, through the addition of constants, key material, and data blocks. We define the function $\mathrm{XorState}(\mathtt{pathP}, \mathtt{pathC}) \in \{0,1\}^b$ for any parent-child pair $(\mathtt{pathP}, \mathtt{pathC}) \in \mathtt{P}^2$, which handles the inner parts as follows:

$$\begin{aligned}&\mathrm{XorState}\left((O, m, N, A, B, f), (O, m, N, A', B', f')\right) = \\ &\qquad \begin{Bmatrix} 0^{b-k} \| K_m & \text{if } A = B = \epsilon \\ 0^b & \text{otherwise} \end{Bmatrix} \oplus \begin{Bmatrix} 0^{b-1}1 & \text{if } A = A', B = \epsilon \\ 0^b & \text{otherwise} \end{Bmatrix} \\ &\qquad\qquad\qquad\qquad\qquad\qquad \oplus \begin{Bmatrix} 0^r \| K_m \| 0^{c-k} & \text{if } f' = 1 \\ 0^b & \text{otherwise} \end{Bmatrix} .\end{aligned}$$

Therefore, we have

$$\mathtt{S}_{\text{in}}[\mathtt{pathC}] \overset{c}{=} \mathtt{S}_{\text{out}}[\mathtt{pathP}] \oplus \text{XorState}\,(\mathtt{pathP}, \mathtt{pathC}) \ .$$

Before moving on, we need one last piece of notation. Let

$$\mathtt{path} = (O, m, N, A, P, f) \in \mathtt{P} \,.$$

We define the set ValidXor ($\mathtt{path}$) of b-bit elements. This set captures all the possible values for the inner part of $\mathtt{S}_{\text{in}}[\mathtt{path}']$, for all potential child / superseded paths $\mathtt{path}'$ of $\mathtt{path}$. It is defined as:

$$\text{ValidXor}\,(\mathtt{path}) = \begin{cases} \{0^b\} & \text{if } f = 1\,, \\ \{0^*\|K_m, (0^*\|K_m) \oplus (0^*\|1)\,, & \\ \ (0^*\|K_m) \oplus (0^*\|1) \oplus \left(0^r\|K_m\|0^{c-k}\right)\} & \text{if } A = P = \epsilon\,, \\ \{0^b, 0^*\|1, (0^*\|1) \oplus \left(0^r\|K_m\|0^{c-k}\right)\} & \text{if } A \neq \epsilon, P = \epsilon\,, \\ \{0^b, \left(0^r\|K_m\|0^{c-k}\right)\} & \text{if } P \neq \epsilon\,. \end{cases}$$

Looking ahead, ValidXor ($\mathtt{path}$) will be useful for the probability computation of the bad events in a query-wise fashion. An important remark for this upcoming bad event analysis is that $|\text{ValidXor}\,(\mathtt{path})| \leq 3$ always holds.

Mock Intermediate States. Our approach will be to establish an extended transcript that releases the intermediate states associated to all construction queries. Therefore, we define a procedure to generate mock intermediate states in the ideal world, in other words define $\mathtt{S}_{\text{in}}[\mathtt{path}]$ and $\mathtt{S}_{\text{out}}[\mathtt{path}]$ for any decryption path $\mathtt{path} \in \mathtt{P}$. Those states mimic the structure of Ascon-AE by using the intermediate states generated by the prior encryption queries, the user keys $(K_m)_{m=1}^{\mu}$, as well as some fresh randomness. The sampling procedure, taking place at the end of the interaction, operates as follows:

- Sample $\mathtt{S}_{\text{out}}[\mathtt{path}]$, for all $\mathtt{path} = (D, m, N, A, C, f)$ as follows:
  - If $\mathtt{path}$ is superseded by an encryption path $\mathtt{path}'$, then $\mathtt{S}_{\text{out}}[\mathtt{path}] \leftarrow \mathtt{S}_{\text{out}}[\mathtt{path}']$;
  - Else, $\mathtt{S}_{\text{out}}[\mathtt{path}] \overset{\$}{\leftarrow} \{0,1\}^b$;
- Then, sample $\mathtt{S}_{\text{in}}[\mathtt{path}]$, for all $\mathtt{path} = (D, m, N, A, C, f)$ as follows:
  - If $P = A = \epsilon$, then necessarily $f = 0$ and $\mathtt{S}_{\text{in}}[(D, m, N, \epsilon, \epsilon, 0)] \leftarrow IV\|K_m\|N$;

  - Else, $\texttt{path}$ has necessarily a decryption parent path $\texttt{path}'$. Then:
    - If $C \neq \epsilon$, let $\tilde{C}$ be the last block of C. Then:

$$\texttt{S}_{\text{in}}[\texttt{path}] \leftarrow \tilde{C} \| \text{inner}_c(\text{XorState}\,(\texttt{path}', \texttt{path}) \oplus \texttt{S}_{\text{out}}[\texttt{path}'])\,;$$

    - Else, let $\tilde{A}$ be the last block of A. Then:

$$\texttt{S}_{\text{in}}[\texttt{path}] \leftarrow \left(\tilde{A} \| 0^c\right) \oplus \text{XorState}\,(\texttt{path}', \texttt{path}) \oplus \texttt{S}_{\text{out}}[\texttt{path}']\,.$$

All these states are generated *after* the interactive phase. However, some of them might use randomness from the non-interactive phase (for instance, the $\texttt{S}_{\text{in}}[(D, m, N, \epsilon, \epsilon, 0)]$s are fixed by the user key K_m). This sampling procedure generates $\mathcal{M}_D - |\texttt{P}_{\text{S}}|$ b-bit random states.

Extended Transcript. Now that the dictionaries $\texttt{S}_{\text{in}}$ and $\texttt{S}_{\text{out}}$ are defined for all labels in $\texttt{P}$ in both worlds, we can define the extended transcript $\tilde{\tau}$ built from τ by adding elements as follows:

- All permutation queries (X, Y, d) are kept untouched;
- A construction query tuple $(O, m, N, B, -) \in \tau$ is followed by tuples of the form $(\texttt{path}, \texttt{S}_{\text{in}}[\texttt{path}], \texttt{S}_{\text{out}}[\texttt{path}])$ for all $\texttt{path}$s generated by the aforementioned query. If a path repeats due to the nonce-misuse setting, the repeating tuples are removed from the transcript. Moreover, any decryption path that is superseded by an encryption path is removed from the transcript;
- At the end of $\tilde{\tau}$, a tuple containing the keys $(K_1, \dots, K_\mu)$ is added.

Summarizing, from $\tilde{\tau}$ we can reconstruct the list of all permutation queries (X, Y, d), the user's keys, the two dictionaries $\texttt{S}_{\text{in}}$ and $\texttt{S}_{\text{out}}$, and the sets $\texttt{P}$ and $\texttt{P}_{\text{S}}$. This extended transcript is released at the end of the interaction, right before the distinguisher outputs its decision bit.

Bad Events. We introduce the following bad events:

$$\begin{aligned}
\mathsf{ColK}^{\mathsf{m}}_{\mathsf{a}} &: \exists^{\neq} m_1, m_2 \in [\![1,\mu]\!] \text{ such that } K_{m_1} = K_{m_2}\,;\\
\mathsf{GueK}^{\mathsf{m}}_{\mathsf{a}} &: \exists m \in [\![1,\mu]\!], (X,Y,d) \in \tilde{\tau} \text{ such that } X[b-k-n+1 : b-n] = K_m, \text{ or}\\
&\quad \exists m \in [\![1,\mu]\!], \mathtt{path}' = (m', N', A', P', f') \in \mathtt{P} \setminus \mathtt{P}_{\mathrm{S}}\\
&\quad \text{such that } (A' \| P' \neq \epsilon) \text{ and } \mathtt{S}_{\mathrm{in}}[(\mathtt{path}')][b-k-n+1 : b-n] = K_m\,;\\
\mathsf{ColS}^{\mathsf{m}}_{\mathsf{a}} &: \exists^{\neq} \mathtt{path} = (m, N, A, P, 0), \mathtt{path}' = (m', N', A', P', 0) \in \mathtt{P} \setminus \mathtt{P}_{\mathrm{S}}\,,\\
&\quad \exists \delta \in \mathrm{ValidXor}(\mathtt{path})\,, \delta' \in \mathrm{ValidXor}(\mathtt{path}')\\
&\quad \text{such that } \mathtt{S}_{\mathrm{out}}[\mathtt{path}] \oplus \delta \overset{c}{=} \mathtt{S}_{\mathrm{out}}[\mathtt{path}'] \oplus \delta'\,;\\
\mathsf{GueS}^{\mathsf{m}}_{\mathsf{a}} &: \exists (X,Y,d) \in \tilde{\tau}, \mathtt{path} = (O, m, N, A, P, f) \in \mathtt{P} \setminus \mathtt{P}_{\mathrm{S}}, \delta \in \mathrm{ValidXor}(\mathtt{path})\\
&\quad \text{such that } \Big(f = 0 \text{ and } X \overset{c}{=} \mathtt{S}_{\mathrm{out}}[\mathtt{path}] \oplus \delta\Big) \text{ or } Y = \mathtt{S}_{\mathrm{out}}[\mathtt{path}]\,;\\
\mathsf{Dec}^{\mathsf{m}}_{\mathsf{a}} &: \exists (D, m, N, A, C, T, \tilde{P}) \in \tilde{\tau} \text{ such that}\\
&\quad \text{either } \mathrm{inner}_t(\mathtt{S}_{\mathrm{out}}[(D, m, N, A, C, 1)]) \oplus \mathrm{inner}_t(K_m) = T\,,\\
&\quad \text{or } \exists \mathtt{path} \in \mathtt{P} \text{ superseding } (D, m, N, A, C, 1)\\
&\quad \text{with } \mathrm{inner}_t(\mathtt{S}_{\mathrm{out}}[\mathtt{path}]) \oplus \mathrm{inner}_t(K_m) = T\,;\\
\mathsf{BAD}^{\mathsf{m}}_{\mathsf{a}} &: \mathsf{ColK}^{\mathsf{m}}_{\mathsf{a}} \vee \mathsf{GueK}^{\mathsf{m}}_{\mathsf{a}} \vee \mathsf{ColS}^{\mathsf{m}}_{\mathsf{a}} \vee \mathsf{GueS}^{\mathsf{m}}_{\mathsf{a}} \vee \mathsf{Dec}^{\mathsf{m}}_{\mathsf{a}}\,.
\end{aligned}$$

The sub-/superscript in the bad events indicates the proof setting, where "**m**" indicates that we are in the nonce-misuse resistance setting and "**a**" that we focus on authenticity. (Here, we remark that upcoming proofs extend over this main proof, and so do their bad events.) $\mathsf{ColK}^{\mathsf{m}}_{\mathsf{a}}$ pinpoints collisions between two user keys, $\mathsf{GueK}^{\mathsf{m}}_{\mathsf{a}}$ corresponds to the event that the adversary guesses a user key via a permutation query, or indirectly via a construction query. $\mathsf{ColS}^{\mathsf{m}}_{\mathsf{a}}$ refers to the event that two potential intermediate states collide on their inner parts, and $\mathsf{GueS}^{\mathsf{m}}_{\mathsf{a}}$ corresponds to the adversary guessing an intermediate (or potential future intermediate) state. Finally, $\mathsf{Dec}^{\mathsf{m}}_{\mathsf{a}}$ corresponds to the event that a decryption query is rejected, but the corresponding intermediate state returns the tag that was guessed by the adversary. This covers two situations, as outlined in the bad event: (i) a decryption query with a tag T is made, but the corresponding mock final state generated at the end of the interaction matches the tag, and (ii) a decryption query with a tag T is made, but later the corresponding encryption is made and returns the tag T.

Probability of Good Transcripts. As long as $\mathsf{BAD}^{\mathsf{m}}_{\mathsf{a}}$ is not set, there are no collisions between intermediate states, and no overlap exists between the permutation evaluations stemming from construction queries and those from permutation queries. Consequently, by the design of the sampling procedure in the ideal world, the intermediate states generated in the ideal world adhere to the structure of the mode Ascon-AE. Moreover, $\neg\mathsf{BAD}_{\mathsf{a}}^{\mathsf{m}}$ guarantees that the real world rejects all decryption queries. Therefore, every good transcript which is reachable in the real world is also reachable in the ideal world, and vice-versa.

Let $\tilde{\tau}$ be a transcript that does not set $\mathsf{BAD}_{\mathsf{a}}^{\mathsf{m}}$. In the real world, this transcript might not induce exactly $\mathcal{N} + \mathcal{M}_E + \mathcal{M}_D$ permutation calls, as the permutation evaluations from encryption and decryption queries might overlap.[7] Let $\ell_E(\tilde{\tau})$, $\ell_D(\tilde{\tau})$ be such that, in the real world, the encryption (resp., decryption) queries induce exactly $\mathcal{M}_E - \ell_E(\tilde{\tau})$ (resp., $\mathcal{M}_D - \ell_D(\tilde{\tau})$) distinct permutation evaluations, and let $\ell(\tilde{\tau}) = \ell_E(\tilde{\tau}) + \ell_D(\tilde{\tau})$. We have

$$\mathbf{Pr}\left(\mathcal{A}\left[W_R\right] \text{ generates } \tilde{\tau}\right) = \frac{1}{(2^b)_{\mathcal{N}+\mathcal{M}_E+\mathcal{M}_D-\ell(\tilde{\tau})}} \frac{1}{(2^k)^{\mu}} .$$

In the ideal world, the decryption queries do not generate permutation evaluations, and the overlap between $\mathcal{M}_E$ and $\mathcal{M}_D$ impacts only the number of mock intermediate states. Remarking that $\ell(\tilde{\tau})$ is the number of superseded decryption paths, a good transcript induces $\mathcal{M}_E + \mathcal{N}$ distinct permutation evaluations, and $\mathcal{M}_D - \ell(\tilde{\tau})$ distinct random states $\mathtt{S}_{\mathrm{out}}[\mathtt{path}]$. Therefore,

$$\mathbf{Pr}\left(\mathcal{A}\left[W_I\right] \text{ generates } \tilde{\tau}\right) = \frac{1}{(2^b)_{\mathcal{N}+\mathcal{M}_E} (2^b)^{\mathcal{M}_D-\ell(\tilde{\tau})}} \frac{1}{(2^k)^{\mu}} .$$

We therefore obtain

$$\frac{\mathbf{Pr}\left(\mathcal{A}\left[W_R\right] \text{ generates } \tilde{\tau}\right)}{\mathbf{Pr}\left(\mathcal{A}\left[W_I\right] \text{ generates } \tilde{\tau}\right)} = \frac{(2^b)_{\mathcal{N}+\mathcal{M}_E} (2^b)^{\mathcal{M}_D-\ell(\tilde{\tau})}}{(2^b)_{\mathcal{N}+\mathcal{M}_E+\mathcal{M}_D-\ell(\tilde{\tau})}} \geq 1 . \tag{9.10}$$

Probability of $\mathsf{BAD}_{\mathsf{a}}^{\mathsf{m}}$ in the Ideal World. We do this evaluation in a query-wise fashion, upper bounding on-the-fly the probability that any fresh permutation evaluation or mock intermediate state triggers $\mathsf{BAD}_{\mathsf{a}}^{\mathsf{m}}$. Permutation evaluations are considered in the order they occur, while mock intermediate states are considered at the end. Let $i \in [\![1, \mathcal{M}_E + \mathcal{N}]\!]$, and for an event $\mathbf{Evt}$, $\mathbf{Evt}[i]$ denotes the probability that $\mathbf{Evt}$ is set after i *fresh* permutation *evaluations* (coming either from permutation or encryption queries).

[7] Indeed, the quantities $\mathcal{M}_E$ and $\mathcal{M}_D$ are defined separately, but do not account for potential repeated permutation evaluations. For instance, if encryption query (E, m, N, A, P, C, T) is followed by a decryption query $(D, m, N, A, C\|\tilde{C}, \tilde{T})$, then fresh permutation evaluations begin only from the moment of absorbing $\tilde{C}$.

Let $\mathsf{BAD}_\mathsf{a}^\mathsf{m}[0]$ be $\mathsf{ColK}_\mathsf{a}^\mathsf{m}$, as this is the only event that can be set before any query from the distinguisher. Therefore, $\mathsf{BAD}_\mathsf{a}^\mathsf{m} \wedge \neg\mathsf{BAD}_\mathsf{a}^\mathsf{m}[\mathcal{M}_E + \mathcal{N}]$ denotes the event that one of the intermediate states generated after the interaction (i.e., by decryption queries that are not superseded) sets $\mathsf{BAD}_\mathsf{a}^\mathsf{m}$.

Let $\mathbf{1}_\mathsf{C}[i]$ denote the indicator function equal to one if and only if the evaluation number i is *fresh* and made in the context of a construction query (here, necessarily an encryption query). Similarly, $\mathbf{1}_\mathsf{P}[i]$ equals one if and only if the evaluation number i is *fresh* and made from a permutation query. Note that whenever an encryption query (E, m, N, A, P, C, T) is made, there can exist earlier decryption queries of the form $(D, m, N, A, C, T_j, \perp)_j$. Define $\eta_{\mathsf{enc},\mathsf{dec}}[i]$ as follows:

- If evaluation i originates from a permutation query or an encryption query with a non-final path, then $\eta_{\mathsf{enc},\mathsf{dec}}[i] = 0$;
- Otherwise, let $(E, m, N, A, P, C, T) \in \tau$ be the associated encryption query, then $\eta_{\mathsf{enc},\mathsf{dec}}[i]$ counts the number of (necessarily earlier) decryption queries of the form $(D, m, N, A, C, T_j, \perp)$.

Since two encryption queries with the same user, nonce, associated data, but with different plaintexts cannot have the same ciphertexts, one single decryption query cannot contribute to increment two distinct $\eta_{\mathsf{enc},\mathsf{dec}}[i]$ and $\eta_{\mathsf{enc},\mathsf{dec}}[j]$. Therefore, we have $\sum_{i=1}^{\mathcal{M}_E+\mathcal{N}} \eta_{\mathsf{enc},\mathsf{dec}}[i] \leq Q_D$.

We break down the probability of $\mathsf{BAD}_\mathsf{a}^\mathsf{m}$ by using basic probability as follows:

1. $\mathsf{BAD}_\mathsf{a}^\mathsf{m}[0]$, or $\mathsf{ColK}_\mathsf{a}^\mathsf{m}$ by definition;
2. $\mathsf{BAD}_\mathsf{a}^\mathsf{m}[i]$ for $i \in [\![1, \mathcal{M}_E + \mathcal{N}]\!]$, in more detail:
   (a) $\mathsf{GueK}_\mathsf{a}^\mathsf{m}[i]$, assuming $\neg\mathsf{BAD}_\mathsf{a}^\mathsf{m}[i-1]$;
   (b) $\mathsf{Dec}_\mathsf{a}^\mathsf{m}[i]$, assuming $\neg\mathsf{BAD}_\mathsf{a}^\mathsf{m}[i-1] \wedge \neg\mathsf{GueK}_\mathsf{a}^\mathsf{m}[i]$;[8]
   (c) $\mathsf{ColS}_\mathsf{a}^\mathsf{m}[i]$, assuming $\neg\mathsf{BAD}_\mathsf{a}^\mathsf{m}[i-1] \wedge \neg\mathsf{GueK}_\mathsf{a}^\mathsf{m}[i] \wedge \neg\mathsf{Dec}_\mathsf{a}^\mathsf{m}[i]$;
   (d) $\mathsf{GueS}_\mathsf{a}^\mathsf{m}[i]$, assuming $\neg\mathsf{BAD}_\mathsf{a}^\mathsf{m}[i-1] \wedge \neg\mathsf{GueK}_\mathsf{a}^\mathsf{m}[i] \wedge \neg\mathsf{Dec}_\mathsf{a}^\mathsf{m}[i] \wedge \neg\mathsf{ColS}_\mathsf{a}^\mathsf{m}[i]$;
3. $\mathsf{BAD}_\mathsf{a}^\mathsf{m}$ after the interaction, which is equivalent to $\mathsf{BAD}_\mathsf{a}^\mathsf{m}$, assuming $\neg\mathsf{BAD}_\mathsf{a}^\mathsf{m}[\mathcal{M}_E + \mathcal{N}]$.

[8]Note that, although $\mathsf{Dec}_\mathsf{a}^\mathsf{m}$ involves a decryption query, any permutation evaluation that triggers this bad event during the interactive phase comes from an encryption query.

Case 2 can be set only with intermediate states from encryption queries while case 3 involves additionally intermediate states from decryption queries.

Case 1. Let us start with the bounding of case 1. We have

$$\mathbf{Pr}\left(\mathsf{BAD}^{\mathsf{m}}_{\mathsf{a}}[0]\right) = \mathbf{Pr}\left(\mathsf{ColK}^{\mathsf{m}}_{\mathsf{a}}\right) \leq \frac{\mu(\mu-1)}{2^{k+1}} \,. \tag{9.11}$$

Case 2. In the following, let $i \in [\![1, \mathcal{M}_E + \mathcal{N}]\!]$. We first focus on the conditioned $\mathsf{GueK}^{\mathsf{m}}_{\mathsf{a}}[i]$ probability of case 2a. In order to set this event, the adversary must be able to guess one of the μ uniform random keys, either via a direct permutation call, or via a permutation evaluation made from an encryption query. In the second case, we can out of generosity *for this event only* assume that the adversary has full control on the input of the intermediate states. Moreover, each failed guess eliminates μ elements in $\{0,1\}^k$ from the set of candidate keys. Therefore,

$$\begin{aligned}\mathbf{Pr}\left(\mathsf{GueK}^{\mathsf{m}}_{\mathsf{a}}[i] \mid \neg\mathsf{BAD}^{\mathsf{m}}_{\mathsf{a}}[i-1]\right) &\leq \left(\mathbf{1}_{\mathsf{P}}[i] + \mathbf{1}_{\mathsf{C}}[i]\right) \frac{\mu}{2^k - \mu\left(\mathcal{M}_E + \mathcal{N}\right)} \\ &\leq \left(\mathbf{1}_{\mathsf{P}}[i] + \mathbf{1}_{\mathsf{C}}[i]\right) \frac{2\mu}{2^k} \,, \end{aligned} \tag{9.12}$$

where we used that $\mu\left(\mathcal{M} + \mathcal{N}\right) \leq 2^{k-1}$.

Regarding the conditioned $\mathsf{Dec}^{\mathsf{m}}_{\mathsf{a}}[i]$ of case 2b, this event during the interactive phase can be set only during the generation of a final state of an encryption query. Those aforementioned states are sampled uniformly in a permutation-consistent way, added to the key, and truncated before output. The keys are uniformly random and hidden from the adversary, and there are by definition $\eta_{\mathsf{enc,dec}}[i]$ candidate tags to hit. Therefore,

$$\mathbf{Pr}\left(\mathsf{Dec}^{\mathsf{m}}_{\mathsf{a}}[i] \mid \neg\mathsf{BAD}^{\mathsf{m}}_{\mathsf{a}}[i-1] \wedge \neg\mathsf{GueK}^{\mathsf{m}}_{\mathsf{a}}[i]\right) \leq \eta_{\mathsf{enc,dec}}[i] \frac{1}{2^t} \,. \tag{9.13}$$

Then, we focus on the conditioned $\mathsf{ColS}^{\mathsf{m}}_{\mathsf{a}}[i]$ probability in case 2c. This event can be set only by an evaluation from an encryption query. Let $\mathtt{path} \in \mathtt{P}$ be the associated encryption path. Assuming $\neg\mathsf{BAD}^{\mathsf{m}}_{\mathsf{a}}$, the state $\mathtt{S}_{\mathrm{out}}[\mathtt{path}]$ is sampled uniformly at random from a set of size at least $2^b - \mathcal{M}_E - \mathcal{N}$. Therefore, the probability that $\mathtt{S}_{\mathrm{out}}[\mathtt{path}]$ collides on its inner part with any other state $\mathtt{S}_{\mathrm{out}}[\mathtt{path}']$, modulo XORing key material or the domain separator bits (captured by the presence of δ and δ'), can be upper bounded by

$$\mathbf{1}_{\mathsf{C}}[i] \frac{9 \cdot 2^r \cdot \mathcal{M}_E}{2^b - \mathcal{M}_E - \mathcal{N}} \,,$$

where we used that $|\text{ValidXor}(\texttt{path})| \leq 3$. Therefore,

$$\mathbf{Pr}\left(\mathsf{ColS}_{\mathsf{a}}^{\mathsf{m}}[i] \mid \neg\mathsf{BAD}_{\mathsf{a}}^{\mathsf{m}}[i-1] \wedge \neg\mathsf{GueK}_{\mathsf{a}}^{\mathsf{m}}[i] \wedge \neg\mathsf{Dec}_{\mathsf{a}}^{\mathsf{m}}[i]\right) \leq \mathbf{1}_{\mathsf{C}}[i]\frac{18\mathcal{M}_E}{2^c}\,, \quad (9.14)$$

where we used that $\mathcal{M}_E + \mathcal{N} \leq 2^{b-1}$.

Then, let us focus on the conditioned $\mathsf{GueS}_{\mathsf{a}}^{\mathsf{m}}[i]$ from case 2d. First, if the evaluation number i comes from a permutation query $(X, Y, d) \in \tau$, then for every existing intermediate state $\mathtt{S}_{\text{in}}[\texttt{path}]$ and $\mathtt{S}_{\text{out}}[\texttt{path}]$, we evaluate the probability that the query (X, Y, d) paired with the state sets $\mathsf{BAD}_{\mathsf{a}}^{\mathsf{m}}$, and count the number of candidate states to be guessed:

- If $d = \mathit{fwd}$ (resp., $d = \mathit{inv}$), then the state Y (resp., X) is sampled uniformly at random from a set of size at least $2^b - \mathcal{M}_E - \mathcal{N}$, thus it collides with a $\mathtt{S}_{\text{out}}[\texttt{path}] \oplus \delta$ on its inner part with probability at most $\frac{3\cdot 2^r}{2^b - \mathcal{M}_E - \mathcal{N}}$. There are at most $\mathcal{M}_E$ such states, thus we get a probability at most

$$\mathbf{1}_{\mathsf{P}}[i]\frac{6\mathcal{M}_E}{2^c}\,,$$

  where we used that $\mathcal{M}_E + \mathcal{N} \leq 2^{b-1}$. From now on, we consider the other cases, where the direction of the permutation aligns with the state to guess;

- If the state to guess is of the form $\mathtt{S}_{\text{out}}[(E, m, N, A, P, 1)]$, then conditioned on $\neg\mathsf{BAD}_{\mathsf{a}}^{\mathsf{m}}[i-1]$, this state is sampled uniformly from a set of size at least $2^b - \mathcal{M}_E - \mathcal{N}$. The outer $b - t$ bits are completely hidden from the adversary. The rightmost t bits are added to $\text{inner}_t(K_m)$ before being returned as tag. Since the keys are random and hidden, access to those t bits is of no help for the adversary. The total number of distinct states $\mathtt{S}_{\text{out}}[(E, m, N, A, P, 1)]$ is upper bounded by Q_E. Therefore, this event is set with probability at most

$$\mathbf{1}_{\mathsf{P}}[i]\frac{Q_E}{2^b - \mathcal{M}_E - \mathcal{N}} \leq \mathbf{1}_{\mathsf{P}}[i]\frac{2Q_E}{2^b}\,,$$

  where we used that $\mathcal{M}_E + \mathcal{N} \leq 2^{b-1}$;

- Otherwise, the state to guess is an internal state (either the output of the first permutation evaluation, the input of the last permutation evaluation, or any input or output of a middle permutation evaluation). Since the adversary is allowed to repeat nonces, we can without loss of accuracy in the bounding assume that the states have their outer part set to a value of the adversary's choice, thus we consider equality on the inner part. Each of the states is sampled uniformly from a set of size at least $2^b - \mathcal{M}_E - \mathcal{N}$, and as long as $\neg\mathsf{BAD}^{\mathsf{m}}_{\mathsf{a}}[i-1]$ holds, their inner part remains secret from the adversary. There are at most $3\,(\mathcal{M}_E - Q_E)$ states concerned (including potential future states), thus we obtain a probability at most

$$\mathbf{1}_{\mathsf{P}}[i]\frac{3\,(\mathcal{M}_E - Q_E)\cdot 2^r}{2^b - \mathcal{M}_E - \mathcal{N}} \leq \mathbf{1}_{\mathsf{P}}[i]\frac{6\,(\mathcal{M}_E - Q_E)}{2^c}\,,$$

where we used that $\mathcal{M}_E + \mathcal{N} \leq 2^{b-1}$.

Then, if the evaluation number i comes from an encryption query and is associated to a path `path`, in all cases the bad event can be triggered only by the randomness of $\mathtt{S}_{\text{out}}[\mathtt{path}]$. There are at most 3 different possible values for δ, and $\mathtt{S}_{\text{out}}[\mathtt{path}]$ is sampled uniformly at random from a set of size at least $2^b - \mathcal{M}_E - \mathcal{N}$. Therefore, this event is set with probability at most

$$\mathbf{1}_{\mathsf{C}}[i]\frac{6\mathcal{N}}{2^c}\,,$$

where we used that $\mathcal{M}_E + \mathcal{N} \leq 2^{b-1}$. Therefore,

$$\begin{aligned}\mathbf{Pr}\left(\mathsf{GueS}^{\mathsf{m}}_{\mathsf{a}}[i] \mid \neg\mathsf{BAD}^{\mathsf{m}}_{\mathsf{a}}[i-1] \wedge \neg\mathsf{GueK}^{\mathsf{m}}_{\mathsf{a}}[i] \wedge \neg\mathsf{Dec}^{\mathsf{m}}_{\mathsf{a}}[i] \wedge \neg\mathsf{ColS}^{\mathsf{m}}_{\mathsf{a}}[i]\right)&\\ \leq \mathbf{1}_{\mathsf{P}}[i]\frac{12\mathcal{M}_E}{2^c} + \mathbf{1}_{\mathsf{C}}[i]\frac{6\mathcal{N}}{2^c}\,.&\end{aligned} \tag{9.15}$$

Case 3. Finally, we can focus on the conditioned bad event in case 3. Because $\neg\mathsf{BAD}^{\mathsf{m}}_{\mathsf{a}}[\mathcal{M}_E + \mathcal{N}]$ holds, then in order to be set, the bad event necessarily involves an intermediate state $\mathtt{S}_{\text{out}}[\mathtt{path}]$, for `path` a decryption path not superseded. The inner part of these states are sampled uniformly at random, with at most $\mathcal{M}_D$ such states in total. We evaluate all sub-events as follows:

9

- $\mathsf{ColK}^{\mathsf{m}}_{\mathsf{a}}$: this event cannot be set after the interaction;
- $\mathsf{GueK}^{\mathsf{m}}_{\mathsf{a}}$: this event is set with probability at most $\frac{2\mu\mathcal{M}_D}{2^k}$, where we used that $\mu\,(\mathcal{N} + \mathcal{M}) \leq 2^{k-1}$;
- $\mathsf{ColS}^{\mathsf{m}}_{\mathsf{a}}$: this event is set with probability at most $\frac{9\mathcal{M}_D(\mathcal{M}_E+\mathcal{M}_D)}{2^c}$;
- $\mathsf{GueS}^{\mathsf{m}}_{\mathsf{a}}$: this event is set with probability at most $\frac{6\mathcal{M}_D\mathcal{N}}{2^c}$;

- $\mathsf{Dec}_{\mathsf{a}}^{\mathsf{m}}$: this event is set with probability at most $\frac{Q_D}{2^t}$.

Therefore,

$$\mathbf{Pr}\left(\mathsf{BAD}_{\mathsf{a}}^{\mathsf{m}} \mid \neg\mathsf{BAD}_{\mathsf{a}}^{\mathsf{m}}[\mathcal{M}_E + \mathcal{N}]\right) \leq \frac{2\mu\mathcal{M}_D}{2^k} + \frac{9\mathcal{M}_D(\mathcal{M}_E + \mathcal{M}_D)}{2^c} + \frac{6\mathcal{M}_D\mathcal{N}}{2^c} + \frac{Q_D}{2^t}. \tag{9.16}$$

Conclusion. By combining the conditioned probabilities of (9.11) to (9.16), we obtain

$$\mathbf{Pr}\left(\mathsf{BAD}_{\mathsf{a}}^{\mathsf{m}}\right) \leq \frac{\mu(\mu-1)}{2^{k+1}} + \frac{2\mu\left(\mathcal{M}+\mathcal{N}\right)}{2^k} + \frac{18\mathcal{M}\left(\mathcal{M}+\mathcal{N}\right)}{2^c} + \frac{2Q_D}{2^t}.$$

We obtained an upper bound for the probability that the ideal world generates a bad transcript. Using the H-coefficient technique, and the fact that the ratio of good transcript is lower-bounded by one (cf., (9.10)), we conclude. □

9.6.2 Proof of Theorem 9.4.3

Authenticity in the nonce-misuse resilience setting is implied by authenticity in the nonce-misuse setting (i.e., Theorem 9.4.2). This proof is therefore dedicated to confidentiality. This proof will re-use a significant part of the notation defined in Section 9.6.1, and we will explicitly highlight the adaptations made here.

Let $\mathcal{A}$ be an adversary that makes at most $\mathcal{N}$ permutation queries, and Q_E encryption queries of at most $\mathcal{M}_E$ blocks, as in the theorem statement. Our goal is to upper bound $\mathbf{Adv}_{\text{Ascon}}^{\mu\text{-mr-conf}}(\mathcal{A})$. The adversary interacts either with the *real world* W_R, which gives access to $\left[\left(\mathbf{Enc}_{K_m}^{\mathcal{P}}, \mathbf{Enc}_{K_m}^{\mathcal{P}}\right)_{m=1}^{\mu}, \mathcal{P}^{\pm}\right]$, or with the *ideal world* W_I, which gives access to $\left[\left(\mathbf{Enc}_{K_m}^{\mathcal{P}}, \$_m\right)_{m=1}^{\mu}, \mathcal{P}^{\pm}\right]$. $\mathcal{A}$ may misuse nonces with the queries to $\mathcal{O}_{1,m}$, but not with $\mathcal{O}_{2,m}$. We will refer to a query to $\mathcal{O}_{2,m}$ as a *challenge* query, and to $\mathcal{O}_{1,m}$ as a *learning* query. Let $\mathcal{M}_{E,C}$ be the data complexity of challenge queries, and $\mathcal{M}_{E,L}$ the one of learning queries, so that $\mathcal{M}_E = \mathcal{M}_{E,C} + \mathcal{M}_{E,L}$.

Transcript Notation. We define below notation for the transcript τ:

- A forward (resp., inverse) permutation query with input X and output Y generates the transcript element (X, Y, fwd) (resp., (X, Y, inv));

- A challenge encryption query with user m, input (N, A, P), and output (C, T) generates the transcript element (E_C, m, N, A, P, C, T);
- A learning encryption query with user m, input (N, A, P), and output (C, T) generates the transcript element (E_L, m, N, A, P, C, T).

Paths and Intermediate States. We will re-use the path notation from Section 9.6.1. This time, all construction queries are encryption queries, so that the paths take the form $\texttt{path} = (O, m, N, A, P, f) \in \texttt{P}$, where $O = E_C$ if the associated query is a challenge query, otherwise $O = E_L$. Thanks to the fact that the queries to $\mathcal{O}_{1,m}$ and $\mathcal{O}_{2,m}$ do not have overlapping nonces, a challenge path cannot be the parent of a learning path, and vice-versa. Because the queries to the challenge oracle are nonce-respecting, each challenge path can have at most one child. We define the dictionaries $\texttt{S}_{\text{in}}$ and $\texttt{S}_{\text{out}}$ similarly to Section 9.6.1. The labels of the dictionaries are in $\texttt{P}$ in the real world, or the set of learning paths in the ideal world.

Mock Intermediate States. Again, our approach will be to establish an extended transcript that releases the intermediate states associated to all construction queries. Therefore, we define in the following a procedure to generate mock intermediate states in the ideal world for $\texttt{S}_{\text{in}}[\texttt{path}]$ and $\texttt{S}_{\text{out}}[\texttt{path}]$ for any challenge path $\texttt{path}$. The sampling procedure, taking place at the end of the interaction, operates as follows:

- Sample $\texttt{S}_{\text{in}}[\texttt{path}]$, for all $\texttt{path} = (E_C, m, N, A, P, f)$ as follows:
  - If $P = A = \epsilon$, then necessarily $f = 0$ and $\texttt{S}_{\text{in}}[\texttt{path}] \leftarrow IV \| K_m \| N$;
  - Else, if $P = \epsilon$, then $\texttt{S}_{\text{in}}[\texttt{path}] \xleftarrow{\$} \{0,1\}^b$;
  - Else, let $l \in \mathbb{N}^*$ be the length of P, find the (unique) encryption query with user m and nonce N, get the ciphertext block number l (name it C_l), and sample $Z \xleftarrow{\$} \{0,1\}^c$. Then: $\texttt{S}_{\text{in}}[\texttt{path}] \leftarrow C_l \| Z$;
- Then, sample $\texttt{S}_{\text{out}}[\texttt{path}]$, for all $\texttt{path} = (E_C, m, N, A, P, f)$ as follows:
  - If $f = 1$, the existence of such a path means that there exists $(E_C, m, N, A, P, C, T) \in \tau$. Then, sample $Z \xleftarrow{\$} \{0,1\}^{b-t}$, and: $\texttt{S}_{\text{out}}[\texttt{path}] \leftarrow Z \| \left(\text{inner}_t(K_m) \oplus T\right)$;
  - Otherwise, let $\texttt{pathC} = (m, N, A', P', f')$ be the (necessarily unique) child of $\texttt{path}$, and let $B \in \{0,1\}^r$ be the last block of $A' \| P'$, then:

$$\texttt{S}_{\text{out}}[\texttt{path}] \leftarrow (B \| 0^c) \oplus \text{XorState}\left(\texttt{path}, \texttt{pathC}\right) \oplus \texttt{S}_{\text{in}}[\texttt{pathC}]\,.$$

All of these states are generated *after* the interactive phase. However, some of them might use randomness from the non-interactive phase (for instance, the $\mathtt{S}_{\mathrm{in}}[(E_C, m, N, \epsilon, \epsilon, 0)]$s are fixed by the user key K_m, and the $\mathtt{S}_{\mathrm{out}}[(E_C, m, N, A, P, 1)]$ have their rightmost t bits set by the encryption query output, along with the key K_m).

Extended Transcript. Now that the dictionaries $\mathtt{S}_{\mathrm{in}}$ and $\mathtt{S}_{\mathrm{out}}$ are defined for all labels in $\mathtt{P}$ in both worlds, we can define the extended transcript $\tilde{\tau}$ built from τ by replacing adding elements as follows:

- All permutation queries (X, Y, d) are kept untouched;
- An encryption query tuple $(O, m, N, A, P, C, T) \in \tau$ is now followed by tuples of the form $(\mathtt{path}, \mathtt{S}_{\mathrm{in}}[\mathtt{path}], \mathtt{S}_{\mathrm{out}}[\mathtt{path}])$ for all $\mathtt{path}$ generated by (m, N, A, P). If some paths repeat (due to the nonce-misuse setting), then the duplicates are removed from the transcript;
- At the end of $\tilde{\tau}$, a tuple containing the keys $(K_1, \ldots, K_\mu)$ is added.

Summarizing, from $\tilde{\tau}$ we can reconstruct the two dictionaries $\mathtt{S}_{\mathrm{in}}$ and $\mathtt{S}_{\mathrm{out}}$, the set $\mathtt{P}$, the list of all permutation queries (X, Y, d), and the user's keys. This transcript is released at the end of the interaction, right before the distinguisher outputs its decision bit.

Bad Events. We will re-use the bad events defined in Section 9.6.1, except Dec, which does not apply there. In order to completely inherit the notation from this section, let us define $\mathtt{P}_{\mathrm{S}} = \emptyset$. The bad events are as follows:

$$\begin{aligned}
&\mathsf{ColK}^{\mathsf{mr}}_{\mathsf{c}} : \mathsf{ColK}^{\mathsf{m}}_{\mathsf{a}} \text{ (of Section 9.6.1)}\,; \quad \mathsf{GueK}^{\mathsf{mr}}_{\mathsf{c}} : \mathsf{GueK}^{\mathsf{m}}_{\mathsf{a}} \text{ (of Section 9.6.1)}\,;\\
&\mathsf{ColS}^{\mathsf{mr}}_{\mathsf{c}} : \mathsf{ColS}^{\mathsf{m}}_{\mathsf{a}} \text{ (of Section 9.6.1)}\,; \quad \mathsf{GueS}^{\mathsf{mr}}_{\mathsf{c}} : \mathsf{GueS}^{\mathsf{m}}_{\mathsf{a}} \text{ (of Section 9.6.1)}\,;\\
&\mathsf{BAD}^{\mathsf{mr}}_{\mathsf{c}} : \mathsf{ColK}^{\mathsf{mr}}_{\mathsf{c}} \vee \mathsf{GueK}^{\mathsf{mr}}_{\mathsf{c}} \vee \mathsf{ColS}^{\mathsf{mr}}_{\mathsf{c}} \vee \mathsf{GueS}^{\mathsf{mr}}_{\mathsf{c}}\,.
\end{aligned}$$

Probability of Good Transcripts. As in Section 9.6.1, the absence of $\mathsf{BAD}^{\mathsf{mr}}_{\mathsf{c}}$ guarantees that the intermediate states generated in the ideal world are consistent with the structure of the mode Ascon-AE. Therefore, every transcript which is reachable in the real world is also reachable in the ideal world, and vice-versa. Let $\tilde{\tau}$ be a transcript that does not set $\mathsf{BAD}^{\mathsf{mr}}_{\mathsf{c}}$. In the real world, this transcript induces $\mathcal{N} + \mathcal{M}_{E,L} + \mathcal{M}_{E,C}$ permutation calls and μ keys, thus

$$\mathbf{Pr}\left(\mathcal{A}\left[W_R\right] \text{ generates } \tilde{\tau}\right) = \frac{1}{(2^b)_{\mathcal{N}+\mathcal{M}_{E,L}+\mathcal{M}_{E,C}}} \frac{1}{(2^k)^\mu}\,.$$

In the ideal world, this transcript induces $\mathcal{N} + \mathcal{M}_{E,L}$ permutation calls, $\mathcal{M}_{E,C}$ random values, and μ keys. Therefore,

$$\mathbf{Pr}\left(\mathcal{A}\left[W_I\right] \text{ generates } \tilde{\tau}\right) = \frac{1}{\left(2^b\right)_{\mathcal{N}+\mathcal{M}_{E,L}} \left(2^b\right)^{\mathcal{M}_{E,C}}} \frac{1}{\left(2^k\right)^{\mu}} \,.$$

Therefore,

$$\frac{\mathbf{Pr}\left(\mathcal{A}\left[W_R\right] \text{ generates } \tilde{\tau}\right)}{\mathbf{Pr}\left(\mathcal{A}\left[W_I\right] \text{ generates } \tilde{\tau}\right)} = \frac{\left(2^b\right)_{\mathcal{N}+\mathcal{M}_{E,L}} \left(2^b\right)^{\mathcal{M}_{E,C}}}{\left(2^b\right)_{\mathcal{N}+\mathcal{M}_{E,L}+\mathcal{M}_{E,C}}} \geq 1 \,. \tag{9.17}$$

Probability of $\mathsf{BAD}_{\mathsf{c}}^{\mathsf{mr}}$ in the Ideal World. This will be done in a query-wise fashion, similarly to Section 9.6.1. In the ideal world, the only encryption queries that trigger permutation evaluations are learning queries. Let $i \in [\![0, \mathcal{M}_{E,L} + \mathcal{N}]\!]$, and denote by $\mathsf{BAD}_{\mathsf{c}}^{\mathsf{mr}}[i]$ the probability that $\mathsf{BAD}_{\mathsf{c}}^{\mathsf{mr}}$ is set after i *fresh* permutation *evaluations* (coming either from permutation or learning construction queries). Here, as before, $\mathsf{BAD}_{\mathsf{c}}^{\mathsf{mr}}[0] = \mathsf{ColK}_{\mathsf{c}}^{\mathsf{mr}}$.

By basic probability, we have

$$\mathbf{Pr}\left(\mathsf{BAD}_{\mathsf{c}}^{\mathsf{mr}}\right) \leq \mathbf{Pr}\left(\mathsf{BAD}_{\mathsf{c}}^{\mathsf{mr}}[\mathcal{M}_{E,L} + \mathcal{N}]\right) + \mathbf{Pr}\left(\mathsf{BAD}_{\mathsf{c}}^{\mathsf{mr}} \mid \neg\mathsf{BAD}_{\mathsf{c}}^{\mathsf{mr}}[\mathcal{M}_{E,L} + \mathcal{N}]\right) \,. \tag{9.18}$$

We remark that the bounding of the query-wise event $\mathsf{BAD}_{\mathsf{c}}^{\mathsf{mr}}[\mathcal{M}_{E,L}+\mathcal{N}]$ can be performed the same way as in the proof from Section 9.6.1, the differences being that (i) at most $\mathcal{M}_{E,L}$ permutation evaluations from encryption queries are made during the interactive phase (as opposed to $\mathcal{M}_E$ in Section 9.6.1), and (ii) the bad event Dec does not apply here. Therefore,

$$\mathbf{Pr}\left(\mathsf{BAD}_{\mathsf{c}}^{\mathsf{mr}}[\mathcal{M}_{E,L} + \mathcal{N}]\right) \leq \frac{\mu(\mu-1)}{2^{k+1}} + \frac{2\mu\left(\mathcal{M}_{E,L} + \mathcal{N}\right)}{2^k} + \frac{18\left(\mathcal{M}_{E,L}\right)^2}{2^c} + \frac{18\mathcal{M}_{E,L}\mathcal{N}}{2^c} \,. \tag{9.19}$$

The bad events in the second term appearing in (9.18) can also be upper bounded the same way, noticing again that this phase involves $\mathcal{M}_{E,C}$ fresh intermediate states (and not $\mathcal{M}_D$). This time, the intermediate states have their entire b bits generated randomly, but since the bad events are defined only on the inner part, we can use the same bounding technique. We obtain

$$\mathbf{Pr}\left(\mathsf{BAD}_{\mathsf{c}}^{\mathsf{mr}} \mid \neg\mathsf{BAD}_{\mathsf{c}}^{\mathsf{mr}}[\mathcal{M}_{E,L} + \mathcal{N}]\right) \leq \frac{2\mu\mathcal{M}_{E,C}}{2^k} + \frac{9\mathcal{M}_{E,C}(\mathcal{M}_{E,L} + \mathcal{M}_{E,C})}{2^c} + \frac{6\mathcal{M}_{E,C}\mathcal{N}}{2^c} \,. \tag{9.20}$$

Finally, combining (9.19) and (9.20) into (9.18), we obtain

$$\mathbf{Pr}\left(\mathsf{BAD}_{\mathsf{c}}^{\mathsf{mr}}\right) \leq \frac{\mu(\mu-1)}{2^{k+1}} + \frac{2\mu\left(\mathcal{M}_E+\mathcal{N}\right)}{2^k} + \frac{18\mathcal{M}_E\left(\mathcal{M}_E+\mathcal{N}\right)}{2^c}\,.$$

We obtained an upper bound for probability that the ideal world generates a bad transcript. Using the H-coefficient technique, and the fact that the ratio of good transcript is lower-bounded by one (cf., (9.17)), we conclude. □

9.6.3 Proof of Theorem 9.5.1

We consider authenticity in Section 9.6.3.1 and confidentiality in Section 9.6.3.2.

9.6.3.1 Authenticity

Let $\mathcal{A}$ be an adversary that makes at most $\mathcal{N}$ permutation queries, Q_E encryption queries of at most $\mathcal{M}_E$ blocks, and Q_D decryption queries of at most $\mathcal{M}_D$ blocks, as in the theorem statement. Our goal is to upper bound $\mathbf{Adv}^{\mu\text{-lr-auth}}_{\text{Ascon},\mathcal{L}}(\mathcal{A})$, for any set of leakage functions that do not leak any information about the two outer permutation calls during the initialization and finalization phases. We therefore assume maximal leakage, so that the leakage function leaks all inner permutation calls. The adversary interacts either with the *real world* W_R, which gives access to

$$\left[\left(\left[\mathbf{Enc}^{\mathcal{P}}_{K_m}\right]_L, \mathbf{Enc}^{\mathcal{P}}_{K_m}, \left[\mathbf{Dec}^{\mathcal{P}}_{K_m}\right]_L, \mathbf{Dec}^{\mathcal{P}}_{K_m}\right)_{m=1}^{\mu}, \mathcal{P}^{\pm}\right],$$

or with the *ideal world* W_I, which gives access to

$$\left[\left(\left[\mathbf{Enc}^{\mathcal{P}}_{K_m}\right]_L, \mathbf{Enc}^{\mathcal{P}}_{K_m}, \left[\mathbf{Dec}^{\mathcal{P}}_{K_m}\right]_L, \perp\right)_{m=1}^{\mu}, \mathcal{P}^{\pm}\right].$$

The adversary must be nonce-respecting with its queries to $\mathcal{O}_{2,m}$. The oracles $\mathcal{O}_{1,m}$ and $\mathcal{O}_{3,m}$ will be referred to as the *leaky oracles*, while the oracles $\mathcal{O}_{2,m}$ and $\mathcal{O}_{4,m}$ will be referred to as the *challenge oracles*. Although $\mathcal{O}_{2,m}$ does not produce real-or-random strings, we still categorize it as a challenge oracle because the nonces used with $\mathcal{O}_{2,m}$ can be re-used with the decryption oracle $\mathcal{O}_{4,m}$. To accurately track the adversarial resources, we refine the online complexity $(Q, \mathcal{M})$ as follows: it is split into the queries to the challenge oracles $(Q_C, \mathcal{M}_C)$ and queries to the leaky oracles $(Q_L, \mathcal{M}_L)$. These are further divided into encryption and decryption queries, with the symbols D or E prepended to the subscript. For example, $(Q_{E,L}, \mathcal{M}_{E,L})$ represents the online complexity of queries made to the leaky encryption oracle.

Transcript Notation. We define below a notation for the transcript τ:

- A forward (resp., inverse) permutation query with input X and output Y generates the transcript element (X, Y, fwd) (resp., (X, Y, inv));
- A leaky query with user m generates the following transcript elements:
  - If the query is an encryption query with input (N, A, P), and output (C, T), the element (E_L, m, N, A, P, C, T) is added to the transcript;
  - If the query is a decryption query with input (N, A, C, T), and output $\tilde{P}$, the element $(D_L, m, N, A, C, T, \tilde{P})$ is added to the transcript;
  - An element (X, Y, cons), for all inner permutation evaluations made with input X and output Y from the construction. In other words, all permutation evaluations are added to the transcript, *except the two outer ones*. As before, duplicate evaluations are removed from the transcript;
- A query to the challenge encryption oracle with user m, input (N, A, P), and output (C, T), generates the transcript element (E_C, m, N, A, P, C, T);
- A query to the challenge decryption oracle with user m, input (N, A, C, T), and output $\tilde{P}$, generates the transcript element $(D_C, m, N, A, C, T, \tilde{P})$.

Paths and Intermediate States. We adapt the path notation from Section 9.6.1 as follows:

- The paths generated by the challenge queries inherit the inductive path definition from Section 9.6.1, with one administrative modification: the very first element of the path is adjusted to distinguish it as a challenge path. In detail, a challenge path takes the form $\texttt{path} = (O, m, N, A, B, f)$, where $O = E_C$ if the associated query is an encryption query, and $O = D_C$ if the associated query is a decryption query. Denote by $\texttt{P}_C$ the set of paths generated by this procedure, and let $\texttt{P}_S$ be the set of decryption paths in $\texttt{P}_C$ that are superseded;
- A leaky query $(O, m, N, A, B, -) \in \tau$ with $O \in \{E_L, D_L\}$ generates exactly two paths: $(O, m, N, \epsilon, \epsilon, 0)$ and $(O, m, N, A, B, 1)$. All other intermediate states are given to the adversary and treated as direct permutation queries in the transcript. Let $\texttt{P}_L$ be the set of paths generated by leaky construction queries.

Let $\mathtt{P} = \mathtt{P}_L \cup \mathtt{P}_C$ be the set of all paths generated by construction queries. We define the dictionaries $\mathtt{S}_{\text{in}}$ and $\mathtt{S}_{\text{out}}$ as in Section 9.6.1, with labels in $\mathtt{P}$ in the real world. In the ideal world, however, the challenge decryption paths are (not yet) present. We stress that the leaked intermediate states (X, Y, cons) are absent in the set of paths $\mathtt{P}$.

Mock Intermediate States. Again, we aim to establish an extended transcript that releases the intermediate states, hence we need to specify a procedure to sample $\mathtt{S}_{\text{in}}[\mathtt{path}]$ and $\mathtt{S}_{\text{out}}[\mathtt{path}]$ for any challenge decryption path $\mathtt{path}$. The procedure is the same as the one defined in Section 9.6.1, and we can safely ignore leaky paths, since they do not have overlapping nonces.

Extended Transcript. Now that the dictionaries $\mathtt{S}_{\text{in}}$ and $\mathtt{S}_{\text{out}}$ are defined for all keys in $\mathtt{P}$ in both worlds, we can define the extended transcript $\tilde{\tau}$ built from τ as follows:

- All permutation evaluations from permutation queries or from leaky construction queries (X, Y, d) are kept untouched, except that if a tuple (X, Y) repeats, then it is removed from $\tilde{\tau}$;

- A challenge construction query $(O, m, N, A, P, C, T) \in \tau$ is now followed by tuples of the form $(\mathtt{path}, \mathtt{S}_{\text{in}}[\mathtt{path}], \mathtt{S}_{\text{out}}[\mathtt{path}])$ for all $\mathtt{path}$s descendant of the aforementioned query. If some paths repeat (due to the nonce-misuse setting, or if a decryption path is superseded by a challenge encryption path), then the duplicates are removed from the transcript;

- A leaky construction query $(O, m, N, A, P, C, T) \in \tau$ is now followed by tuples of the form $(\mathtt{path}, \mathtt{S}_{\text{in}}[\mathtt{path}], \mathtt{S}_{\text{out}}[\mathtt{path}])$, for the two elements $\mathtt{path}$ generated by the aforementioned query. Again, if some paths are redundant, then the duplicates are removed from the transcript;

- At the end of $\tilde{\tau}$, a tuple containing the keys $(K_1, \ldots, K_\mu)$ is added.

This transcript is released at the end of the interaction, right before the distinguisher outputs its decision bit.

Bad Events. We introduce the following bad events:

$$\begin{aligned}
\mathsf{ColK}_{\mathsf{a}}^{\mathsf{lr}} &: \mathsf{ColK}_{\mathsf{a}}^{\mathsf{m}} \text{ (of Section 9.6.1)}\,; \qquad \mathsf{GueK}_{\mathsf{a}}^{\mathsf{lr}} : \mathsf{GueK}_{\mathsf{a}}^{\mathsf{m}} \text{ (of Section 9.6.1)}\,;\\
\mathsf{Dec}_{\mathsf{a}}^{\mathsf{lr}} &: \exists (D_C, m, N, A, C, T, \tilde{P}) \in \tilde{\tau} \text{ such that}\\
&\quad \text{either } \mathrm{inner}_t(\mathtt{S}_{\mathrm{out}}[(D_C, m, N, A, C, 1)]) \oplus \mathrm{inner}_t(K_m) = T\,,\\
&\quad \text{or } \exists \mathtt{path} \in \mathtt{P} \text{ superseding } (D_C, m, N, A, C, 1)\\
&\quad \text{with } \mathrm{inner}_t(\mathtt{S}_{\mathrm{out}}[\mathtt{path}]) \oplus \mathrm{inner}_t(K_m) = T\,;\\
\mathsf{ColS}_{\mathsf{a}}^{\mathsf{lr}} &: \exists^{\neq} \mathtt{path} \in \mathtt{P}_C \setminus \mathtt{P}_{\mathrm{S}}, \mathtt{path}' \in \mathtt{P} \setminus \mathtt{P}_{\mathrm{S}}\\
&\quad \text{such that } \mathtt{S}_{\mathrm{in}}[\mathtt{path}] \overset{c}{=} \mathtt{S}_{\mathrm{in}}[\mathtt{path}'] \text{ or } \mathtt{S}_{\mathrm{out}}[\mathtt{path}] \overset{c}{=} \mathtt{S}_{\mathrm{out}}[\mathtt{path}']\,;\\
\mathsf{GueS}_{\mathsf{a}}^{\mathsf{lr}} &: \exists (X, Y, d) \in \tilde{\tau}, \mathtt{path} \in \mathtt{P} \setminus \mathtt{P}_{\mathrm{S}} \text{ such that } X \overset{c}{=} \mathtt{S}_{\mathrm{in}}[\mathtt{path}] \text{ or } Y \overset{c}{=} \mathtt{S}_{\mathrm{out}}[\mathtt{path}]\,;\\
\mathsf{BAD}_{\mathsf{a}}^{\mathsf{lr}} &: \mathsf{ColK}_{\mathsf{a}}^{\mathsf{lr}} \vee \mathsf{GueK}_{\mathsf{a}}^{\mathsf{lr}} \vee \mathsf{ColS}_{\mathsf{a}}^{\mathsf{lr}} \vee \mathsf{GueS}_{\mathsf{a}}^{\mathsf{lr}} \vee \mathsf{Dec}_{\mathsf{a}}^{\mathsf{lr}}\,.
\end{aligned}$$

Compared to the bad events from Section 9.6.1, $\mathsf{ColS}_{\mathsf{a}}^{\mathsf{lr}}$ now only considers collisions involving a challenge intermediate state. Moreover, $\mathsf{GueS}_{\mathsf{a}}^{\mathsf{lr}}$ has been adjusted to account for the fact that not all intermediate states are added to the set $\mathtt{P}$. Finally, $\mathsf{Dec}_{\mathsf{a}}^{\mathsf{lr}}$ accounts for the fact that the decryption queries are clearly distinguished as challenge queries, though this adjustment is purely administrative.

Probability of Good Transcripts. The absence of $\mathsf{BAD}_{\mathsf{a}}^{\mathsf{lr}}$ prevents the ideal world from generating states which do not adhere to the structure of the Ascon-AE mode. In addition, the absence of $\mathsf{BAD}_{\mathsf{a}}^{\mathsf{lr}}$ (or, more precisely, of $\mathsf{Dec}_{\mathsf{a}}^{\mathsf{lr}}$) in the real world prevents the challenge decryption oracle from returning a string $\tilde{P}$ different from $\perp$. Therefore, every good transcript which is reachable in the real world is also reachable in the ideal world, and vice-versa. Let $\tilde{\tau}$ be a good transcript. Such a transcript induces a certain number of permutation evaluations from the queries to the leaky oracles and the permutation queries. These evaluations do not overlap with evaluations from the challenge oracles, and the count is identical in both the real and ideal worlds. Moreover, in the real world, the challenge queries induce $\mathcal{M}_{D,C} + \mathcal{M}_{E,C} - |\mathtt{P}_{\mathrm{S}}|$ additional permutation evaluations, while in the ideal world this induces $\mathcal{M}_{E,C}$ additional permutation evaluations and $\mathcal{M}_{D,C} - |\mathtt{P}_{\mathrm{S}}|$ random b-bit states. Therefore, similarly to Section 9.6.1, we have

$$\frac{\mathbf{Pr}\left(\mathcal{A}\left[W_R\right] \text{ generates } \tilde{\tau}\right)}{\mathbf{Pr}\left(\mathcal{A}\left[W_I\right] \text{ generates } \tilde{\tau}\right)} \geq 1\,. \tag{9.21}$$

Probability of $\mathsf{BAD}_{\mathsf{a}}^{\mathsf{lr}}$ in the Ideal World. We will again use a query-wise approach to evaluate the probability to set $\mathsf{BAD}_{\mathsf{a}}^{\mathsf{lr}}$. The number of permutation evaluations done during the interactive phase is at most $\mathcal{M}_E + \mathcal{M}_{D,L} + \mathcal{N}$. Let $i \in [\![0, \mathcal{M}_E + \mathcal{M}_{D,L} + \mathcal{N}]\!]$. We will re-use the notation $\eta_{\texttt{enc,dec}}[i]$ from Section 9.6.1. Since part of the leaky evaluations is treated the same way as permutation queries, we refine the indicator functions $\mathbf{1}_{\texttt{C}}[i]$ and $\mathbf{1}_{\texttt{P}}[i]$ into the following functions:

- $\mathbf{1}_{\texttt{P,CL}}[i]$ is equal to 1 if and only if the evaluation number i is fresh, and (i) either the evaluation is from a direct permutation query, or (ii) the evaluation comes from a leaky construction evaluation. `CL` stands for "construction leaky". The total number of is setting this function to 1 is the number of tuples (X, Y, d) in the extended transcript, thus at most $\mathcal{N} + \mathcal{M}_L - Q_L$;

- $\mathbf{1}_{\texttt{C,KB}}[i]$ is equal to 1 if and only if the evaluation number i is fresh, and comes from a construction evaluation during the leftmost or rightmost permutation evaluation. $\mathbf{1}_{\texttt{C,KB}}[i]$ is itself refined into $\mathbf{1}_{\texttt{C,KBI}}[i]$ and $\mathbf{1}_{\texttt{C,KBF}}[i]$, for respectively the initial and the final evaluation. Each of these functions is set to 1 by at most $Q_E + Q_{D,L}$ different indexes. `KBI` and `KBF` stand for respectively "initial key blinding" and "final key blinding";

- $\mathbf{1}_{\texttt{CH}}[i]$ is equal to 1 if and only if the permutation evaluation number i is fresh, and originates from an internal state generated during a challenge permutation query (excluding thus the key blindings). Thanks to the nonce-respecting setting, the number of is that set this function to 1 is at most $\mathcal{M}_{E,C} - 2Q_{E,C}$.

We will derive two distinct bounds, with the second involving an additional auxiliary event, which helps to manage inner collisions. Depending on the adversarial resources, the tighter of the two bounds will apply.

Probability of $\mathsf{BAD}_{\mathsf{a}}^{\mathsf{lr}}$ in the Ideal World, First Bound. In this bounding, we evaluate $\mathsf{BAD1}_{\mathsf{a}}^{\mathsf{lr}} := \mathsf{BAD}_{\mathsf{a}}^{\mathsf{lr}}$, without an additional auxiliary event. We break down the probability of $\mathsf{BAD1}_{\mathsf{a}}^{\mathsf{lr}}$ by using basic probability as follows:

1. $\mathsf{BAD1}_{\mathsf{a}}^{\mathsf{lr}}[0]$, or $\mathsf{ColK}_{\mathsf{a}}^{\mathsf{lr}}$ by definition;
2. $\mathsf{BAD1}_{\mathsf{a}}^{\mathsf{lr}}[i]$ for $i \in [\![1, \mathcal{M}_E + \mathcal{M}_{D,L} + \mathcal{N}]\!]$, in more detail:
   (a) $\mathsf{GueK}_{\mathsf{a}}^{\mathsf{lr}}[i]$, assuming $\neg\mathsf{BAD1}_{\mathsf{a}}^{\mathsf{lr}}[i-1]$;
   (b) $\mathsf{GueS}_{\mathsf{a}}^{\mathsf{lr}}[i]$, assuming $\neg\mathsf{BAD1}_{\mathsf{a}}^{\mathsf{lr}}[i-1] \wedge \neg\mathsf{GueK}_{\mathsf{a}}^{\mathsf{lr}}[i]$;

(c) $\mathsf{ColS}^{\mathsf{lr}}_{\mathsf{a}}[i]$, assuming $\neg\mathsf{BAD1}^{\mathsf{lr}}_{\mathsf{a}}[i-1] \wedge \neg\mathsf{GueK}^{\mathsf{lr}}_{\mathsf{a}}[i] \wedge \neg\mathsf{GueS}^{\mathsf{lr}}_{\mathsf{a}}[i]$;

(d) $\mathsf{Dec}^{\mathsf{lr}}_{\mathsf{a}}[i]$, assuming $\neg\mathsf{BAD1}^{\mathsf{lr}}_{\mathsf{a}}[i-1] \wedge \neg\mathsf{GueK}^{\mathsf{lr}}_{\mathsf{a}}[i] \wedge \neg\mathsf{ColS}^{\mathsf{lr}}_{\mathsf{a}}[i] \wedge \neg\mathsf{GueS}^{\mathsf{lr}}_{\mathsf{a}}[i]$;

3. $\mathsf{BAD1}^{\mathsf{lr}}_{\mathsf{a}}$ at the end of the interaction, which is equivalent to $\mathsf{BAD1}^{\mathsf{lr}}_{\mathsf{a}}$ assuming $\neg\mathsf{BAD1}^{\mathsf{lr}}_{\mathsf{a}}[\mathcal{M}_E + \mathcal{M}_{D,L} + \mathcal{N}]$.

Cases 1, 2a, 2d, and 3 can be upper bounded the same way as in Section 9.6.1, so that

$$\mathbf{Pr}\left(\mathsf{BAD1}^{\mathsf{lr}}_{\mathsf{a}}[0]\right) \leq \frac{\mu(\mu-1)}{2^{k+1}}, \tag{9.22}$$

$$\mathbf{Pr}\left(\mathsf{GueK}^{\mathsf{lr}}_{\mathsf{a}}[i] \mid \neg\mathsf{BAD1}^{\mathsf{lr}}_{\mathsf{a}}[i-1]\right) \leq \left(\mathbf{1}_{\mathrm{P,CL}}[i] + \mathbf{1}_{\mathrm{CH}}[i] + \mathbf{1}_{\mathrm{C,KBF}}[i]\right)\frac{2\mu}{2^k}, \tag{9.23}$$

$$\begin{aligned}\mathbf{Pr}\left(\mathsf{Dec}^{\mathsf{lr}}_{\mathsf{a}}[i] \mid \neg\mathsf{BAD1}^{\mathsf{lr}}_{\mathsf{a}}[i-1] \wedge \neg\mathsf{GueK}^{\mathsf{lr}}_{\mathsf{a}}[i] \wedge \neg\mathsf{ColS}^{\mathsf{lr}}_{\mathsf{a}}[i] \wedge \neg\mathsf{GueS}^{\mathsf{lr}}_{\mathsf{a}}[i]\right)&\\ \leq \eta_{\mathtt{enc,dec}}[i]\frac{1}{2^t}&,\end{aligned} \tag{9.24}$$

$$\begin{aligned}\mathbf{Pr}\left(\mathsf{BAD1}^{\mathsf{lr}}_{\mathsf{a}} \mid \neg\mathsf{BAD1}^{\mathsf{lr}}_{\mathsf{a}}[\mathcal{M}_E + \mathcal{M}_{D,L} + \mathcal{N}]\right) &\leq \frac{2\mu\mathcal{M}_{D,C}}{2^k} + \frac{Q_{D,C}}{2^t}\\ &+ \frac{2\mathcal{M}_{D,C}\left(\mathcal{M}+\mathcal{N}\right)}{2^c}.\end{aligned} \tag{9.25}$$

where we used that $\mu\left(\mathcal{N}+\mathcal{M}\right) \leq 2^{k-1}$. Now, we have two cases left.

Case 2b. We focus on the conditioned $\mathsf{GueS}^{\mathsf{lr}}_{\mathsf{a}}[i]$. If the evaluation is from a permutation query or a leaky internal state evaluation, it can target either an input/output of a leftmost/rightmost permutation call, or a challenge intermediate state. For challenge intermediate states, using that the inner part of challenge intermediate states is secret and hidden from the adversary, we obtain a probability of at most

$$\mathbf{1}_{\mathrm{P,CL}}[i]\frac{4\mathcal{M}_{E,C}}{2^c},$$

where we used that $\mathcal{M}+\mathcal{N} \leq 2^{b-1}$. For the key blinding input/outputs, the adversary has, in the best case, access to the input of the evaluation before the key additions and the output after the key additions. It therefore remains to guess the state after key addition.[9] There are 3 different places where the key blinding is applied,[10] thus in total at most $3(Q_E + Q_{D,L})$ states to be guessed. Therefore, this event is set with a probability of at most

$$\mathbf{1}_{\mathrm{P,CL}}[i]\frac{6\left(Q_E+Q_{D,L}\right)}{2^k},$$

[9]For the output of the rightmost key blinding, we take a lossy step here.

[10]Note that due to the absence of $\mathsf{GueK}^{\mathsf{lr}}_{\mathsf{a}}[i]$, the event can never be set with the input of the leftmost permutation call.

where we used that $Q(\mathcal{M}+\mathcal{N}) \leq 2^{k-1}$. Else, if the evaluation i is from an internal state evaluation due to a challenge query, we obtain a probability of at most

$$\mathbf{1}_{\texttt{CH}}[i]\frac{4(\mathcal{N}+\mathcal{M}_L-Q_L)}{2^c},$$

where we used that $\mathcal{M}+\mathcal{N} \leq 2^{b-1}$. Else, if the evaluation i is from a construction query during the leftmost or rightmost evaluation, we obtain a probability of at most

$$\mathbf{1}_{\texttt{C,KB}}[i]\frac{4(\mathcal{N}+\mathcal{M}_L-Q_L)}{2^k},$$

where we used that $Q(\mathcal{M}+\mathcal{N}) \leq \mu 2^{k-1}$. Therefore,

$$\mathbf{Pr}\left(\mathsf{GueS}_{\mathsf{a}}^{\mathsf{lr}}[i] \mid \neg\mathsf{BAD1}_{\mathsf{a}}^{\mathsf{lr}}[i-1] \wedge \neg\mathsf{GueK}_{\mathsf{a}}^{\mathsf{lr}}[i]\right) \leq \mathbf{1}_{\texttt{P,CL}}[i]\frac{4\mathcal{M}_{E,C}}{2^c}$$
$$+\mathbf{1}_{\texttt{CH}}[i]\frac{4(\mathcal{N}+\mathcal{M}_L-Q_L)}{2^c}+\mathbf{1}_{\texttt{P,CL}}[i]\frac{6(Q_E+Q_{D,L})}{2^k}+\mathbf{1}_{\texttt{C,KB}}[i]\frac{4(\mathcal{N}+\mathcal{M}_L-Q_L)}{2^k}. \tag{9.26}$$

Case 2c. We focus on the conditioned $\mathsf{ColS}_{\mathsf{a}}^{\mathsf{lr}}[i]$. With this event, only collisions with a challenge intermediate state matter. Those states have a secret inner part, and can be evaluated similarly to the conditioned $\mathsf{GueS}_{\mathsf{a}}^{\mathsf{lr}}[i]$. Therefore,

$$\mathbf{Pr}\left(\mathsf{ColS}_{\mathsf{a}}^{\mathsf{lr}}[i] \mid \neg\mathsf{BAD1}_{\mathsf{a}}^{\mathsf{lr}}[i-1] \wedge \neg\mathsf{ColKS}_{\mathsf{a}}^{\mathsf{lr}}[i] \wedge \neg\mathsf{GueS}_{\mathsf{a}}^{\mathsf{lr}}[i]\right)$$
$$\leq \mathbf{1}_{\texttt{C,KB}}[i]\frac{4\mathcal{M}_{E,C}}{2^c}+\mathbf{1}_{\texttt{CH}}[i]\left(\frac{6Q_L}{2^c}+\frac{4\mathcal{M}_{E,C}}{2^c}\right), \tag{9.27}$$

where we used $\mathcal{M}+\mathcal{N} \leq 2^{b-1}$.

Combining (9.22) to (9.27), and simplifying the bounds with constant factor losses, we obtain

$$\mathbf{Pr}\left(\mathsf{BAD1}_{\mathsf{a}}^{\mathsf{lr}}\right) \leq \frac{\mu(\mu-1)}{2^{k+1}}+\frac{2\mu(\mathcal{N}+\mathcal{M})}{2^k}+\frac{14Q(\mathcal{N}+\mathcal{M})}{2^k}$$
$$+\frac{2Q_D}{2^t}+\frac{18\mathcal{M}(\mathcal{M}+\mathcal{N})}{2^c}. \tag{9.28}$$

Probability of $\mathsf{BAD}_{\mathsf{a}}^{\mathsf{lr}}$ in the Ideal World, Second Bound. We introduce the following auxiliary bad event $\mathsf{Inner}_{\mathsf{a}}^{\mathsf{lr}}$, set whenever there exists $i \in [\![1, \mathcal{M}_E + \mathcal{M}_{D,L} + \mathcal{N}]\!]$ such that the output of the evaluation i collides on its inner part with a prior permutation evaluation input or output.

Let $\mathsf{BAD2}_{\mathsf{a}}^{\mathsf{lr}} = \mathsf{BAD}_{\mathsf{a}}^{\mathsf{lr}} \vee \mathsf{Inner}_{\mathsf{a}}^{\mathsf{lr}}$, we will evaluate the probability of $\mathsf{BAD2}_{\mathsf{a}}^{\mathsf{lr}}$. Let $i \in [\![0, \mathcal{M}_E + \mathcal{M}_{D,L} + \mathcal{N}]\!]$. We introduce the following random variables:

$$\begin{aligned}\Theta_{\mathrm{KBI}}[i] = \max_{Z \in \{0,1\}^{c-k}} \Big|\Big\{ &\mathtt{path} = (m, N, \epsilon, \epsilon, 0) \in \mathtt{P}[1:i] \mid \exists \mathcal{P}\left(\mathtt{S}_{\mathrm{in}}[\mathtt{path}]\right) \\ &\text{in forward direction such that } \mathtt{S}_{\mathrm{out}}[\mathtt{path}][c+1 : c-k] = Z \Big\}\Big| ,\\ \Theta_{\mathrm{KBF}}[i] = \max_{Z \in \{0,1\}^{c-k}} \Bigg|\Big\{ &(X, Y, d) \in \tilde{\tau}[1:i] \mid d \in \{\mathit{fwd}, \mathrm{cons}\} \text{ and } \mathrm{inner}_{c-k}(Y) = Z \Big\} \\ &\cup \Big\{\mathtt{path} = (m, N, \epsilon, \epsilon, 0) \in \mathtt{P}[1:i] \mid \exists \mathcal{P}\left(\mathtt{S}_{\mathrm{in}}[\mathtt{path}]\right) \\ &\text{in forward direction such that } \mathrm{inner}_{c-k}(\mathtt{S}_{\mathrm{out}}[\mathtt{path}]) = Z \Big\}\Bigg| ,\end{aligned}$$

where we abuse notation with $\mathtt{P}[1:i]$ to denote the set of path descendants of the queries made up to and including the evaluation number i.

Moreover, let Θ_{KBI} and Θ_{KBF} denote respectively $\Theta_{\mathrm{KBI}}[\mathcal{N} + \mathcal{M}_E + \mathcal{M}_{D,L}]$ and $\Theta_{\mathrm{KBF}}[\mathcal{N} + \mathcal{M}_E + \mathcal{M}_{D,L}]$. The variable Θ_{KBI} counts the maximal size of multicollisions on the middle $c - k$ bits made from the first permutation evaluation in the construction queries. The variable Θ_{KBF} counts the maximal size of multicollisions on the lowest $c - k$ bits, over (almost) all permutation evaluations made in the forward direction. It is constructed from two sets: one that gathers the states generated from challenge queries or construction queries with a single plaintext/ciphertext block, and another one that gathers forward permutation queries and leaky permutation evaluations. All forward permutation evaluations are included since they can potentially be presented as input to the final permutation evaluation. Looking ahead, since $\mathsf{Inner}_{\mathsf{a}}^{\mathsf{lr}}$ is included in the bad events, this means that, as long as $\mathsf{BAD2}_{\mathsf{a}}^{\mathsf{lr}}$ does not occur, the internal states generated from construction queries are only from forward permutation evaluations, and we can use Θ_{KBI} and Θ_{KBF} to tame multicollisions. We thus have

$$\mathsf{E}\left(\Theta_{\mathrm{KBI}}\right) \le \mathrm{mucol}(Q, 2^{c-k}) , \qquad \mathsf{E}\left(\Theta_{\mathrm{KBF}}\right) \le \mathrm{mucol}(\mathcal{M} + \mathcal{N}, 2^{c-k}) . \quad (9.29)$$

We will evaluate the probability of $\mathsf{BAD2}_{\mathsf{a}}^{\mathsf{lr}}$. Again, we break down the probability of $\mathsf{BAD2}_{\mathsf{a}}^{\mathsf{lr}}$ using basic probabilities as follows:

1. $\mathsf{BAD2}_{\mathsf{a}}^{\mathsf{lr}}[0]$: same bounding as (9.22);
2. $\mathsf{BAD2}_{\mathsf{a}}^{\mathsf{lr}}[i]$, for all $i \in [\![1, \mathcal{N} + \mathcal{M}_E + \mathcal{M}_{D,L}]\!]$, we evaluate:

(a) $\mathsf{GueK}_\mathsf{a}^\mathsf{lr}[i]$, assuming $\neg\mathsf{BAD2}_\mathsf{a}^\mathsf{lr}[i-1]$: same bounding as (9.23);

(b) $\mathsf{Inner}_\mathsf{a}^\mathsf{lr}[i]$, assuming $\neg\mathsf{BAD2}_\mathsf{a}^\mathsf{lr}[i-1] \wedge \neg\mathsf{GueK}_\mathsf{a}^\mathsf{lr}[i] \wedge \neg\mathsf{ColKS}_\mathsf{a}^\mathsf{lr}[i]$: the reasoning and bounding is done later in (9.30);

(c) $\mathsf{GueS}_\mathsf{a}^\mathsf{lr}[i] \wedge \neg\mathsf{BAD2}_\mathsf{a}^\mathsf{lr}[i-1] \wedge \neg\mathsf{Inner}_\mathsf{a}^\mathsf{lr}[i] \wedge \neg\mathsf{GueK}_\mathsf{a}^\mathsf{lr}[i]$: the reasoning and bounding is done later in (9.32);

(d) $\mathsf{ColS}_\mathsf{a}^\mathsf{lr}[i]$, assuming $\neg\mathsf{BAD2}_\mathsf{a}^\mathsf{lr}[i-1] \wedge \neg\mathsf{GueK}_\mathsf{a}^\mathsf{lr}[i] \wedge \neg\mathsf{Inner}_\mathsf{a}^\mathsf{lr}[i] \wedge \neg\mathsf{GueS}_\mathsf{a}^\mathsf{lr}[i]$: same bounding as (9.27);

(e) $\mathsf{Dec}_\mathsf{a}^\mathsf{lr}[i]$, assuming $\neg\mathsf{BAD2}_\mathsf{a}^\mathsf{lr}[i-1] \wedge \neg\mathsf{GueK}_\mathsf{a}^\mathsf{lr}[i] \wedge \neg\mathsf{Inner}_\mathsf{a}^\mathsf{lr}[i] \wedge \neg\mathsf{ColS}_\mathsf{a}^\mathsf{lr}[i] \wedge \neg\mathsf{GueS}_\mathsf{a}^\mathsf{lr}[i]$: same bounding as (9.24);

3. $\mathsf{BAD2}_\mathsf{a}^\mathsf{lr}$ at the end of the interaction, which is equivalent to $\mathsf{BAD2}_\mathsf{a}^\mathsf{lr}$ assuming $\neg\mathsf{BAD2}_\mathsf{a}^\mathsf{lr}[\mathcal{N} + \mathcal{M}_E + \mathcal{M}_{D,L}]$: $\mathsf{Inner}_\mathsf{a}^\mathsf{lr}$ cannot be set at the end of the interaction, and the same bound as (9.25) can be derived.

We are left with two cases.

Case 2b. We start to evaluate the conditioned $\mathsf{Inner}_\mathsf{a}^\mathsf{lr}[i]$. This corresponds to the probability that a b-bit string generated in a permutation-consistent way collides on its inner part with another b-bit string:

$$\mathbf{Pr}\left(\mathsf{Inner}_\mathsf{a}^\mathsf{lr}[i] \mid \neg\mathsf{BAD2}_\mathsf{a}^\mathsf{lr}[i-1] \wedge \neg\mathsf{GueK}_\mathsf{a}^\mathsf{lr}[i] \wedge \neg\mathsf{ColKS}_\mathsf{a}^\mathsf{lr}[i]\right) \leq \left(\mathbf{1}_{\mathsf{P,CL}}[i] + \mathbf{1}_{\mathsf{C,KB}}[i]\right) \frac{4(i-1)}{2^c}\,, \quad (9.30)$$

where we used that $\mathcal{M} + \mathcal{N} \leq 2^{b-1}$.

Case 2c. We then evaluate the event $\mathsf{GueS}_\mathsf{a}^\mathsf{lr}[i] \wedge \neg\mathsf{BAD2}_\mathsf{a}^\mathsf{lr}[i-1] \wedge \neg\mathsf{Inner}_\mathsf{a}^\mathsf{lr}[i] \wedge \neg\mathsf{GueK}_\mathsf{a}^\mathsf{lr}[i]$. We will introduce the multicollision random variables Θ_{KBI} and Θ_{KBF} only during the sub-case that requires it, which is the reason why we did not switch to conditioned probabilities directly. If the i^{th} evaluation is from a permutation query or a leaky internal state evaluation, we upper bound the probability that the evaluation guesses correctly one of the following states:

- A state that is input of the leftmost key blinding: this case has already been handled with the bad event $\mathsf{GueK}_\mathsf{a}^\mathsf{lr}[i]$;
- A state that is output of the leftmost key blinding: the states in question are XORed with the keys before being leaked, and we can use multicollisions on the middle $c-k$ bits, since the key blinding evaluations

are made in the forward direction. Let $\theta_{\mathrm{KBI}} \in \mathbb{N}$, then conditioned on $\Theta_{\mathrm{KBI}}[i] = \theta_{\mathrm{KBI}}$, this event is set with probability at most

$$\mathbf{1}_{\mathtt{P,CL}}[i]\frac{2\theta_{\mathrm{KBI}}}{2^k} \ ;$$

- A state that is input of the rightmost key blinding: the same reasoning applies here. Let $\theta_{\mathrm{KBF}} \in \mathbb{N}$, then conditioned on $\Theta_{\mathrm{KBF}}[i] = \theta_{\mathrm{KBF}}$, this event is set with probability at most

$$\mathbf{1}_{\mathtt{P,CL}}[i]\frac{2\theta_{\mathrm{KBF}}}{2^k} \ ;$$

- A challenge intermediate state, or output of the rightmost key blinding: the states in question are sampled uniformly and remain secret from the adversary. Therefore, this event is set with probability at most

$$\mathbf{1}_{\mathtt{P,CL}}[i]\frac{4\left(\mathcal{M}_{E,C} - Q_{E,C} + Q_L\right)}{2^c} \ .$$

On the other hand, if the i^{th} evaluation is from an internal evaluation due to a challenge query, the event is set with a probability of at most

$$\mathbf{1}_{\mathtt{CH}}[i]\frac{4\left(\mathcal{N} + \mathcal{M}_L - Q_{D,L}\right)}{2^c} \ .$$

If the i^{th} evaluation corresponds to a construction evaluation and is the output of the leftmost or rightmost permutation evaluation, then the output state is once again sampled in a permutation-consistent manner, and the event occurs with a probability of at most

$$\mathbf{1}_{\mathtt{C,KB}}[i]\frac{4\left(\mathcal{N} + \mathcal{M}_L - Q_{D,L}\right)}{2^c} \ .$$

We next consider the case when the i^{th} evaluation is from a construction evaluation, and the event is evaluated on the rightmost permutation input. Then, in the best case, the adversary can choose among a certain set of states with full control on their outer parts, and then the key addition is applied on the middle k bits of the chosen state S. For all existing $(X, Y, d) \in \tilde{\tau}[1 : i-1]$, the only candidates to set this event must have their outer part and rightmost $c-k$ bits set to those of S. Let $Z \in \{0,1\}^{b-k}$, and define C_Z as follows:

$$C_Z = \{(X, Y, d) \in \tilde{\tau} \mid \mathrm{outer}_r(X)\|\mathrm{inner}_{c-k}(X) = Z\} \ .$$

C_Z counts the number of $(X, Y, d) \in \tilde{\tau}$ such that X has its top r bits concatenated with its bottom $c-k$ bit fixed to Z. By the absence of $\mathsf{Inner}_{\mathsf{a}}^{\mathsf{lr}}[i]$, the state S was obtained only from forward permutation evaluations, so that multicollisions can be used. Let $\theta_{\text{KBF}} \in \mathbb{N}$. We define $\mathbf{1}_{\texttt{C,KB}_\texttt{F},Z}[i \mid \theta_{\text{KBF}}]$ to be the indicator function equal to 1 if and only if, conditioned on $\Theta_{\text{KBF}} = \theta_{\text{KBF}}$, the evaluation number i is a rightmost key blinding, has it top r bits concatenated with its bottom $c-k$ bits equal to Z. Conditioned on $\Theta_{\text{KBF}} = \theta_{\text{KBF}}$, this event is set with a probability of at most

$$\sum_{Z\in\{0,1\}^{b-k}} \mathbf{1}_{\texttt{KB}_\texttt{f},Z}[i \mid \theta_{\text{KBF}}]\frac{2C_Z}{2^k}\,.$$

Looking ahead, when summing over all queries and all possible values for multicollisions, we will have a term of the form:

$$\begin{aligned}
&\sum_{i}\sum_{\theta_{\text{KBF}}}\sum_{Z} \mathbf{1}_{\texttt{KB}_\texttt{f},Z}[i \mid \theta_{\text{KBF}}]\frac{2C_Z}{2^k}\mathbf{Pr}\left(\Theta_{\text{KBF}}[i] = \theta_{\text{KBF}}\right)\\
&= \sum_{Z}\frac{2C_Z}{2^k}\sum_{\theta_{\text{KBF}}}\mathbf{Pr}\left(\Theta_{\text{KBF}} = \theta_{\text{KBF}}\right)\sum_{i}\mathbf{1}_{\texttt{C,KB}_\texttt{F},Z}[i \mid \theta_{\text{KBF}}]\\
&\le \sum_{Z}\frac{2C_Z}{2^k}\sum_{\theta_{\text{KBF}}}\mathbf{Pr}\left(\Theta_{\text{KBF}} = \theta_{\text{KBF}}\right)\cdot\theta_{\text{KBF}}\\
&\le \frac{2(\mathcal{N}+\mathcal{M})\cdot \text{mucol}(\mathcal{N}+\mathcal{M}, 2^{c-k})}{2^k}\,, \qquad (9.31)
\end{aligned}$$

where we used (9.29). Therefore,

$$\begin{aligned}
&\mathbf{Pr}\left(\mathsf{GueS}_{\mathsf{a}}^{\mathsf{lr}}[i] \wedge \neg\mathsf{BAD2}_{\mathsf{a}}^{\mathsf{lr}}[i-1] \wedge \neg\mathsf{GueK}_{\mathsf{a}}^{\mathsf{lr}}[i] \wedge \neg\mathsf{Inner}_{\mathsf{a}}^{\mathsf{lr}}[i]\right)\\
&\le \mathbf{1}_{\texttt{P,CL}}[i]\frac{2\left(\mathcal{M}_{E,C} - Q_{E,C} + Q_L\right)}{2^c} + \left(\mathbf{1}_{\texttt{CH}}[i] + \mathbf{1}_{\texttt{C,KBI}}[i]\right)\frac{2\left(\mathcal{N}+\mathcal{M}_L\right)}{2^c}\\
&+ \sum_{\theta_{\text{KBF}}\in\mathbb{N}}\left(\mathbf{1}_{\texttt{P,CL}}[i]\frac{2\theta_{\text{KBF}}}{2^k} + \sum_{Z\in\{0,1\}^{b-k}}\mathbf{1}_{\texttt{C,KB}_\texttt{F},Z}[i \mid \theta_{\text{KBF}}]\frac{2C_Z}{2^k}\right)\cdot\mathbf{Pr}\left(\Theta_{\text{KBF}} = \theta_{\text{KBF}}\right)\\
&+ \sum_{\theta_{\text{KBI}}\in\mathbb{N}}\mathbf{1}_{\texttt{P,CL}}[i]\frac{2\theta_{\text{KBI}}}{2^k}\mathbf{Pr}\left(\Theta_{\text{KBI}} = \theta_{\text{KBI}}\right)\\
&\le \mathbf{1}_{\texttt{P,CL}}[i]\frac{2\left(\mathcal{M}_{E,C} - Q_{E,C} + Q_L\right)}{2^c} + \left(\mathbf{1}_{\texttt{CH}}[i] + \mathbf{1}_{\texttt{C,KBI}}[i]\right)\frac{2\left(\mathcal{N}+\mathcal{M}_L\right)}{2^c}\\
&+ \mathbf{1}_{\texttt{P,CL}}[i]\frac{2\text{mucol}(\mathcal{N}+\mathcal{M}, 2^{c-k})}{2^k} + \mathbf{1}_{\texttt{P,CL}}[i]\frac{2\text{mucol}(Q, 2^{c-k})}{2^k}
\end{aligned}$$

$$+ \sum_{\theta_{\mathrm{KBF}} \in \mathbb{N}} \mathbf{Pr}\left(\Theta_{\mathrm{KBF}} = \theta_{\mathrm{KBF}}\right) \sum_{Z \in \{0,1\}^{b-k}} \mathbf{1}_{\mathsf{C},\mathsf{KB}_\mathsf{F},Z}[i \mid \theta_{\mathrm{KBF}}] \frac{2C_Z}{2^k} \,. \tag{9.32}$$

Combining (9.22) to (9.25), (9.27) and (9.30) to (9.32), we obtain

$$\begin{aligned} \mathbf{Pr}\left(\mathsf{BAD2}^{\mathsf{lr}}_{\mathsf{a}}\right) \leq{}& \frac{\mu(\mu-1)}{2^{k+1}} + \frac{2\mu\left(\mathcal{N}+\mathcal{M}\right)}{2^k} + \frac{2Q_D}{2^t} + \frac{18\mathcal{M}\left(\mathcal{M}+\mathcal{N}\right)}{2^c} \\ &+ \frac{2\left(\mathcal{M}+\mathcal{N}\right)^2}{2^c} + \frac{6(\mathcal{N}+\mathcal{M}) \cdot \mathrm{mucol}(\mathcal{N}+\mathcal{M}, 2^{c-k})}{2^k} \,. \end{aligned} \tag{9.33}$$

Since both bounds (9.28) and (9.33) are valid upper bounds of the probability of $\mathsf{BAD}^{\mathsf{lr}}_{\mathsf{a}}$, we take the minimum of these two bounds and obtain

$$\begin{aligned} \mathbf{Pr}\left(\mathsf{BAD}^{\mathsf{lr}}_{\mathsf{a}}\right) &\leq \min\left\{\mathbf{Pr}\left(\mathsf{BAD1}^{\mathsf{lr}}_{\mathsf{a}}\right), \mathbf{Pr}\left(\mathsf{BAD2}^{\mathsf{lr}}_{\mathsf{a}}\right)\right\} \\ &\leq \frac{\mu(\mu-1)}{2^{k+1}} + \frac{2\mu\left(\mathcal{N}+\mathcal{M}\right)}{2^k} + \frac{2Q_D}{2^t} + \frac{18\mathcal{M}\left(\mathcal{M}+\mathcal{N}\right)}{2^c} \\ &+ \min\left\{\frac{14Q\left(\mathcal{N}+\mathcal{M}\right)}{2^k}, \frac{4\left(\mathcal{M}+\mathcal{N}\right)^2}{2^c} + \frac{6(\mathcal{N}+\mathcal{M}) \cdot \mathrm{mucol}(\mathcal{N}+\mathcal{M}, 2^{c-k})}{2^k}\right\} . \end{aligned}$$

We obtained an upper bound for the probability that the ideal world generates a bad transcript. Using the H-coefficient technique, and the fact that the ratio of good transcript is lower-bounded by one (cf., (9.21)), we conclude. □

9.6.3.2 Confidentiality

Let $\mathcal{A}$ be an adversary that makes at most $\mathcal{N}$ permutation queries, Q_E encryption queries of at most $\mathcal{M}_E$ blocks, and Q_D decryption queries of at most $\mathcal{M}_D$ blocks, as in the theorem statement. Our goal is to upper bound $\mathbf{Adv}^{\mu\text{-lr-conf}}_{\mathrm{Ascon},\mathcal{L}}(\mathcal{A})$, for any set of leakage functions that do not leak any information about the two outer permutation calls during the initialization and finalization phases. The adversary interacts either with the *real world* W_R, which gives access to

$$\left[\left(\left[\mathbf{Enc}^{\mathcal{P}}_{K_m}\right]_L, \mathbf{Enc}^{\mathcal{P}}_{K_m}, \left[\mathbf{Dec}^{\mathcal{P}}_{K_m}\right]_L\right)^{\mu}_{m=1}, \mathcal{P}^{\pm}\right] ,$$

or with the *ideal world* W_I, which gives access to

$$\left[\left(\left[\mathbf{Enc}^{\mathcal{P}}_{K_m}\right]_L, \$_m, \left[\mathbf{Dec}^{\mathcal{P}}_{K_m}\right]_L\right)^{\mu}_{m=1}, \mathcal{P}^{\pm}\right] ,$$

where the adversary must be nonce-respecting with its queries to $\mathcal{O}_{2,m}$.

Similarly to the fact that the nonce-misuse authenticity proof can be used for bounding confidentiality in the nonce-misuse resilience setting, the proof of leakage resilience authenticity can be reused for leakage resilience confidentiality. We adopt the same terminology as in Section 9.6.3.1, with $\mathcal{O}_{1,m}$ and $\mathcal{O}_{3,m}$ will be referred to as *leaky oracles*, and $\mathcal{O}_{2,m}$ and $\mathcal{O}_{4,m}$ referred to as *challenge oracles.*

Setup. We adopt the same transcript and path notation as in Section 9.6.3.1, with the difference that there are no challenge decryption queries, which thus do not appear in the transcript, nor in the paths. We also generate mock intermediate states, following the same procedure as in the nonce-misuse resilience confidentiality proof in Section 9.6.1, and we can safely ignore leaky paths, since they do not have overlapping nonces. The extended transcript, released at the end of the interaction, can be derived as in Section 9.6.3.1. From $\tilde{\tau}$ we can reconstruct the two dictionaries $\mathtt{S}_{\text{in}}$ and $\mathtt{S}_{\text{out}}$, the set $\mathtt{P}$, the list of all permutation queries (X, Y, d), and the users' keys. Finally, we re-use the bad events defined in Section 9.6.3.1, except Dec, which does not apply there. In order to inherit the notation from this section, let us define $\mathtt{P}_{\mathrm{S}} = \emptyset$. The bad events are as follows:

$$\begin{aligned}
&\mathsf{ColK}_{\mathsf{c}}^{\mathsf{lr}} : \mathsf{ColK}_{\mathsf{a}}^{\mathsf{lr}} \text{ (of Section 9.6.1)}\,; && \mathsf{GueK}_{\mathsf{c}}^{\mathsf{lr}} : \mathsf{GueK}_{\mathsf{a}}^{\mathsf{lr}} \text{ (of Section 9.6.1)}\,;\\
&\mathsf{ColS}_{\mathsf{c}}^{\mathsf{lr}} : \mathsf{ColS}_{\mathsf{a}}^{\mathsf{lr}} \text{ (of Section 9.6.1)}\,; && \mathsf{GueS}_{\mathsf{c}}^{\mathsf{lr}} : \mathsf{GueS}_{\mathsf{a}}^{\mathsf{lr}} \text{ (of Section 9.6.1)}\,;\\
&\mathsf{Inner}_{\mathsf{c}}^{\mathsf{lr}} : \mathsf{Inner}_{\mathsf{a}}^{\mathsf{lr}} \text{ (of Section 9.6.1)}\,; &&\\
&\mathsf{BAD}_{\mathsf{c}}^{\mathsf{lr}} : \mathsf{ColK}_{\mathsf{c}}^{\mathsf{lr}} \vee \mathsf{GueK}_{\mathsf{c}}^{\mathsf{lr}} \vee \mathsf{ColS}_{\mathsf{c}}^{\mathsf{lr}} \vee \mathsf{GueS}_{\mathsf{c}}^{\mathsf{lr}}\,; &&\\
&\mathsf{BAD1}_{\mathsf{c}}^{\mathsf{lr}} : \mathsf{BAD}_{\mathsf{c}}^{\mathsf{lr}}\,; && \mathsf{BAD2}_{\mathsf{c}}^{\mathsf{lr}} : \mathsf{BAD}_{\mathsf{c}}^{\mathsf{lr}} \vee \mathsf{Inner}_{\mathsf{c}}^{\mathsf{lr}}\,.
\end{aligned}$$

Probability of Good Transcripts. Since the bad events from Section 9.6.3.1 already handled collisions with challenge states from encryption queries, the same reasoning can be applied, so that the absence of $\mathsf{BAD}_{\mathsf{c}}^{\mathsf{lr}}$ guarantees that the intermediate states generated in the ideal world are consistent with the structure of the mode Ascon-AE. Therefore, every transcript which is reachable in the real world is also reachable in the ideal world, and vice-versa, and for any transcript $\tilde{\tau}$ that does not set $\mathsf{BAD}_{\mathsf{c}}^{\mathsf{lr}}$, we have

$$\frac{\mathbf{Pr}\left(\mathcal{A}\left[W_R\right] \text{ generates } \tilde{\tau}\right)}{\mathbf{Pr}\left(\mathcal{A}\left[W_I\right] \text{ generates } \tilde{\tau}\right)} \geq 1\,. \tag{9.34}$$

Probability of $\mathsf{BAD}_\mathsf{c}^\mathsf{lr}$ in the Ideal World. Compared to the evaluation done in Section 9.6.3.1, all challenge intermediate states are sampled at the end of the interaction, thus the evaluation of $\mathsf{BAD1}_\mathsf{c}^\mathsf{lr}$ (or $\mathsf{BAD2}_\mathsf{c}^\mathsf{lr}$) involving these challenge intermediate states is postponed at the end of the interaction. Overall, the same technique can be applied, and we obtain

$$\mathbf{Pr}\left(\mathsf{BAD}_\mathsf{c}^\mathsf{lr}\right) \leq \frac{\mu(\mu-1)}{2^{k+1}} + \frac{2\mu\left(\mathcal{N}+\mathcal{M}\right)}{2^k} + \frac{18\mathcal{M}\left(\mathcal{M}+\mathcal{N}\right)}{2^c}$$
$$+ \min\left\{\frac{14Q\left(\mathcal{N}+\mathcal{M}\right)}{2^k}, \frac{4\left(\mathcal{M}+\mathcal{N}\right)^2}{2^c} + \frac{6(\mathcal{N}+\mathcal{M})\cdot \mathrm{mucol}(\mathcal{N}+\mathcal{M}, 2^{c-k})}{2^k}\right\}.$$

We obtained an upper bound for probability that the ideal world generates a bad transcript. Using the H-coefficient technique, and the fact that the ratio of good transcript is lower-bounded by one (cf., (9.34)), we conclude. □

9.6.4 Proof of Theorem 9.5.2

Let $\mathcal{A}$ be an adversary that makes at most $\mathcal{N}$ permutation queries, Q_E encryption queries of at most $\mathcal{M}_E$ blocks, and Q_D decryption queries of at most $\mathcal{M}_D$ blocks, as in the theorem statement. Our goal is to upper bound $\mathbf{Adv}_{\text{Ascon}}^{\mu\text{-sr-auth}}(\mathcal{A})$.

Let $\mathbf{BotDec}_{K_m}^{\mathcal{P}}$ be the function that takes as input a tuple $(N, A, C, T) \in \{0,1\}^n \times \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^t$, computes internally $\mathbf{Dec}_{K_m}^{\mathcal{P}}$, but always returns $\perp$. Therefore, the function $\left[\mathbf{BotDec}_{K_m}^{\mathcal{P}}\right]_L$ always returns $\perp$, but leaks the internal states in the same way as $\left[\mathbf{Dec}_{K_m}^{\mathcal{P}}\right]_L$. With this notation, the adversary interacts either with the *real world* W_R, which gives access to $\left[\left(\left[\mathbf{Enc}_{K_m}^{\mathcal{P}}\right]_L, \left[\mathbf{Dec}_{K_m}^{\mathcal{P}}\right]_L\right)_{m=1}^{\mu}, \mathcal{P}^{\pm}\right]$, or with the *ideal world* W_I, which gives access to $\left[\left(\left[\mathbf{Enc}_{K_m}^{\mathcal{P}}\right]_L, \left[\mathbf{BotDec}_{K_m}^{\mathcal{P}}\right]_L\right)_{m=1}^{\mu}, \mathcal{P}^{\pm}\right]$. The following proof will be heavily based on the one of Section 9.6.3.1, the difference being that challenge queries are not present.

Setup. We adopt the same transcript and path notation as in Section 9.6.3.1, with the difference that there are no challenge queries. All intermediate states come from genuine permutation evaluations, so that no mock intermediate states are needed. However, we need to revise the definition of *superseded* *paths.* Recall that a decryption path $\texttt{path} = (D_L, m, N, A, C, f)$ from a previous query is considered superseded by a later encryption path $\texttt{path}' = (E_L, m, N, A, P, f)$ associated to an encryption query $(E_L, m, N, A', P', C', T) \in \tau$, if $|P| = |C|$ and $C = C'[1 : |C|]$. In the previous proofs, only decryption queries could be superseded by encryption queries, because intermediate states for decryption queries were generated at the end of the interaction. In contrast, now decryption queries can supersede encryption queries. In more detail, an encryption path (E_L, m, N, A, P, f) associated to an encryption query $(E_L, m, N, A', P', C, T)$ is superseded by a later decryption path $\texttt{path}' = (D_L, m, N, A, C', f)$ if $|P| = |C'|$ and $C' = C[1 : |C'|]$. Let $\texttt{P}_\text{S}$ denote the set of paths that have been superseded according to this revised definition.

The extended transcript, released at the end of the interaction, can be derived as in Section 9.6.3.1. From $\tilde{\tau}$ we can reconstruct the two dictionaries $\texttt{S}_\text{in}$ and $\texttt{S}_\text{out}$, the set $\texttt{P}$, the list of all permutation queries (X, Y, d), and the users' keys. Finally, we re-use the bad events defined in Section 9.6.3.1 as follows:

$$\mathsf{ColK}_\mathsf{a}^\mathsf{sr} : \mathsf{ColK}_\mathsf{a}^\mathsf{lr} \text{ (of Section 9.6.3.1)}\,; \qquad \mathsf{GueK}_\mathsf{a}^\mathsf{sr} : \mathsf{GueK}_\mathsf{a}^\mathsf{lr} \text{ (of Section 9.6.3.1)}\,;$$
$$\mathsf{GueS}_\mathsf{a}^\mathsf{sr} : \mathsf{GueS}_\mathsf{a}^\mathsf{lr} \text{ (of Section 9.6.3.1)}\,; \qquad \mathsf{Inner}_\mathsf{a}^\mathsf{sr} : \mathsf{Inner}_\mathsf{a}^\mathsf{lr} \text{ (of Section 9.6.3.1)}\,;$$
$$\begin{aligned}\mathsf{Dec}_\mathsf{a}^\mathsf{sr} : {} & \exists (D_L, m, N, A, C, T, \tilde{P}) \in \tilde{\tau} \text{ such that} \\ & \text{either } \mathrm{inner}_t(\texttt{S}_\text{out}[(D_L, m, N, A, C, 1)]) \oplus \mathrm{inner}_t(K_m) = T \\ & \text{or } \exists \texttt{path} \in \texttt{P} \text{ superseding } (D_L, m, N, A, C, 1) \\ & \text{with } \mathrm{inner}_t(\texttt{S}_\text{out}[\texttt{path}]) \oplus \mathrm{inner}_t(K_m) = T\,; \\ \mathsf{BAD}_\mathsf{a}^\mathsf{sr} : {} & \mathsf{ColK}_\mathsf{a}^\mathsf{sr} \vee \mathsf{GueK}_\mathsf{a}^\mathsf{sr} \vee \mathsf{GueS}_\mathsf{a}^\mathsf{sr} \vee \mathsf{Inner}_\mathsf{a}^\mathsf{sr} \vee \mathsf{Dec}_\mathsf{a}^\mathsf{sr}\,.\end{aligned}$$

Because there are no more challenge paths, there is no bad event of the form ColS, and $\mathsf{Dec}_\mathsf{a}^\mathsf{sr}$ now includes leaky decryption queries. Moreover, now $\mathsf{Inner}_\mathsf{a}^\mathsf{sr}$ is incorporated in the main bad event.

Probability of Good Transcripts. As long as $\mathsf{Dec}_\mathsf{a}^\mathsf{sr}$ does not occur, the decryption queries in real world will always output $\perp$, so that for any good transcript $\tilde{\tau}$,

$$\mathbf{Pr}\left(\mathcal{A}\left[W_R\right] \text{ generates } \tilde{\tau}\right) = \mathbf{Pr}\left(\mathcal{A}\left[W_I\right] \text{ generates } \tilde{\tau}\right). \tag{9.35}$$

Note that here, using the fundamental lemma of game playing [BR06] would have been sufficient, but we continue to use the H-coefficient technique for the sake of consistency.

Probability of $\mathsf{BAD}_\mathsf{a}^\mathsf{sr}$. The bounding of $\mathsf{BAD}_\mathsf{a}^\mathsf{sr}$ can be handled similarly to the second bounding in Section 9.6.3.1, with one key difference for the bad event $\mathsf{Dec}_\mathsf{a}^\mathsf{sr}$: this time, the adversary can submit forgeries with nonces associated to states that leaked. However, as long as $\neg\mathsf{GueS}_\mathsf{a}^\mathsf{sr}[i]$ holds, the adversary cannot predict the input to the key blinding call using its permutation queries. Moreover, as long a $\neg\mathsf{Inner}_\mathsf{a}^\mathsf{sr}[i]$ holds, every rightmost permutation evaluation made during a decryption query is unique, so that no two distinct decryption queries share the same input final state. Therefore, the bounding of the conditioned $\mathsf{Dec}_\mathsf{a}^\mathsf{sr}[i]$ is the same, and we have

$$\begin{aligned}\mathbf{Pr}\left(\mathsf{BAD}_\mathsf{a}^\mathsf{sr}\right) \le \frac{\mu(\mu-1)}{2^{k+1}} + \frac{2\mu\left(\mathcal{N}+\mathcal{M}\right)}{2^k} + \frac{2Q_D}{2^t} + \frac{18\mathcal{M}\left(\mathcal{M}+\mathcal{N}\right)}{2^c} \\ + \frac{4\left(\mathcal{M}+\mathcal{N}\right)^2}{2^c} + \frac{6(\mathcal{N}+\mathcal{M})\cdot \mathrm{mucol}(\mathcal{N}+\mathcal{M},2^{c-k})}{2^k}\,.\end{aligned}$$

We obtained an upper bound for the probability that the ideal world generates a bad transcript. Using the H-coefficient technique, and the fact that the ratio of good transcript is lower-bounded by one (cf., (9.35)), we conclude. □

9.6.5 Proof of Theorem 9.5.4

Let $\mathcal{A}$ be an adversary with complexity $(\mathcal{N}, Q_E, \mathcal{M}_E, Q_D, \mathcal{M}_D)$, as in the theorem statement. Our goal is to upper bound $\mathbf{Adv}^{\mu\text{-rup-auth}}_{\mathrm{Ascon}}(\mathcal{A})$. Out of generosity, we assume that a query to $\mathbf{V}$ triggers first an evaluation to $\mathbf{D}$, and a query to $\mathbf{D}$ is followed by an evaluation of $\mathbf{V}$, and doing so only increases the adversary's success probability. Therefore, we can combine the oracles $\mathbf{D}$ and $\mathbf{V}$ back into a single oracle $\mathbf{DV}$, which first evaluates $\mathbf{D}$, then $\mathbf{V}$. Denote by $\mathbf{DBot}$ the oracle that computes $\mathbf{DV}$, releases the unverified plaintext, but instead of returning the output of $\mathbf{V}$, it always returns $\perp$.

With this notation, the adversary interacts either with the *real world* W_R, which gives access to $\left[\left(\mathbf{Enc}^{\mathcal{P}}_{K_m}, \mathbf{DV}^{\mathcal{P}}_{K_m}\right)^{\mu}_{m=1}, \mathcal{P}^{\pm}\right]$, or with the *ideal world* W_I, which gives access to $\left[\left(\mathbf{Enc}^{\mathcal{P}}_{K_m}, \mathbf{DBot}^{\mathcal{P}}_{K_m}\right)^{\mu}_{m=1}, \mathcal{P}^{\pm}\right]$. Here, the adversarial complexity is $(Q_D, \mathcal{M}_D)$ for the oracle $\mathcal{O}_{2,m}$. By abuse of notation, we will continue to refer to the oracle $\mathcal{O}_{2,m}$ as a decryption oracle.

The following proof builds on the nonce-misuse authenticity proof from Section 9.6.1. The key difference lies in how decryption queries are handled: in the current case, all states are computed on-the-fly rather than at the end of the interaction. We will make adjustments similar to those made when adapting the leakage resilience proof (in Section 9.6.3.1) for the state-recovery proof (in Section 9.6.4).

Setup. The transcript notation now needs to account for the unverified plaintext released by the decryption oracle. It is defined as follows:

- A forward (resp., inverse) permutation query with input X and output Y generates the transcript element (X, Y, fwd) (resp., (X, Y, inv));
- An encryption query with user m, input (N, A, P), and output (C, T) generates the transcript element (E, m, N, A, P, C, T);
- A decryption query with user m, input (N, A, C, T), output unverified plaintext P, and verification output V generates the transcript element (D, m, N, A, C, P, T, V).

We adopt the same path notation as in Section 9.6.1, but adapt the notion of *superseded paths* as done in Section 9.6.4. Let P_S denote the set of paths that have been superseded. All intermediate states come from genuine permutation evaluations, so that no mock intermediate states are needed. The extended transcript, released at the end of the interaction, can be derived as in Section 9.6.1. Finally, we re-use the bad events defined in Section 9.6.1 as follows:

$$\begin{aligned}
\mathsf{ColK}_\mathsf{a}^\mathsf{rup} &: \mathsf{ColK}_\mathsf{a}^\mathsf{m} \text{ (of Section 9.6.1)}\,; & \mathsf{GueK}_\mathsf{a}^\mathsf{rup} &: \mathsf{GueK}_\mathsf{a}^\mathsf{m} \text{ (of Section 9.6.1)}\,;\\
\mathsf{ColS}_\mathsf{a}^\mathsf{rup} &: \mathsf{ColS}_\mathsf{a}^\mathsf{m} \text{ (of Section 9.6.1)}\,; & \mathsf{GueS}_\mathsf{a}^\mathsf{rup} &: \mathsf{GueS}_\mathsf{a}^\mathsf{m} \text{ (of Section 9.6.1)}\,;\\
\mathsf{Dec}_\mathsf{a}^\mathsf{rup} &: \mathsf{Dec}_\mathsf{a}^\mathsf{m} \text{ (of Section 9.6.1)}\,;\\
\mathsf{BAD}_\mathsf{a}^\mathsf{rup} &: \mathsf{ColK}_\mathsf{a}^\mathsf{rup} \vee \mathsf{GueK}_\mathsf{a}^\mathsf{rup} \vee \mathsf{ColS}_\mathsf{a}^\mathsf{rup} \vee \mathsf{GueS}_\mathsf{a}^\mathsf{rup} \vee \mathsf{Dec}_\mathsf{a}^\mathsf{rup}\,.
\end{aligned}$$

Probability of Good Transcripts. Similar to the state-recovery proof of Section 9.6.4, as long as $\mathsf{Dec}_\mathsf{a}^\mathsf{rup}$ does not occur, the decryption queries in real world will always output $\perp$, so that for any good transcript $\tilde{\tau}$,

$$\mathbf{Pr}\left(\mathcal{A}\left[W_R\right] \text{ generates } \tilde{\tau}\right) = \mathbf{Pr}\left(\mathcal{A}\left[W_I\right] \text{ generates } \tilde{\tau}\right)\,. \tag{9.36}$$

Again, using the fundamental lemma of game playing [BR06] would have been sufficient, but we continue to use the H-coefficient technique for the sake of consistency.

Probability of BAD in the Ideal World. The same bounding technique as in Section 9.6.1 can be done. Indeed, the bad events defined previously are general enough to treat encryption and decryption states the same way, focusing only on equality on the inner part of the states. In this case, there are no more events that can be set at the end of the interaction. Therefore,

$$\mathbf{Pr}\left(\mathsf{BAD}_{\mathsf{a}}^{\mathsf{rup}}\right) \leq \frac{\mu(\mu-1)}{2^{k+1}} + \frac{2\mu\left(\mathcal{M}+\mathcal{N}\right)}{2^{k}} + \frac{18\mathcal{M}\left(\mathcal{M}+\mathcal{N}\right)}{2^{c}} + \frac{2Q_D}{2^{t}}\,.$$

We obtained an upper bound for the probability that the ideal world generates a bad transcript. Using the H-coefficient technique, and the fact that the ratio of good transcript is lower-bounded by one (cf., (9.36)), we conclude. □

9.7 Generic Attacks for Ascon-AE

We include the attacks of Propositions 9.4.1, 9.4.3–9.5.1, and 9.5.4 here, in Sections 9.7.1–9.7.5, respectively.

9.7.1 Proof of Proposition 9.4.1

In each of the following sections, we describe the adversaries as stated in the proposition. These are all forgery attacks, but they can be easily converted into distinguishing attacks by returning "ideal" when the attack fails, and "real" otherwise (see for instance the last step in Section 9.7.3).

9.7.1.1 Adversary $\mathcal{A}_1$

Recall that $\mathcal{A}_1$ has resources satisfying $Q_D \approx 2^t$. A simple matching attack consists of submitting $\approx 2^t$ arbitrary decryption queries, each with a different tag.

9.7.1.2 Adversary $\mathcal{A}_2$

Recall that $\mathcal{A}_2$ has resources satisfying $\mathcal{N} \approx 2^k/\mu$. This term corresponds to the probability that the adversary guesses one of the user keys. Consider the following attack:

1. Let $N \in \{0,1\}^n$. For $m \in [\![1,\mu]\!]$ make an encryption query, with user m, with input $(N, \epsilon, 0^r)$, and denote the ciphertext by C^m and the tag by T^m;

2. For $i \in [\![1,\mathcal{N}]\!]$, sample $L_i \xleftarrow{\$} \{0,1\}^k$, and compute $\mathbf{Enc}_{L_i}^{\mathcal{P}}\left(N, \epsilon, 0^r\right)$, get a ciphertext $\tilde{C}^i$;

3. With probability $\approx \frac{\mu\mathcal{N}}{2^k} \approx 1$, there exists a L_i from step 2 that collides with a key K_m of a user m. The adversary can check this by observing that $\tilde{C}^i = C^m$. In the following, we assume that this is the case;

4. Let $P \in \{0,1\}^*$, and $N' \neq N$. Compute $(C,T) = \mathbf{Enc}^{\mathcal{P}}_{L_i}(N', \epsilon, P)$;

5. Submit a forgery for user m with input (N', ϵ, C, T).

In order to reduce the probability of a false positive in step 3 to a negligible probability, the encryption queries in step 1 can be extended with $\gamma = \tilde{\mathcal{O}}\left(\lceil \frac{b}{r} \rceil\right)$ plaintext blocks.

9.7.1.3 Adversary $\mathcal{A}_3$

Recall that $\mathcal{A}_3$ has resources satisfying $(\mathcal{M}_E + 2^r)\mathcal{N} \approx 2^b$. At a high level, the following attack involves guessing an intermediate state generated during encryption queries and using it to create a forgery. Let $\gamma \in \mathbb{N}^*$ be a parameter. As in the previous attack, γ serves to mitigate the probability to obtain false positives, and we suggest taking $\gamma = \tilde{\mathcal{O}}\left(\lceil \frac{b}{r} \rceil\right)$. For simplicity of presentation, we will assume that $\mu \cdot 2^n \geq \mathcal{M}_E$, but the attack can be easily adapted by stretching the block length of the encryption queries by a logarithmic factor. Let $\left(m^i, N^i\right)_i$ be a family of $\mathcal{M}_E$ pairwise distinct user indices and nonces. The attack operates a follows:

1. **Encryption queries:** for $i = 1, \ldots, \mathcal{M}_E$, make an encryption query with user m^i with input $\left(N^i, \epsilon, 0^{r(\gamma+1)}\right)$, and denote the ciphertexts by $C^i_0, C^i_1, \ldots, C^i_\gamma$.

   This step requires $\tilde{\mathcal{O}}(\mathcal{M}_E)$ encryption queries with $\tilde{\mathcal{O}}(\gamma)$ blocks each;

2. **State guessing:**

   (a) Denote by $\nu \in \{0,1\}^r$ the outer part that occurs most frequently among $(C^i_0)_{i \in [\![1, \mathcal{M}_E]\!]}$;

   (b) For $l \in [\![1, \mathcal{N}]\!]$, sample $X^l \xleftarrow{\$} \{0,1\}^c$, and compute $\mathcal{P}^f(\nu \| X^l)$ for $f \in [\![1, \gamma]\!]$;

   (c) If there exists some $l \in [\![1, \mathcal{N}]\!]$, $i \in [\![1, \mathcal{M}_E]\!]$ such that for all $f \in [\![0, \gamma]\!]$, $\mathrm{outer}_r(\mathcal{P}^f(\nu \| X^l)) = C^i_f$, then we consider that the adversary has successfully guessed the intermediate state with user m^i, nonce N^i, empty associated data, and after having absorbed the plaintext block 0^r;

(d) Let $S_{\text{in}} := \nu \| X^l$, S_{fin} be the state of the associated encryption query right before the last key blinding, T be the tag of the associated encryption query, $N := N^i$, and $m := m^i$.

This step requires $\gamma\mathcal{N}$ permutation queries. As for the success probability, note that γ allows to reduce the influence of false positives. Therefore, the success probability is almost the probability that the adversary permutation query history contains a successful guess, which is itself $\approx \frac{\mathcal{M}_E\mathcal{N}}{2^b} + \frac{\mathcal{N}}{2^c} \approx 1$;

3. **State binding:** The goal of this step is to connect S_{in} and S_{fin} by a sequence of message blocks different from $(0^{r(\gamma+1)}, 1\|0^{r-1})$. To achieve this, we use an attack introduced by the designers of the sponge [BDPV07], which relies on finding inner collisions. Let us first for simplicity assume that $r > c/2$. It consists of the following steps:

   (a) Compute $X^i := \mathcal{P}\left(S_{\text{in}} \oplus P_1^i \| 0^c\right)$ for $\mathcal{N}$ different P_1^is, distinct from 0^r;[11]

   (b) Compute $Y^i := \mathcal{P}^{-1}\left(S_{\text{fin}} \oplus P_3^i \| 0^c\right)$ for $\mathcal{N}$ different P_3^is, distinct from 0^r;

   (c) If there exists $i, j \in [\![1, \mathcal{N}]\!]$ such that $\text{inner}_c(X_i) = \text{inner}_c(Y_j)$, let $P_2 := \text{outer}_r(X_i) \oplus \text{outer}_r(Y_j)$, and output (P_1^i, P_2, P_1^j); else abort.

   Note that the returned sequence of message blocks corresponds to a valid padding. In the setting where $\mathcal{N} \geq 2^r$ and $r \leq c/2$, the attack can be extended by making multiple sequential absorb calls in steps 3a and 3b. In the following, we denote the output sequence of message blocks by $(P_1, \ldots, P_d)$.

   This step requires $2\mathcal{N}$ permutation queries. The step succeeds with probability $\approx \min\left\{1, \frac{\mathcal{N}^2}{2^c}\right\}$. Given that $(\mathcal{M}_E + 2^r)\mathcal{N} \approx 2^b$ and $\mathcal{M}_E \ll \mathcal{N}$, this gives a success probability of ≈ 1;

4. **Compute the ciphertexts:** Initialize $S \leftarrow S_{\text{in}}$. Then, for $f \in [\![1, d]\!]$, compute $\tilde{C}_f \leftarrow \text{outer}_r(S) \oplus P_f$, and update $S \leftarrow \mathcal{P}(S \oplus (P_f \| 0^c))$;

5. **Forgery:** Let $l = |unpad_r(P_1\| \cdots \|P_d)|$. Submit a forgery with input $(m, N, \epsilon, (\tilde{C}_1\| \cdots \|\tilde{C}_d)[0:l], T)$.

Overall, the attack requires $\tilde{\mathcal{O}}(\mathcal{M}_E)$ encryption queries, one decryption query, and $\tilde{\mathcal{O}}(\mathcal{N})$ permutation queries, and succeeds with high probability. □

[11] To generalize the attack to any input sequence of message blocks $(B_1, \ldots, B_l)$, the P_1^is can be chosen different from B_1.

9.7.2 Proof of Proposition 9.4.3

Recall that $\mathcal{A}$ has resources satisfying $\mathcal{M}_E\mathcal{N} \approx 2^c$. The attack is similar to the one described in Section 9.7.1.3, with the main difference being that in this case, the adversary makes use of the nonce-misuse setting to set the outer parts of the states to a value of its choice. Let $\gamma \in \mathbb{N}^*$ be a parameter. As in the previous attacks, γ serves to mitigate the probability to obtain false positives, and we suggest taking $\gamma = \tilde{\mathcal{O}}\left(\lceil \frac{b}{r} \rceil\right)$. For simplicity, we describe the attack in the case where $\mu \cdot 2^n \geq \mathcal{M}_E$, similar to the nonce-respecting case. Let $\left(m^i, N^i\right)_i$ be a family of $\mathcal{M}_E$ pairwise distinct user indices and nonces. The attack operates a follows:

1. **Encryption queries:** For $i = 1, \ldots \mathcal{M}_E$, do the following:

   (a) Make an encryption query with user m^i with input $\left(N^i, \epsilon, 0^r\right)$, and denote by C^i the obtained ciphertext;

   (b) Make an encryption query with user m^i with input $\left(N^i, \epsilon, C^i \| 0^{\gamma r}\right)$, get ciphertexts $\left(C_0^i, C_1^i, \ldots, C_\gamma^i\right)$, and tag T^i.

   Note that in the second permutation evaluation made in the context of the query from step (b), the outer part of the state is set to 0^r, so that necessarily $C_0^i = 0^r$. This step requires $\tilde{\mathcal{O}}\left(\mathcal{M}_E\right)$ encryption queries, each with $\tilde{\mathcal{O}}\left(\gamma\right)$ blocks, and each nonce is re-used twice;

2. **State guessing:** This step is identical to step 2 of Section 9.7.1.3, except that in this case, the permutation queries of the adversary now have all their outer r bits fixed to 0^r. Thus, the adversary only needs to guess one of the rightmost c bits of the intermediate states obtained during encryption queries. This speeds up the success probability to $\approx \frac{\mathcal{M}_E\mathcal{N}}{2^c} \approx 1$;

3. **State binding:** Same as step 3 of Section 9.7.1.3;

4. **Compute the ciphertexts:** Same as step 4 of Section 9.7.1.3;

5. **Forgery:** Same as step 5 of Section 9.7.1.3.

Overall, the attack requires $\tilde{\mathcal{O}}\left(\mathcal{M}_E\right)$ encryption queries where each nonce is repeated twice, one decryption query, and $\tilde{\mathcal{O}}\left(\mathcal{N}\right)$ permutation evaluations. The state guessing attack (step 2) succeeds with probability $\approx \frac{\mathcal{M}_E\mathcal{N}}{2^c} \approx 1$, the state binding attack (step 3) succeeds with probability $\approx \min\left\{1, \frac{\mathcal{N}^2}{2^c}\right\} \approx 1$, so that this attack succeeds with probability close to 1. $\square$

9.7.3 Proof of Proposition 9.4.4

We give below a key recovery attack that exploits nonce-misuse encryption queries. Recall that $\mathcal{A}^{\text{conf}}$ and $\mathcal{A}^{\text{auth}}$ have resources satisfying $\mathcal{M}_E\mathcal{N} \approx 2^c$. The strategies of $\mathcal{A}^{\text{conf}}$ and $\mathcal{A}^{\text{auth}}$ are identical, except in the final step. In the phases of the attack shared with $\mathcal{A}^{\text{auth}}$, $\mathcal{A}^{\text{conf}}$ makes encryption queries only to the non-challenge oracle (i.e., $\mathcal{O}_{1,m}$), allowing thus nonce reuse. The attack operates as follows:

1. **Encryption queries and state guessing:** Apply steps 1 and 2 from the attack described in Section 9.7.2. If the attack succeeds, denote by S_{in} the state guessed and let $S_{\text{out}} = \mathcal{P}(S_{\text{in}})$. Let m, N, and P be the user, nonce, and plaintext block sequence associated to this state (i.e., after absorbing P, one gets the permutation input S_{in}), respectively.

   This step requires $\tilde{\mathcal{O}}(\mathcal{M}_E)$ nonce-misuse encryption queries with $\tilde{\mathcal{O}}(1)$ blocks each, where each nonce is repeated twice, and $\tilde{\mathcal{O}}(\mathcal{N})$ permutation queries. The step succeeds with probability $\approx \frac{\mathcal{M}_E\mathcal{N}}{2^c} \approx 1$;

2. **State expansion:** For $i \in [\![1, \mathcal{M}_E]\!]$, make a forward permutation query $\mathcal{P}(S_{\text{out}} \oplus (P_2^i \| 0^c))$.[12] We obtain a family of states along with their paths:

$$\begin{aligned} S^i &= 0^r \| \text{inner}_c(\mathcal{P}(S_{\text{out}} \oplus (P_2^i \| 0^c)))\,, \\ \texttt{path}^i &= \big(m, N, \epsilon, (P, P_2^i, \text{outer}_r(\mathcal{P}(S_{\text{out}} \oplus (P_2^i \| 0^c))))\big)\ . \end{aligned}$$

   Note that all states S^is have their outer r bits fixed to 0^r. Some of the states S^i will later be interpreted as states right before the last key blinding.

   This step requires $\mathcal{M}_E$ permutation queries;

3. **Filtering:** Let mucol $\in \{0,1\}^{c-k}$ be such that the size of the following set is maximal:

$$I_{\text{mucol}} = \big\{ i \in [\![1, \mathcal{M}_E]\!] \mid \text{inner}_{c-k}(S^i) = \text{mucol} \big\}\ .$$

   In other words, the states S^i are filtered by selecting those with the rightmost $c-k$ bits to the value that maximizes the number of S^is. We expect that $|I_{\text{mucol}}| \geq \max\big\{1, \frac{\mathcal{M}_E}{2^{c-k}}\big\}$ with high probability. Looking ahead, in the regime where $\frac{\mathcal{M}_E}{2^{c-k}} \leq 1$, then the attack targeting the term $\frac{\mu\mathcal{N}}{2^k}$ from Section 9.7.1.2 is better. Therefore, we assume $\frac{\mathcal{M}_E}{2^{c-k}} > 1$ in the following;

[12] If $2^r < \mathcal{M}_E$, this step can be extended by making cascaded permutation calls.

4. **Construction queries:** For each $i \in I_{\text{mucol}}$, make an encryption query with input `path`^{i}, and get tag T^i.

   This step requires on expectation $\frac{\mathcal{M}_E}{2^{c-k}}$ encryption queries, all using the same nonce;

5. **Key guessing:** From the previous step, the adversary has made $\approx \frac{\mathcal{M}_E}{2^{c-k}}$ encryption queries, where the inputs to their last permutation query (during the key blinding) have $b-k$ bits set to a value known by the adversary. These bits are always in fixed positions, i.e., the leftmost r bits and the rightmost $c-k$ bits. The following step consists of trying to guess the remaining k bits of those states.

   For $j = 1, \ldots, \mathcal{N}$, do the following:

   (a) Sample $X^j \xleftarrow{\$} \{0,1\}^k$, and make the permutation query $\mathcal{P}\left(0^r \| X^j \| \text{mucol}\right)$;

   (b) For each $i \in I_{\text{mucol}}$:

      i. Compute $K^{ij} = \text{outer}_k(\text{inner}_c(S^i)) \oplus X^j$;

      ii. If $\left\lfloor \text{inner}_k(\mathcal{P}\left(0^r \| X^j \| \text{mucol}\right)) \oplus K^{ij} \right\rfloor_t = T^i$, then K^{ij} is a key candidate;

      iii. For each key candidate, the adversary checks whether the obtained ciphertexts and the state S^i from the encryption query number i match those obtained with a direct evaluation of $\mathbf{Enc}^{\mathcal{P}}_{K^{ij}}$. This step allows to determine whether K^{ij} is a false positive or a correct key.

   Checking one false positive in step 5(b)iii costs $\tilde{\mathcal{O}}(1)$ permutation queries. In order to not blow up the permutation complexity of the adversary, we allow a small number of false positives per permutation query, hence the requirement $\frac{\mathcal{M}_E}{2^{c-k+t}} \leq 1$, or equivalently $\mathcal{N} \geq 2^{k-t}$, as stated in the proposition.

   The probability that the adversary, via one of the permutation calls made in this step, guesses the k bits of one of the $i \in I_{\text{mucol}}$ encryption queries is

   $$\approx \frac{\mathcal{N}\,|I_{\text{mucol}}|}{2^k} \approx \frac{\mathcal{M}_E \mathcal{N}}{2^c} \approx 1\,;$$

6. **Last step:** Let K be the guessed key. Depending on whether the adversary is against confidentiality or authenticity, do the following:

- $\mathcal{A}^{\text{conf}}$: make an evaluation with $\mathbf{Enc}_K^{\mathcal{P}}$ with a new nonce, obtain several ciphertext blocks (e.g., take $\gamma = \tilde{\mathcal{O}}\left(\lceil \frac{b}{r} \rceil\right)$), and make the same query to the challenge oracle. If the ciphertext blocks coincide with the challenge oracle, return 1, else return 0. The distinguishing advantage of the adversary is close to 1;
- $\mathcal{A}^{\text{auth}}$: make an evaluation with $\mathbf{Enc}_K^{\mathcal{P}}$ with a new nonce, obtain a ciphertext and tag, and submit it to the challenge decryption oracle.

Overall, the attack requires $\tilde{\mathcal{O}}(\mathcal{M}_E)$ encryption queries and $\tilde{\mathcal{O}}(\mathcal{N})$ permutation queries, and succeeds with probability close to 1. $\square$

9.7.4 Proof of Proposition 9.5.1

The attack shares similarities with the nonce-misuse resilience key-recovery attack presented in Section 9.7.3, with the key difference being that the adversary has direct access to the intermediate states, and thus does not need to do step 1 of this attack. This gives more freedom in the parametrization of the attack. The strategies of $\mathcal{A}^{\text{conf}}$ and $\mathcal{A}^{\text{auth}}$ are identical, except in the final step. In the phases of the attack shared with $\mathcal{A}^{\text{auth}}$, $\mathcal{A}^{\text{conf}}$ makes encryption queries only to the non-challenge oracle (i.e., $\mathcal{O}_{1,m}$), thus leaky encryption queries. Here, the attack is parametrized by the maximum number of allowed encryption queries, thus allowing for a tradeoff between offline and online complexity. We make a case distinction depending on the chosen parameter regime.

Case 1, when $\max\left\{2^{c/2}, 2^k/Q_E\right\} = 2^{c/2}$**.** In this situation, the number of allowed encryption queries is not a limiting factor. In that case, we have $\mathcal{N} \approx 2^{c/2}$, and the attack operates as follows:

1. **State leaking:** Let $m \in \{0,1\}^\mu$, $N \in \{0,1\}^n$, $\mathrm{PT} \in \{0,1\}^*$, and $P \leftarrow pad_r^{10^*}(\mathrm{PT})$. Make an encryption query to the leaky oracle with user m and input $(N, \epsilon, \mathrm{PT})$, and obtain S_{in}, the state after having absorbed P. Let $S_{\text{out}} = \mathcal{P}(S_{\text{in}})$;
2. **State expansion:** For $i \in [\![1, \mathcal{N}]\!]$, make a forward permutation query $\mathcal{P}(S_{\text{out}} \oplus (P_2^i \| 0^c))$.[13] We obtain a family of states along with their paths:

$$\begin{aligned} S^i &= 0^r \| \mathrm{inner}_c(\mathcal{P}(S_{\text{out}} \oplus (P_2^i \| 0^c)))\,, \\ \mathtt{path}^i &= \left(m, N, \epsilon, (P, P_2^i, \mathrm{outer}_r(\mathcal{P}(S_{\text{out}} \oplus (P_2^i \| 0^c))))\right)\,. \end{aligned}$$

[13] If $2^r < \mathcal{N}$, this step can be extended by making cascaded permutation calls.

We will interpret S^i as a state right before the last key blinding.

This step requires $\mathcal{N}$ permutation queries;

3. **Filtering:** Let mucol $\in \{0,1\}^{c-k}$ be such that the size of the following set is maximal:

$$I_{\text{mucol}} = \left\{ i \in [\![1, \mathcal{N}]\!] \mid \text{inner}_{c-k}(S^i) = \text{mucol} \right\} .$$

We expect that $|I_{\text{mucol}}| \geq \frac{\mathcal{N}}{2^{c-k}}$ with high probability;

4. **Construction queries:** For each $i \in I_{\text{mucol}}$, make an encryption query with input $\texttt{path}^i$, and get tag T^i.

This step requires on expectation $\frac{\mathcal{N}}{2^{c-k}} \approx 2^{k-c/2} \leq Q_E$ encryption queries;

5. **Key guessing:** Same as step 5 of Section 9.7.3.

Since the size of I_{mucol} is different, the cost computation is a bit different as well. The number of false positives per permutation query is equal to

$$\frac{|I_{\text{mucol}}|}{2^t} = \frac{\mathcal{N}}{2^{c-k+t}} .$$

Similarly to the prior attack, we limit the number of false positives per permutation query to at most a small constant, hence the requirement $\mathcal{N} \geq 2^{k-t}$, as stated in the proposition.

The probability that the adversary, via one of the permutation calls made in this step, guesses the k bits of one of the $i \in I_{\text{mucol}}$ encryption queries is

$$\approx \frac{\mathcal{N}\,|I_{\text{mucol}}|}{2^k} \approx \frac{\mathcal{N}^2}{2^c} \approx 1 ;$$

6. **Last step:** Same as step 6 from Section 9.7.3.

Overall, this attack requires $\approx 2^{k-c/2}$ encryption queries and $\tilde{\mathcal{O}}(\mathcal{N})$ permutation queries, and succeeds with probability close to 1.

Case 2, when $\max\left\{2^{c/2}, 2^k/Q_E\right\} = 2^k/Q_E$**.** In this situation, the number of allowed encryption queries is a limiting factor, so that the offline complexity needs to compensate for that, so that $\mathcal{N} \approx 2^k/Q_E$. The performed steps are the same as in case 1, but the parametrization is different. The attack operates as follows:

1. **State leaking:** Same as step 1 of case 1;
2. **State expansion:** Same as step 2 of case 1, with the difference that the number of permutation calls made here is $Q_E 2^{c-k}$. Note that, since $\mathcal{N} \approx 2^k / Q_E$, this step costs $2^c/\mathcal{N} \leq 2^{c/2} \leq \mathcal{N}$ permutation queries;
3. **Filtering:** Same as step 3 of case 1. This time, we expect that $|I_{\text{mucol}}| \geq Q_E$ with high probability;
4. **Construction queries:** Same as step 4 of case 1. This time, the number of encryption queries is Q_E;
5. **Key guessing:** Same as step 5 of case 1. This time, the number of false positives per permutation query is equal to
$$\frac{|I_{\text{mucol}}|}{2^t} = \frac{Q_E}{2^t}\,.$$
Again, the constraint $\mathcal{N} \geq 2^{k-t}$ guarantees that every permutation query has no more than a small number of false positives.
The probability that the adversary, via one of the permutation calls made in this step, guesses the k bits of one of the $i \in I_{\text{mucol}}$ encryption queries is
$$\approx \frac{\mathcal{N}\,|I_{\text{mucol}}|}{2^k} \approx \frac{Q_E \mathcal{N}}{2^k} \approx 1\,.$$
6. **Last step:** Same as step 6 of case 1.

Overall, this attack requires Q_E encryption queries and $\tilde{\mathcal{O}}(\mathcal{N})$ permutation queries, and succeeds with probability close to 1.

With the case distinction performed depending on whether the number of allowed encryption queries is limiting or not, we obtained an attack with complexities as stated in the proposition. □

9

9.7.5 Proof of Proposition 9.5.4

At a high level, this is the forgery attack of Section 9.7.1.3, but with the state guessing step removed, as the leaky oracles give access to the states.

1. **State leaking:** Let $m \in \{0,1\}^\mu$, $N \in \{0,1\}^n$, $\mathrm{PT} \in \{0,1\}^*$, and $P \leftarrow \mathit{pad}_r^{10^*}(\mathrm{PT})$. Make an encryption oracle to the leaky oracle with user m and input $(N, \epsilon, \mathrm{PT})$, and obtain S, the state right after the first key blinding. Let S_{fin} be the state after having absorbed P (i.e., right before the last key blinding);

2. **State binding:** Same as step 3 of Section 9.7.1.3 with states S and S_{fin}, and plaintext block P;

3. **Compute the ciphertexts:** Same as step 4 of Section 9.7.1.3;

4. **Forgery:** Same as step 5 of Section 9.7.1.3.

Overall, the attack requires one encryption query, one decryption query, and $\tilde{\mathcal{O}}(\mathcal{N})$ permutation queries. The state binding attack (step 2) succeeds with probability $\approx \frac{\mathcal{N}^2}{2^c} \approx 1$. □

9.8 Ascon-Hash/Ascon-(C)XOF Modes and Their Security

We describe the mode underlying Ascon-Hash/Ascon-(C)XOF in Section 9.8.1, and the security of the constructions in Section 9.8.2.

9.8.1 Description of the Modes

Ascon-Hash, Ascon-XOF, as well as Ascon-CXOF are based on the sponge construction [BDPV07], which was itself extensively described in Section 3.1.1. Let $b, c, r \in \mathbb{N}^*$ such that $b = r + c$, and let $\mathcal{P}$ be a cryptographic permutation over b bits. The sponge construction takes as input a plaintext $P \in \{0,1\}^*$, and a length $\nu \in \mathbb{N}^*$. It returns a digest $Z \in \{0,1\}^*$ with $|Z| = \nu$. Then, Ascon-XOF$^{\mathcal{P}}$ is defined as follows:

$$\begin{aligned} \text{Ascon-XOF}^{\mathcal{P}} : \{0,1\}^* \times \mathbb{N}^* &\longrightarrow \{0,1\}^* \\ (P, \nu) \quad &\longrightarrow Z \in \{0,1\}^{\nu} . \end{aligned}$$

The construction is illustrated in Figure 9.4. Here, $IV \in \{0,1\}^b$ is a fixed initialization value. The Ascon specification [DEMS21a, SMC$^+$25] also specifies Ascon-Hash and Ascon-CXOF, the main difference being in the choice of initialization value IV. Ascon-CXOF then further distinguishes itself from the others in the fact that it additionally takes a customization string $C \in \{0,1\}^{\leq 2048}$, and incorporates it into the padded message with a length encoding at the beginning. The Ascon standard [SMC$^+$25] specifies Ascon-Hash, Ascon-XOF, and Ascon-CXOF to operate with capacity $c = 256$ and rate $r = 64$. The digest size of Ascon-Hash is 256 whereas for Ascon-XOF and Ascon-CXOF it is unlimited.

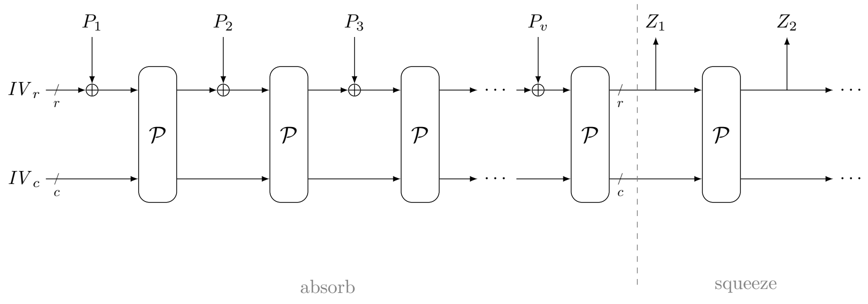


Figure 9.4: The Ascon-XOF mode of operation. Here, P is injectively padded as $(P_1, \ldots, P_v) \leftarrow pad_r^{10^*}(P)$. Here, for graphical convenience, the IV is split as $IV = (IV_r, IV_c)$, with $IV_r \in \{0,1\}^r$, $IV_c \in \{0,1\}^c$.

9.8.2 Security

We consider the security models from Section 3.1.2, namely indifferentiability, collision, and (second) preimage resistance, which were discussed in Sections 3.1.2.1 and 3.1.2.2 in the context of the sponge.

For indifferentiability, we repeat below the theorem of Bertoni et al. [BDPV08].

Theorem 9.8.1 ([BDPV08])**.** *Let $b, c, r, \mathcal{N} \in \mathbb{N}^*$ with $b = r + c$. Consider the* Ascon-XOF *mode of Section 9.8.1 with parameters b, c, r. Let $\mathcal{A}$ be an adversary with complexity $\mathcal{N}$. There exists a simulator $\mathbf{S}$ with complexity $\tilde{\mathcal{O}}(\mathcal{N})$ queries such that*

$$\mathbf{Adv}^{\text{indif}}_{\text{Ascon-XOF},\mathbf{S}}(\mathcal{A}) \leq \frac{\mathcal{N}(\mathcal{N}+1)}{2^c} .$$

This result directly implies collision, and (second) preimage resistance using Lemma 2.4.1. Moreover, the results from Chapter 4 allow to derive an improved bound for preimage resistance. We apply those results in the context of Ascon-XOF.

Theorem 9.8.2 ([BDPV08, AMP10b], Chapter 4)**.** *Let $b, c, r, \kappa, \nu \in \mathbb{N}^*$ with $b = r + c$, and let $\ell = \lceil \frac{\nu}{r} \rceil$. Let $\mathcal{N} \in \mathbb{N}^*$ be such that, for the case of preimage resistance, $\mathcal{N} \leq 2^{c-1}/3$ and $(\ell - 1)^2 \leq 2^b$. Consider the* Ascon-XOF *mode of Section 9.8.1 with parameters b, c, r. Let $\mathcal{A}$ be an adversary with complexity $\mathcal{N}$. We have*

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{col}[\nu]}_{\text{Ascon-XOF}}(\mathcal{A}) &\le \frac{\mathcal{N}(\mathcal{N}+1)}{2^c} + \frac{\mathcal{N}(\mathcal{N}-1)}{2^{\nu+1}}\,,\\
\mathbf{Adv}^{\mathrm{ePre}[\nu]}_{\text{Ascon-XOF}}(\mathcal{A}) &\le \min\left\{\frac{\mathcal{N}(\mathcal{N}+1)}{2^c}, \frac{4\ell\mathcal{N}}{2^{\nu-r}}\right\} + \frac{4\mathcal{N}}{2^{\nu}}\,,\\
\mathbf{Adv}^{\mathrm{eSec}[\kappa,\nu]}_{\text{Ascon-XOF}}(\mathcal{A}) &\le \frac{\mathcal{N}(\mathcal{N}+1)}{2^c} + \frac{\mathcal{N}}{2^{\nu}}\,.
\end{aligned}$$

For the specific parameters of Ascon-Hash, Theorem 9.8.2 implies 128-bit collision and second preimage resistance but 192-bit preimage resistance. For Ascon-(C)XOF, Theorem 9.8.2 implies $\min\{\nu/2, 128\}$-bit collision resistance, $\min\{\nu, 128\}$-bit second preimage resistance, and $\min\{\nu, 192\}$-bit preimage resistance, where ν is the minimal output size.

The bounds of Theorem 9.8.1 and Theorem 9.8.2 are tight, with matching attacks already given in the original specification [BDPV07, Section 5], and we described them previously in Propositions 3.1.1, 3.1.2 and 4.2.1. For completeness, we restate these results below.

Proposition 9.8.1 ([BDPV07])**.** *Let $b, c, r, \kappa, \nu \in \mathbb{N}^*$ with $b = r + c$. Consider the* Ascon-XOF *mode of Section 9.8.1 with parameters b, c, r. There exists an adversary $\mathcal{A}$ with $\mathcal{N} \approx 2^{c/2}$, such that for any simulator* **S**,

$$\mathbf{Adv}^{\mathrm{indif}}_{\text{Ascon-XOF},\mathbf{S}}(\mathcal{A}) \approx 1\,.$$

In addition, there exist adversaries $\mathcal{A}^{\mathrm{col}}_1$ with $\mathcal{N} \approx 2^{c/2}$, $\mathcal{A}^{\mathrm{col}}_2$ with $\mathcal{N} \approx 2^{\nu/2}$, $\mathcal{A}^{\mathrm{pre}}_1$ with $\mathcal{N} \approx \max\{2^{c/2}, 2^{\nu-r}\}$, $\mathcal{A}^{\mathrm{pre}}_2$ with $\mathcal{N} \approx 2^{\nu}$, $\mathcal{A}^{\mathrm{sec}}_1$ with $\mathcal{N} \approx 2^{c/2}$, and $\mathcal{A}^{\mathrm{sec}}_2$ with $\mathcal{N} \approx 2^{\nu}$, such that

$$\begin{aligned}
\mathbf{Adv}^{\mathrm{col}[\nu]}_{\text{Ascon-XOF}}\left(\mathcal{A}^{\mathrm{col}}_1\right), \mathbf{Adv}^{\mathrm{col}[\nu]}_{\text{Ascon-XOF}}\left(\mathcal{A}^{\mathrm{col}}_2\right) &\approx 1\,,\\
\mathbf{Adv}^{\mathrm{ePre}[\nu]}_{\text{Ascon-XOF}}\left(\mathcal{A}^{\mathrm{pre}}_1\right), \mathbf{Adv}^{\mathrm{ePre}[\nu]}_{\text{Ascon-XOF}}\left(\mathcal{A}^{\mathrm{pre}}_2\right) &\approx 1\,,\\
\mathbf{Adv}^{\mathrm{eSec}[\kappa,\nu]}_{\text{Ascon-XOF}}\left(\mathcal{A}^{\mathrm{sec}}_1\right), \mathbf{Adv}^{\mathrm{eSec}[\kappa,\nu]}_{\text{Ascon-XOF}}\left(\mathcal{A}^{\mathrm{sec}}_2\right) &\approx 1\,.
\end{aligned}$$

9.9 Ascon-PRF Mode and Its Security

We describe the mode underlying Ascon-PRF in Section 9.9.1, the security model in Section 9.9.2, and the security of the construction in Section 9.9.3.

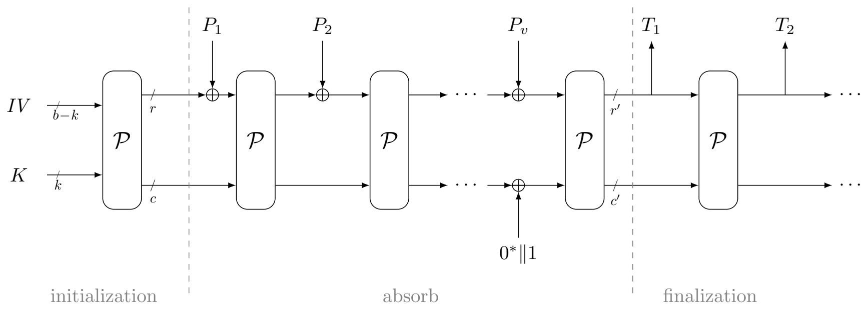


Figure 9.5: The Ascon-PRF mode of operation. Here, P is injectively padded as $(P_1, \ldots, P_v) \leftarrow pad_r^{10^*}(P)$.

9.9.1 Description of the Mode

The construction underlying Ascon-PRF is a tweaked version of the full-state keyed sponge (cf., Section 3.2.2).[14] Let $b, c, r, c', r', k \in \mathbb{N}^*$ such that $b = r + c = r' + c'$, $k \leq b$, and let $\mathcal{P}$ be a cryptographic permutation over b bits. The function Ascon-PRF$^{\mathcal{P}}$ takes as input a key $K \in \{0,1\}^k$, a plaintext $P \in \{0,1\}^*$, and a length $\nu \in \mathbb{N}^*$. It returns a tag $T \in \{0,1\}^*$ with $|T| = \nu$. We have

$$\begin{aligned} \text{Ascon-PRF}_K^{\mathcal{P}} : \{0,1\}^* \times \mathbb{N}^* &\longrightarrow \{0,1\}^* \,, \\ (P, \nu) &\longrightarrow T \in \{0,1\}^\nu \,. \end{aligned}$$

The most notable difference with the full-state keyed sponge is that Ascon-PRF features domain separation between the absorption phase and the squeezing phase. The construction is illustrated in Figure 9.5. Ascon-PRF is suggested to be instantiated with key size $k = 128$, absorption capacity and rate $(c, r) = (64, 256)$, and squeezing capacity and rate $(c', r') = (192, 128)$.

[14]The Ascon-PRF specification also comes with a small-input variant Ascon-PRFshort, that is basically a truncated permutation construction with key blinding.

9.9.2 Security Model

We consider plain multi-user PRF security for Ascon-PRF. We adapt Definition 2.4.5 to the context of Ascon-PRF.

Definition 9.9.1. *Consider the* Ascon-PRF *mode of Section 9.9.1. Let* $(\$_m)_{m=1}^{\mu}$ *be a family of* μ *independent random functions,* $\mathcal{P} \xleftarrow{\$} \texttt{Perm}(b)$*, and* $K_1, \ldots, K_\mu \xleftarrow{\$} \{0,1\}^k$*. The PRF security of* Ascon-PRF *against an adversary* $\mathcal{A}$ *is defined as*

$$\mathbf{Adv}^{\mu\text{-prf}}_{\text{Ascon-PRF}}(\mathcal{A}) = \Delta_{\mathcal{A}}\left(\left(\text{Ascon-PRF}^{\mathcal{P}}_{K_m}\right)_{m=1}^{\mu}, \mathcal{P}^{\pm} \; ; \; (\$_m)_{m=1}^{\mu}, \mathcal{P}^{\pm}\right) .$$

9.9.3 Overview

Ascon-PRF is basically a full-state keyed sponge. Therefore, as discussed in Section 3.2.2.2, a security bound of Ascon-PRF follows from the analysis of Daemen et al. [DMV17]. However, in our terminology, their bound (stated with a big-O notation in (3.9)) has a term of the form $\mathcal{MN}/2^{c'}$, but the attack matching this term is actually thwarted by the domain separation between absorption and squeezing in Ascon-PRF. Because of this, Mennink [Men23] dived into the existing duplex proofs [DMV17, DM19b] and improved them to obtain a bound specifically tailored to Ascon-PRF. However, with respect to multicollision handling, their proofs adopt the strategy (2.5) in Section 2.5.2, whereas the analysis of Ascon-AE of Sections 9.4 and 9.5 follows the strategy (2.6). We thus take the bound of Mennink, but adapt it by taking the second multicollision strategy. (This boils down to replacing the fraction $2\theta(\mathcal{N}+1)/2^{c'}$ by $2\theta\mathcal{N}/2^{c'}$, though for a different meaning of the value θ.)

Theorem 9.9.1 ([Men23])**.** *Let* $b, c, r, c', r', k \in \mathbb{N}^*$ *with* $b = r + c = r' + c'$ *and* $k \leq b$*. Let* $\mathcal{N}, \mathcal{M}, Q \in \mathbb{N}$*. Consider the* Ascon-PRF *mode of Section 9.9.1 with parameters* b, c, r, c', r', k*. Let* $\mathcal{A}$ *be an adversary with complexity* $(\mathcal{N}, Q, \mathcal{M})$*. We have*

$$\mathbf{Adv}^{\mu\text{-prf}}_{\text{Ascon-PRF}}(\mathcal{A}) \leq \frac{2\text{mucol}(\mathcal{M}, 2^{r'})\mathcal{N}}{2^{c'}} + \frac{(\mathcal{M}-Q)Q}{2^b - Q} + \frac{2\binom{\mathcal{M}}{2}}{2^b} + \frac{Q(\mathcal{M}-Q)}{2^{\min\{c'+k, b\}}} + \frac{\mu\mathcal{N}}{2^k} + \frac{\binom{\mu}{2}}{2^k} .$$

Using that $\mu \leq \mathcal{M} \ll \mathcal{N}$ (cf., Section 9.3.2) and $\text{mucol}(q, R) = \tilde{\mathcal{O}}\left(1 + \frac{q}{R}\right)$, we obtain a bound of the order

$$\mathbf{Adv}^{\mu\text{-prf}}_{\text{Ascon-PRF}}(\mathcal{A}) = \tilde{\mathcal{O}}\left(\frac{\mu\mathcal{N}}{2^k} + \frac{\mathcal{N}}{2^{c'}} + \frac{\mathcal{MN}}{2^b}\right) . \tag{9.37}$$

The bound (9.37) is tight. The same type of attacks as those in Proposition 9.4.1 apply here, as we show in Proposition 9.9.1.

Proposition 9.9.1. *Let $b, c, r, c', r', k \in \mathbb{N}^*$ with $b = r + c = r' + c'$ and $k \leq c$. Consider the* Ascon-PRF *mode of Section 9.9.1 with parameters b, c, r, c', r', k. There exist adversaries $\mathcal{A}_1$ with $\mathcal{N} \approx 2^k/\mu$, and $\mathcal{A}_2$ with $(\mathcal{M}+2^{r'})\mathcal{N} \approx 2^b$, such that*

$$\mathbf{Adv}^{\mu\text{-prf}}_{\text{Ascon-PRF}}(\mathcal{A}_1)\,,\ \mathbf{Adv}^{\mu\text{-prf}}_{\text{Ascon-PRF}}(\mathcal{A}_2) \approx 1\,.$$

Proof. The adversary $\mathcal{A}_1$ acts similarly to one described in Section 9.7.1.2: it tries to guess one of the user's keys by doing permutation queries, and checks them against the outputs of each of the μ Ascon-PRF instances to determine whether one of its guesses is correct. Similarly, adversary $\mathcal{A}_2$ follows the strategy from Section 9.7.1.3, as it aims to guess an internal state. This internal state must be during the squeezing phase, and successfully guessing it directly implies key recovery. □

9.10 Conclusion

We presented a general discussion of existing, and new, results on the generic security of the Ascon-AE authenticated encryption scheme, as well as the Ascon-Hash/Ascon-(C)XOF hash/(C)XOF function, and the Ascon-PRF pseudorandom function. In particular, we investigated formally what happens in Ascon if an inner state is recovered by the adversary. Using our tailormade definition, we proved that Ascon even guarantees authenticity in this case, unlike typical duplex-style authenticated encryption modes that only use the key upon initialization. On the way, we observed that, even though state-of-the-art appeared quite broad at first sight, there were still many gaps to be filled. In particular, some existing results were not entirely correct/accurate, and results in leaky settings.

9.10.1 What We Do Not Cover

Even though our treatment is rather broad, we do not cover *all* possible security models. We elaborate on three directions that we do not cover:

- We do not cover related-key security. We do, in fact, consider multi-user security, where the adversary has access to μ instances of the scheme, but we assume independence of these μ keys. If we had stretched the analysis to arbitrary distributions, e.g., as in Daemen et al. [DMV17, Section 2.1], this would imply *some forms of* related-key security, but this would significantly add to the complexity of the proofs;

- While we cover leakage resilience in Section 9.5.1, we do not cover security under fault attacks, where the adversary may introduce faults in the implementation and that way retrieve secret information. Fruitful starting points for the analysis of Ascon-AE in this setting are [DMP22, SKP22, BGP+23];
- There has been significant recent interest in committing security of authenticated encryption schemes, where the adversary has freedom in choosing the keys [GLR17, LGR21, ADG+22, CR22, BH22]. Clearly, there is a relation between hash function security and key committing security of authenticated encryption schemes, but the blinding of the keys in Ascon-AE makes this relation not directly applicable. Naito et al. [NSS23] proved key committing security of the Ascon-AE mode with zero-padding, where security (in part) depends on the size of the zero-padding. An alternative security proof up to a comparable bound is given by Krämer et al. [KSW24].

In addition, our analysis does not cover variations to the Ascon-AE scheme of Section 9.2:

- The standard published by NIST [SMC+25] also specifies an option to implement Ascon-AE with nonce masking, where the nonce gets blinded with additional key material, following Dobraunig and Mennink [DM24]. The goal of this mechanism is to enhance the multi-user security of the mode, i.e., to replace the term $\frac{\mu \mathcal{N}}{2^k}$ with $\frac{\mathcal{N}}{2^k}$. We do not cover this mechanism in our proofs, but remark that the change only affects the initialization, that Dobraunig and Mennink proposed their mechanisms as extension to the original duplex proofs [DMV17, DM19a], and that our results may similarly be generalized by an isolated analysis;
- The encryption in duplex-based authenticated encryption, and Ascon-AE in particular, consists of bitwise addition of plaintext into the outer part, but upon decryption, the adversary chooses the ciphertext and has free choice in selecting the r outer bits of the state. This gives the adversary additional power, especially in settings such as release of unverified plaintext (Section 9.5.3). One way to mitigate this decrease in security is to apply a more advanced mechanism to absorb plaintext and squeeze ciphertext, e.g., following the approach of Beetle [CDNY18]. A general

analysis of security of Beetle-style authenticated encryption was given by Chakraborty et al. [CJN20]. However, such mechanism cannot be implemented with Ascon-AE as a black-box (unlike above variation); instead, it would basically be a different scheme.

9.10.2 Future Research

Finally, we wish to point out two directions where our models and assumptions limit, and where further research on Ascon-AE would be worthwhile:

- For leakage resilience (Section 9.5.1), we started from the notion of nonce-misuse resilience, and we additionally covered any possible leakage function that is independent of the permutation $\mathcal{P}$. These two modeling decisions, together, implied that limited and unlimited leakage are equivalent. However, while the first modeling decision is rather fair, the second modeling decision is fairly liberal as in practice, an adversary cannot freely choose the leakage function. Therefore, *for specific leakage functions*, we expect much better security bounds. For example, if we limit the leakage function to leak the Hamming weight of the first byte, the security bound may be closer to security with no leakage than security with unlimited leakage. However, performing security analysis for more complex leakage functions is very subtle and technical. For example, Berendsen and Mennink [BM24] considered the security of the suffix keyed sponge under Hamming weight leakage, and it seems not trivial to generalize our analysis (of Theorem 9.5.1) to Hamming weight leakage, nor to generalize the analysis of [BM24] on the suffix keyed sponge to Ascon-AE;

- We can be reasonably confident that the Ascon permutation (and, in the context of Ascon-AE, the outer permutation $\mathcal{P}_\mathrm{o}$ in particular) is sufficiently strong. That said, they are definitely not perfectly random, and the more one considers round-reduced variants (e.g., the inner permutation $\mathcal{P}_\mathrm{i}$ versus the outer permutation $\mathcal{P}_\mathrm{o}$), the more one diverges from this assumption. Harshly said, our results do not apply to the actual Ascon schemes. For instance, Baudrin et al. [BCP22] performed a practical attack against the actual Ascon authenticated encryption scheme in the setting of a nonce-misusing adversary to recover the internal state (after the initialization), with a complexity of around 2^{40}. This attack exploits weaknesses of the underlying permutation. However, our results do give some certainty, namely that no generic attacks are possible beyond the proven bounds, and also state-recovery security says that some level of security is still achieved even if one of the inner states leaks. Having said that, it is a very interesting research question to try to prove security of Ascon under weaker assumptions on the permutation.

Chapter 10

Kirby and MacaKey: Sponge-Based Pseudorandom Functions with Enhanced Squeezing Rate

10.1 Introduction

In Section 3.2, we presented various sponge-based PRF constructions, in particular the full-state keyed sponge [MRV15] (FSKS) in Section 3.2.2. It is one of the most optimized variants that absorbs *over the entire state*, and in Section 3.2.2.3 we discussed several refinements of FSKS to maximize security. Nevertheless, even with these optimizations, the output rate of sponge-based PRFs is limited to r bits at a time, with maximal generic security up to $\min\left\{2^{b-r}, 2^k\right\}$ permutation queries, where b is the permutation size, and k the key size.

In the broader landscape of permutation-based cryptography, several dedicated designs achieve *full-state squeezing*. Early examples include the stream cipher family Salsa [Ber08b], its extended-nonce version XSalsa [Ber11], as well as its variant ChaCha [Ber08a], all of which use a feed-forward before outputting the state, as in the Davies-Meyer construction employed in Merkle-Damgård hashing [Mer89,Dam89,PGV93]. An alternative approach is to blind the state with a key, such as Chaskey [MMH$^+$14]. Those designs are limited to a fixed output length of b bits (i.e., one squeeze). A more recent permutation-based construction that combines the advantages of Salsa/ChaCha and sponge/duplex is Farfalle [BDH+17]. It builds a so-called doubly extendable cryptographic keyed (deck) function that supports both variable-length input and output.

10.1.1 Contributions

We propose in this chapter two novel permutation-based PRF constructions that squeeze *over the entire state*: Kirby and MacaKey.

Kirby. Kirby is a variable-input-length PRF construction inspired by the sponge construction and Salsa, and is illustrated in Figure 10.1. Kirby can be seen as a generalization of the construction underlying the stream cipher family Salsa, with some subtle differences explained in Section 10.1.2. With Kirby, full-state squeezing is possible thanks to continuous feed-forward, which also allows for strong security guarantees: the leakage of a state does not compromise the recovery of the key or earlier states.[1]

To enhance multi-user security, Kirby adopts a flexible state initialization mechanism using identifiers, as was seen in several prior works, e.g., [DHP+20, BBN22,DM24]. We prove tight generic security of Kirby up to $\min\left\{2^b/\mathcal{M}, 2^k\right\}$ permutation evaluations, where b denotes the permutation size, k the key size, and $\mathcal{M}$ the online complexity.

MacaKey. MacaKey is a variable-input-length and variable-output-length PRF construction, and is illustrated in Figure 10.2. The scheme combines ideas of the full-state keyed sponge with those of the summation-truncation hybrid of Gunsing and Mennink [GM20]. The core observation in Gunsing and Mennink's approach is that, when one considers the truncation of a secret permutation with output size of $a < b$ bits per call, one can pairwise group the evaluations and XOR the truncated parts together, allowing to output $(b-a)$ extra bits *without sacrificing security.* The authors also suggest that this method can be extended further by grouping more permutation evaluations, similar to the CENC construction [Iwa06]. Building on this idea, MacaKey integrates this summation-truncation technique into the squeezing part of FSKS. More precisely, the inner parts during squeezing are summed

[1] We omit the leakage-resilience proof in this thesis; this chapter instead focuses on the two constructions. The security model and a proof of leakage resilience can be found in [LBD23].

pairwise in a CENC fashion before being output. This increases the throughput to b bits per permutation call during the squeezing phase.

We prove generic security of MacaKey in the multi-user setting, with a flexible state initialization mechanism based on identifiers, as with Kirby. We obtain a tight security bound, comparable to that of FSKS with domain separation (cf. Section 3.2.2.3).

Comparison. While Kirby and MacaKey share similarities, they are motivated by different design goals. Kirby aims for simplicity and suitability for lightweight implementations. On the other hand, MacaKey is designed to improve over FSKS, and is more general than Kirby, as it supports arbitrarily long output lengths. We provide a detailed comparison in Section 10.4.

10.1.2 Comparison to Existing Permutation-Based PRFs

It is relevant to compare Kirby and MacaKey against modes of operation instantiated with the Even-Mansour construction [EM91,EM97] (EM). In the context of keystream generation, natural comparisons are with OFB or CTR modes [Dwo01] instantiated with EM (referred to as OFB-EM and CTR-EM, respectively). Although EM-based constructions can output the entire state, they fundamentally differ in how they handle key material and in their security bounds. Indeed, EM blinds both the input and the output with the key, and guarantees birthday-bound security *in the key size*, or in other words security is guaranteed up to approximately $2^k/\mathcal{M}$ permutation calls. In contrast, Kirby and MacaKey (and keyed sponges in general) inject the key only at the first permutation call, and achieve linear security in the key size and birthday bound security in the capacity or permutation size, respectively. For example, in a single-user scenario with a maximal online complexity $\mathcal{M}$ of 2^{64}, achieving 128-bit security with EM would require keys as large as 192 bits. In contrast, MacaKey and Kirby can maintain the desired security level with a key of size 128 bits with the same permutation size as EM.

Other EM-based constructions, such as SoEM [CLM19], take a fundamentally different approach by aiming to achieve security beyond the birthday bound. However, this comes at the cost of significantly increasing the number of permutation calls.

Kirby. Kirby has a significantly smaller memory footprint than Farfalle [BDH+17]. Over sponge and duplex, Kirby has the advantage that it offers full-state squeezing, and over Salsa it has the advantage that it supports arbitrarily long inputs. Although Kirby is more general than (X)Salsa, it also differs from it in several other important aspects. Firstly, Kirby systematically applies a feed-forward with bitwise addition after each permutation call, making it leakage resilient. Additionally, XSalsa overwrites a part of the state with the data to absorb, while with Kirby the diversifier is added block by block to the state, allowing thus full-state absorption. Lastly, Kirby is designed with multi-user security in mind, and the security analysis covers a wide range of state initialization mechanisms.

MacaKey. MacaKey's main advantage over Kirby is its ability to produce large outputs. It also builds directly on FSKS to improve its throughput. Indeed, MacaKey is more efficient than FSKS with no security cost. Whereas this improvement has a limited impact if the output length is required to be short anyway (e.g., a tag), it becomes much more significant for applications like keystream generation. Consider, for example, the size of the permutation underlying Ascon [DEMS21b], i.e., $b = 320$, and take $c = 256$. Compared to FSKS, the throughput of MacaKey with those parameter sizes is increased by 400%. Moreover, instead of FSKS, one could also use, in an OKS fashion (see Section 3.2.1), a variant of the sponge that absorbs r bits at a time and squeezes at a rate of $r + c/2$ bits [GPP11, NO14] (see also Section 3.1.1). In this context, MacaKey provides a 66% improvement in throughput during the squeezing phase, all while supporting full-state absorption. Refer to Table 10.1 for a detailed comparison.

On Earlier Versions of MacaKey. An earlier version of this work used the same permutation during the absorbing and squeezing phase. This version was attacked by Chakraborty and Dhar [CD25], showing that the original scheme was insecure. The root cause is that the internal state immediately before the first squeeze was used as a mask on outputs. This lets an adversary reabsorb previously output blocks in a way that can force collisions between an intermediate state during absorption and a state during squeezing. On the proof side, the flaw comes from the distance between RS and RO (cf., Section 10.6.3), where it was missed that the set of absorbing and squeezing states could be non-disjoint, and the intermediate states released in the RO world ignored that aspect. As a result, the bad event **IColl** could trivially be set in the RS world. The issue can be corrected by enforcing a strong separation between the absorbing and squeezing phases. In this thesis, we use two different permutations during the absorbing and squeezing phases. The rationale for this choice is discussed in more detail in Section 10.3.1.

Table 10.1: Comparison of MacaKey with previous constructions. Security bounds are expressed in big-O notation and in the single-user setting, where $\mathcal{N}$ and $\mathcal{M}$ denote the offline and online complexities, respectively.

Design	Throughput	Security	Reference
FSKS	$\frac{r}{b}$	$\frac{\mathcal{M}\mathcal{N}}{2^c} + \frac{\mathcal{N}}{2^k}$	[DMV17]
PHOTON	$\frac{r+c/2}{b}$	at best $\frac{\mathcal{M}\mathcal{N}}{2^c} + \frac{\mathcal{N}}{2^k}$[2]	[GPP11, NO14]
OFB-EM, CTR-EM	1	$\frac{\mathcal{M}\mathcal{N}}{2^k}$	[EM91, EM97]
MacaKey	1	$\frac{\mathcal{N}}{2^c} + \frac{\mathcal{M}\mathcal{N}}{2^b} + \frac{\mathcal{N}}{2^k}$	[This work]

10.1.3 Outline

Sections 10.2 and 10.3 introduce respectively Kirby and MacaKey. Section 10.4 compares and details possible use cases of both constructions. Sections 10.5 and 10.6 are dedicated to security proofs, and finally, Section 10.7 concludes.

10.2 Kirby Construction

10.2.1 Specification

Let $k, b, \mu \in \mathbb{N}^*$ such that $k \leq b$. Let $\mathcal{P} \in \texttt{Perm}(b)$ be a cryptographic permutation, and let

$$pad_b^{\mathrm{PF}} : \{0,1\}^* \to \left(\{0,1\}^b\right)^*$$

be a prefix-free padding function. Moreover, let $\boldsymbol{IV}$ be a public array of μ elements from $\{0,1\}^{b-k}$, and $\boldsymbol{K}$ an array of μ elements from $\{0,1\}^k$. The construction Kirby based on the key array $\boldsymbol{K}$, the IVs $\boldsymbol{IV}$, and the permutation $\mathcal{P}$, takes as input an index $m \in [\![1, \mu]\!]$, a message $M \in \{0,1\}^*$, and it outputs a string $Z \in \{0,1\}^b$:

$$\begin{aligned} \mathrm{Kirby}^{\mathcal{P}}_{\boldsymbol{K},\boldsymbol{IV}} : [\![1,\mu]\!] \times \{0,1\}^* &\longrightarrow \{0,1\}^b \\ (m, M) &\longrightarrow Z\,. \end{aligned}$$

The algorithmic details of Kirby are provided in Algorithm 24, with an illustration in Figure 10.1. At a high level, Kirby resembles FSKS, but it (i)

[2]This bound has not been formally derived, but by analogy with the outer-keyed sponge construction (cf., Section 3.2.1), we conjecture the same sort of bounds.

replaces the permutation evaluation $\mathcal{P}(S)$ with a permutation evaluation and a feed-forward $S \oplus \mathcal{P}(S)$, which allows Kirby to (ii) output the *entire* b-bit state. Because of (ii), in order to avoid length-extension attacks, the padding function pad_b^{PF} must be prefix-free, and the output is limited to b bits.

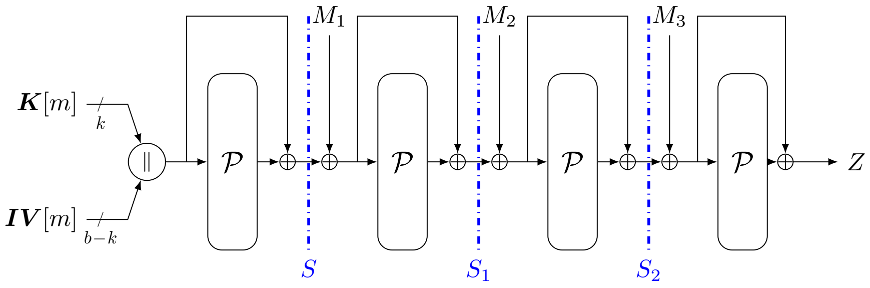


Figure 10.1: Illustration of the Kirby construction with intermediate states. Here, the input is padded as $(M_1, M_2, M_3) \leftarrow pad_b^{\mathrm{PF}}(M)$.

Algorithm 24 Kirby construction. pad_b^{PF} is a prefix-free padding function.

```
1: function $\mathrm{Kirby}^{\mathcal{P}}_{\boldsymbol{K},\boldsymbol{IV}}(m, M)$
       input: $(m, M) \in [\![1, \mu]\!] \times \{0,1\}^*$
       output: $Z \in \{0,1\}^b$
2:     $(M_1, \ldots, M_\ell) \leftarrow pad_b^{\mathrm{PF}}(M)$
3:     $S \leftarrow \boldsymbol{K}[m] \| \boldsymbol{IV}[m]$ // State of the Kirby construction
4:     $S \leftarrow S \oplus \mathcal{P}(S)$ // Initialization
5:     for $i = 1$ to $\ell$ do
6:         $S \leftarrow S \oplus M_i$
7:         $S \leftarrow S \oplus \mathcal{P}(S)$
8:     return $S$
```

10.2.2 Security

We consider multi-user PRF security as in Definition 2.4.5. In the distinguishing game, the real world is implemented as

$$W_R := \left(\mathrm{Kirby}^{\mathcal{P}}_{\boldsymbol{K},\boldsymbol{IV}}, \mathcal{P}\right) ,$$

where $\mathcal{P} \xleftarrow{\$} \texttt{Perm}(b)$ and $\boldsymbol{K}[1], \ldots, \boldsymbol{K}[\mu] \xleftarrow{\$} \{0,1\}^k$, and the ideal world as

$$W_I := (\mathcal{RO}^*, \mathcal{P}) ,$$

where $\mathcal{P} \xleftarrow{\$} \texttt{Perm}(b)$ and $\mathcal{RO}^* : [\![1, \mu]\!] \times \{0,1\}^* \to \{0,1\}^b$ is a random oracle. Moreover, we will need to describe $\boldsymbol{IV}$ using two particular metrics. To facilitate this, let us represent $\boldsymbol{IV}$ with two lists, L_1 and L_2, defined as follows: L_1 contains all elements appearing in $\boldsymbol{IV}$ without any repetition. L_2, having the same length as L_1, records the *multiplicity* of each element. In other words, the s^{th} element in L_1 appears $L_2[s]$ times in $\boldsymbol{IV}$. From this, we define:

$$u_s^{\boldsymbol{IV}} = L_2[s] , \qquad u_{\max}^{\boldsymbol{IV}} = \max_s u_s^{\boldsymbol{IV}} .$$

We measure the adversary's query resources similarly to Section 3.2.1.2:

- $\mathcal{N}$ denotes the total number of distinct queries to the primitive oracle;
- $\mathcal{M}$ denotes the number of *non-duplicate* queries to the construction oracles, measured as the minimal number of permutation calls that these construction queries would induce with the construction oracle in the real world.

From this, we define the PRF advantage of Kirby as:

$$\mathbf{Adv}_{\text{Kirby}}^{\mu\text{-prf}}(\boldsymbol{IV}, \mathcal{N}, \mathcal{M}) = \max_{\mathcal{A}} \mathbf{Adv}_{\text{Kirby}}^{\mu\text{-prf}}(\mathcal{A}) ,$$

where the maximum is taken over all adversaries $\mathcal{A}$ having access to $\mu = |\boldsymbol{IV}|$ instances of Kirby with identifiers spread according to $\boldsymbol{IV}$, making at most $\mathcal{N}$ permutation queries, and construction queries with a cost of at most $\mathcal{M}$.

We bound the quantity $\mathbf{Adv}_{\text{Kirby}}^{\mu\text{-prf}}(\boldsymbol{IV}, \mathcal{N}, \mathcal{M})$ in the following theorem.

Theorem 10.2.1. *Let $b, k \in \mathbb{N}^*$ be such that $k \leq b$. Let $\mu, \mathcal{M}, \mathcal{N} \in \mathbb{N}^*$. Moreover, let $\boldsymbol{IV}$ be an array of μ elements from $\{0,1\}^{b-k}$. Consider the construction* Kirby *of Algorithm 24 with parameters b, k, and with a prefix-free padding function. We have*

$$\mathbf{Adv}_{\text{Kirby}}^{\mu\text{-prf}}(\boldsymbol{IV}, \mathcal{N}, \mathcal{M}) \leq \frac{\sum_s u_s^{\boldsymbol{IV}}(u_s^{\boldsymbol{IV}} - 1)}{2^{k+1}} + \frac{\mathcal{N} \cdot u_{\max}^{\boldsymbol{IV}}}{2^k} + \frac{\mathcal{M}(\mathcal{M}-1)}{2^b} + \frac{2\mathcal{N}\mathcal{M}}{2^b} .$$

Interpretation of the Bound. This security bound captures a wide range of use cases regarding the identifiers. In particular, when all identifiers are distinct, the bound simplifies to

$$\mathbf{Adv}_{\text{Kirby}}^{\mu\text{-prf}}(\boldsymbol{IV}, \mathcal{N}, \mathcal{M}) \leq \frac{\mathcal{N}}{2^k} + \frac{\mathcal{M}(\mathcal{M}-1)}{2^b} + \frac{2\mathcal{N}\mathcal{M}}{2^b} . \qquad (10.1)$$

Assuming that $\mathcal{M} \leq 2^{100}$, Kirby is secure as long as $\mathcal{N} \ll \min\left\{2^k, 2^{b-100}\right\}$ and $b \geq 200$. In that case, if we take $b = 256$ and $k = 128$, we obtain 128 bits of security.

On the other hand, when no distinct identifier is used, the bound becomes

$$\mathbf{Adv}^{\mu\text{-prf}}_{\text{Kirby}}(\boldsymbol{IV}, \mathcal{N}, \mathcal{M}) \leq \frac{\mu(\mu-1)}{2^{k+1}} + \frac{\mathcal{N}\mu}{2^k} + \frac{\mathcal{M}(\mathcal{M}-1)}{2^b} + \frac{2\mathcal{N}\mathcal{M}}{2^b} \, .$$

Compared to (10.1), there is a security degradation in the key length compared to the case where all identifiers are distinct. More precisely, targeting the same security strength, the key length may need to be increased by up to $\log(\mu)$ bits, and the key length should be larger than $2\log(\mu)$. For example, with the same assumptions as previously, if aiming for an equivalent security strength (i.e., 128 bits), the key length should be increased to at least 192.

The bound is tight, and attacks similar to the ones described in Section 9.7.2 apply.

10.3 MacaKey Construction

10.3.1 Specification

Let $k, b, r, c \in \mathbb{N}^*$ such that $b = c + r$ and $k \leq b$. Let $\mathcal{P}, \mathcal{P}' \in \texttt{Perm}(b)$ be two cryptographic permutations, and let

$$\begin{aligned} pad_b : \{0,1\}^* &\longrightarrow (\{0,1\}^b)^* \\ M &\longrightarrow M_1 \| \cdots \| M_u \, , \end{aligned}$$

be a sponge-compliant padding function (cf., Section 3.1.1) Moreover, let $\boldsymbol{IV}$ be a public array of μ elements from $\{0,1\}^{b-k}$, and $\boldsymbol{K}$ an array of μ elements from $\{0,1\}^k$. The construction MacaKey based on the key array $\boldsymbol{K}$, the IVs $\boldsymbol{IV}$, and the permutations $\mathcal{P}$ and $\mathcal{P}'$, takes as input an index $m \in [\![1, \mu]\!]$, a message $M \in \{0,1\}^*$, the output length $\nu \in \mathbb{N}^*$, and it outputs a string $Z \in \{0,1\}^\nu$:

$$\begin{aligned} \text{MacaKey}^{\mathcal{P},\mathcal{P}'}_{\boldsymbol{K},\boldsymbol{IV}} : [\![1, \mu]\!] \times \{0,1\}^* \times \mathbb{N}^* &\longrightarrow \{0,1\}^* \\ (m, M, \nu) &\longrightarrow Z \in \{0,1\}^\nu \, . \end{aligned}$$

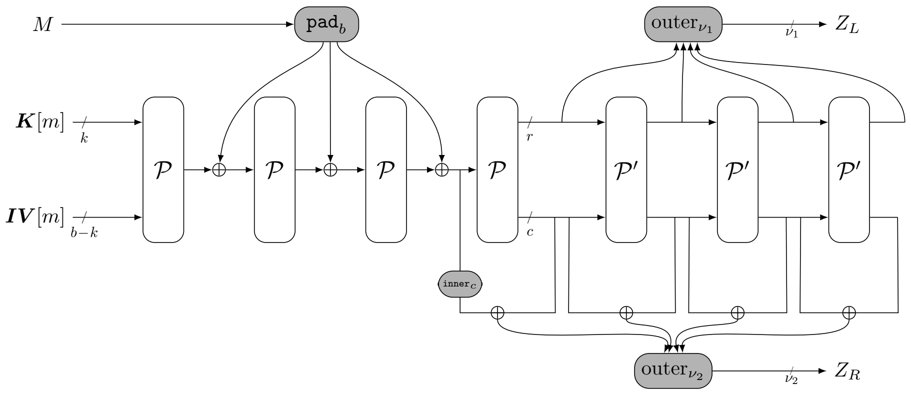


Figure 10.2: MacaKey construction. The output string is $Z_L \| Z_R$.

Algorithm 25 MacaKey construction.

```
1:  function $\mathrm{MacaKey}^{\mathcal{P},\mathcal{P}'}_{\boldsymbol{K},\boldsymbol{IV}}(m, M, \nu)$
        input: $(m, M, \nu) \in [\![1, \mu]\!] \times \{0,1\}^* \times \mathbb{N}^*$
        output: $Z \in \{0,1\}^\nu$
2:      $S \leftarrow \boldsymbol{K}[m] \| \boldsymbol{IV}[m]$
3:      $S \leftarrow \mathcal{P}(\boldsymbol{K}[m] \ \| \ \boldsymbol{IV}[m])$ // State of the MacaKey construction
4:      $(M_1, \dots, M_\ell) \leftarrow pad_b(M)$
5:      for $1 \le i \le \ell$ do // Absorb
6:          $Y_R \leftarrow \mathrm{inner}_c(S)$ // XORing base string
7:          $S \leftarrow \mathcal{P}(S \oplus M_i)$
8:      $Z_L \leftarrow \epsilon$ // Leftmost part of output string
9:      $Z_R \leftarrow \epsilon$ // Rightmost part of output string
10:     for $1 \le i \le \lceil \frac{\nu}{b} \rceil$ do //  Squeeze
11:         $Z_L \leftarrow Z_L \ \| \ \mathrm{outer}_r(S)$
12:         $Z_R \leftarrow Z_R \ \| \ (\mathrm{inner}_c(S) \oplus Y_R)$
13:         $Y_R \leftarrow \mathrm{inner}_c(S)$
14:         $S \leftarrow \mathcal{P}'(S)$
15:     $Z \leftarrow Z_L \ \| \ Z_R$
16:     return $Z[1 : \nu]$
```

The algorithmic details of MacaKey are provided in Algorithm 25, with an illustration in Figure 10.2. At a high level, MacaKey builds on FSKS by augmenting the squeezing phase through the integration of the summation-truncation hybrid technique. This is done by XORing together two consecutive inner states, and appending them to the output. This process can also be implemented in a CENC fashion [Iwa06], where the state before the squeezing phase is XORed with all subsequent states. Both approaches can be seen as equivalent, since the output contains the same information, i.e., the output of one can be reconstructed from the output of the other.

Using two different permutations during the absorbing and squeezing phases prevents that the adversary makes use of the collisions that Chakraborty and Dhar exploited in their attack [CD25]. Alternatively, a similar effect can be achieved by using the same permutation but reserving the lowest bit of the input to indicate 0 for absorbing and 1 for squeezing, as done, e.g., in Section 6.3.5, or by employing carefully chosen non-cryptographic permutations, as done in Chapter 8. Due to this strict separation during absorbing and squeezing phases, the adversary cannot overwrite the outer part of a state anymore. This improves the security, similarly to the guarantees achieved by `Ascon-PRF` (see Section 3.2.2.3).

Parametrization of r. Looking ahead (cf., Theorem 10.3.1), maximum security is reached when $r = 0$. This case is illustrated in Figure 10.3, where each output block is the XOR of two consecutive states. Thus, from a purely generic security viewpoint, $r = 0$ is optimal. However, we kept the analysis general because the memory overhead is c bits, hence choosing a larger r might make sense in memory-constrained applications, or whenever b is itself large enough to exceed the target security level.

10.3.2 Security

We consider multi-user PRF security as in Definition 2.4.5. In the distinguishing game, the real world is implemented as

$$W_R := \left(\text{MacaKey}^{\mathcal{P},\mathcal{P}'}_{\boldsymbol{K},\boldsymbol{IV}}, \mathcal{P}, \mathcal{P}'\right) ,$$

where $\mathcal{P}, \mathcal{P}' \xleftarrow{\$} \texttt{Perm}(b)$ and $\boldsymbol{K}[1], \ldots, \boldsymbol{K}[\mu] \xleftarrow{\$} \{0,1\}^k$, and the ideal world as

$$W_I := (\mathcal{RO}, \mathcal{P}, \mathcal{P}') ,$$

where $\mathcal{P}, \mathcal{P}' \xleftarrow{\$} \texttt{Perm}(b)$ and $\mathcal{RO} : [\![1, \mu]\!] \times \{0,1\}^* \times \mathbb{N}^* \to \{0,1\}^*$ is a random oracle. The adversary's resources are quantified as in Section 10.2.2. Notably,

(i) we use the same notation for the array $\boldsymbol{IV}$, (ii) the number of adversarial primitive queries (to $\mathcal{P}$ and $\mathcal{P}'$ combined) is denoted by $\mathcal{N}$, and (iii) the adversary's construction queries are quantified by the minimum number of permutation evaluations needed in the real world to compute them, and is also denoted by $\mathcal{M}$.

From this, we define the PRF advantage of MacaKey as:

$$\mathbf{Adv}^{\mu\text{-prf}}_{\mathrm{MacaKey}}(\boldsymbol{IV}, \mathcal{N}, \mathcal{M}) = \max_{\mathcal{A}} \mathbf{Adv}^{\mu\text{-prf}}_{\mathrm{MacaKey}}(\mathcal{A}) \,,$$

where the maximum is taken over all adversaries $\mathcal{A}$ having access to $\mu = |\boldsymbol{IV}|$ instances of MacaKey with identifiers spread according to $\boldsymbol{IV}$, and such that $\mathcal{A}$ has complexity $(\mathcal{N}, \mathcal{M})$.

Theorem 10.3.1. *Let $b, c, r, k \in \mathbb{N}^*$ be such that $b = r + c$ and $k \leq b$. Let $\mu, \mathcal{M}, \mathcal{N} \in \mathbb{N}$. Moreover, let $\boldsymbol{IV}$ be an array of μ elements from $\{0,1\}^{b-k}$. Consider the construction* MacaKey *of Algorithm 25 with parameters b, c, r, k. We have*

$$\mathbf{Adv}^{\mu\text{-prf}}_{\mathrm{MacaKey}}(\boldsymbol{IV}, \mathcal{N}, \mathcal{M}) \leq \frac{\sum_s u^{\boldsymbol{IV}}_s (u^{\boldsymbol{IV}}_s - 1)}{2^{k+1}} + \frac{(\mathcal{N} + \mathcal{M}) u^{\boldsymbol{IV}}_{\max}}{2^k} + \frac{3\mathcal{M}^2}{2^b} + \frac{3\mathcal{N} \cdot \mathrm{mucol}(\mathcal{M}, 2^r)}{2^c} \,.$$

The security bound is of the same order as that of FSKS when overwriting the outer part of the state is forbidden (cf., Section 3.2.2.3), and similar matching attacks applies.

10.4 Comparison and Application

Kirby and MacaKey are structurally similar. In particular, when $r = 0$, MacaKey's structure (see Figure 10.3) closely resembles that of Kirby, and both achieve security bounds of the same order. In that case, the main technical difference between the two constructions lies in how the feed-forward operation is used. Kirby applies a feed-forward after every permutation call (i.e., state update is $S \leftarrow S \oplus \mathcal{P}(S)$), while MacaKey applies the permutation alone during the update phase and uses the feed-forward only when producing the output. Despite these similarities, the two constructions were developed with different goals in mind and target distinct use cases.

Kirby, on the one hand, is suitable for building efficient, lightweight cryptographic primitives with low memory footprint and/or low latency. Its potential is demonstrated by its use in Koala [ABD+24], a low-latency PRF, and Xymmer [AGD24], a deck function suitable for entry-level processors.

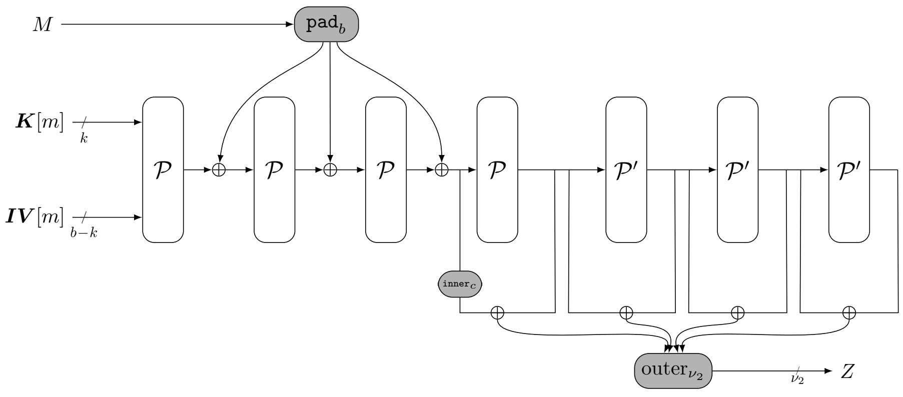


Figure 10.3: The MacaKey construction with $r = 0$.

MacaKey, on the other hand, was designed to improve over FSKS. It natively provides arbitrarily long outputs without requiring an external mode of operation. This makes MacaKey more general and well-suited for high-throughput applications, such as keystream generation, multi-key derivation, or as the core component in authenticated encryption schemes.

10.5 Proof of Theorem 10.2.1

In this section we provide a proof of Theorem 10.2.1. To do that, we will use the H-coefficient technique of Lemma 2.5.1. Let $\mathcal{A}$ be an adversary with resources as stated in the theorem. Without loss of generality, we assume that the messages queried to the construction are already padded, i.e., they are b-bit tuples such that the set of queried tuples is prefix free. Any query that does not meet this criterion will be rejected by the construction oracles. We first start to define a transcript notation, then define bad transcripts, upper bound the ratios, and bound the probability of bad transcripts in the ideal world.

Transcript Notation. In the following, we define the transcript induced by the interaction between the distinguisher and the oracles. Denote by Q the total number of construction queries. The transcript is an ordered list with $\mathcal{N} + Q$ elements. Each permutation query results in the addition of a tuple

(X, Y, d) to the transcript, where $d \in \{\mathit{fwd}, \mathit{inv}\}$ denotes the direction of the query, and $Y = \mathcal{P}(X)$. Similarly, each construction query appends to the list a tuple $(m, \text{path} = M, Z)$, where $1 \leq m \leq \mu$ refers to the user, the path M is the input block sequence absorbed, and Z is the output of the construction oracle.

For the sake of the proof, we allow the oracles to release additional information after the interactive phase, right before the distinguisher outputs its decision bit. More precisely, we define the extended transcript built from the transcript as follows:

- The elements (X, Y, d) associated to permutation queries are kept unchanged;
- The states $\boldsymbol{K}[m] \| \boldsymbol{IV}[m]$ are appended, where in the ideal world, the keys are dummy keys uniformly sampled;
- Each construction query $(m, \text{path} = (M_1, \ldots, M_\ell), Z)$ is split into $\ell + 1$ transcript elements:
    - A construction initialization element: $(m, \text{path} = \epsilon, S)$;
    - ℓ different construction absorb elements. For instance if $\ell = 4$ we have
        * $(m, \text{path} = (M_1), S)$;
        * $(m, \text{path} = (M_1, M_2), S)$;
        * $(m, \text{path} = (M_1, M_2, M_3), S)$;
        * $(m, \text{path} = (M_1, M_2, M_3, M_4), S)$.

We call a path M *final* if the construction has presented an output Z for it. A path which is not final is called *intermediate.*

Given $(m, \text{path} = M, S)$ in the (extended) transcript, the method to generate the state S varies depending on the adversary's world:

- In the real world, S is the state after having absorbed the path M with the user m. For instance, following the notation of Figure 10.1, S is the state after having absorbing ϵ, S_1 after absorbing (M_1), S_2 after absorbing (M_1, M_2), and Z after absorbing (M_1, M_2, M_3). In particular, if M is intermediate, then S is called an intermediate state;
- In the ideal world, S equals to $\mathcal{RO}^*(m, M)$.

To simplify the notation, when the path M equals ϵ, we omit it. There may be duplicates among the construction absorbs elements and these are removed from the transcript.

The construction absorb elements in the transcript can be arranged in a graph and form a forest. Each of the μ initialization elements is a root of a tree, and its nodes are the construction absorb elements, where the path is the sequence of edges one has to follow to get to the root (in reverse order). Quite naturally, the blocks of path M form the labels of the edges. For example, node $[m, \text{path} = (M_1, M_2, M_3)]$ is the parent of node $[m, \text{path} = (M_1, M_2, M_3, M_4)]$. The nodes are labeled with the state S, and we denote the state of a node reached by following the path M in tree m by $S[m, M]$. If $M \neq \epsilon$, we denote the label of the parent of the node in position $[m, M]$ by $[m, \texttt{par}(M)]$ and the last block of a path M by $M_{\texttt{las}}$. The total number of nodes in the graph is equal to $\mathcal{M}$.

One important remark is that every transcript that can be produced in the real world is also reachable in the ideal world. However, the converse is not true as the ideal world can produce intermediate states that do not conform to Kirby. Indeed, in the ideal world the permutation queries and construction queries are independent, and the intermediate states are generated randomly and uniformly, so that they might be incompatible with the bijectivity of a permutation. These transcripts are referred to as *permutation-inconsistent.*

Bad Transcripts Definition. We define here two bad events called respectively Coll and Guess, each split into sub-events. To facilitate notation, in the following, we implicitly assume the existence of the nodes listed at the beginning of each bad event.

$$\begin{aligned}
\mathsf{Coll_F} : {} & [m] \neq [m'] \text{ with } \boldsymbol{K}[m] \| \boldsymbol{IV}[m] = \boldsymbol{K}[m'] \| \boldsymbol{IV}[m'] \text{ or} \\
& [m, M] \neq [m', M'] \text{ with } S[m, \texttt{par}(M)] \oplus M_{\texttt{las}} = \\
& \qquad\qquad S[m', \texttt{par}(M')] \oplus M'_{\texttt{las}} \text{ or} \\
& [m, M], [m'] \text{ with } \boldsymbol{K}[m'] \| \boldsymbol{IV}[m'] = S[m, \texttt{par}(M)] \oplus M_{\texttt{las}}\,, \\
\mathsf{Coll_B} : {} & [m, M] \neq [m', M'] \text{ with } S[m, M] \oplus S[m, \texttt{par}(M)] \oplus M_{\texttt{las}} = \\
& \qquad\qquad S[m', M'] \oplus S[m', \texttt{par}(M')] \oplus M'_{\texttt{las}} \text{ or} \\
& [m, M], [m'] \text{ with } \boldsymbol{K}[m'] \| \boldsymbol{IV}[m'] \oplus S[m'] = \\
& \qquad\qquad S[m, M] \oplus S[m, \texttt{par}(M)] \oplus M_{\texttt{las}} \text{ or} \\
& [m] \neq [m'] \text{ with } \boldsymbol{K}[m] \| \boldsymbol{IV}[m] \oplus S[m] = \boldsymbol{K}[m'] \| \boldsymbol{IV}[m'] \oplus S[m']\,, \\
\mathsf{Coll} : {} & \mathsf{Coll_F} \vee \mathsf{Coll_B}\,,
\end{aligned}$$

$$\begin{aligned}
\mathsf{Guess_F} &: [m], (X, Y, d) \text{ with } X = \boldsymbol{K}[m] \| \boldsymbol{IV}[m] \text{ or} \\
&\quad [m, M], (X, Y, d) \text{ with } X = S[m, \mathtt{par}(M)] \oplus M_{\mathtt{las}}\,, \\
\mathsf{Guess_B} &: [m, M], (X, Y, d) \text{ with } Y = S[m, M] \oplus S[m, \mathtt{par}(M)] \oplus M_{\mathtt{las}} \text{ or} \\
&\quad [m], (X, Y, d) \text{ with } Y = \boldsymbol{K}[m] \| \boldsymbol{IV}[m] \oplus S[m]\,, \\
\mathsf{Guess} &: \mathsf{Guess_F} \vee \mathsf{Guess_B}\,.
\end{aligned}$$

In the real world, $\mathsf{Coll_F}$ (resp., $\mathsf{Coll_B}$) concerns the state right before (resp., right after) a permutation evaluation, so that Coll prevents collisions between intermediate states (including collisions between users' keys). Coll guarantees that every permutation call at the construction level associated to a path that is not a prefix of another path has a new permutation call. On the other hand, Guess corresponds to the adversary in the real world being able to guess a permutation evaluation that was used by the construction. More precisely, $\mathsf{Guess_F}$ (resp., $\mathsf{Guess_B}$) concerns the state right before (resp., right after) a permutation evaluation. Moreover, both of these events prevent the ideal world from generating permutation-inconsistent transcripts. A transcript is called *bad* if it sets $\mathsf{Coll} \vee \mathsf{Guess}$.

Bounding the Probability of Good Transcripts. Let τ be a good transcript. In particular the transcript is permutation-consistent, thus reachable in the real world. Moreover, in the real world, with a good transcript, every permutation call which does not correspond to a repeated subpath is fresh, so that τ induces $\mathcal{N} + \mathcal{M}$ different permutation calls. On the other side, in the ideal world, one transcript corresponds to $\mathcal{N}$ permutation outputs, and $\mathcal{M}$ random states. Therefore,

$$\frac{\mathbf{Pr}\left(\mathcal{A}^{W_R} \text{ generates } \tau\right)}{\mathbf{Pr}\left(\mathcal{A}^{W_I} \text{ generates } \tau\right)} = \frac{(2^b)_{\mathcal{N}} \times (2^b)^{\mathcal{M}}}{(2^b)_{\mathcal{M}+\mathcal{N}}} \geq 1\,. \tag{10.2}$$

Bounding the Probability of Bad Transcripts in the Ideal World. Note that the bad events involve either a key, or a *non-final* state, which are both generated at the end of the interaction. As a result, the adversary is non-adaptive for those events. There are $\mathcal{M}$ b-bit states that are generated independently, uniformly at random, and the k-bit keys are also uniformly distributed. Therefore,

$$\mathbf{Pr}\left(\mathcal{A}^{W_I} \text{ sets } \mathsf{Coll}\right) \leq \frac{\sum_s u_s^{\boldsymbol{IV}}(u_s^{\boldsymbol{IV}} - 1)}{2^{k+1}} + \frac{\mathcal{M}(\mathcal{M}-1)}{2^b}\,, \tag{10.3}$$

$$\mathbf{Pr}\left(\mathcal{A}^{W_I} \text{ sets } \mathsf{Guess}\right) \leq \frac{2\mathcal{N}\mathcal{M}}{2^b} + \frac{\mathcal{N} \cdot u_{\max}^{\boldsymbol{IV}}}{2^k}\,. \tag{10.4}$$

We can therefore plug (10.3) and (10.4), which together upper bound the probability that a bad transcript occurs in the ideal world. Combining this result with (10.2) and Lemma 2.5.1 completes the proof. □

10.6 Proof of Theorem 10.3.1

Let $\mathcal{A}$ be an adversary that has access to either $\left(\text{MacaKey}^{\mathcal{P},\mathcal{P}'}_{\boldsymbol{K},\boldsymbol{IV}}, \mathcal{P}, \mathcal{P}'\right)$, or to $(\mathcal{RO}, \mathcal{P}, \mathcal{P}')$. Without loss of generality, we assume that the messages queried to the construction are already padded, i.e., they are b-bit tuples that correspond to a valid padding. Any query that does not meet this criterion will be rejected by the construction oracles. We also assume that $\mathcal{A}$ requests outputs of size a multiple of b bits, as this does not increase its query complexity.

To prove an upper bound on the PRF advantage of MacaKey, we follow the approach of Daemen et al. [DMV17] by introducing an idealized version of MacaKey named $\mathcal{RS}$ in Section 10.6.1. Bounding the distinguishing advantage between $\mathcal{RS}$ and $\mathcal{RO}$ is more involved than in [DMV17], and requires an original proof. This added complexity comes from the unique feature of MacaKey that outputs the XOR of two consecutive inner parts, introducing additional constraints during the combinatorial analysis of the transcripts in the ideal world.

The distance is split into two distances Section 10.6.2, which are themselves analyzed in respectively Sections 10.6.3 and 10.6.4.

10.6.1 Randomized Sponge Construction

Introducing a hybrid algorithm that lies in-between MacaKey and $\mathcal{RO}$ will greatly simplify our analysis: the full-state randomized sponge ($\mathcal{RS}$) construction. $\mathcal{RS}$ behaves similarly to MacaKey with the exception that calls to $\mathcal{P}$ and $\mathcal{P}'$ have been replaced by three primitives: φ, λ, and Λ. φ is a uniformly-random injective mapping: $\varphi : [\![1, \mu]\!] \to \{0,1\}^b$ that replaces the initialization calls to $\mathcal{P}$ and λ and Λ are random b-bit permutations that replace respectively $\mathcal{P}$ and $\mathcal{P}'$ in the remainder of the calls.

φ, λ, and Λ are secret primitives, i.e., $\mathcal{A}$ cannot query them. Refer to Algorithm 26 for an algorithmic description of $\mathcal{RS}$ and the leftmost picture of Figure 10.4 for an illustration.

Algorithm 26 Description of $\mathcal{RS}^{\varphi,\lambda,\Lambda}$ based on the injection φ and the permutations λ and Λ.

```
1:  function $\mathcal{RS}^{\varphi,\lambda,\Lambda}(m, M, \nu)$
        input: $(m, M, \nu) \in [\![1, \mu]\!] \times \{0,1\}^* \times \mathbb{N}^*$
        output: $Z \in \{0,1\}^\nu$
2:      $S \leftarrow \varphi(m)$ // State of the construction
3:      $(M_1, \ldots, M_\ell) \leftarrow pad_b(M)$
4:      for $1 \leq i \leq \ell$ do // Absorb
5:          $Y_R \leftarrow \mathrm{inner}_c(S)$ // XORing base string
6:          $S \leftarrow \lambda(S \oplus M_i)$
7:      $Z_L \leftarrow \epsilon$// Leftmost part of output string
8:      $Z_R \leftarrow \epsilon$ // Rightmost part of output string
9:      for $1 \leq i \leq \lceil \frac{\nu}{b} \rceil$ do //  Squeeze
10:         $Z_L \leftarrow Z \parallel \mathrm{outer}_r(S)$
11:         $Z_R \leftarrow Z_R \parallel (\mathrm{inner}_c(S) \oplus Y_R)$
12:         $Y_R \leftarrow \mathrm{inner}_c(S)$
13:         $S \leftarrow \Lambda(S)$
14:     $Z \leftarrow Z_L \parallel Z_R$
15:     return $Z[1 : \nu]$
```

10.6.2 Distance Splitting

By the triangle inequality, we have:

$$\begin{aligned}&\Delta_{\mathcal{A}}(\mathrm{MacaKey}^{\mathcal{P},\mathcal{P}'}_{\boldsymbol{K},\boldsymbol{IV}}, \mathcal{P}, \mathcal{P}' \ ; \ \mathcal{RO}, \mathcal{P}, \mathcal{P}') \\ &\leq \underbrace{\Delta_{\mathcal{A}}(\mathrm{MacaKey}^{\mathcal{P},\mathcal{P}'}_{\boldsymbol{K},\boldsymbol{IV}}, \mathcal{P}, \mathcal{P}' \ ; \ \mathcal{RS}^{\varphi,\lambda,\Lambda}, \mathcal{P}, \mathcal{P}')}_{(10.5.\mathrm{A})} + \underbrace{\Delta_{\mathcal{A}}(\mathcal{RS}^{\varphi,\lambda,\Lambda}, \mathcal{P}, \mathcal{P}' \ ; \ \mathcal{RO}, \mathcal{P}, \mathcal{P}')}_{(10.5.\mathrm{B})},\end{aligned} \tag{10.5}$$

where $\mathcal{RS}^{\varphi,\lambda,\Lambda}$ denotes the construction from Section 10.6.1 based on a random injection function φ and random permutations λ and Λ.

Furthermore, we can simplify (10.5.B) by noting that both $\mathcal{RS}^{\varphi,\lambda,\Lambda}$ and $\mathcal{RO}$ do not depend on $\mathcal{P}$ and $\mathcal{P}'$. Hence:

$$\Delta_{\mathcal{A}}(\mathcal{RS}^{\varphi,\lambda,\Lambda}, \mathcal{P}, \mathcal{P}' \ ; \ \mathcal{RO}, \mathcal{P}, \mathcal{P}') \leq \Delta_{\mathcal{B}}(\mathcal{RS}^{\varphi,\lambda,\Lambda} \ ; \ \mathcal{RO}), \tag{10.6}$$

where $\mathcal{B}$ is an adversary with the same resources as $\mathcal{A}$, except with no access to $\mathcal{P}$ and $\mathcal{P}'$.

It remains to bound (10.6) and (10.5.A). This follows from Lemmas 10.6.1 and 10.6.2, whose proofs will be given in Sections 10.6.3 and 10.6.4, respectively.

Lemma 10.6.1. *We have*

$$\Delta_{\mathcal{B}}(\mathcal{RS}^{\varphi,\lambda,\Lambda}\ ;\ \mathcal{RO}) \leq \frac{\mathcal{M}^2}{2^b}\,. \tag{10.7}$$

Lemma 10.6.2. *We have*

$$\Delta_{\mathcal{A}}(\mathrm{MacaKey}^{\mathcal{P},\mathcal{P}'}_{\boldsymbol{K},\boldsymbol{IV}},\mathcal{P},\mathcal{P}'\ ;\ \mathcal{RS}^{\varphi,\lambda,\Lambda},\mathcal{P},\mathcal{P}') \leq \frac{\sum_s u_s^{\boldsymbol{IV}}(u_s^{\boldsymbol{IV}}-1)}{2^{k+1}} + \frac{(\mathcal{N}+\mathcal{M})u_{\max}^{\boldsymbol{IV}}}{2^k} + \frac{2\mathcal{M}\mu}{2^b} + \frac{3\mathcal{N}\cdot\mathrm{mucol}(\mathcal{M},2^r)}{2^c}\,. \tag{10.8}$$

Noticing that $\mu \leq \mathcal{M}$, we simplify (10.7) and (10.8) using:

$$\frac{\mathcal{M}^2}{2^b} + \frac{2\mathcal{M}\mu}{2^b} \leq \frac{3\mathcal{M}^2}{2^b}\,. \tag{10.9}$$

Finally, combining (10.6) to (10.9) together with (10.5.A) ends the proof. □

10.6.3 Distance Between RS and RO

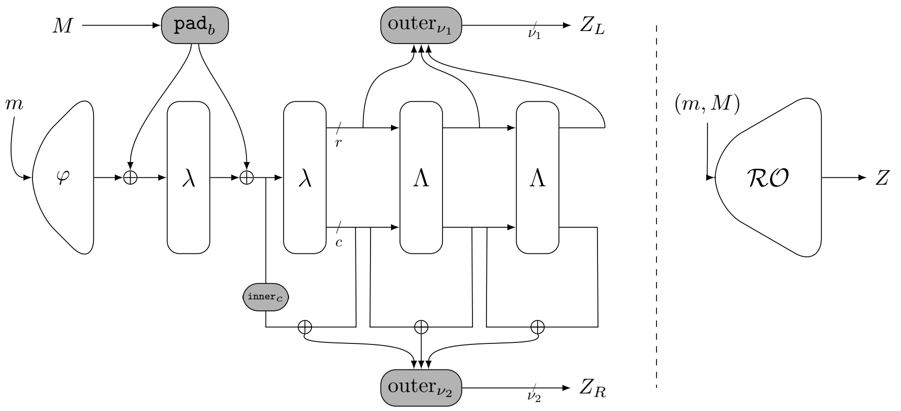


Figure 10.4: Distinguishing experiment between $\mathcal{RS}^{\varphi,\lambda,\Lambda}$ and $\mathcal{RO}$.

In this section, we bound the distinguishing advantage between the randomized sponge $\mathcal{RS}^{\varphi,\lambda,\Lambda}$ and a random oracle $\mathcal{RO}$ (bound (10.7) of Lemma 10.6.1).

Let $\mathcal{B}$ be an adversary with query complexity $\mathcal{M}$. The distinguishing experiment is illustrated in Figure 10.4.

For the derivation we rely on the H-coefficient technique of Lemma 2.5.1. That means that we must first find a suitable description of the transcripts.

Counting the Queries Made by $\mathcal{B}$. We denote by $\mathcal{Q}$ the number of queries made by $\mathcal{B}$ to its oracle, *without doubly counting queries with repeated user and message* (cf., examples 10.6.1 and 10.6.2). We note that $\mathcal{Q}$ depends on the specific behavior of $\mathcal{B}$, while $\mathcal{M}$ is a fixed resource of $\mathcal{B}$. Nevertheless, it always holds that $\mathcal{Q} \leq \mathcal{M}$.

Example 10.6.1. *For simplicity, in the examples of this section we use the* 10^* *padding. Suppose* $\mathcal{B}$ *makes the following two queries in the* $\mathcal{RS}^{\varphi,\lambda,\Lambda}$ *world:*

$$(m_1, M_1, \nu_1) = (1, 0^{2b-1} \parallel 1, b), \text{ and}$$
$$(m_1, M_2, \nu_2) = (1, 0^{2b-1} \parallel 1, 2b)\,.$$

Computing the first query requires evaluating: $\varphi(1), \lambda \circ \varphi(1)$ *during the absorbing phase, and* $S := \lambda\left(\lambda \circ \varphi(1) \oplus 0^{b-1} \| 1\right)$ *during the squeezing phase. To compute the second query, we need to compute the aforementioned values, together with* $\Lambda(S)$. *So in this case,* $\mathcal{M} = 4$ *and* $\mathcal{Q} = 1$.

Example 10.6.2. *Assume that* $b > 1$, *and suppose* $\mathcal{B}$ *makes only the following two queries in the* $\mathcal{RS}^{\varphi,\lambda,\Lambda}$ *world:*

$$(m_1, M_1, \nu_1) = (1, 0^{2b-1} \| 1, b), \text{ and}$$
$$(m_1, M_2, \nu_2) = (1, 0^{b} \parallel 1^{b}, b)\,.$$

To compute the first query, we proceed as in example 10.6.1. To compute the second query, we again need to compute the previously computed values, together with $\lambda\left(\lambda \circ \varphi(1) \oplus 1^b\right)$. *So in this case, we also have* $\mathcal{M} = 4$, *but* $\mathcal{Q} = 2$.

Transcript Notation. During the interaction, $\mathcal{B}$ makes queries of the form (m, M, ν), where m is the index of the user, ν is the output length in bits, and M is the message queried. The response is of the form $Z = Z_L \| Z_R \in \{0,1\}^{\nu}$, where Z_L gathers the part of Z originating from truncation, while Z_R gathers the part originating from summation (cf., Figure 10.4).

To simplify our analysis, at the end of the interaction, but before $\mathcal{B}$ outputs its decision, we disclose additional tuples, denoted by $T_1, \ldots, T_{\mathcal{Q}}$:

- In the $\mathcal{RS}^{\varphi,\lambda,\Lambda}$ world, for $i \in [\![1,\mathfrak{Q}]\!]$, T_i is of the form $\left(t_i^0, t_i^1, \ldots\right) \in (\{0,1\}^b)^*$. It consists of the outputs of φ, λ, and Λ generated while answering the i-th query of $\mathcal{B}$. We do not include in the transcript any value t_i^j that is generated from a repeated evaluation to φ, λ, or Λ (cf., example 10.6.3);
- In the $\mathcal{RO}$ world, we will specify a procedure to generate t_i^j, in such a way that they are uniformly random and match the intrinsic properties of the $\mathcal{RS}^{\varphi,\lambda,\Lambda}$ world.

Before we describe how the transcript is generated in the $\mathcal{RO}$ world we need to introduce additional concepts. For a given i-th query, we can split T_i into two tuples: T_i^a, made of the values that are generated during the absorbing phase, and T_i^b, made of the values that are generated during the squeezing phase. We denote $\theta(i) = |T_i^a|$, and $\chi(i) = |T_i| - 1$, so that:

$$\begin{aligned} T_i^a &= \left(t_i^0, t_i^1, \ldots, t_i^{\theta(i)-1}\right) \,, \\ T_i^b &= \left(t_i^{\theta(i)}, \ldots, t_i^{\chi(i)}\right) \,. \end{aligned} \tag{10.10}$$

The following example serves to clarify this concept.

Example 10.6.3. *If we consider the queries made in example 10.6.2, this gives* $T_1^a = (\varphi(1), \lambda \circ \varphi(1))$*,* $T_1^b = \left(\lambda(\lambda \circ \varphi(1) \oplus 0^{b-1} \| 1)\right)$*,* $T_2^a = \emptyset$*, and* $T_2^b = \left(\lambda(\lambda \circ \varphi(1) \oplus 1^b)\right)$*.* ♣

Putting all together, we define the extended transcript[3] τ as a set consisting of tuples:

$$\tau = \{(m_i, \nu_i, M_i, T_i)\}_{i=1}^{\mathfrak{Q}} \,,$$

where $m_i \in [\![1,\mu]\!]$ is the index of the user, $\nu_i \in \mathbb{N}^*$ is the length of the output in bits, M_i is the message queried and T_i is as in (10.10). For $i \in [\![1,\mathfrak{Q}]\!]$, $j \in [\![1,\theta(i)]\!]$, we define $M_i^j \in \{0,1\}^b$ to be the message block absorbed before obtaining the state t_i^j. Finally, for $j > 0$ we define

$$\mathbf{inSt}(i,j) = \begin{cases} M_i^j \oplus t_i^{j-1} & \text{if } j \le \theta(i) \text{ and } t_i^j \text{ appears in } \tau \\ \epsilon & \text{otherwise}\,. \end{cases}$$

In other words, $\mathbf{inSt}(i,j)$ is defined only when the j^{th} permutation evaluation during query i is not redundant and occurs during the absorbing phase, in which case it represents the input to the corresponding λ-evaluation.

[3]From now on, we will simply call it the transcript.

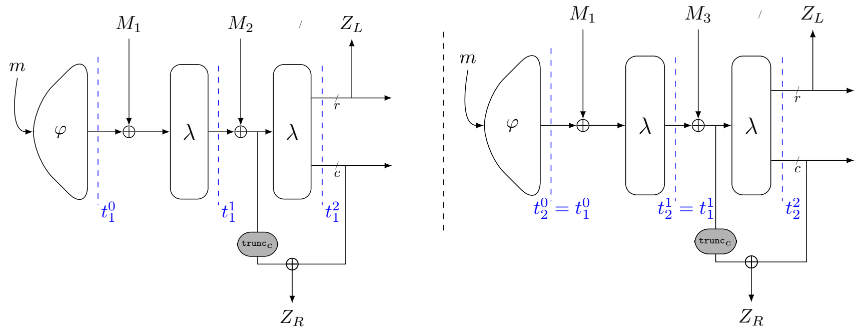


Figure 10.5: Comparison of queries sharing a *common maximal prefix.* Here, the second (rightmost) query shares a common maximal prefix with the first (leftmost) one.

Common Maximal Prefixes. Before we explain how to generate transcripts in the $\mathcal{RO}$ world, we need one last concept. A query (m, M, ν) is said to share a *common maximal prefix* with an *earlier query* (m', M', ν') whenever $m = m'$ and there exist $M_1, M_2 \in \left(\{0,1\}^b\right)^*$, $M_B \in \{0,1\}^b$ such that $M = M_1 \| M_B$ and $M' = M_1 \| M_2$. In other words, M deprived of its last b-bit block is a prefix of M'.

Example 10.6.4. *Assume that $\mathcal{B}$ makes two queries with input $(1, M_1\|M_2, b)$ and $(1, M_1\|M_3, b)$, where $M_1, M_2, M_3 \in \{0,1\}^b$ and M_2 is distinct from M_3. Then $\theta(1) = 2$, $\theta(2) = 0$ and a fortiori $T_2^a = \epsilon$. This is because the second query only differs from the first one in the last b-bit block. This means that the second query simply makes the same absorb calls as the first one, hence all the intermediate states from this phase are repeated and therefore discarded. See Figure 10.5 for reference. More generally, for the i-th query, (m_i, ν_i, M_i, T_i), we have that $T_i^a = \epsilon$ if and only if there is a previous query sharing a maximal common prefix with the query i.*

We define the common prefix function $\mathrm{CP} : [\![1, \mathcal{Q}]\!] \to \{0,1\}^b$ as:

$$\mathrm{CP}(i) = \begin{cases} t_i^{\theta(i)-1}, & \text{if } \theta(i) \geq 1\,, \\ t_i^{\theta(i')-1}, & \text{otherwise, where } i' < i \text{ is the earliest query made} \\ & \text{sharing a common maximal prefix with query i.} \end{cases} \tag{10.11}$$

Generating Transcripts in the $\mathcal{RO}$ World. In order to maintain consistency with the $\mathcal{RS}^{\varphi,\lambda,\Lambda}$ world, the transcripts in the $\mathcal{RO}$ world must satisfy certain constraints.

Fix any $i \in [\![1, \mathcal{Q}]\!]$, let Z_L^i and Z_R^i be respectively the response to the query generated via truncation and via summation (cf., Figure 10.2). We split Z_L^i (resp., Z_R^i) into r-bits (resp., c-bits) blocks: $Z_L^{i,\theta(i)} \| \cdots \| Z_L^{i,\chi(i)}$ (resp., $Z_R^{i,\theta(i)} \| \cdots \| Z_R^{i,\chi(i)}$).

Now, to keep consistent answers, the transcript released in the $\mathcal{RO}$ world must satisfy the following equations:

$$\begin{cases} Z_L^{i,j} & = \mathrm{outer}_r(t_i^j)\,, & \forall\, \theta(i) \le j \le \chi(i)\,, \\ Z_R^{i,\theta(i)} & = \mathrm{inner}_c(\mathrm{CP}(i)) \oplus \mathrm{inner}_c(t_i^{\theta(i)}) \oplus \mathrm{inner}_c(M_i^{\theta(i)})\,, & \\ Z_R^{i,j} & = \mathrm{inner}_c(t_i^{j-1}) \oplus \mathrm{inner}_c(t_i^j)\,, & \forall\, \theta(i)+1 \le j \le \chi(i)\,. \end{cases} \tag{10.12}$$

Every choice of $\mathrm{CP}(i)$ fixes all the $t_{i'}^j$ such that $\mathrm{CP}(i) = \mathrm{CP}(i')$. Hence, we choose $\mathrm{CP}(i) \xleftarrow{\$} \{0,1\}^b$, for every query i that does not share a *common maximal prefix* with an earlier query.

We now describe how we generate $T_1, \ldots, T_{\mathcal{Q}}$ in the $\mathcal{RO}$ world.

- If $j < \theta(i) - 1$, then the values t_i^j would, in the $\mathcal{RS}^{\varphi,\lambda,\Lambda}$ world, correspond to an intermediate state during the absorbing phase, but not in the last permutation call of the absorbing phase. In this case t_i^j does not appear in (10.12), and we sample it uniformly at random:

$$t_i^j \xleftarrow{\$} \{0,1\}^b\,;$$

- If $j = \theta(i) - 1$, then only $\mathrm{inner}_c(t_i^j)$ is constrained by (10.12). Therefore, we compute t_i^j as follows:

$$s \xleftarrow{\$} \{0,1\}^r\,, \quad t_i^j \leftarrow s \| \mathrm{inner}_c(\mathrm{CP}(i))\,;$$

- Else, then necessarily $j \ge \theta(i)$, and t_i^j is completely determined by (10.12):
  - We have $\mathrm{outer}_r(t_i^j) = Z_R^{i,j}$;
  - We have $\mathrm{inner}_c(t_i^j) = \mathrm{inner}_c(t_i^{j-1}) \oplus Z_R^{i,j}$, which we can solve iteratively.

Bad Transcripts Definition. Let $\mathcal{T}$ be the set of all attainable transcripts. We say that a transcript τ is in $\mathcal{T}_{bad}$ if it triggers the bad event **IColl** defined as follows:

$$\begin{aligned}
\mathbf{IColl}_\lambda : \exists i, i', j, j', \in \mathbb{N}, &\left(m_i, \nu_i, M_i, \left(t_i^j\right)_{j=0}^{\chi(i)}\right), \left(m_{i'}, \nu_{i'}, M_{i'}, \left(t_{i'}^{j'}\right)_{j=0}^{\chi(i')}\right) \in \tau \\
&\text{with } (i,j) \neq (i',j'), j \leq \theta(i), \text{ and } j' \leq \theta(i') \\
&\text{and either } t_i^j = t_{i'}^{j'} \\
&\text{or } j, j' > 0, \mathbf{inSt}(i,j), \mathbf{inSt}(i',j') \neq \epsilon, \text{ and } \mathbf{inSt}(i,j) = \mathbf{inSt}(i',j'), \\
\mathbf{IColl}_\Lambda : \exists i, i', j, j', \in \mathbb{N}, &\left(m_i, \nu_i, M_i, \left(t_i^j\right)_{j=0}^{\chi(i)}\right), \left(m_{i'}, \nu_{i'}, M_{i'}, \left(t_{i'}^{j'}\right)_{j=0}^{\chi(i')}\right) \in \tau \\
&\text{with } (i,j) \neq (i',j'), j \geq \theta(i), \text{ and } j' \geq \theta(i') \text{ and } t_i^j = t_{i'}^{j'}, \\
\mathbf{IColl} : \mathbf{IColl}_\lambda \vee \mathbf{IColl}_\Lambda \,.
\end{aligned}$$

In words, $\mathbf{IColl}_\lambda$ (resp., $\mathbf{IColl}_\Lambda$) is set whenever two inputs/outputs to λ (resp., Λ) collide together. Note that cross-collisions between λ and Λ do not set **IColl**.

Bounding the Probability of Good Transcripts. Let

$$\tau = \left\{\left(m_i, \nu_i, M_i, \left(t_i^j\right)_{j=1}^{\chi(i)}\right)\right\}_{i=1}^{\mathcal{Q}},$$

be any good transcript. Fix any query $i \in [\![1, \mathcal{Q}]\!]$, and let $j \in [\![0, \chi(i)]\!]$. To begin, we consider the case $\mathcal{B}$ is in the $\mathcal{RO}$ world. By definition, the intermediate states t_i^j are generated as follows:

- If $j < \theta(i) - 1$, then t_i^j is uniformly generated;
- If $j = \theta(i) - 1$, then necessarily the query does not share a common maximal prefix with an earlier query, hence t_i^j is also uniformly generated;
- Else, necessarily $j \geq \theta(i)$. First note that the output string $Z^i = Z_L^i \| Z_R^i$ is generated from a $\mathcal{RO}$ call, whose entries are never accessed twice. Hence, $\text{outer}_r(t_i^j) = Z_L^{i,j}$ and $\text{inner}_c(t_i^j) = \text{inner}_c(t_i^{j-1}) \oplus Z_R^{i,j}$ are uniformly random and independent.

Therefore, every t_i^j is uniformly distributed and independent of the other values. Finally, we have $\mathcal{M}$ different (i,j) tuples in τ. Hence,

$$\mathbf{Pr}\left(\mathcal{B}^{\mathcal{RO}} \text{ generates } \tau\right) = \frac{1}{2^{b\mathcal{M}}}\,. \tag{10.13}$$

Consider now the case where $\mathcal{B}$ is placed in the $\mathcal{RS}^{\varphi,\lambda,\Lambda}$ world. In particular, τ is attainable in the $\mathcal{RS}^{\varphi,\lambda,\Lambda}$ world. Each state value t_i^j in τ is equal to an evaluation of φ of λ, or of Λ. Furthermore, precisely μ of these values come from evaluations of φ on the user indexes, so they are sampled uniformly at *random without replacement.* The remaining $\mathcal{M} - \mu$ come from evaluations of λ and Λ. If we split these into $\mathcal{M}_1$ evaluations to λ and $\mathcal{M}_2$ evaluations to Λ, we obtain

$$\mathbf{Pr}\left(\mathcal{B}^{\mathcal{RS}^{\varphi,\lambda,\Lambda}} \text{ generates } \tau\right) = \frac{1}{(2^b)_\mu (2^b)_{\mathcal{M}_1} (2^b)_{\mathcal{M}_2}}\,, \tag{10.14}$$

where $\mathcal{M}_1 + \mathcal{M}_2 = \mathcal{M} - \mu$. Hence, from (10.13) and (10.14), we get:

$$\frac{\mathbf{Pr}\left(\mathcal{B}^{\mathcal{RS}^{\varphi,\lambda,\Lambda}} \text{ generates } \tau\right)}{\mathbf{Pr}\left(\mathcal{B}^{\mathcal{RO}} \text{ generates } \tau\right)} \geq 1\,. \tag{10.15}$$

Bounding the Probability of Bad Transcripts in the Ideal World. Let $\tau \in \mathcal{T}_{bad}$, $i, i' \in [\![1, \mathcal{Q}]\!], j, j' \in \mathbb{N}$ with $(i,j) \neq (i',j')$ and either $(j \leq \theta(i)$ and $j' \leq \theta(i'))$ or $(j \geq \theta(i)$ and $j' \geq \theta(i'))$. The probability that $t_i^{j'}$ and t_i^j collide (or that $\mathbf{inSt}(i,j) = \mathbf{inSt}(i',j') \neq \epsilon$) is at most $\frac{1}{2^b}$. There are at most $2\binom{\mathcal{M}}{2} \leq \mathcal{M}^2$ such possible colliding pairs. Therefore,

$$\mathbf{Pr}\left(\mathcal{B}^{\mathcal{RO}} \text{ generates } \tau \in \mathcal{T}_{bad}\right) \leq \frac{\mathcal{M}^2}{2^b}\,. \tag{10.16}$$

We finaly combine (10.15) and (10.16) with Lemma 2.5.1, which finishes the proof of Lemma 10.6.1. $\square$

10.6.4 Distance Between MacaKey and RS

In this section, we prove Lemma 10.6.2, i.e., the distance (10.5.A), which is the statistical distance between $\mathrm{MacaKey}_{\boldsymbol{K},\boldsymbol{IV}}^{\mathcal{P},\mathcal{P}'}$ and $\mathcal{RS}^{\varphi,\lambda,\Lambda}$, as illustrated in Figure 10.6. Consider a distinguisher $\mathcal{A}$ with resources $(\mathcal{N}, \mathcal{M})$. We will use again the H-coefficient technique of Lemma 2.5.1.

To upper bound this distance, we will build upon the technique which of Daemen et al. [DMV17], while accounting for the additional complexity introduced by XORing with the extra c inner bits, which requires a subtler analysis. For simplicity, we will denote the $\mathcal{RS}$ world $\mathcal{O}_{\mathrm{RS}}$ and the MacaKey world by $\mathcal{O}_{\mathrm{MK}}$.

We will focus first on $\mathcal{O}_{\mathrm{RS}}$. Denote by $\mathcal{M}_1$ the online complexity induced by evaluations to λ (excluding the first initialization call), and $\mathcal{M}_2$ induced by

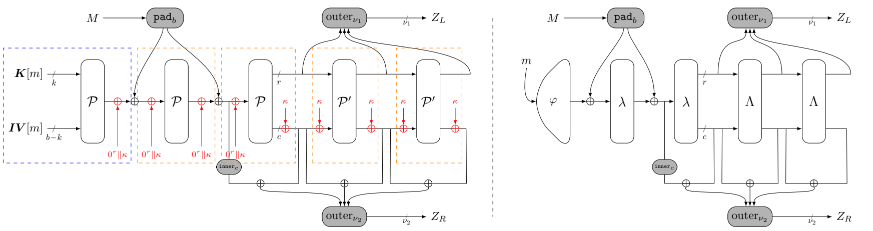


Figure 10.6: Distinguishing experiment between $\mathrm{MacaKey}^{\mathcal{P},\mathcal{P}'}_{\boldsymbol{K},\boldsymbol{IV}}$ and $\mathcal{RS}^{\varphi,\lambda,\Lambda}$. The leftmost blue dashed box indicates $\mathrm{IM}^{\mathcal{P}}_{\boldsymbol{K},\kappa,\boldsymbol{IV}}$, while the others in dashed orange represent $\mathrm{EM}^{\mathcal{P}}_{\kappa}$ and $\mathrm{EM}^{\mathcal{P}'}_{\kappa}$. Besides construction queries, the adversary has additionally oracle access to $\mathcal{P}$ and $\mathcal{P}'$.

evaluations to Λ, so that $\mathcal{M}_1 + \mathcal{M}_2 = \mathcal{M} - \mu$. Moreover, we split $\mathcal{N}$ as $\mathcal{N}_1 + \mathcal{N}_2$, where $\mathcal{N}_1$ (resp., $\mathcal{N}_2$) records the number of queries to $\mathcal{P}$ (resp., $\mathcal{P}'$). Note that $\mathcal{A}$ does not have access to φ, λ, and Λ except through queries to $\mathcal{RS}$. We split the transcript into the queries made to φ, λ, Λ, $\mathcal{P}$, and $\mathcal{P}'$ as follows:

- τ_φ contains the initialization part of construction queries. That is:
$$\tau_\varphi = \{(m, I_m)\}_{m=1}^{\mu} ;$$
- τ_λ contains the input-output pairs to λ made during the construction queries. That is:
$$\tau_\lambda = \{(s_i, t_i)\}_{i=1}^{\mathcal{M}_1} ;$$
- τ_Λ contains the input-output pairs to Λ made during the construction queries. That is:
$$\tau_\Lambda = \{(s_i, t_i)\}_{i=1}^{\mathcal{M}_2} ;$$
- $\tau_{\mathcal{P}}$ contains the primitive queries to $\mathcal{P}$. That is:
$$\tau_{\mathcal{P}} = \{(x_i, y_i)\}_{i=1}^{\mathcal{N}_1} ;$$
- $\tau_{\mathcal{P}'}$ contains the primitive queries to $\mathcal{P}'$. That is:
$$\tau_{\mathcal{P}'} = \{(x_i, y_i)\}_{i=1}^{\mathcal{N}_2} .$$

Moreover, to simplify our analysis, after the interaction, right before $\mathcal{A}$ outputs their decision bit, we release dummy keys $\boldsymbol{K}$ and κ, sampled as $\boldsymbol{K}[1], \dots, \boldsymbol{K}[m] \xleftarrow{\$} \{0,1\}^k$ and $\kappa \xleftarrow{\$} \{0,1\}^c$. Therefore, the extended transcript is $\tau = (\tau_\varphi, \tau_\lambda, \tau_\Lambda, \tau_{\mathcal{P}}, \tau_{\mathcal{P}'}, \boldsymbol{K}, \kappa)$.

We now focus on $\mathcal{O}_{\mathrm{MK}}$. Given $\kappa \in \{0,1\}^c$, define $\mathrm{IM}^{\mathcal{P}}_{\boldsymbol{K},\kappa,\boldsymbol{IV}}$, $\mathrm{EM}^{\mathcal{P}}_{\kappa}$, and $\mathrm{EM}^{\mathcal{P}'}_{\kappa}$ as follows:

$$\begin{aligned}
\mathrm{EM}^{\mathcal{P}}_{\kappa}(x) &= \mathcal{P}\left(x \oplus (0^r \| \kappa)\right) \oplus (0^r \| \kappa) \ , \\
\mathrm{EM}^{\mathcal{P}'}_{\kappa}(x) &= \mathcal{P}'\left(x \oplus (0^r \| \kappa)\right) \oplus (0^r \| \kappa) \ , \\
\mathrm{IM}^{\mathcal{P}}_{\boldsymbol{K},\kappa,\boldsymbol{IV}}(m) &= \mathcal{P}\left(\boldsymbol{K}[m] \| \boldsymbol{IV}[m]\right) \oplus (0^r \| \kappa) \ .
\end{aligned}$$

For every construction query, we can replace the initialization and first $\mathcal{P}$-evaluation with $\mathrm{IM}^{\mathcal{P}}_{\boldsymbol{K},\kappa,\boldsymbol{IV}}$, and the remaining $\mathcal{P}$-evaluations (resp., $\mathcal{P}'$-evaluations) with $\mathrm{EM}^{\mathcal{P}}_{\kappa}$ (resp., $\mathrm{EM}^{\mathcal{P}'}_{\kappa}$) This modification does not alter the construction's behavior, but allows to view MacaKey as a cascade of Even-Mansour evaluations. Intuitively, this consists of artificially adding a random *dummy* c-bit key in-between the permutation calls made in MacaKey. This dummy keying can be safely done as the c-bit masks cancel each other out before producing MacaKey's output. A visual representation of this process is shown in the leftmost illustration of Figure 10.6.

Let $\kappa \xleftarrow{\$} \{0,1\}^c$. We now define the transcript in the MacaKey world analogously to how it is defined in the $\mathcal{RS}$ world:

- τ_φ records the outputs of all calls to $\mathrm{IM}^{\mathcal{P}}_{\boldsymbol{K},\kappa,\boldsymbol{IV}}$;
- τ_λ records the inputs and outputs of all calls to $\mathrm{EM}^{\mathcal{P}}_{\kappa}$;
- τ_Λ records the inputs and outputs of all calls to $\mathrm{EM}^{\mathcal{P}'}_{\kappa}$;
- $\tau_{\mathcal{P}}$ and $\tau_{\mathcal{P}'}$ contains the queries to $\mathcal{P}$ and $\mathcal{P}'$.

Bad Transcripts Definition. We will label a transcript as *bad* if there exists a collision between elements from $\tau_{\mathcal{P}}$, $\tau_{\mathcal{P}'}$, τ_φ, τ_λ, or τ_Λ that lead to a degenerate behavior in $\mathcal{O}_{\mathrm{RS}}$. More precisely, a transcript τ is *bad* if any of those events occur:

- Coll^{φ}: $\exists m \neq m' \in [\![1, \mu]\!]$ such that: $\boldsymbol{K}[m'] \| \boldsymbol{IV}[m'] = \boldsymbol{K}[m] \| \boldsymbol{IV}[m]$;
- $\mathsf{Connect}^{\mathcal{P},\varphi}_{F}$: $\exists (x,y) \in \tau_{\mathcal{P}}, (m, I) \in \tau_\varphi$ such that: $x = \boldsymbol{K}[m] \| \boldsymbol{IV}[m]$;

- $\mathsf{Connect}_B^{\mathcal{P},\varphi}$: $\exists (x,y) \in \tau_{\mathcal{P}}, (m,I) \in \tau_\varphi$ such that: $y = \boldsymbol{I} \oplus (0^r \| \kappa)$;
- $\mathsf{Connect}_F^{\mathcal{P},\lambda}$: $\exists (x,y) \in \tau_{\mathcal{P}}, (u,v) \in \tau_\lambda$ such that: $x = u \oplus (0^r \| \kappa)$;
- $\mathsf{Connect}_B^{\mathcal{P},\lambda}$: $\exists (x,y) \in \tau_{\mathcal{P}}, (u,v) \in \tau_\lambda$ such that: $y = v \oplus (0^r \| \kappa)$;
- $\mathsf{Connect}_F^{\mathcal{P}',\Lambda}$: $\exists (x,y) \in \tau_{\mathcal{P}'}, (u,v) \in \tau_\Lambda$ such that: $x = u \oplus (0^r \| \kappa)$;
- $\mathsf{Connect}_B^{\mathcal{P}',\Lambda}$: $\exists (x,y) \in \tau_{\mathcal{P}'}, (u,v) \in \tau_\Lambda$ such that: $y = v \oplus (0^r \| \kappa)$;
- $\mathsf{Connect}_F^{\lambda,\varphi}$: $\exists (s,t) \in \tau_\lambda, (m,I) \in \tau_\varphi$ such that: $s \oplus (0^r \| \kappa) = \boldsymbol{K}[m] \| \boldsymbol{IV}[m]$;
- $\mathsf{Connect}_B^{\lambda,\varphi}$: $\exists (s,t) \in \tau_\lambda, (m,I) \in \tau_\varphi$ such that: $I \oplus (0^r \| \kappa) = t$;
- Bad : $\mathsf{Coll}^\varphi \vee \mathsf{Connect}_F^{\mathcal{P},\varphi} \vee \mathsf{Connect}_B^{\mathcal{P},\varphi} \vee \mathsf{Connect}_F^{\mathcal{P},\lambda} \vee \mathsf{Connect}_B^{\mathcal{P},\lambda} \vee \mathsf{Connect}_F^{\mathcal{P}',\Lambda} \vee \mathsf{Connect}_B^{\mathcal{P}',\Lambda} \vee \mathsf{Connect}_F^{\lambda,\varphi} \vee \mathsf{Connect}_B^{\lambda,\varphi}$.

Note that collisions between $\tau_{\mathcal{P}}$ and τ_Λ, between $\tau_{\mathcal{P}'}$ and τ_λ, as well as between τ_φ and $\tau_{\mathcal{P}'}$, do not lead to a degenerate behavior in $\mathcal{O}_{\mathrm{RS}}$.

Bounding the Probability of Good Transcripts. By definition of $\mathcal{O}_{\mathrm{RS}}$, every transcript which is attainable in $\mathcal{O}_{\mathrm{MK}}$ is also attainable in $\mathcal{O}_{\mathrm{RS}}$. Every transcript in $\mathcal{O}_{\mathrm{RS}}$ determines μ distinct input-outputs pairs for the injection function, $\mathcal{M}_1$ (resp., $\mathcal{M}_2$) distinct input-outputs pairs for λ (resp., Λ), and $\mathcal{N}_1$ (resp., $\mathcal{N}_2$) distinct input-outputs pairs for $\mathcal{P}$ (resp., $\mathcal{P}'$). Moreover, any good transcript which is attainable in $\mathcal{O}_{\mathrm{RS}}$ is also attainable in $\mathcal{O}_{\mathrm{MK}}$, where it defines $\mathcal{M}_1 + \mathcal{N}_1 + \mu$ (resp., $\mathcal{M}_2 + \mathcal{N}_2$) distinct permutation calls to $\mathcal{P}$ (resp., $\mathcal{P}'$). Therefore, for any good transcript τ,

$$\frac{\mathbf{Pr}\left(\mathcal{A}^{\mathcal{O}_{\mathrm{MK}}} \text{ generates } \tau\right)}{\mathbf{Pr}\left(\mathcal{A}^{\mathcal{O}_{\mathrm{RS}}} \text{ generates } \tau\right)} = \frac{(2^b)_{\mathcal{M}_1}(2^b)_{\mathcal{M}_2}(2^b)_\mu(2^b)_{\mathcal{N}_1}(2^b)_{\mathcal{N}_2}}{(2^b)_{\mathcal{M}_1+\mathcal{N}_1+\mu}(2^b)_{\mathcal{M}_2+\mathcal{N}_2}} \geq 1\,. \quad (10.17)$$

Bounding the Probability of Bad Transcripts in $\mathcal{O}_{\mathrm{RS}}$. Now, it remains to evaluate the probability that a bad event occurs in $\mathcal{O}_{\mathrm{RS}}$. In this world, $\boldsymbol{K}$ and κ are sampled uniformly at random at the end of the interaction, thus the adversary cannot set this event adaptively. We evaluate each case below, we start by those that do not benefit from multicollision analysis.

- Coll^{φ}: the probability that two users with the same IV have their key colliding is upper bounded by
$$\sum_s \frac{u_s^{\boldsymbol{IV}}(u_s^{\boldsymbol{IV}}-1)}{2^{k+1}}\,;$$
- $\mathsf{Connect}_F^{\mathcal{P},\varphi}$: each permutation evaluation can target at most $u_{\max}^{\boldsymbol{IV}}$ users at the same time. Therefore, this sub-event is set with a probability of at most
$$\frac{\mathcal{N}\cdot u_{\max}^{\boldsymbol{IV}}}{2^k}\,;$$
- $\mathsf{Connect}_F^{\lambda,\varphi}$: We take a similar approach as the previous event. For each element of τ_λ, it has a probability of triggering this event of at most $u_{\max}^{\boldsymbol{IV}}/2^k$. So this happens with a probability of at most
$$\frac{\mathcal{M}\cdot u_{\max}^{\boldsymbol{IV}}}{2^k}\,;$$
- $\mathsf{Connect}_B^{\lambda,\varphi}$: With a query-wise approach, we can see that this event happens with a probability of at most
$$\frac{2(\mathcal{M}-\mu)\mu}{2^b} \leq \frac{2\mathcal{M}\mu}{2^b}\,.$$

In order to analyze the remaining bad events, we introduce the following random variables:

$$\begin{aligned}
\texttt{Col}_F &= \max_{z\in\{0,1\}^r} \#\left\{X \in \{0,1\}^b \mid \exists (X,Y)\in\tau_\lambda\cup\tau_\Lambda \text{ such that } \mathrm{outer}_r(X)=z\right\}\,,\\
\texttt{Col}_B &= \max_{z\in\{0,1\}^r} \#\left\{Y \in \{0,1\}^b \mid \exists (X,Y)\in\tau_\lambda\cup\tau_\Lambda \text{ such that } \mathrm{outer}_r(Y)=z\right\}\,,\\
\texttt{Col}_I &= \max_{z\in\{0,1\}^r} \#\left\{I \in \{0,1\}^b \mid \exists (m,I)\in\tau_\varphi \text{ such that } \mathrm{outer}_r(I)=z\right\}\,.
\end{aligned}$$

The inputs to λ (in particular their outer part) cannot be chosen by the adversary. Therefore, $\texttt{Col}_F$, $\texttt{Col}_B$, and $\texttt{Col}_I$ correspond to the bins-and-balls experiment described in Lemma 2.5.2, where respectively $\mathcal{M}-\mu$, $\mathcal{M}-\mu$, and μ balls are thrown uniformly in 2^r bins. Therefore,

$$\begin{aligned}
\mathsf{E}\left(\texttt{Col}_F\right) &\leq \mathrm{mucol}(\mathcal{M}-\mu, 2^r)\,,\\
\mathsf{E}\left(\texttt{Col}_B\right) &\leq \mathrm{mucol}(\mathcal{M}-\mu, 2^r)\,,\\
\mathsf{E}\left(\texttt{Col}_I\right) &\leq \mathrm{mucol}(\mu, 2^r)\,,
\end{aligned}$$

where mucol$(\cdot)$ is defined in Lemma 2.5.2. Therefore,

$$\mathsf{E}\left(\mathtt{Col}_B + \mathtt{Col}_I + \mathtt{Col}_F\right) \leq 3\mathrm{mucol}(\mathcal{M}, 2^r)\,. \tag{10.18}$$

We are now ready to upper bound the remaining bad events:

- $\mathsf{Connect}_B^{\mathcal{P},\varphi}$: Let us assume $\mathtt{Col}_I = \theta$ for some $\theta \in \mathbb{N}$. For each element of $\tau_{\mathcal{P}}$, there are by assumption at most θ elements of τ_φ that can trigger this event. For each of these, the probability of having the same inner part is at most $1/2^c$. Therefore, $\mathbf{Pr}\left(\mathsf{Connect}_B^{\mathcal{P},\varphi} \mid \mathtt{Col}_I = \theta\right) \leq \theta\mathcal{N}/2^c$. Hence, the probability of triggering this event is at most:

$$\begin{aligned}\mathbf{Pr}\left(\mathsf{Connect}_B^{\mathcal{P},\varphi}\right) &\leq \sum_{\theta\geq 0} \mathbf{Pr}\left(\mathsf{Connect}_B^{\mathcal{P},\varphi} \mid \mathtt{Col}_I = \theta\right)\mathbf{Pr}\left(\mathtt{Col}_I = \theta\right)\\ &\leq \frac{\mathcal{N}}{2^c}\sum_{\theta\geq 0}\theta\mathbf{Pr}\left(\mathtt{Col}_I = \theta\right) = \frac{\mathsf{E}\left(\mathtt{Col}_I\right)\mathcal{N}}{2^c};\end{aligned}$$

- $\mathsf{Connect}_F^{\mathcal{P},\lambda}$, $\mathsf{Connect}_B^{\mathcal{P},\lambda}$, $\mathsf{Connect}_F^{\mathcal{P}',\Lambda}$, and $\mathsf{Connect}_B^{\mathcal{P}',\Lambda}$: We can repeat the same reasoning and get

$$\begin{aligned}\mathbf{Pr}\left(\mathsf{Connect}_F^{\mathcal{P},\lambda} \vee \mathsf{Connect}_F^{\mathcal{P}',\Lambda}\right) &\leq \frac{\mathsf{E}\left(\mathtt{Col}_F\right)\mathcal{N}}{2^c}\\ \mathbf{Pr}\left(\mathsf{Connect}_B^{\mathcal{P},\lambda} \vee \mathsf{Connect}_B^{\mathcal{P},\lambda}\right) &\leq \frac{\mathsf{E}\left(\mathtt{Col}_B\right)\mathcal{N}}{2^c}\,.\end{aligned}$$

Therefore, by grouping all bad events together, we obtain:

$$\begin{aligned}\mathbf{Pr}\left(\mathsf{Bad}\right) &\leq \frac{\sum_s u_s^{\boldsymbol{IV}}(u_s^{\boldsymbol{IV}}-1)}{2^{k+1}} + \frac{(\mathcal{N}+\mathcal{M})u_{\max}^{\boldsymbol{IV}}}{2^k} + \frac{2\mathcal{M}\mu}{2^b} + \\ &\quad \frac{\left(\mathsf{E}\left(\mathtt{Col}_I\right) + \mathsf{E}\left(\mathtt{Col}_F\right) + \mathsf{E}\left(\mathtt{Col}_B\right)\right)\mathcal{N}}{2^c}\\ &\leq \frac{\sum_s u_s^{\boldsymbol{IV}}(u_s^{\boldsymbol{IV}}-1)}{2^{k+1}} + \frac{(\mathcal{N}+\mathcal{M})u_{\max}^{\boldsymbol{IV}}}{2^k}\\ &\quad + \frac{2\mathcal{M}\mu}{2^b} + \frac{3\mathcal{N}\cdot\mathrm{mucol}(\mathcal{M},2^r)}{2^c}\,,\end{aligned}$$

where the last inequality uses (10.18). This concludes the proof. □

10

10.7 Conclusion

In this chapter, we introduced Kirby and MacaKey, two permutation-based PRF constructions that support both full-state absorption and full-state squeezing. For both constructions, we proved tight security bounds. The bound for MacaKey is of the same order as that of FSKS, and, depending on the choice of the rate, can match the bound of Kirby. Kirby is structually simpler and has already been instantiated in lightweight schemes, while MacaKey is more general. An interesting future work would be to extend the duplex construction [BDPV11b] using the idea underlying MacaKey. This would yield broader applicability, e.g., as an authenticated encryption scheme. However, extending the duplex construction using the idea underlying MacaKey is far from trivial. In particular, designing an authenticated encryption scheme would require a careful treatment of nonces and domain separation to ensure that the resulting scheme is both secure and optimal. Achieving tight, fine-grained security bounds in this setting would necessitate novel analysis.

Part V

Conclusions and Future Directions

Chapter 11

Conclusion and Perspectives

In this thesis, we sharpened the state of the art in provable security for permutation-based cryptography by closing security gaps, introducing new proof techniques, designing constructions that address existing limitations, and systematizing prior knowledge. While the work is theoretical in nature, many of the results support deployed or emerging permutation-based schemes.

Tightened Security Bounds. One direction we explored was closing security gaps for sponge-based hashing. In Chapter 4, we established a tight bound for the preimage resistance of the sponge construction, which indicates that several lightweight sponge-based hash functions enjoy stronger generic preimage security than previously understood. In particular, the hash function of the Ascon suite enjoys 192 bits of *generic* preimage resistance, an improvement over the previously believed 128 bits of security. This raises the concrete question of whether Ascon's preimage resistance can be broken in fewer than 2^{192} queries using cryptanalytic techniques. Some round-reduced analyses have been conducted in this direction; see, e.g., [NHS+24, DZQ+24]. Moreover, the proof technique developed in this chapter serves as a versatile building block. We applied it again in Chapter 7, and it has since been reused in follow-up works [Foe23, SLZ+25].

In Chapter 5 we proved a tight upper bound on the indifferentiability of the sponge when the number of absorbed message blocks is restricted. We also established a tight bound for public indifferentiability of the sponge and showed its implications for the classical properties of collision and second preimage resistance. Both results contribute to a more fine-grained understanding of the sponge's security, and may be used to increase the efficiency of sponge-based schemes in the aforementioned use cases.

New Constructions. In another line of work, we designed constructions that surpass current security and efficiency limits. In Chapter 6, we introduced the double sponge, a permutation-based double block length hash function construction. It is based on a permutation of size $b = r + c$ bits, absorbs r bits per compression function call and achieves $2c/3$ bits of indifferentiability security. When the absorption rate is small relative to the permutation size, the double sponge achieves security beyond the birthday bound in the permutation size. A natural next step is to investigate the double sponge's collision resistance. In Section 6.5.1, we described a collision attack with complexity $2^{\frac{2c+r}{2}}$, and it remains an open question whether collision resistance can be proven close to this bound or whether a better generic attack can be mounted. Another promising direction is to extend this line of research by designing provably secure sponge-based double block length hash functions, aiming to (i) reduce the number of permutation calls per compression function call and/or (ii) achieve security beyond $2c/3$ bits.

In Chapter 10, we introduced two novel full-state-squeezing PRFs: Kirby, a lightweight construction, and MacaKey, which enhances the full-state keyed sponge by integrating the summation-truncation hybrid. Both constructions offer high throughput, full-state absorption, and output generation at the full-state rate with minimal overhead. They also come with strong generic security guarantees, making them compelling alternatives to existing permutation-based PRFs.

Application-Specific Permutation-Based Hashing. In another direction, we explored specific use cases of permutation-based hashing. In Chapter 8, we improved efficiency and security of the sponge and the duplex in arithmetization-oriented settings over large finite fields. This was achieved by partially replacing the padding with lightweight, non-cryptographic permutations applied to the inner state, thereby eliminating the overhead induced by padding. We showed that these techniques, like the sponge and duplex constructions, achieve birthday-bound security in the capacity, but with a constant loss that is independent of the field size.

In Chapter 7, we gave a refined security analysis of hash chains, in particular when the hash function is permutation-based. The bounds that we obtained go significantly beyond what existing general security notions would yield, and show that small permutations can suffice when instantiating the hash function. Central to the security model was the splitting of the adversarial computational power into offline and online phases, which allowed to argue that the adversary's online query access is significantly limited.

A natural extension of this work is to consider memory-bounded adversaries. Indeed, by modeling preprocessing as an offline phase that outputs a limited-size advice string, we can more realistically capture an attacker's capabilities. Although results on time-memory bounds for sponges already exist (i.e., [CDG18, FGK22, ADGL23] for collision resistance of the sponge with random initial values) and fine-grained bounded-memory security models have been established (cf., [TT18]), a systematic treatment of security of permutation-based hashing with memory-bounded adversaries remains open, and would tighten the gap between the adversarial resources captured by provable security bounds and those of practical attacks [Lef24].

Generic Security of the Ascon Modes. Lastly, in Chapter 9 we performed a comprehensive overview on the generic security of the Ascon authenticated encryption mode. Our analysis covered a broad range of models, including nonce-respecting security, nonce-misuse resistance and resilience, leakage resilience, release of unverified plaintext, and we introduced the notion of state recovery security. Notably, these results highlight, from a provable security perspective, the benefits of Ascon's key-blinding mechanism. We also organized and summarized the existing results for Ascon-Hash and Ascon-PRF. Future research could explore Ascon's security under related-key settings and fault attacks, as well as specialize the leakage resilience analysis for concrete leakage functions.

Caveat on Provable Security. The provable security guarantees given in this thesis rest on idealized assumptions and must therefore be interpreted with care. In particular, the ideal permutation model, by definition, abstracts away the internal structure of the permutations used in schemes. For example, the designers of Ascon explicitly acknowledge that their permutations exhibit non-random behavior [DEMS21b, Section 3.1]. Nonetheless, these security bounds remain useful as reference points, as they are viewed as the best possible security levels that can be achieved. Therefore, designers use them for parameter selection, and cryptanalysts use them to identify meaningful attacks. We refer to the works of Bellare [Bel97], Koblitz and Menezes [KM07, KM19], and Damgård [Dam07] for in-depth discussions around provable security.

It would be valuable to explore how far one can move beyond idealized assumptions and still obtain meaningful security proofs. In the keyless setting, this is especially challenging. A related approach, somewhat connected to our work in Chapter 6, is to analyze the actual design structure of the primitives and prove security accordingly. For example, indifferentiability proofs have been carried out for confusion-diffusion networks [DSSL16] and Feistel ciphers [CPS08,Seu09]. In these works, however, the proofs still rely on idealized assumptions, as the underlying building blocks are modeled as ideal components. Thus, the idealized assumptions are not removed but shifted from the whole construction to its parts, with the focus placed on structural soundness rather than standard-model security.

In the keyed setting, several constructions discussed in this thesis can be reduced to the pseudorandom permutation security of the underlying Even-Mansour (EM) block cipher. For instance, the proof strategy used in MacaKey (see Section 10.6.4) follows this approach, as do other works such as [CDH+12, ADMV15, DMV17]. This therefore implicitly brings the security proofs to the standard model, reducing the security of the construction to the security of the EM constructions. However, (i) the reduction typically targets a *family* of EM constructions, for example, in the case of MacaKey, it involves two EM instances, where the second key of the first instance is reused in the second; (ii) stopping at the EM level can hide certain terms, particularly those reflecting the adversary's fine-grained capabilities (such as the parameters $\mathcal{L}$ and Ω, which capture the adversary's ability to overwrite the outer part of the state, see Section 3.2.2.3); and (iii) in some cases, dedicated proofs can yield significantly tighter bounds, see for example the discussion in Section 9.4.3.2 regarding state-recovery security.

Another interesting direction lies in getting closer to differential cryptanalysis, which, among others, studies difference probabilities and establishes bounds on them. For instance, bounds on the differential probabilities of the Ascon permutation have been established (see, e.g., [HMMD22]). Building on this, it would be interesting to investigate in detail under what conditions a permutation-based construction preserves XOR-universality. Fuchs et al. [FRD23] addressed this question for the Farfalle construction [BDH+17] and the construction underlying Pelican MAC [DR05], and prove their universality in terms of the probability of differentials over the public permutation. This was done under the assumption that long, independent keys were applied at each permutation call. A natural open question is whether these key assumptions can be removed or significantly relaxed. Preliminary results suggest that such relaxation fails for keyed sponge constructions; for example, a random involution may be XOR-universal, but completely insecure when used in the sponge construction. Thus, additional assumptions, possibly involving the cyclic properties of the permutation, might be necessary.

Bibliography

[AAG$^+$19] Mark Aagaard, Riham AlTawy, Guang Gong, Kalikinkar Mandal, and Raghvendra Rohit. ACE: An Authenticated Encryption and Hash Algorithm. Second Round Submission to NIST Lightweight Cryptography, 2019.

[AB24] Tomer Ashur and Amit Singh Bhati. Generalized Indifferentiable Sponge and its Application to Polygon Miden VM. Cryptology ePrint Archive, Paper 2024/911, 2024. `https://eprint.iacr.org/2024/911`.

[ABD$^+$23] Roberto Avanzi, Subhadeep Banik, Orr Dunkelman, Maria Eichlseder, Shibam Ghosh, Marcel Nageler, and Francesco Regazzoni. The QARMAv2 Family of Tweakable Block Ciphers. *IACR Trans. Symmetric Cryptol.*, 2023(3):25–73, 2023.

[ABD$^+$24] Parisa Amiri-Eliasi, Yanis Belkheyar, Joan Daemen, Santosh Ghosh, Daniël Kuijsters, Alireza Mehrdad, Silvia Mella, Shahram Rasoolzadeh, and Gilles Van Assche. Koala: A Low-Latency Pseudorandom Function. In Maria Eichlseder and Sébastien Gambs, editors, *Selected Areas in Cryptography - SAC 2024 - 31st International Conference, Montreal, QC, Canada, August 28-30, 2024, Revised Selected Papers, Part II*, volume 15517 of *Lecture Notes in Computer Science*, pages 239–266. Springer, 2024.

[ABK$^+$23] Tomer Ashur, Amit Singh Bhati, Al Kindi, Mohammad Mahzoun, and Léo Perrin. XHash: Efficient STARK-friendly Hash Function. Cryptology ePrint Archive, Paper 2023/1045, 2023. `https://eprint.iacr.org/2023/1045`.

[ABL$^+$13] Elena Andreeva, Andrey Bogdanov, Atul Luykx, Bart Mennink, Elmar Tischhauser, and Kan Yasuda. Parallelizable and Authenticated Online Ciphers. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part I*, volume 8269 of *Lecture Notes in Computer Science*, pages 424–443. Springer, 2013.

[ABL$^+$14] Elena Andreeva, Andrey Bogdanov, Atul Luykx, Bart Mennink, Nicky Mouha, and Kan Yasuda. How to Securely Release Unverified Plaintext in Authenticated Encryption. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 105–125. Springer, 2014.

[ADG$^+$22] Ange Albertini, Thai Duong, Shay Gueron, Stefan Kölbl, Atul Luykx, and Sophie Schmieg. How to Abuse and Fix Authenticated Encryption Without Key Commitment. In Kevin R. B. Butler and Kurt Thomas, editors, *31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022*, pages 3291–3308. USENIX Association, 2022.

[ADGL23] Akshima, Xiaoqi Duan, Siyao Guo, and Qipeng Liu. On Time-Space Lower Bounds for Finding Short Collisions in Sponge Hash Functions. In Guy N. Rothblum and Hoeteck Wee, editors, *Theory of Cryptography - 21st International Conference, TCC 2023, Taipei, Taiwan, November 29 - December 2, 2023, Proceedings, Part III*, volume 14371 of *Lecture Notes in Computer Science*, pages 237–270. Springer, 2023.

[ADL17] 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 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III*, volume 10403 of *Lecture Notes in Computer Science*, pages 3–33. Springer, 2017.

[ADMV15] 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.

[AFK$^+$11] Frederik Armknecht, Ewan Fleischmann, Matthias Krause, Jooyoung Lee, Martijn Stam, and John P. Steinberger. The Preimage Security of Double-Block-Length Compression Functions. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, volume 7073 of *Lecture Notes in Computer Science*, pages 233–251. Springer, 2011.

[AGD24] Parisa Amiri-Eliasi, Koustabh Ghosh, and Joan Daemen. Mystrium: Wide Block Encryption Efficient on Entry-Level Processors. In Clemente Galdi and Duong Hieu Phan, editors, *Security and Cryptography for Networks - 14th International Conference, SCN 2024, Amalfi, Italy, September 11-13, 2024, Proceedings, Part II*, volume 14974 of *Lecture Notes in Computer Science*, pages 71–96. Springer, 2024.

[AGR$^+$16] Martin R. Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen. MiMC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 191–219, 2016.

[AHMN10] Jean-Philippe Aumasson, Luca Henzen, Willi Meier, and María Naya-Plasencia. Quark: A Lightweight Hash. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*, volume 6225 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2010.

[AJN14a] Jean-Philippe Aumasson, Philipp Jovanovic, and Samuel Neves. NORX: Parallel and Scalable AEAD. In Miroslaw Kutylowski and Jaideep Vaidya, editors, *Computer Security - ESORICS 2014 - 19th European Symposium on Research in Computer Security, Wroclaw, Poland, September 7-11, 2014. Proceedings, Part II*, volume 8713 of *Lecture Notes in Computer Science*, pages 19–36. Springer, 2014.

[AJN14b] Jean-Philippe Aumasson, Philipp Jovanovic, and Samuel Neves. NORX v1. Submission to CAESAR competition, 2014.

[AKMQ23] Jean-Philippe Aumasson, Dmitry Khovratovich, Bart Mennink, and Porçu Quine. SAFE: Sponge API for Field Elements. Cryptology ePrint Archive, Paper 2023/522, 2023. `https://eprint.iacr.org/2023/522`.

[AMP10a] Elena Andreeva, Bart Mennink, and Bart Preneel. On the Indifferentiability of the Grøstl Hash Function. In Juan A. Garay and Roberto De Prisco, editors, *Security and Cryptography for Networks, 7th International Conference, SCN 2010, Amalfi, Italy, September 13-15, 2010. Proceedings*, volume 6280 of *Lecture Notes in Computer Science*, pages 88–105. Springer, 2010.

[AMP10b] Elena Andreeva, Bart Mennink, and Bart Preneel. Security Reductions of the Second Round SHA-3 Candidates. In Mike Burmester, Gene Tsudik, Spyros S. Magliveras, and Ivana Ilic, editors, *Information Security - 13th International Conference, ISC 2010, Boca Raton, FL, USA, October 25-28, 2010, Revised Selected Papers*, volume 6531 of *Lecture Notes in Computer Science*, pages 39–53. Springer, 2010.

[AMP12] Elena Andreeva, Bart Mennink, and Bart Preneel. The parazoa family: generalizing the sponge hash functions. *Int. J. Inf. Sec.*, 11(3):149–165, 2012.

[ANPS07] Elena Andreeva, Gregory Neven, Bart Preneel, and Thomas Shrimpton. Seven-Property-Preserving Iterated Hashing: ROX. In Kaoru Kurosawa, editor, *Advances in Cryptology - ASIACRYPT 2007, 13th International Conference on the Theory* *and Application of Cryptology and Information Security, Kuching, Malaysia, December 2-6, 2007, Proceedings*, volume 4833 of *Lecture Notes in Computer Science*, pages 130–146. Springer, 2007.

[AS11] Elena Andreeva and Martijn Stam. The Symbiosis between Collision and Preimage Resistance. In Liqun Chen, editor, *Cryptography and Coding - 13th IMA International Conference, IMACC 2011, Oxford, UK, December 12-15, 2011. Proceedings*, volume 7089 of *Lecture Notes in Computer Science*, pages 152–171. Springer, 2011.

[Ava17] 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 Trans. Symmetric Cryptol.*, 2017(1):4–44, 2017.

[BBC$^+$19] Christof Beierle, Alex Biryukov, Luan Cardoso dos Santos, Johann Großschädl, Léo Perrin, Aleksei Udovenko, Vesselin Velichkov, Qingju Wang, Amir Moradi, and Aein Rezaei Shahmirzadi. Schwaemm and Esch: Lightweight Authenticated Encryption and Hashing using the Sparkle Permutation Family. Final Round Submission to NIST Lightweight Cryptography, 2019.

[BBC$^+$23] Clémence Bouvier, Pierre Briaud, Pyrros Chaidos, Léo Perrin, Robin Salen, Vesselin Velichkov, and Danny Willems. New Design Techniques for Efficient Arithmetization-Oriented Hash Functions: Anemoi Permutations and Jive Compression Mode. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part III*, volume 14083 of *Lecture Notes in Computer Science*, pages 507–539. Springer, 2023.

[BBLT18] Subhadeep Banik, Andrey Bogdanov, Atul Luykx, and Elmar Tischhauser. SUNDAE: Small Universal Deterministic Authenticated Encryption for the Internet of Things. *IACR Trans. Symmetric Cryptol.*, 2018(3):1–35, 2018.

[BBM00] Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-Key Encryption in a Multi-user Setting: Security Proofs and Improvements. In Bart Preneel, editor, *Advances in Cryptology -*

EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding, volume 1807 of *Lecture Notes in Computer Science*, pages 259–274. Springer, 2000.

[BBN22] Arghya Bhattacharjee, Ritam Bhaumik, and Mridul Nandi. A Sponge-Based PRF with Good Multi-user Security. In Benjamin Smith and Huapeng Wu, editors, *Selected Areas in Cryptography - 29th International Conference, SAC 2022, Windsor, ON, Canada, August 24-26, 2022, Revised Selected Papers*, volume 13742 of *Lecture Notes in Computer Science*, pages 459–478. Springer, 2022.

[BCD$^+$19] Zhenzhen Bao, Avik Chakraborti, Nilanjan Datta, Jian Guo, Mridul Nandi, Thomas Peyrin, and Kan Yasuda. PHOTON-Beetle. Final Round Submission to NIST Lightweight Cryptography, 2019.

[BCDM20] 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.

[BCG$^+$12] Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knezevic, Lars R. Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar, Christian Rechberger, Peter Rombouts, Søren S. Thomsen, and Tolga Yalçin. PRINCE - A Low-Latency Block Cipher for Pervasive Computing Applications - Extended Abstract. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, volume 7658 of *Lecture Notes in Computer Science*, pages 208–225. Springer, 2012.

[BCK96] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying Hash Functions for Message Authentication. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 1996.

[BCP22] Jules Baudrin, Anne Canteaut, and Léo Perrin. Practical Cube Attack against Nonce-Misused Ascon. *IACR Trans. Symmetric Cryptol.*, 2022(4):120–144, 2022.

[BD08] Steve Babbage and Matthew Dodd. The MICKEY Stream Ciphers. In Matthew J. B. Robshaw and Olivier Billet, editors, *New Stream Cipher Designs - The eSTREAM Finalists*, volume 4986 of *Lecture Notes in Computer Science*, pages 191–209. Springer, 2008.

[BDD$^+$17] Ritam Bhaumik, Nilanjan Datta, Avijit Dutta, Nicky Mouha, and Mridul Nandi. The Iterated Random Function Problem. 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 667–697. Springer, 2017.

[BDH$^+$17] 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.

[BDP$^+$16a] Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer. Ketje v2. Submission to CAESAR Competition, 2016.

[BDP$^+$16b] Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer. Keyak v2. Submission to CAESAR Competition, 2016.

[BDPA10] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Sponge-Based Pseudo-Random Number Generators. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*, volume 6225 of *Lecture Notes in Computer Science*, pages 33–47. Springer, 2010.

[BDPV06] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. RadioGatún, a belt-and-mill hash function. Second

Cryptographic Hash Workshop, Santa Barbara, August 2006. `http://radiogatun.noekeon.org/`.

[BDPV07] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Sponge functions. Ecrypt Hash Workshop 2007, May 2007.

[BDPV08] 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.

[BDPV11a] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Cryptographic sponge functions, 2011. `https://keccak.team/files/CSF-0.1.pdf`.

[BDPV11b] 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.

[BDPV11c] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. On the Security of the Keyed Sponge Construction. Symmetric Key Encryption Workshop (SKEW 2011), 2011.

[BDPV11d] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. The Keccak reference, January 2011. `https://keccak.team/files/Keccak-reference-3.0.pdf`.

[BDPV12] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Permutation-based encryption, authentication and authenticated encryption. Directions in Authenticated Ciphers, July 2012.

[BEK$^+$20] Dusan Bozilov, Maria Eichlseder, Miroslav Knezevic, Baptiste Lambin, Gregor Leander, Thorben Moos, Ventzislav Nikov, Shahram Rasoolzadeh, Yosuke Todo, and Friedrich Wiemer.

PRINCEv2 - More Security for (Almost) No Overhead. In Orr Dunkelman, Michael J. Jacobson Jr., and Colin O'Flynn, editors, *Selected Areas in Cryptography - SAC 2020 - 27th International Conference, Halifax, NS, Canada (Virtual Event), October 21-23, 2020, Revised Selected Papers*, volume 12804 of *Lecture Notes in Computer Science*, pages 483–511. Springer, 2020.

[Bel97] Mihir Bellare. Practice-Oriented Provable-Security. In Eiji Okamoto, George I. Davida, and Masahiro Mambo, editors, *Information Security, First International Workshop, ISW '97, Tatsunokuchi, Japan, September 17-19, 1997, Proceedings*, volume 1396 of *Lecture Notes in Computer Science*, pages 221–231. Springer, 1997.

[Ber08a] Daniel J. Bernstein. ChaCha, a variant of Salsa20. In *Workshop record of SASC*, 2008.

[Ber08b] Daniel J. Bernstein. The Salsa20 Family of Stream Ciphers. In Matthew J. B. Robshaw and Olivier Billet, editors, *New Stream Cipher Designs - The eSTREAM Finalists*, volume 4986 of *Lecture Notes in Computer Science*, pages 84–97. Springer, 2008.

[Ber09] Daniel J. Bernstein. CubeHash specification (2.B.1). Second Round Submission to NIST SHA-3 Competition, 2009.

[Ber11] Daniel J. Bernstein. Extending the Salsa20 nonce. In *Workshop record of Symmetric Key Encryption Workshop*, volume 2011, 2011.

[BGP$^+$20] Francesco Berti, Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. TEDT, a Leakage-Resist AEAD Mode for High Physical Security Applications. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(1):256–320, 2020.

[BGP$^+$23] Francesco Berti, Chun Guo, Thomas Peters, Yaobin Shen, and François-Xavier Standaert. Secure Message Authentication in the Presence of Leakage and Faults. *IACR Trans. Symmetric Cryptol.*, 2023(1):288–315, 2023.

[BH22] Mihir Bellare and Viet Tung Hoang. Efficient Schemes for Committing Authenticated Encryption. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology - EUROCRYPT 2022 - 41st Annual International Conference on the Theory and*

Applications of Cryptographic Techniques, Trondheim, Norway, May 30 - June 3, 2022, Proceedings, Part II, volume 13276 of *Lecture Notes in Computer Science*, pages 845–875. Springer, 2022.

[Bih02] Eli Biham. How to decrypt or even substitute DES-encrypted messages in 2^{28} steps. *Inf. Process. Lett.*, 84(3):117–124, 2002.

[BJK$^+$16] 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 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 123–153. Springer, 2016.

[BJK$^+$19] 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 v1.1. Second Round Submission to NIST Lightweight Cryptography, 2019.

[BKL$^+$07] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007.

[BKL$^+$11] 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, *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, volume 6917 of *Lecture Notes in Computer Science*, pages 312–325. Springer, 2011.

[BKL$^+$19] Daniel J. Bernstein, Stefan Kölbl, Stefan Lucks, Pedro Maat Costa Massolino, Florian Mendel, Kashif Nawaz, Tobias Schneider, Peter Schwabe, François-Xavier Standaert, Yosuke

Todo, and Benoît Viguier. Gimli. Second Round Submission to NIST Lightweight Cryptography, 2019.

[BKR94] Mihir Bellare, Joe Kilian, and Phillip Rogaway. The Security of Cipher Block Chaining. In Yvo Desmedt, editor, *Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings*, volume 839 of *Lecture Notes in Computer Science*, pages 341–358. Springer, 1994.

[BKR98] Mihir Bellare, Ted Krovetz, and Phillip Rogaway. Luby-Rackoff Backwards: Increasing Security by Making Block Ciphers Non-invertible. In Kaisa Nyberg, editor, *Advances in Cryptology - EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding*, volume 1403 of *Lecture Notes in Computer Science*, pages 266–280. Springer, 1998.

[Blo25] Blockchain website, total hash rate. `https://www.blockchain.com/explorer/charts/hash-rate`, 2025. Accessed: 2025-10-23.

[BM24] Henk Berendsen and Bart Mennink. Tightening Leakage Resilience of the Suffix Keyed Sponge. *IACR Trans. Symmetric Cryptol.*, 2024(1):459–496, 2024.

[BMN10] Rishiraj Bhattacharyya, Avradip Mandal, and Mridul Nandi. Security Analysis of the Mode of JH Hash Function. In Seokhie Hong and Tetsu Iwata, editors, *Fast Software Encryption, 17th International Workshop, FSE 2010, Seoul, Korea, February 7-10, 2010, Revised Selected Papers*, volume 6147 of *Lecture Notes in Computer Science*, pages 168–191. Springer, 2010.

[BMOS17] Guy Barwell, Daniel P. Martin, Elisabeth Oswald, and Martijn Stam. Authenticated Encryption in the Face of Protocol and Side Channel Leakage. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 693–723. Springer, 2017.

[BN00] 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.

[BN08] Mihir Bellare and Chanathip Namprempre. Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. *J. Cryptol.*, 21(4):469–491, 2008.

[BN18] Srimanta Bhattacharya and Mridul Nandi. A note on the chi-square method: A tool for proving cryptographic security. *Cryptogr. Commun.*, 10(5):935–957, 2018.

[BPP$^+$17] Subhadeep Banik, Sumit Kumar Pandey, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Yosuke Todo. GIFT: A Small Present - Towards Reaching the Limit of Lightweight Encryption. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, volume 10529 of *Lecture Notes in Computer Science*, pages 321–345. Springer, 2017.

[BR93] Mihir Bellare and Phillip Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993*, pages 62–73. ACM, 1993.

[BR96] Mihir Bellare and Phillip Rogaway. The Exact Security of Digital Signatures - How to Sign with RSA and Rabin. In Ueli M. Maurer, editor, *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416. Springer, 1996.

[BR06] Mihir Bellare and Phillip Rogaway. The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. 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 409–426. Springer, 2006.

[BS90] Eli Biham and Adi Shamir. Differential Cryptanalysis of DES-like Cryptosystems. In Alfred Menezes and Scott A. Vanstone, editors, *Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings*, volume 537 of *Lecture Notes in Computer Science*, pages 2–21. Springer, 1990.

[BS91] Eli Biham and Adi Shamir. Differential Cryptanalysis of DES-like Cryptosystems. *J. Cryptol.*, 4(1):3–72, 1991.

[BSS+13] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK Families of Lightweight Block Ciphers. Cryptology ePrint Archive, Paper 2013/404, 2013. `https://eprint.iacr.org/2013/404`.

[CAE14] CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness, May 2014. `http://competitions.cr.yp.to/caesar.html`.

[CB18] Paul Crowley and Eric Biggers. Adiantum: length-preserving encryption for entry-level processors. *IACR Trans. Symmetric Cryptol.*, 2018(4):39–61, 2018.

[CD25] Bishwajit Chakraborty and Chandranan Dhar. Breaking and Fixing MacaKey. Cryptology ePrint Archive, Paper 2025/2038, 2025.

[CDD+19] Donghoon Chang, Nilanjan Datta, Avijit Dutta, Bart Mennink, Mridul Nandi, Somitra Sanadhya, and Ferdinand Sibleyras. Release of Unverified Plaintext: Tight Unified Model and Application to ANYDAE. *IACR Trans. Symmetric Cryptol.*, 2019(4):119–146, 2019.

[CDD+25] Yu Long Chen, Michael Davidson, Morris Dworkin, John Kelsey, Yu Sasaki, Meltem Sönmez Turan, Alyssa Thompson, Nicky Mouha, and Donghoon Chang. Requirements for Cryptographic Accordions. NIST IR 8552, 2025.

[CDG18] Sandro Coretti, Yevgeniy Dodis, and Siyao Guo. Non-Uniform Bounds in the Random-Permutation, Ideal-Cipher, and Generic-Group Models. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 693–721. Springer, 2018.

[CDH$^+$12] Donghoon Chang, Morris Dworkin, Seokhie Hong, John Kelsey, and Mridul Nandi. A Keyed Sponge Construction with Pseudorandomness in the Standard Model. NIST SHA–3 Workshop, March 2012.

[CDK09] Christophe De Cannière, Orr Dunkelman, and Miroslav Knezevic. KATAN and KTANTAN - A Family of Small and Efficient Hardware-Oriented Block Ciphers. In Christophe Clavier and Kris Gaj, editors, *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*, volume 5747 of *Lecture Notes in Computer Science*, pages 272–288. Springer, 2009.

[CDMP05] Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-Damgård Revisited: How to Construct a Hash Function. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *Lecture Notes in Computer Science*, pages 430–448. Springer, 2005.

[CDN23a] Bishwajit Chakraborty, Nilanjan Datta, and Mridul Nandi. Designing Full-Rate Sponge Based AEAD Modes. In Anupam Chattopadhyay, Shivam Bhasin, Stjepan Picek, and Chester Rebeiro, editors, *Progress in Cryptology - INDOCRYPT 2023 - 24th International Conference on Cryptology in India, Goa, India, December 10-13, 2023, Proceedings, Part I*, volume 14459 of *Lecture Notes in Computer Science*, pages 89–110. Springer, 2023.

[CDN23b] Bishwajit Chakraborty, Chandranan Dhar, and Mridul Nandi. Exact Security Analysis of ASCON. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, December*

4-8, 2023, Proceedings, Part III, volume 14440 of *Lecture Notes in Computer Science*, pages 346–369. Springer, 2023.

[CDN24] Bishwajit Chakraborty, Chandranan Dhar, and Mridul Nandi. Tight Multi-user Security of Ascon and Its Large Key Extension. In Tianqing Zhu and Yannan Li, editors, *Information Security and Privacy - 29th Australasian Conference, ACISP 2024, Sydney, NSW, Australia, July 15-17, 2024, Proceedings, Part I*, volume 14895 of *Lecture Notes in Computer Science*, pages 57–76. Springer, 2024.

[CDNY18] 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.

[CFN$^+$12] Anne Canteaut, Thomas Fuhr, María Naya-Plasencia, Pascal Paillier, Jean-René Reinhard, and Marion Videau. A Unified Indifferentiability Proof for Permutation- or Block Cipher-Based Hash Functions. Cryptology ePrint Archive, Report 2012/363, 2012. `http://eprint.iacr.org/2012/363`.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The Random Oracle Methodology, Revisited (Preliminary Version). In Jeffrey Scott Vitter, editor, *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 209–218. ACM, 1998.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.

[CJN20] Bishwajit Chakraborty, Ashwin Jha, and Mridul Nandi. On the Security of Sponge-type Authenticated Encryption Modes. *IACR Trans. Symmetric Cryptol.*, 2020(2):93–119, 2020.

[CLL19] Wonseok Choi, ByeongHak Lee, and Jooyoung Lee. Indifferentiability of Truncated Random Permutations. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part I*, volume 11921 of *Lecture Notes in Computer Science*, pages 175–195. Springer, 2019.

[CLM19] Yu Long Chen, Eran Lambooij, and Bart Mennink. How to Build Pseudorandom Functions from Public Random Permutations. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 266–293. Springer, 2019.

[CN08] Donghoon Chang and Mridul Nandi. Improved Indifferentiability Security Analysis of chopMD Hash Function. In Kaisa Nyberg, editor, *Fast Software Encryption, 15th International Workshop, FSE 2008, Lausanne, Switzerland, February 10-13, 2008, Revised Selected Papers*, volume 5086 of *Lecture Notes in Computer Science*, pages 429–443. Springer, 2008.

[CN19] Bishwajit Chakraborty and Mridul Nandi. ORANGE. Second Round Submission to NIST Lightweight Cryptography, 2019.

[CP08] Christophe De Cannière and Bart Preneel. Trivium. In Matthew J. B. Robshaw and Olivier Billet, editors, *New Stream Cipher Designs - The eSTREAM Finalists*, volume 4986 of *Lecture Notes in Computer Science*, pages 244–266. Springer, 2008.

[CPS08] Jean-Sébastien Coron, Jacques Patarin, and Yannick Seurin. The Random Oracle Model and the Ideal Cipher Model Are Equivalent. In David A. Wagner, editor, *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, volume 5157 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2008.

[CR22] John Chan and Phillip Rogaway. On Committing Authenticated-Encryption. In Vijayalakshmi Atluri, Roberto Di Pietro, Christian Damsgaard Jensen, and Weizhi Meng, editors, *Computer Security - ESORICS 2022 - 27th European Symposium on Research in Computer Security, Copenhagen, Denmark, September 26-30, 2022, Proceedings, Part II*, volume 13555 of *Lecture Notes in Computer Science*, pages 275–294. Springer, 2022.

[CS14] Shan Chen and John P. Steinberger. Tight Security Bounds for Key-Alternating Ciphers. 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 327–350. Springer, 2014.

[Dam89] Ivan Damgård. A Design Principle for Hash Functions. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 416–427. Springer, 1989.

[Dam07] Ivan Damgård. A "proof-reading" of Some Issues in Cryptography. In Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki, editors, *Automata, Languages and Programming, 34th International Colloquium, ICALP 2007, Wroclaw, Poland, July 9-13, 2007, Proceedings*, volume 4596 of *Lecture Notes in Computer Science*, pages 2–11. Springer, 2007.

[DDNT23] Nilanjan Datta, Avijit Dutta, Mridul Nandi, and Suprita Talnikar. Tight Multi-User Security Bound of DbHtS. *IACR Trans. Symmetric Cryptol.*, 2023(1):192–223, 2023.

[DEM$^+$17] 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.

[DEM$^+$19] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, Bart Mennink, Robert Primas, and Thomas Unterluggauer. ISAP v2. Final Round Submission to NIST Lightweight Cryptography, 2019.

[DEM$^+$20] 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.

[DEMS14] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1. Submission to CAESAR competition, 2014.

[DEMS21a] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2. Winning submission to NIST Lightweight Cryptography, 2021.

[DEMS21b] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2: Lightweight Authenticated Encryption and Hashing. *J. Cryptol.*, 34(3):33, 2021.

[DEMS24] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon MAC, PRF, and Short-Input PRF - Lightweight, Fast, and Efficient Pseudorandom Functions. In Elisabeth Oswald, editor, *Topics in Cryptology - CT-RSA 2024 - Cryptographers' Track at the RSA Conference 2024, San Francisco, CA, USA, May 6-9, 2024, Proceedings*, volume 14643 of *Lecture Notes in Computer Science*, pages 381–403. Springer, 2024.

[DFG23] Jean Paul Degabriele, Marc Fischlin, and Jérôme Govinden. The Indifferentiability of the Duplex and Its Practical Applications. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, December 4-8, 2023, Proceedings, Part VIII*, volume 14445 of *Lecture Notes in Computer Science*, pages 237–269. Springer, 2023.

[DFL12] Marion Daubignard, Pierre-Alain Fouque, and Yassine Lakhnech. Generic Indifferentiability Proofs of Hash Designs. In Stephen Chong, editor, *25th IEEE Computer Security Foundations Symposium, CSF 2012, Cambridge, MA, USA, June 25-27, 2012*, pages 340–353. IEEE Computer Society, 2012.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theory*, 22(6):644–654, 1976.

[DHP$^+$20] Joan Daemen, Seth Hoffert, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer. Xoodyak, a lightweight cryptographic scheme. *IACR Trans. Symmetric Cryptol.*, 2020(S1):60–87, 2020.

[DHP$^+$21] Joan Daemen, Seth Hoffert, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer. Xoodyak, a lightweight cryptographic scheme. Final Round Submission to NIST Lightweight Cryptography, 2021.

[DHT17] Wei Dai, Viet Tung Hoang, and Stefano Tessaro. Information-Theoretic Indistinguishability via the Chi-Squared Method. In

Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III*, volume 10403 of *Lecture Notes in Computer Science*, pages 497–523. Springer, 2017.

[Din20] Itai Dinur. Tight Time-Space Lower Bounds for Finding Multiple Collision Pairs and Their Applications. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 405–434. Springer, 2020.

[Din24] Itai Dinur. Tight Indistinguishability Bounds for the XOR of Independent Random Permutations by Fourier Analysis. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part I*, volume 14651 of *Lecture Notes in Computer Science*, pages 33–62. Springer, 2024.

[DJS19] Jean Paul Degabriele, Christian Janson, and Patrick Struck. Sponges Resist Leakage: The Case of Authenticated Encryption. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part II*, volume 11922 of *Lecture Notes in Computer Science*, pages 209–240. Springer, 2019.

[DKL09] Yevgeniy Dodis, Yael Tauman Kalai, and Shachar Lovett. On cryptography with auxiliary input. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 621–630. ACM, 2009.

[DM19a] Christoph Dobraunig and Bart Mennink. Leakage Resilience of the Duplex Construction. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019 - 25th*

International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part III, volume 11923 of *Lecture Notes in Computer Science*, pages 225–255. Springer, 2019.

[DM19b] Christoph Dobraunig and Bart Mennink. Security of the Suffix Keyed Sponge. *IACR Trans. Symmetric Cryptol.*, 2019(4):223–248, 2019.

[DM20] Christoph Dobraunig and Bart Mennink. Tightness of the Suffix Keyed Sponge Bound. *IACR Trans. Symmetric Cryptol.*, 2020(4):195–212, 2020.

[DM24] Christoph Dobraunig and Bart Mennink. Generalized Initialization of the Duplex Construction. In Christina Pöpper and Lejla Batina, editors, *Applied Cryptography and Network Security - 22nd International Conference, ACNS 2024, Abu Dhabi, United Arab Emirates, March 5-8, 2024, Proceedings, Part II*, volume 14584 of *Lecture Notes in Computer Science*, pages 460–484. Springer, 2024.

[DMM20] Christoph Dobraunig, Florian Mendel, and Bart Mennink. Practical forgeries for ORANGE. *Inf. Process. Lett.*, 159-160:105961, 2020.

[DMMS21] Sébastien Duval, Pierrick Méaux, Charles Momin, and François-Xavier Standaert. Exploring Crypto-Physical Dark Matter and Learning with Physical Rounding Towards Secure and Efficient Fresh Re-Keying. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(1):373–401, 2021.

[DMP22] Christoph Dobraunig, Bart Mennink, and Robert Primas. Leakage and Tamper Resilient Permutation-Based Cryptography. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, pages 859–873. ACM, 2022.

[DMR19] Joan Daemen, Pedro Maat Costa Massolino, and Yann Rotella. The Subterranean 2.0 cipher suite. Second Round Submission to NIST Lightweight Cryptography, 2019.

[DMV17] 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.

[DP08] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-Resilient Cryptography. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 293–302. IEEE Computer Society, 2008.

[DP10] Yevgeniy Dodis and Krzysztof Pietrzak. Leakage-Resilient Pseudorandom Functions and Side-Channel Attacks on Feistel Networks. In Tal Rabin, editor, *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, volume 6223 of *Lecture Notes in Computer Science*, pages 21–40. Springer, 2010.

[DPVR00] Joan Daemen, Michaël Peeters, Gilles Van Assche, and Vincent Rijmen. Nessie Proposal: NOEKEON. First Open NESSIE Workshop, 2000.

[DR02] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard.* Information Security and Cryptography. Springer, 2002.

[DR05] Joan Daemen and Vincent Rijmen. The MAC function Pelican 2.0. Cryptology ePrint Archive, Paper 2005/088, 2005. `https://eprint.iacr.org/2005/088`.

[DRS09] Yevgeniy Dodis, Thomas Ristenpart, and Thomas Shrimpton. Salvaging Merkle-Damgård for Practical Applications. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, volume 5479 of *Lecture Notes in Computer Science*, pages 371–388. Springer, 2009.

[DSSL16] Yevgeniy Dodis, Martijn Stam, John P. Steinberger, and Tianren Liu. Indifferentiability of Confusion-Diffusion Networks. In Marc

Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 679–704. Springer, 2016.

[Dwo01] Morris Dworkin. Recommendation for Block Cipher Modes of Operation. NIST SP 800-38A, 2001. `https://doi.org/10.6028/NIST.SP.800-38A`.

[Dwo04] Morris Dworkin. Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality. NIST SP 800-38C, 2004. `https://doi.org/10.6028/NIST.SP.800-38C`.

[Dwo05] Morris Dworkin. Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication. NIST SP 800-38B, 2005. `https://doi.org/10.6028/NIST.SP.800-38B-2005`.

[Dwo07] Morris Dworkin. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. NIST SP 800-38D, 2007. `https://doi.org/10.6028/NIST.SP.800-38D`.

[DZQ$^+$24] Xiaoyang Dong, Boxin Zhao, Lingyue Qin, Qingliang Hou, Shun Zhang, and Xiaoyun Wang. Generic MitM Attack Frameworks on Sponge Constructions. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part IV*, volume 14923 of *Lecture Notes in Computer Science*, pages 3–37. Springer, 2024.

[ECR04] ECRYPT. Stream Cipher Project, 2004.

[EM91] Shimon Even and Yishay Mansour. A Construction of a Cipher From a Single Pseudorandom Permutation. In Hideki Imai, Ronald L. Rivest, and Tsutomu Matsumoto, editors, *Advances in Cryptology - ASIACRYPT '91, International Conference on the Theory and Applications of Cryptology, Fujiyoshida, Japan, November 11-14, 1991, Proceedings*, volume 739 of *Lecture Notes in Computer Science*, pages 210–224. Springer, 1991.

[EM97] Shimon Even and Yishay Mansour. A Construction of a Cipher from a Single Pseudorandom Permutation. *J. Cryptol.*, 10(3):151–162, 1997.

[FFL12] Ewan Fleischmann, Christian Forler, and Stefan Lucks. McOE: A Family of Almost Foolproof On-Line Authenticated Encryption Schemes. In Anne Canteaut, editor, *Fast Software Encryption - 19th International Workshop, FSE 2012, Washington, DC, USA, March 19-21, 2012. Revised Selected Papers*, volume 7549 of *Lecture Notes in Computer Science*, pages 196–215. Springer, 2012.

[FGK22] Cody Freitag, Ashrujit Ghoshal, and Ilan Komargodski. Time-Space Tradeoffs for Sponge Hashing: Attacks and Limitations for Short Collisions. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part III*, volume 13509 of *Lecture Notes in Computer Science*, pages 131–160. Springer, 2022.

[FGL09] Ewan Fleischmann, Michael Gorski, and Stefan Lucks. Security of Cyclic Double Block Length Hash Functions. In Matthew Geoffrey Parker, editor, *Cryptography and Coding, 12th IMA International Conference, Cryptography and Coding 2009, Cirencester, UK, December 15-17, 2009. Proceedings*, volume 5921 of *Lecture Notes in Computer Science*, pages 153–175. Springer, 2009.

[FH15] Benjamin Fuller and Ariel Hamlin. Unifying Leakage Classes: Simulatable Leakage and Pseudoentropy. In Anja Lehmann and Stefan Wolf, editors, *Information Theoretic Security - 8th International Conference, ICITS 2015, Lugano, Switzerland, May 2-5, 2015. Proceedings*, volume 9063 of *Lecture Notes in Computer Science*, pages 69–86. Springer, 2015.

[FID25] FIDO Alliance – Open Authentication Standards More Secure than Passwords. `https://fidoalliance.org`, 2025. Accessed: 2025-07-25.

[Foe23] Robin Foekens. Security of the Sponge Construction with a Random Transformation. Bachelor's Thesis, 2023.

[FPS12] Sebastian Faust, Krzysztof Pietrzak, and Joachim Schipper. Practical Leakage-Resilient Symmetric Cryptography. In Emmanuel

Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, volume 7428 of *Lecture Notes in Computer Science*, pages 213–232. Springer, 2012.

[FRD23] Jonathan Fuchs, Yann Rotella, and Joan Daemen. On the Security of Keyed Hashing Based on Public Permutations. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part III*, volume 14083 of *Lecture Notes in Computer Science*, pages 607–627. Springer, 2023.

[FS86] Amos Fiat and Adi Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

[GBKR23] Henri Gilbert, Rachelle Heim Boissier, Louiza Khati, and Yann Rotella. Generic Attack on Duplex-Based AEAD Modes Using Random Function Statistics. 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 348–378. Springer, 2023.

[GDM19] Aldo Gunsing, Joan Daemen, and Bart Mennink. Deck-Based Wide Block Cipher Modes and an Exposition of the Blinded Keyed Hashing Model. *IACR Trans. Symmetric Cryptol.*, 2019(4):1–22, 2019.

[Gen09] Craig Gentry. *A fully homomorphic encryption scheme.* PhD thesis, Stanford University, USA, 2009.

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. On the Cryptographic Applications of Random Functions. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August*

19-22, 1984, Proceedings, volume 196 of *Lecture Notes in Computer Science*, pages 276–288. Springer, 1984.

[GIM22] Chun Guo, Tetsu Iwata, and Kazuhiko Minematsu. New indifferentiability security proof of MDPH hash function. *IET Inf. Secur.*, 16(4):262–281, 2022.

[GKL$^+$22] Lorenzo Grassi, Dmitry Khovratovich, Reinhard Lüftenegger, Christian Rechberger, Markus Schofnegger, and Roman Walch. Reinforced Concrete: A Fast Hash Function for Verifiable Computation. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, pages 1323–1335. ACM, 2022.

[GKL$^+$24] Lorenzo Grassi, Dmitry Khovratovich, Reinhard Lüftenegger, Christian Rechberger, Markus Schofnegger, and Roman Walch. Monolith: Circuit-Friendly Hash Functions with New Nonlinear Layers for Fast and Constant-Time Implementations. *IACR Trans. Symmetric Cryptol.*, 2024(3):44–83, 2024.

[GKM$^+$11] Praveen Gauravaram, Lars Knudsen, Krystian Matusiewicz, Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren Thomsen. Grøstl – a SHA-3 candidate, 2011. Submission to NIST's SHA-3 competition.

[GKR$^+$21] Lorenzo Grassi, Dmitry Khovratovich, Christian Rechberger, Arnab Roy, and Markus Schofnegger. Poseidon: A New Hash Function for Zero-Knowledge Proof Systems. In Michael D. Bailey and Rachel Greenstadt, editors, *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, pages 519–535. USENIX Association, 2021.

[GKS23] Lorenzo Grassi, Dmitry Khovratovich, and Markus Schofnegger. Poseidon2: A Faster Version of the Poseidon Hash Function. In Nadia El Mrabet, Luca De Feo, and Sylvain Duquesne, editors, *Progress in Cryptology - AFRICACRYPT 2023 - 14th International Conference on Cryptology in Africa, Sousse, Tunisia, July 19-21, 2023, Proceedings*, volume 14064 of *Lecture Notes in Computer Science*, pages 177–203. Springer, 2023.

[GLR17] Paul Grubbs, Jiahui Lu, and Thomas Ristenpart. Message Franking via Committing Authenticated Encryption. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III*, volume 10403 of *Lecture Notes in Computer Science*, pages 66–97. Springer, 2017.

[GM82] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information. In Harry R. Lewis, Barbara B. Simons, Walter A. Burkhard, and Lawrence H. Landweber, editors, *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, May 5-7, 1982, San Francisco, California, USA*, pages 365–377. ACM, 1982.

[GM20] Aldo Gunsing and Bart Mennink. The Summation-Truncation Hybrid: Reusing Discarded Bits for Free. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 187–217. Springer, 2020.

[GM22] Lorenzo Grassi and Bart Mennink. Security of Truncated Permutation Without Initial Value. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part II*, volume 13792 of *Lecture Notes in Computer Science*, pages 620–650. Springer, 2022.

[GMS$^+$15] Danilo Gligoroski, Hristina Mihajloska, Simona Samardjiska, Håkon Jacobsen, Mohamed El-Hadedy, Rune Erlend Jensen, and Daniel Otte. π-Cipher v2.0. Submission to CAESAR Competition, 2015.

[Gon81] Gaston H. Gonnet. Expected Length of the Longest Probe Sequence in Hash Code Searching. *J. ACM*, 28(2):289–304, 1981.

[GPP11] Jian Guo, Thomas Peyrin, and Axel Poschmann. The PHOTON Family of Lightweight Hash Functions. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011.*

Proceedings, volume 6841 of *Lecture Notes in Computer Science*, pages 222–239. Springer, 2011.

[GPPS19a] Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Authenticated Encryption with Nonce Misuse and Physical Leakage: Definitions, Separation Results and First Construction - (Extended Abstract). In Peter Schwabe and Nicolas Thériault, editors, *Progress in Cryptology - LATINCRYPT 2019 - 6th International Conference on Cryptology and Information Security in Latin America, Santiago de Chile, Chile, October 2-4, 2019, Proceedings*, volume 11774 of *Lecture Notes in Computer Science*, pages 150–172. Springer, 2019.

[GPPS19b] Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Towards Low-Energy Leakage-Resistant Authenticated Encryption from the Duplex Sponge Construction. Cryptology ePrint Archive, Report 2019/193, 2019. `http://eprint.iacr.org/2019/193` (full version of [GPPS20]).

[GPPS20] Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Towards Low-Energy Leakage-Resistant Authenticated Encryption from the Duplex Sponge Construction. *IACR Trans. Symmetric Cryptol.*, 2020(1):6–42, 2020.

[GPT15] Peter Gazi, 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, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 368–387. Springer, 2015.

[GT23] Ashrujit Ghoshal and Stefano Tessaro. The Query-Complexity of Preprocessing Attacks. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part II*, volume 14082 of *Lecture Notes in Computer Science*, pages 482–513. Springer, 2023.

[GTW24] Felix Günther, Martin Thomson, and Christopher A. Wood. Usage Limits on AEAD Algorithms. Internet Engineering Task Force, Internet-Draft, draft-irtf-cfrg-aead-limits-09, October 2024.

[Hal95] Neil Haller. The S/KEY One-Time Password System. Request for Comments (RFC) 1760, February 1995.

[HBHW23] Daira Hopwood, Sean Bowe, Taylor Hornby, and Nathan Wilcox. ZCash protocol specification, 2023. `https://github.com/zcash/zips/blob/master/protocol/protocol.pdf`.

[Hir04] Shoichi Hirose. Provably Secure Double-Block-Length Hash Functions in a Black-Box Model. In Choonsik Park and Seongtaek Chee, editors, *Information Security and Cryptology - ICISC 2004, 7th International Conference, Seoul, Korea, December 2-3, 2004, Revised Selected Papers*, volume 3506 of *Lecture Notes in Computer Science*, pages 330–342. Springer, 2004.

[Hir06] Shoichi Hirose. Some Plausible Constructions of Double-Block-Length Hash Functions. In Matthew J. B. Robshaw, editor, *Fast Software Encryption, 13th International Workshop, FSE 2006, Graz, Austria, March 15-17, 2006, Revised Selected Papers*, volume 4047 of *Lecture Notes in Computer Science*, pages 210–225. Springer, 2006.

[Hir18] Shoichi Hirose. Sequential Hashing with Minimum Padding. *Cryptogr.*, 2(2):11, 2018.

[HJMM08] Martin Hell, Thomas Johansson, Alexander Maximov, and Willi Meier. The Grain Family of Stream Ciphers. In Matthew J. B. Robshaw and Olivier Billet, editors, *New Stream Cipher Designs - The eSTREAM Finalists*, volume 4986 of *Lecture Notes in Computer Science*, pages 179–190. Springer, 2008.

[HMMD22] Solane El Hirch, Silvia Mella, Alireza Mehrdad, and Joan Daemen. Improved Differential and Linear Trail Bounds for ASCON. *IACR Trans. Symmetric Cryptol.*, 2022(4):145–178, 2022.

[Hoe94] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. In *The collected works of Wassily Hoeffding*, pages 409–426. Springer, 1994.

[HPY07] Shoichi Hirose, Je Hong Park, and Aaram Yun. A Simple Variant of the Merkle-Damgård Scheme with a Permutation. In Kaoru Kurosawa, editor, *Advances in Cryptology - ASIACRYPT 2007, 13th International Conference on the Theory and Application of*

Cryptology and Information Security, Kuching, Malaysia, December 2-6, 2007, Proceedings, volume 4833 of *Lecture Notes in Computer Science*, pages 113–129. Springer, 2007.

[HRRV15] Viet Tung Hoang, Reza Reyhanitabar, Phillip Rogaway, and Damian Vizár. Online Authenticated-Encryption and its Nonce-Reuse Misuse-Resistance. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 493–517. Springer, 2015.

[HSH$^+$06] Deukjo Hong, Jaechul Sung, Seokhie Hong, Jongin Lim, Sangjin Lee, Bonseok Koo, Changhoon Lee, Donghoon Chang, Jesang Lee, Kitae Jeong, Hyun Kim, Jongsung Kim, and Seongtaek Chee. HIGHT: A New Block Cipher Suitable for Low-Resource Device. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, volume 4249 of *Lecture Notes in Computer Science*, pages 46–59. Springer, 2006.

[HT16] 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 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 3–32. Springer, 2016.

[IK03] Tetsu Iwata and Kaoru Kurosawa. OMAC: One-Key CBC MAC. In Thomas Johansson, editor, *Fast Software Encryption, 10th International Workshop, FSE 2003, Lund, Sweden, February 24-26, 2003, Revised Papers*, volume 2887 of *Lecture Notes in Computer Science*, pages 129–153. Springer, 2003.

[IKMP20] Tetsu Iwata, Mustafa Khairallah, Kazuhiko Minematsu, and Thomas Peyrin. Duel of the Titans: The Romulus and Remus Families of Lightweight AEAD Algorithms. *IACR Trans. Symmetric Cryptol.*, 2020(1):43–120, 2020.

[IS09] Kevin M. Igoe and Jerome A. Solinas. AES Galois Counter Mode for the Secure Shell Transport Layer Protocol. Request for Comments (RFC) 5647, 2009.

[Iwa06] Tetsu Iwata. New Blockcipher Modes of Operation with Beyond the Birthday Bound Security. In Matthew J. B. Robshaw, editor, *Fast Software Encryption, 13th International Workshop, FSE 2006, Graz, Austria, March 15-17, 2006, Revised Selected Papers*, volume 4047 of *Lecture Notes in Computer Science*, pages 310–327. Springer, 2006.

[JK77] Norman Lloyd Johnson and Samuel Kotz. Urn models and their application; an approach to modern discrete probability theory. 1977.

[JLM14] 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, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 85–104. Springer, 2014.

[JLM$^+$19] Philipp Jovanovic, Atul Luykx, Bart Mennink, Yu Sasaki, and Kan Yasuda. Beyond Conventional Security in Sponge-Based Authenticated Encryption Modes. *J. Cryptol.*, 32(3):895–940, 2019.

[JN20] Ashwin Jha and Mridul Nandi. Tight Security of Cascaded LRW2. *J. Cryptol.*, 33(3):1272–1317, 2020.

[JÖS12] Dimitar Jetchev, Onur Özen, and Martijn Stam. Collisions Are Not Incidental: A Compression Function Exploiting Discrete Geometry. In Ronald Cramer, editor, *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, volume 7194 of *Lecture Notes in Computer Science*, pages 303–320. Springer, 2012.

[Jou04] Antoine Joux. Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions. In Matthew K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California,* *USA, August 15-19, 2004, Proceedings*, volume 3152 of *Lecture Notes in Computer Science*, pages 306–316. Springer, 2004.

[JT19] Joseph Jaeger and Stefano Tessaro. Tight Time-Memory Trade-Offs for Symmetric Encryption. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 467–497. Springer, 2019.

[Kah96] David Kahn. *The Codebreakers: The comprehensive history of secret communication from ancient times to the internet.* Simon and Schuster, 1996.

[KBM23] Dmitry Khovratovich, Mario Marhuenda Beltrán, and Bart Mennink. Generic Security of the SAFE API and Its Applications. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, December 4-8, 2023, Proceedings, Part VIII*, volume 14445 of *Lecture Notes in Computer Science*, pages 301–327. Springer, 2023.

[Ker83] Auguste Kerckhoffs. La cryptographie militaire, journal des sciences militaires, vol, 1883.

[KKW18] Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. Improved Non-Interactive Zero Knowledge with Applications to Post-Quantum Signatures. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 525–537. ACM, 2018.

[KM07] Neal Koblitz and Alfred Menezes. Another Look at "Provable Security". *J. Cryptol.*, 20(1):3–37, 2007.

[KM19] Neal Koblitz and Alfred Menezes. Critical perspectives on provable security: Fifteen years of "another look" papers. *Adv. Math. Commun.*, 13(4):517–558, 2019.

[KMB17] Dmitry Kogan, Nathan Manohar, and Dan Boneh. T/Key: Second-Factor Authentication From Secure Hash Chains. In Bhavani Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 983–999. ACM, 2017.

[KP97] Lars R. Knudsen and Bart Preneel. Fast and Secure Hashing Based on Codes. In Burton S. Kaliski Jr., editor, *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*, volume 1294 of *Lecture Notes in Computer Science*, pages 485–498. Springer, 1997.

[KR11] Ted Krovetz and Phillip Rogaway. The Software Performance of Authenticated-Encryption Modes. In Antoine Joux, editor, *Fast Software Encryption - 18th International Workshop, FSE 2011, Lyngby, Denmark, February 13-16, 2011, Revised Selected Papers*, volume 6733 of *Lecture Notes in Computer Science*, pages 306–327. Springer, 2011.

[KR19] Yael Tauman Kalai and Leonid Reyzin. A Survey of Leakage-Resilient Cryptography. Cryptology ePrint Archive, Report 2019/302, 2019. `http://eprint.iacr.org/2019/302`.

[KRS19] Mustafa Khairallah, Raghvendra Rohit, and Sumanta Sarkar. Round 2 official comments: ORANGE. Submission to NIST LWC Standardization Process, 2019. `https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/official-comments/orange-round2-official-comment.pdf`.

[KRT07] Lars R. Knudsen, Christian Rechberger, and Søren S. Thomsen. The Grindahl Hash Functions. In Alex Biryukov, editor, *Fast Software Encryption, 14th International Workshop, FSE 2007, Luxembourg, Luxembourg, March 26-28, 2007, Revised Selected Papers*, volume 4593 of *Lecture Notes in Computer Science*, pages 39–57. Springer, 2007.

[KSW24] Juliane Krämer, Patrick Struck, and Maximiliane Weishäupl. Committing AE from Sponges Security Analysis of the NIST

LWC Finalists. *IACR Trans. Symmetric Cryptol.*, 2024(4):191–248, 2024.

[Lam81] Leslie Lamport. Password Authentification with Insecure Communication. *Commun. ACM*, 24(11):770–772, 1981.

[LAMP12] Atul Luykx, Elena Andreeva, Bart Mennink, and Bart Preneel. Impossibility Results for Indifferentiability with Resets. Cryptology ePrint Archive, Paper 2012/644, 2012. `https://eprint.iacr.org/2012/644`.

[LB25] Charlotte Lefevre and Mario Marhuenda Beltrán. MacaKey: Full-State Keyed Sponge Meets the Summation-Truncation Hybrid. Cryptology ePrint Archive, Paper 2025/893, 2025. `https://eprint.iacr.org/2025/893`.

[LBD23] Charlotte Lefevre, Yanis Belkheyar, and Joan Daemen. Kirby: A Robust Permutation-Based PRF Construction. Cryptology ePrint Archive, Paper 2023/1520, 2023. `https://eprint.iacr.org/2023/1520`.

[LBM25] Charlotte Lefevre, Mario Marhuenda Beltrán, and Bart Mennink. To Pad or Not to Pad? Padding-Free Arithmetization-Oriented Sponges. *IACR Trans. Symmetric Cryptol.*, 2025(1):97–137, 2025.

[Lef23] Charlotte Lefevre. Indifferentiability of the Sponge Construction with a Restricted Number of Message Blocks. *IACR Trans. Symmetric Cryptol.*, 2023(1):224–243, 2023.

[Lef24] Charlotte Lefevre. A Note on Adversarial Online Complexity in Security Proofs of Duplex-Based Authenticated Encryption Modes. Cryptology ePrint Archive, Report 2024/213, 2024. `http://eprint.iacr.org/2024/213`.

[LGR21] Julia Len, Paul Grubbs, and Thomas Ristenpart. Partitioning Oracle Attacks. In Michael D. Bailey and Rachel Greenstadt, editors, *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, pages 195–212. USENIX Association, 2021.

[LK11] Jooyoung Lee and Daesung Kwon. The Security of Abreast-DM in the Ideal Cipher Model. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, 94-A(1):104–109, 2011.

[LM92] Xuejia Lai and James L. Massey. Hash Function Based on Block Ciphers. In Rainer A. Rueppel, editor, *Advances in Cryptology - EUROCRYPT '92, Workshop on the Theory and Application of of Cryptographic Techniques, Balatonfüred, Hungary, May 24-28, 1992, Proceedings*, volume 658 of *Lecture Notes in Computer Science*, pages 55–70. Springer, 1992.

[LM22] Charlotte Lefevre and Bart Mennink. Tight Preimage Resistance of the Sponge Construction. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part IV*, volume 13510 of *Lecture Notes in Computer Science*, pages 185–204. Springer, 2022.

[LM24a] Charlotte Lefevre and Bart Mennink. Generic Security of the Ascon Mode: On the Power of Key Blinding. In Maria Eichlseder and Sébastien Gambs, editors, *Selected Areas in Cryptography - SAC 2024 - 31st International Conference, Montreal, QC, Canada, August 28-30, 2024, Revised Selected Papers, Part II*, volume 15517 of *Lecture Notes in Computer Science*, pages 3–32. Springer, 2024.

[LM24b] Charlotte Lefevre and Bart Mennink. Permutation-Based Hash Chains with Application to Password Hashing. *IACR Trans. Symmetric Cryptol.*, 2024(4):249–286, 2024.

[LM24c] Charlotte Lefevre and Bart Mennink. Permutation-Based Hashing Beyond the Birthday Bound. *IACR Trans. Symmetric Cryptol.*, 2024(1):71–113, 2024.

[LM25a] Charlotte Lefevre and Bart Mennink. SoK: Security of the Ascon Modes. *IACR Trans. Symmetric Cryptol.*, 2025(1):138–210, 2025.

[LM25b] Gaëtan Leurent and César Mathéus. Generic Attacks on Double Block Length Sponge Hashing. *IACR Trans. Symmetric Cryptol.*, 2025(4), 2025. to appear.

[LMO$^+$14] Jake Longo, Daniel P. Martin, Elisabeth Oswald, Daniel Page, Martijn Stam, and Michael Tunstall. Simulatable Leakage: Analysis, Pitfalls, and New Constructions. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th*

International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I, volume 8873 of *Lecture Notes in Computer Science*, pages 223–242. Springer, 2014.

[LMP17] Atul Luykx, Bart Mennink, and Kenneth G. Paterson. Analyzing Multi-key Security Degradation. 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 575–605. Springer, 2017.

[LNS18] Gaëtan Leurent, Mridul Nandi, and Ferdinand Sibleyras. Generic Attacks Against Beyond-Birthday-Bound MACs. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 306–336. Springer, 2018.

[LRW02] Moses D. Liskov, Ronald L. Rivest, and David A. Wagner. Tweakable Block Ciphers. In Moti Yung, editor, *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, volume 2442 of *Lecture Notes in Computer Science*, pages 31–46. Springer, 2002.

[LS15] Jooyoung Lee and Martijn Stam. MJH: a faster alternative to MDC-2. *Des. Codes Cryptogr.*, 76(2):179–205, 2015.

[LSS11] Jooyoung Lee, Martijn Stam, and John P. Steinberger. The Collision Security of Tandem-DM in the Ideal Cipher Model. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*, pages 561–577. Springer, 2011.

[Luc00] Stefan Lucks. The Sum of PRPs Is a Secure PRF. In Bart Preneel, editor, *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic*

Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding, volume 1807 of *Lecture Notes in Computer Science*, pages 470–484. Springer, 2000.

[Mat93] Mitsuru Matsui. Linear Cryptanalysis Method for DES Cipher. In Tor Helleseth, editor, *Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer, 1993.

[Men12] Bart Mennink. Optimal Collision Security in Double Block Length Hashing with Single Length Key. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, volume 7658 of *Lecture Notes in Computer Science*, pages 526–543. Springer, 2012.

[Men13] Bart Mennink. Indifferentiability of Double Length Compression Functions. In Martijn Stam, editor, *Cryptography and Coding - 14th IMA International Conference, IMACC 2013, Oxford, UK, December 17-19, 2013. Proceedings*, volume 8308 of *Lecture Notes in Computer Science*, pages 232–251. Springer, 2013.

[Men14] Bart Mennink. On the collision and preimage security of MDC-4 in the ideal cipher model. *Des. Codes Cryptogr.*, 73(1):121–150, 2014.

[Men18] Bart Mennink. Key Prediction Security of Keyed Sponges. *IACR Trans. Symmetric Cryptol.*, 2018(4):128–149, 2018.

[Men23] Bart Mennink. Understanding the Duplex and Its Security. *IACR Trans. Symmetric Cryptol.*, 2023(2):1–46, 2023.

[Men25] Bart Mennink. Keying Merkle-Damgård at the Suffix. *IACR Trans. Symmetric Cryptol.*, 2025(1):70–96, 2025.

[Mer89] Ralph C. Merkle. One Way Hash Functions and DES. In Gilles Brassard, editor, *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 428–446. Springer, 1989.

[MGH$^+$15] Pawel Morawiecki, Kris Gaj, Ekawat Homsirikamol, Krystian Matusiewicz, Josef Pieprzyk, Marcin Rogawski, Marian Srebrny, and Marcin Wójcik. ICEPOLE v2.0. Submission to CAESAR Competition, 2015.

[Mil82] Frank Miller. *Telegraphic code to insure privacy and secrecy in the transmission of telegrams*. C.M. Cornwell, New York, 1882.

[MK22] Mary Maller and Dmitry Khovratovich. Baloo: open source implementation, 2022. `https://github.com/mmaller/caulk-dev/tree/main/baloo`.

[MMH$^+$14] 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.

[MMPR11] David M'Raïhi, Salah Machani, Mingliang Pei, and Johan Rydell. TOTP: Time-Based One-Time Password Algorithm. Request for Comments (RFC) 6238, May 2011.

[MP12] Bart Mennink and Bart Preneel. Hash Functions Based on Three Permutations: A Generic Security Analysis. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, volume 7417 of *Lecture Notes in Computer Science*, pages 330–347. Springer, 2012.

[MPS12] Avradip Mandal, Jacques Patarin, and Yannick Seurin. On the Public Indifferentiability and Correlation Intractability of the 6-Round Feistel Construction. In Ronald Cramer, editor, *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, volume 7194 of *Lecture Notes in Computer Science*, pages 285–302. Springer, 2012.

[MPS16] Dustin Moody, Souradyuti Paul, and Daniel Smith-Tone. Improved indifferentiability security bound for the JH mode. *Des. Codes Cryptogr.*, 79(2):237–259, 2016.

[MRH04] Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology. In Moni Naor, editor, *Theory of Cryptography, First Theory of Cryptography Conference, TCC 2004, Cambridge, MA, USA, February 19-21, 2004, Proceedings*, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39. Springer, 2004.

[MRV15] 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, *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, volume 9453 of *Lecture Notes in Computer Science*, pages 465–489. Springer, 2015.

[MS88] Carl H Meyer and Michael Schilling. Secure program load with manipulation detection code. In *Proc. Securicom*, volume 88, pages 111–130, 1988.

[MV04] 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, *Progress in Cryptology - INDOCRYPT 2004, 5th International Conference on Cryptology in India, Chennai, India, December 20-22, 2004, Proceedings*, volume 3348 of *Lecture Notes in Computer Science*, pages 343–355. Springer, 2004.

[Nai16] Yusuke Naito. Sandwich Construction for Keyed Sponges: Independence Between Capacity and Online Queries. In Sara Foresti and Giuseppe Persiano, editors, *Cryptology and Network Security - 15th International Conference, CANS 2016, Milan, Italy, November 14-16, 2016, Proceedings*, volume 10052 of *Lecture Notes in Computer Science*, pages 245–261, 2016.

[Nai17] Yusuke Naito. Indifferentiability of Double-Block-Length Hash Function Without Feed-Forward Operations. In Josef Pieprzyk and Suriadi Suriadi, editors, *Information Security and Privacy - 22nd Australasian Conference, ACISP 2017, Auckland, New Zealand, July 3-5, 2017, Proceedings, Part II*, volume 10343 of *Lecture Notes in Computer Science*, pages 38–57. Springer, 2017.

[Nai19] Yusuke Naito. Optimally Indifferentiable Double-Block-Length Hashing Without Post-processing and with Support for Longer Key Than Single Block. In Peter Schwabe and Nicolas Thériault, editors, *Progress in Cryptology - LATINCRYPT 2019 - 6th International Conference on Cryptology and Information Security in Latin America, Santiago de Chile, Chile, October 2-4, 2019, Proceedings*, volume 11774 of *Lecture Notes in Computer Science*, pages 65–85. Springer, 2019.

[Nat77] National Bureau of Standards. FIPS PUB 46: Data Encryption Standard, 1977.

[Nat08] National Institute of Standards and Technology. FIPS 198-1: The Keyed-Hash Message Authentication Code (HMAC), July 2008.

[Nat12] National Institute of Standards and Technology. SHA-3 Competition. In *International Conference on the Theory and Application of Cryptographic Techniques*, 2007-2012.

[Nat15a] National Institute of Standards and Technology. FIPS 180-4: Secure Hash Standard (SHS), August 2015.

[Nat15b] National Institute of Standards and Technology. FIPS PUB 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, August 2015.

[Nat19] National Institute of Standards and Technology. Lightweight Cryptography, February 2019. `https://csrc.nist.gov/Projects/Lightweight-Cryptography`.

[NHS$^+$24] Zhongfeng Niu, Kai Hu, Siwei Sun, Zhiyu Zhang, and Meiqin Wang. Speeding Up Preimage and Key-Recovery Attacks with Highly Biased Differential-Linear Approximations. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part IV*, volume 14923 of *Lecture Notes in Computer Science*, pages 73–104. Springer, 2024.

[NIC25] NICCS. Announcement on Launching the Next-generation Commercial Cryptographic Algorithms Program, 2025. `https://www.niccs.org.cn/tzgg/`.

[NO14] Yusuke Naito and Kazuo Ohta. Improved Indifferentiable Security Analysis of PHOTON. In Michel Abdalla and Roberto De Prisco, editors, *Security and Cryptography for Networks - 9th International Conference, SCN 2014, Amalfi, Italy, September 3-5, 2014. Proceedings*, volume 8642 of *Lecture Notes in Computer Science*, pages 340–357. Springer, 2014.

[NSS21] Yusuke Naito, Yu Sasaki, and Takeshi Sugawara. Double-Block-Length Hash Function for Minimum Memory Size. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part III*, volume 13092 of *Lecture Notes in Computer Science*, pages 376–406. Springer, 2021.

[NSS23] Yusuke Naito, Yu Sasaki, and Takeshi Sugawara. Committing Security of Ascon: Cryptanalysis on Primitive and Proof on Mode. *IACR Trans. Symmetric Cryptol.*, 2023(4):420–451, 2023.

[NY16] Yusuke Naito and Kan Yasuda. New Bounds for Keyed Sponges with Extendable Output: Independence Between Capacity and Message Length. In Thomas Peyrin, editor, *Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers*, volume 9783 of *Lecture Notes in Computer Science*, pages 3–22. Springer, 2016.

[ÖS09] Onur Özen and Martijn Stam. Another Glance at Double-Length Hashing. In Matthew Geoffrey Parker, editor, *Cryptography and Coding, 12th IMA International Conference, Cryptography and Coding 2009, Cirencester, UK, December 15-17, 2009. Proceedings*, volume 5921 of *Lecture Notes in Computer Science*, pages 176–201. Springer, 2009.

[Pat91] Jacques Patarin. *Étude des Générateurs de Permutations Basés sur le Schéma du D.E.S.* PhD thesis, Université Paris 6, Paris, France, November 1991.

[Pat08a] Jacques Patarin. A Proof of Security in $O(2^n)$ for the Xor of Two Random Permutations. In Reihaneh Safavi-Naini, editor, *Information Theoretic Security, Third International Conference, ICITS 2008, Calgary, Canada, August 10-13, 2008, Proceedings*, volume 5155 of *Lecture Notes in Computer Science*, pages 232–248. Springer, 2008.

[Pat08b] 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.

[PDM$^+$18] Damian Poddebniak, Christian Dresen, Jens Müller, Fabian Ising, Sebastian Schinzel, Simon Friedberger, Juraj Somorovsky, and Jörg Schwenk. Efail: Breaking S/MIME and OpenPGP Email Encryption using Exfiltration Channels. In William Enck and Adrienne Porter Felt, editors, *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*, pages 549–566. USENIX Association, 2018.

[PGV93] Bart Preneel, René Govaerts, and Joos Vandewalle. Hash Functions Based on Block Ciphers: A Synthetic Approach. In Douglas R. Stinson, editor, *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings*, volume 773 of *Lecture Notes in Computer Science*, pages 368–378. Springer, 1993.

[Pie09] Krzysztof Pietrzak. A Leakage-Resilient Mode of Operation. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, volume 5479 of *Lecture Notes in Computer Science*, pages 462–482. Springer, 2009.

[PM16] Trevor Perrin and Moxie Marlinspike. The double ratchet algorithm. *GitHub wiki*, 2016. `https://github.com/trevp/double_ratchet/wiki`.

[PSV15] Olivier Pereira, François-Xavier Standaert, and Srinivas Vivek. Leakage-Resilient Authentication and Encryption from Symmetric Cryptographic Primitives. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pages 96–108. ACM, 2015.

[RBB03] 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, 2003.

[RBBK01] 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, *CCS 2001, Proceedings of the 8th ACM Conference on Computer and Communications Security, Philadelphia, Pennsylvania, USA, November 6-8, 2001*, pages 196–205. ACM, 2001.

[Rog02] 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.

[RS98] Martin Raab and Angelika Steger. “Balls into Bins” - A Simple and Tight Analysis. In Michael Luby, José D. P. Rolim, and Maria J. Serna, editors, *Randomization and Approximation Techniques in Computer Science, Second International Workshop, RANDOM'98, Barcelona, Spain, October 8-10, 1998, Proceedings*, volume 1518 of *Lecture Notes in Computer Science*, pages 159–170. Springer, 1998.

[RS04] Phillip Rogaway and Thomas Shrimpton. Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance. In Bimal K. Roy and Willi Meier, editors, *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers*, volume 3017 of *Lecture Notes in Computer Science*, pages 371–388. Springer, 2004.

[RS06] 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.

[RS08] Phillip Rogaway and John P. Steinberger. Constructing Cryptographic Hash Functions from Fixed-Key Blockciphers. In David A. Wagner, editor, *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, volume 5157 of *Lecture Notes in Computer Science*, pages 433–450. Springer, 2008.

[RSS11] Thomas Ristenpart, Hovav Shacham, and Thomas Shrimpton. Careful with Composition: Limitations of the Indifferentiability Framework. In Kenneth G. Paterson, editor, *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, volume 6632 of *Lecture Notes in Computer Science*, pages 487–506. Springer, 2011.

[Sch16] Berry Schoenmakers. Explicit Optimal Binary Pebbling for One-Way Hash Chain Reversal. In Jens Grossklags and Bart Preneel, editors, *Financial Cryptography and Data Security - 20th International Conference, FC 2016, Christ Church, Barbados, February 22-26, 2016, Revised Selected Papers*, volume 9603 of *Lecture Notes in Computer Science*, pages 299–320. Springer, 2016.

[SCM08] Joseph Salowey, Abhijit Choudhury, and David A. McGrew. AES Galois Counter Mode (GCM) Cipher Suites for TLS. Request for Comments (RFC) 5288, 2008.

[Set22] Srinath Setty. Nova: open source implementation, 2022. `https://github.com/microsoft/Nova`.

[Seu09] Yannick Seurin. Primitives et protocoles cryptographiquesa sécurité prouvée. PhD thesis, 2009.

[Sha48] Claude E. Shannon. A mathematical theory of communication. *Bell Syst. Tech. J.*, 27(4):623–656, 1948.

[Sha49] Claude E. Shannon. Communication theory of secrecy systems. *Bell Syst. Tech. J.*, 28(4):656–715, 1949.

[Shr04] Tom Shrimpton. A Characterization of Authenticated-Encryption as a Form of Chosen-Ciphertext Security. Cryptology ePrint Archive, Report 2004/272, 2004. `http://eprint.iacr.org/2004/272`.

[SKP22] Sayandeep Saha, Mustafa Khairallah, and Thomas Peyrin. Exploring Integrity of AEADs with Faults: Definitions and Constructions. *IACR Trans. Symmetric Cryptol.*, 2022(4):291–324, 2022.

[SLS$^+$23] Alan Szepieniec, Alexander Lemmens, Jan Ferdinand Sauer, Bobbin Threadbare, and Al-Kindi. The Tip5 Hash Function for Recursive STARKs. Cryptology ePrint Archive, Paper 2023/107, 2023. `https://eprint.iacr.org/2023/107`.

[SLZ$^+$25] Siwei Sun, Shun Li, Zhiyu Zhang, Charlotte Lefevre, Bart Mennink, Zhen Qin, and Dengguo Feng. Permutation-Based Hashing with Stronger (Second) Preimage Resistance - Application to Hash-Based Signature Schemes. Cryptology ePrint Archive, Paper 2025/963, 2025. `https://eprint.iacr.org/2025/963`.

[SMC$^+$25] Meltem Sönmez Turan, Kerry A. McKay, Donghoon Chang, Jinkeon Kang, and John Kelsey. Ascon-Based Lightweight Cryptography Standards for Constrained Devices. NIST SP 800-232, August 2025. `https://doi.org/10.6028/NIST.SP.800-232`.

[SPY$^+$10] François-Xavier Standaert, Olivier Pereira, Yu Yu, Jean-Jacques Quisquater, Moti Yung, and Elisabeth Oswald. Leakage Resilient Cryptography in Practice. In Ahmad-Reza Sadeghi and David Naccache, editors, *Towards Hardware-Intrinsic Security - Foundations and Practice*, Information Security and Cryptography, pages 99–134. Springer, 2010.

[SPY13] François-Xavier Standaert, Olivier Pereira, and Yu Yu. Leakage-Resilient Symmetric Cryptography under Empirically Verifiable Assumptions. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 335–352. Springer, 2013.

[SS08] Thomas Shrimpton and Martijn Stam. Building a Collision-Resistant Compression Function from Non-compressing Primitives. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, volume 5126 of *Lecture Notes in Computer Science*, pages 643–654. Springer, 2008.

[SSY12] John P. Steinberger, Xiaoming Sun, and Zhe Yang. Stam's Conjecture and Threshold Phenomena in Collision Resistance. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, volume 7417 of *Lecture Notes in Computer Science*, pages 384–405. Springer, 2012.

[Sta08] Martijn Stam. Beyond Uniformity: Better Security/Efficiency Tradeoffs for Compression Functions. In David A. Wagner, editor, *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, volume 5157 of *Lecture Notes in Computer Science*, pages 397–412. Springer, 2008.

[Sta09] Martijn Stam. Blockcipher-Based Hashing Revisited. In Orr Dunkelman, editor, *Fast Software Encryption, 16th International Workshop, FSE 2009, Leuven, Belgium, February 22-25, 2009, Revised Selected Papers*, volume 5665 of *Lecture Notes in Computer Science*, pages 67–83. Springer, 2009.

[Ste10] John P. Steinberger. Stam's Collision Resistance Conjecture. In Henri Gilbert, editor, *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*, pages 597–615. Springer, 2010.

[SY15] Yu Sasaki and Kan Yasuda. How to Incorporate Associated Data in Sponge-Based Authenticated Encryption. In Kaisa Nyberg, editor, *Topics in Cryptology - CT-RSA 2015, The Cryptographer's*

Track at the RSA Conference 2015, San Francisco, CA, USA, April 20-24, 2015. Proceedings, volume 9048 of *Lecture Notes in Computer Science*, pages 353–370. Springer, 2015.

[Tsu92] Gene Tsudik. Message authentication with one-way hash functions. *Comput. Commun. Rev.*, 22(5):29–38, 1992.

[TT18] Stefano Tessaro and Aishwarya Thiruvengadam. Provable Time-Memory Trade-Offs: Symmetric Cryptography Against Memory-Bounded Adversaries. In Amos Beimel and Stefan Dziembowski, editors, *Theory of Cryptography - 16th International Conference, TCC 2018, Panaji, India, November 11-14, 2018, Proceedings, Part I*, volume 11239 of *Lecture Notes in Computer Science*, pages 3–32. Springer, 2018.

[WFW05] Peng Wang, Dengguo Feng, and Wenling Wu. HCTR: A Variable-Input-Length Enciphering Mode. In Dengguo Feng, Dongdai Lin, and Moti Yung, editors, *Information Security and Cryptology, First SKLOIS Conference, CISC 2005, Beijing, China, December 15-17, 2005, Proceedings*, volume 3822 of *Lecture Notes in Computer Science*, pages 175–188. Springer, 2005.

[WHF03] Doug Whiting, Russell Housley, and Niels Ferguson. Counter with CBC-MAC (CCM). 2003.

[Wu11] Hongjun Wu. The Hash Function JH, 2011. Submission to NIST's SHA-3 competition.

[Wu14] Hongjun Wu. ACORN v1. Submission to CAESAR competition, 2014.

[YMO09] Kazuki Yoneyama, Satoshi Miyagawa, and Kazuo Ohta. Leaky Random Oracle. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, 92-A(8):1795–1807, 2009.

[YSPY10] Yu Yu, François-Xavier Standaert, Olivier Pereira, and Moti Yung. Practical leakage-resilient pseudorandom generators. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*, pages 141–151. ACM, 2010.

[ZDY$^+$19] Wentao Zhang, Tianyou Ding, Bohan Yang, Zhenzhen Bao, Zejun Xiang, Fulei Ji, and Xuefeng Zhao. KNOT: Algorithm Specifications and Supporting Document. Second Round Submission to NIST Lightweight Cryptography, 2019.

Research Data Management

This thesis research has been carried out under the research data management policy of the Institute for Computing and Information Science of Radboud University, The Netherlands.

No additional data was created or generated for the research presented in this thesis. All results and security proofs are self-contained and suffice to support the presented results.

Summary

Permutation-based cryptography, in particular the sponge construction, has become central to the design of lightweight symmetric schemes.

In this thesis, we advance the state of provable security for permutation-based designs in several ways.

First, we establish tight bounds on the preimage resistance of the sponge construction using a new proof technique. This result significantly improves generic preimage security for many lightweight schemes. In addition, we prove indifferentiability of the sponge in two scenarios that have received limited attention: (i) when input messages are bounded in size, and (ii) when inputs are not required to remain secret.

Second, we introduce new sponge-based constructions. We propose the double sponge, the first permutation-based hash function to achieve indifferentiability security beyond the birthday bound in the capacity.

We also present two efficient sponge-based pseudorandom function constructions which outperform existing state-of-the-art designs in terms of efficiency.

Next, we explore permutation-based hashing in specific contexts, namely password-based hash chains and finite-field-oriented designs. Our results demonstrate that dedicated analysis and small design tweaks can yield stronger security guarantees and more efficient schemes.

Finally, we unify and sharpen the generic security analysis of the authenticated encryption mode of Ascon, the winner of the Lightweight Cryptography competition organized by the US National Institute of Standards and Technology. We systematize existing knowledge, introduce a novel security model, and fill important gaps, which together complete the picture of its generic security.

Samenvatting

Permutatiegebaseerde cryptografie, met name de sponsconstructie, is centraal komen te staan in het ontwerp van lichtgewicht symmetrische schema's.

In dit proefschrift analyseren en verbeteren we bewijsbare veiligheid van permutatiegebaseerde ontwerpen op verschillende manieren.

Ten eerste stellen we nauwkeurige grenzen aan de eenwegsweerbaarheid van de sponsconstructie met behulp van een nieuwe bewijstechniek. Dit resultaat verbetert de generieke eenwegsweerbaarheid voor veel lichtgewicht schema's aanzienlijk. Daarnaast bewijzen we de ondifferentieerbaarheid van de spons in twee scenario's die tot nu toe weinig aandacht hebben gekregen: (i) wanneer invoerberichten een beperkte grootte hebben, en (ii) wanneer invoer niet geheim hoeft te blijven.

Ten tweede introduceren we nieuwe sponsgebaseerde constructies. We stellen de dubbelspons voor, de eerste permutatiegebaseerde hashfunctie die ondifferentieerbaar is voorbij de verjaardagsgrens op de capaciteit.

We presenteren ook twee sponsgebaseerde constructies voor pseudowillekeurige functies die efficiënter zijn dan bestaande toonaangevende ontwerpen.

Vervolgens onderzoeken we permutatiegebaseerd hashen in specifieke contexten, namelijk wachtwoordgebaseerde hashketens en eindig-lichaam-georiënteerde ontwerpen. Onze resultaten tonen aan dat gerichte analyses en kleine ontwerpaanpassingen kunnen leiden tot sterkere beveiligingsgaranties en efficiëntere schema's.

Tot slot verenigen en verfijnen we de generieke beveiligingsanalyse van de geauthenticeerde encryptiemodus van Ascon, de winnaar van de Lightweight Cryptography-wedstrijd georganiseerd door het Amerikaanse National Institute of Standards and Technology. We systematiseren bestaande kennis, introduceren een nieuw beveiligingsmodel, en vullen belangrijke open problemen aan, die samen het beeld van de generieke beveiliging ervan compleet maken.

List of Publications

Conference Proceedings

1. *Charlotte Lefevre and Bart Mennink.* **SoK: Security of the Ascon Modes.** IACR Transactions on Symmetric Cryptology 2025(1), pp. 138-210.

2. *Charlotte Lefevre and Mario Marhuenda Beltrán and Bart Mennink.* **To Pad or Not to Pad? Padding-Free Arithmetization-Oriented Sponges.** IACR Transactions on Symmetric Cryptology 2025(1), pp. 97-137.

3. *Charlotte Lefevre and Bart Mennink.* **Permutation-Based Hash Chains with Application to Password Hashing.** IACR Transactions on Symmetric Cryptology 2024(4), pp. 249-286. Best paper award.

4. *Charlotte Lefevre and Bart Mennink.* **Generic Security of the Ascon Mode: On the Power of Key Blinding.** Selected Areas in Cryptography 2024 (II). LNCS, vol. 15517, pp. 3-32 Springer (2024).

5. *Charlotte Lefevre and Bart Mennink.* **Permutation-Based Hashing Beyond the Birthday Bound.** IACR Transactions on Symmetric Cryptology 2024(1), pp. 71-113.

6. *Charlotte Lefevre.* **Indifferentiability of the Sponge Construction with a Restricted Number of Message Blocks.** IACR Transactions on Symmetric Cryptology 2023(1), pp. 224-243.

7. *Charlotte Lefevre and Bart Mennink.* **Tight Preimage Resistance of the Sponge Construction.** CRYPTO 2022 (IV). LNCS, vol. 13510, pp. 185-204 Springer (2022).

8. *Pierre Karpman and Charlotte Lefevre.* **Time-Memory Tradeoffs for Large-Weight Syndrome Decoding in Ternary Codes.** PKC 2022, pp. 82-111.

Preprints and Notes

1. *Siwei Sun and Shun Li and Zhiyu Zhang and Charlotte Lefevre and Bart Mennink and Zhen Qin and Dengguo Feng.* **Permutation-Based Hashing with Stronger (Second) Preimage Resistance - Application to Hash-Based Signature Schemes.** Cryptology ePrint Archive, Paper 2025/963. url: `https://eprint.iacr.org/2025/963`.

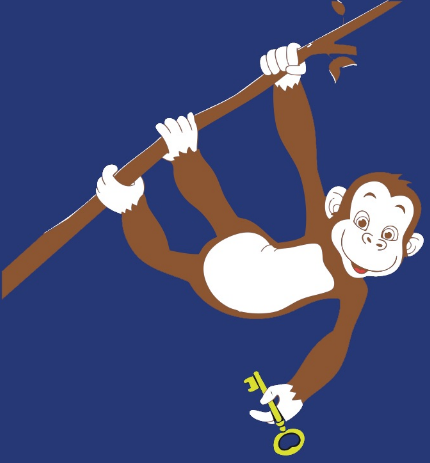
2. *Charlotte Lefevre and Mario Marhuenda Beltrán.* **MacaKey: Full-State Keyed Sponge Meets the Summation-Truncation Hybrid.** Cryptology ePrint Archive, Paper 2025/893. url: `https://eprint.iacr.org/2025/893`.

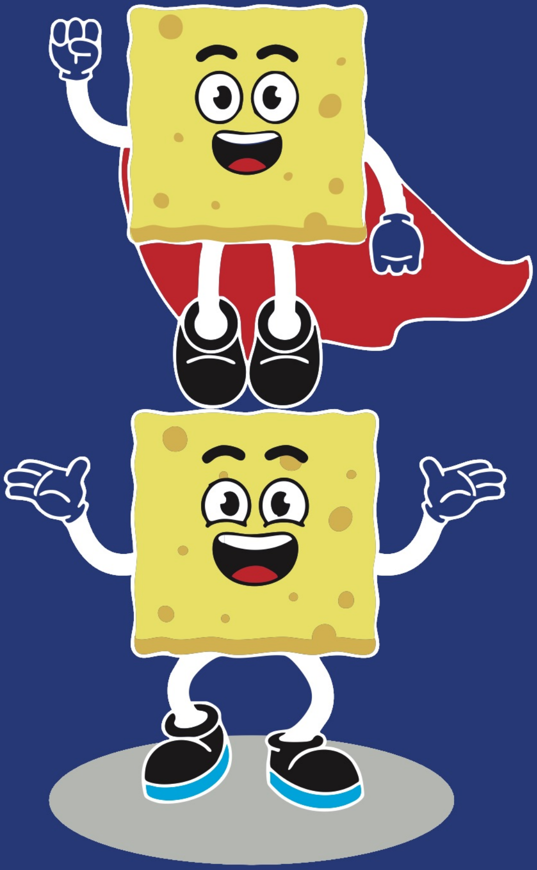
3. *Charlotte Lefevre.* **A Note on Adversarial Online Complexity in Security Proofs of Duplex-Based Authenticated Encryption Modes.** Cryptology ePrint Archive, Paper 2024/213. url: `https://eprint.iacr.org/2024/213`.

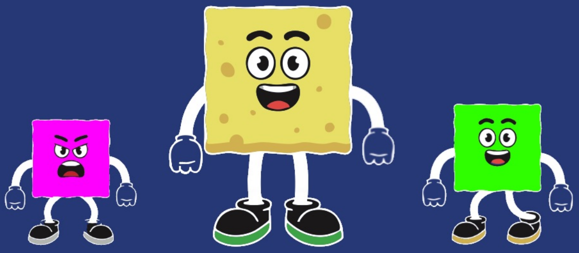
4. *Charlotte Lefevre and Yanis Belkheyar and Joan Daemen.* **Kirby: A Robust Permutation-Based PRF Construction.** Cryptology ePrint Archive, Paper 2023/1520. url: `https://eprint.iacr.org/2023/1520`.

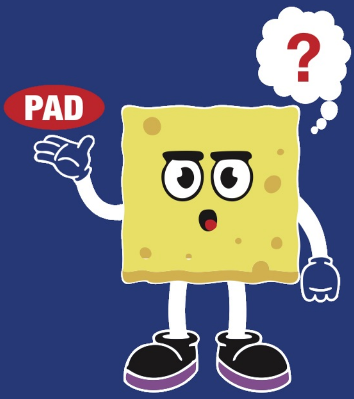
About the Author

Charlotte Lefevre was born in 1997 in Clermont-Ferrand (France). She completed a *DUT mesures physiques* in Clermont-Ferrand in 2017. She completed a bachelor in mathematics in 2019, and a master of science in applied mathematics, last year specialization in cybersecurity in 2021. Both were done at the Université Grenoble Alpes (France) and obtained with the highest distinction (*mention très bien*). Her master thesis, with subject *time-memory tradeoffs for large-weight syndrome decoding in ternary codes*, was supervised by Pierre Karpman at the Laboratoire Jean Kuntzmann, Grenoble (France). In 2021, she started a Ph.D. at Radboud University under the supervision of Bart Mennink and Joan Daemen. This thesis is the result of that research. During her Ph.D., she completed an internship at CSEM (Switzerland) on *post compromise security for IOT settings*, under the supervision of Damian Vizár.








PAD
?