KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Provable Security of Cryptographic Hash Functions

Bart Mennink

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor in Engineering

May 2013

Provable Security of Cryptographic Hash Functions

Bart Mennink

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor in Engineering

May 2013

"Alle grote ontdekkingen komen voort uit onzinnige intuïties."

Harry Mulisch, De Procedure

D/2013/7515/46
ISBN 978-94-6018-660-8

Preface

After months of thesis writing, it is time to start with the hardest but most important chapter of this thesis. I would like to take the opportunity to express my gratitude to all people that have played an important role in the conclusion of this thesis.

At the very first place, I am utterly grateful to my promotors, prof. dr. ir. Bart Preneel and prof. dr. ir. Vincent Rijmen. It has been a great honor to pursuit a PhD under the guidance of two of the world's most notable cryptographers. Back in 2009, when you offered me the opportunity to become a PhD student on symmetric cryptology, I was about to finish my Master's thesis in the area of asymmetric cryptographic protocols, and had barely any knowledge of symmetric cryptology. Particularly, I had no specific research topic in mind, it would just be "symmetric cryptology." However, with the help and encouragement of both of you, I quickly became familiar with symmetric cryptology, and managed to plot my own path in the area of provable security. I am thankful for the full freedom you gave me to choose my own research topics. I believe it is exactly this freedom that gives rise to nonsensical intuitions that are needed for great discoveries.

Besides Bart and Vincent, I would like to express my gratitude to the other members of my jury, prof. dr. Marc Fischlin, prof. dr. ir. Frank Piessens, dr. ir. Martijn Stam, and prof. dr. ir. Joos Vandewalle, for reading my manuscript, attending my private and public defence, and providing valuable feedback. I would also like to thank prof. dr. ir. Pierre Verbaeten for chairing the jury.

A word of appreciation goes to the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT) for their financial support that made this thesis possible.

During my stay at COSIC and at the conferences I visited, I have had the fortune and pleasure to meet some of the brightest cryptographers and had plenty of interesting and inspiring discussions with them. I am especially indebted to my co-authors, Elena Andreeva, Andrey Bogdanov, Yevgeniy Dodis, Jorge Guajardo, Atul Luykx, Florian Mendel, Bart Preneel, Christian Rechberger, Vincent Rijmen, Berry Schoenmakers, Marjan Škrobot, John P. Steinberger, and Elmar Tischhauser. Partially overlapping this list, I would like to thank Elena, Andrey, Atul, Mohammad Reza Reyhanitabar, and Kan Yasuda for the enjoyable and delighting discussions on symmetric provable security. I want to thank Elena in particular also for always being available to discuss research ideas, to read proof write-ups, and to provide feedback on my papers. Also thanks in particular to Atul and Marjan, whose Master's theses I co-supervised. I learned a lot from both of you, and wish you good luck with your own fresh, yet challenging, PhD periods. A sportive thanks to Joan Daemen and Simon Knellwolf for the enlightening and unforgettable discussions while discovering Washington DC or the UCSB coastline on running shoes. Furthermore, I want to thank all remaining COSIC members for the occasional chats, and all other friends and colleagues abroad.

A warm thanks to the administrative fundament of COSIC, Péla Noë, Elsy Vermoesen, and Wim Devroye. Without your help, I would have been stuck in the maze of administration and paperwork for most of the time. Thanks to Sebastiaan Indesteege for providing me with a template of his thesis, it saved me plenty of LaTeX work. Thanks to Elena and Atul for proofreading parts of the thesis, and to Sander for verifying my Limburgish abstract.

I would like to thank my high school classmates, Benno, Dennis, Joost, Rob, Rob, and Ron. Even though high school is already almost a decade ago, the friendship has lasted through all those years. Also thanks to Roger, not only a good friend but also a great physicist. I am quite sure you will end up with a nice PhD thesis.

A very sincere word of thanks goes to my family, my parents and my brother Paul, for the continuous support, the chances you have given me, and for always being there when needed. Thanks to my family (in-law) for all enjoyable moments in my life. Finally, I would like to express my utmost gratefulness to my wife Audrey. You have supported me throughout my entire PhD, and I want to thank you for all your love, support, understanding, humor, and friendship. It is hardly possible to express your importance for this thesis in words.

Bart Mennink
Leuven, April 2013

Abstract

Cryptographic hash functions form the basis of the security of today's digital environment, and find applications in numerous cryptographic systems such as tamper detection, key derivation functions, and digital signatures. Ideally, hash functions behave like a random oracle, a function that returns random outputs for each new input, but in practice such a construction does not exist. Usually, a hash function is designed to give strong confidence that it is indeed secure, and it is presumed secure until it is broken. In 2004-2005, cryptanalytic breakthroughs have raised doubts about the security of many widely employed hash functions, such as MD5 and SHA-1. As a response, in 2007 the US National Institute for Standards and Technology (NIST) announced a call for the design of a new SHA-3 hashing algorithm.

This dissertation deals with the fundamental security properties of hash functions. It is divided into two parts.

In the first part of the dissertation, we analyze existing hash functions and introduce design methodologies. We particularly search for the limits within the provable security framework, by considering minimalist designs with maximal security. Firstly, we look at double block length $3n$-to-$2n$-bit compression functions based on block ciphers with an n-bit message and key space. We consider the MDC-4 hash function, and improve its collision and preimage security bounds. Next, we present a family of designs that make three cipher calls and achieve optimal collision security and very good preimage security. Furthermore, we consider the possibility of compression functions based on permutations, and provide a full security classification of all $2n$-to-n-bit compression functions solely built of XOR operators and three permutations.

As a final contribution of this part, we propose the family of parazoa functions as a generalization of the sponge hash function design, and prove that parazoa functions are indifferentiable from a random oracle. The sponge is a popular hash function design and many derivatives, called sponge-like functions, appeared in literature. However, these sponge-like functions do not automatically enjoy the same security guarantees as the original sponge. Our generalized parazoa family applies to a wide class of sponge-like functions, and the indifferentiability proof for parazoa naturally carries over.

In the second part of the dissertation, we consider NIST's SHA-3 hash function competition from a provable security point of view. We provide a detailed survey of the five SHA-3 finalists, in which we analyze and compare their security guarantees. We consider collision, preimage, and second preimage resistance and indifferentiability of all finalists, and solve open problems where needed.

Samenvatting

Cryptografische hashfuncties liggen ten grondslag aan de beveiliging van de hedendaagse digitale wereld, en worden gebruikt in talrijke cryptografische toepassingen zoals het detecteren van ongeoorloofde datawijzigingen, het afleiden van cryptografische sleutels en digitale handtekeningen. In het ideale geval gedraagt een hashfunctie zich als een volledig willekeurige functionaliteit, een functie die voor iedere invoer een willekeurige uitvoer produceert, maar zulk soort functies bestaan in de praktijk niet. Derhalve worden hashfuncties normaliter op een zodanige wijze ontworpen dat ze veilig genoeg lijken, en ze worden veilig geacht totdat iemand een zwakheid in de functie ontdekt. In 2004-2005 hebben cryptanalytische doorbraken echter de veiligheid van enkele wijdverspreide hashfuncties, zoals MD5 en SHA-1, ter discussie gesteld. Om deze reden lanceerde het National Institute for Standards and Technology (NIST) van de VS in 2007 een internationale competitie voor het ontwerp van een nieuwe SHA-3 hashfunctie.

Deze dissertatie behandelt fundamentele veiligheidsaspecten van cryptografische hashfuncties. Ze bestaat uit twee delen.

In het eerste deel van de dissertatie analyseren we bestaande hashfuncties en introduceren we nieuwe ontwerpmethoden. In het bijzonder gaan we op zoek naar de limieten van het bewijsbareveiligheidskader, waarbij we minimalistische ontwerpen met maximale veiligheid bekijken. Als eerste onderzoeken we compressiefuncties van dubbele bloklengte: $3n$-naar-$2n$-bit compressiefuncties gebaseerd op blokcijfers met een n-bit bericht- en sleutelgrootte. We beschouwen de MDC-4 hashfunctie en verbeteren haar botsings- en éénwegsaanvalsbestendigheidsgrens. Vervolgens introduceren we een nieuwe familie van ontwerpen die drie cijferoproepen doen en die optimale botsingsbestendigheid en zeer goede éénwegsaanvalsbestendigheid bereiken. Voorts analyseren we de mogelijkheid om compressiefuncties te construeren van permutaties, en verschaffen een volledige veiligheidsclassificatie van alle $2n$-naar-n-bit compressiefuncties louter gebouwd op XOR-operators en drie permutaties.

Als laatste bijdrage van dit deel van de dissertatie presenteren we de familie van parazoafuncties als een generalisatie van sponsfuncties, en bewijzen we dat parazoafuncties indifferentieerbaar zijn van een volledig willekeurige functionaliteit. De spons is een populair hashfunctieontwerp en tal van afgeleiden, zogenaamde sponsachtige functies, zijn in de loop der tijd gepubliceerd. Deze sponsachtige functies genieten echter niet automatisch dezelfde veiligheidsgaranties als de oorspronkelijke spons. Onze gegeneraliseerde parazoafamilie behelst een breed spectrum van sponsachtige functies, en het indifferentieerbaarheidsbewijs voor parazoa is overdraagbaar naar deze functies.

In het tweede deel van de dissertatie beschouwen we NIST's SHA-3 hashfunctiecompetitie met betrekking tot haar bewijsbare veiligheid. We presenteren een gedetailleerde beschouwing van de vijf SHA-3-finalisten, waarin we hun veiligheidseigenschappen analyseren en vergelijken. We beschouwen botsings-, éénwegsaanvals- en tweede-éénwegsaanvalsbestendigheid, alsmede indifferentieerbaarheid van alle finalisten, en lossen waar nodig open problemen op.

Samevatting

Cryptografische hashfunkties ligke te gróndjsjlaag aan de beveiliging van de hedendaagse digitale waereld, en waere gebroêk in talrieke cryptografische toepassinge zoès het opsjpaore van onrechmaotige gegaevesverangeringe, het aafleije van cryptografische sjleutels en digitale handjteikeninge. In het ideale geval gedreug ein hashfunktie zich wie ein volledig willekäörige funktionaliteit, ein funktie die veur jedere inveur ein willekäörige oetveur produceert, mèr zo ein funkties besjtaon in de praktijk neet. Daorom waere hashfunkties normaal gesjpraoke op ein zodanige maneer ontworpe dat ze veilig genóg lieke, en ze waere veilig besjouwd totdat emes ein zjwaakheid in de funktie ontdèk. In 2004-2005 höbbe cryptanalytische doorbrake echter de veiligheid van versjillende wiedversjpreide hashfunkties, zoès MD5 en SHA-1, ter discussie gesjtèld. Om dees raeje lanceerde het National Institute for Standards and Technology (NIST) van de VS in 2007 ein internationale competitie veur het ontwerp van ein nuuje SHA-3 hashfunktie.

Dees dissertatie behanjelt fundamentele veiligheidsaspecte van cryptografische hashfunkties. Ze besjteit oet tweë deile.

In het eësjte deil van de dissertatie analysere ver besjtaonde hashfunkties en introducere ver nuuje ontwerpmethodes. In het biezunjer gaon ver op zeuk nao de limiete van het bewiesbareveiligheidskader, wobie ver minimalistische ontwerpe mit maximale veiligheid bekieke. Es eësjte ongerzeuke ver kompressiefunkties van dubbele bloklengte: $3n$-nao-$2n$-bit kompressiefunkties gebaseerd op bloksiefers mit ein n-bit berich- en sjleutelgroatte. Ver besjouwe de MDC-4 hashfunktie en verbaetere häör botsings- en einwaegsaanvalsbesjtendigheidsgrens. Vervolges introducere ver ein nuuje femielie van ontwerpe die drie sieferopreupe doon en die optimale botsingsbesjtendigheid en zeer gooje einwaegsaanvalsbesjtendigheid bereike. Wiejer analysere ver de mäögelikheid om kompressiefunkties te construere van permutaties, en versjaffe ein volledige veiligheidsklassifikatie van alle $2n$-nao-n-bit kompressiefunkties louter geboewd op XOR-operators en drie permutaties.

Es lètste biedraag van dit deil van de dissertatie prizzentere ver de femielie van parazoafunkties es ein generalisatie van sjponsfunkties, en bewieze ver dat parazoafunkties indifferentieerbaar zeen van ein volledig willekäörige funktionaliteit. De sjpons is ein populair hashfunktieontwerp en tal van aafgeleije, zogenaamde sjponsachtige funkties, zeen in de laup der tied gepubliceerd. Dees sjponsachtige funkties genete echter neet automatisch dezelfde veiligheidsgeranties es de oorsjpronkelikke sjpons. Ózze gegeneraliseerde parazoafemielie behels ein breid sjpectrum van sjponsachtige funkties, en het indifferentieerbaarheidsbewies veur parazoa is euverdraagbaar nao dees funkties.

In het tweëde deil van de dissertatie besjouwe ver NIST's SHA-3 hashfunktiecompetitie mit betrèkking tot häör bewiesbare veiligheid. Ver presentere ein gedetailleerde besjouwing van de vief SHA-3-finaliste, wo-in ver hun veiligheidseigensjappe analysere en vergelieke. Ver besjouwe botsings-, einwaegsaanvals- en tweëde-einwaegsaanvalsbesjtendigheid, es auch indifferentieerbaarheid van alle finaliste, en losse wo neudig aope probleme op.

Table of Contents

List of Figures

List of Tables

1 Introduction

"Mooi moment, altijd, net voor de start, alles is gedaan, alles is voorbij, alles moet nog beginnen."

Dolf Jansen, Altijd Verder

In a world increasingly dominated by technology and social media, properties such as secrecy, authenticity, and integrity of data are of paramount importance. Communication of potentially sensitive data, like financial transactions, phone conversations, and e-mails, comes with ultimate confidentiality requirements, especially if transmitted over an insecure channel. On a larger scale, digital infrastructures of companies and governmental institutions should be properly secured to preclude hackers from breaking into the system and obtaining valuable and strategic information. It is here where the science of cryptology comes into use. Traditionally employed by secret services and military — we refer to Kahn [121] for an extensive discussion of the early history of cryptology — cryptology forms a solution to the major security difficulties we face today. In part due to pioneering publications of Kerckhoffs [124] and Shannon [204, 205], but also due to the evolution of computers, cryptology has been lifted to a new level in the past decades. For most security threats to date, mathematical cryptographic systems are known that guarantee the required security and limit the security risks.

1.1 Hash Functions as Fundamental Primitives

At the heart of many cryptographic systems are cryptographic hash functions. A cryptographic hash function is an algorithm that maps arbitrarily long bit strings to digests of a fixed length. The concept dates back to the work of Diffie and Hellman [82], with the main goal to make digital signatures more efficient and compact: instead of signing an integral message, one would sign a hash value of that message instead.

Besides digital signature schemes, cryptographic hash functions nowadays find plenty of applications, such as commitment schemes, key derivation, message authentication, passphrase protection, and tamper detection. This wide use has lead to many stand-alone security definitions for hash functions, undoubtedly the foremost important among them being collision, preimage, and second preimage resistance (as first suggested by Merkle [168]). The first one means that it is "hard" to find two distinct messages with the same digest, the second one that it is "hard" to find a message for a given digest, and the third one that it is "hard" to find, given a certain message, a distinct message with the same digest.

Yet, hash functions are employed for different purposes, and many security proofs of cryptographic protocols or systems (e.g., [34, 73, 186, 201]) are in the random oracle model, where the hash function is assumed to be a function that returns random outputs for each new input query [33, 94]. This is a strictly stronger security requirement that among others implies the above-mentioned properties. Although this approach turns out to be very useful for obtaining security guarantees, random oracles are impossible to construct in practice [29, 49, 65, 66, 178]. The main question then is, how to construct a hash function that behaves *like* a random oracle (and how to prove that it does so).

We refer to Menezes et al. [161] for a more technical discussion of hash functions and other cryptographic primitives, such as block and stream ciphers.

1.2 Design of Cryptographic Hash Functions

Borrowed from the area of public key cryptography, a promising approach to build a hash function is to base it on a hard mathematical problem. Several known hash functions follow this approach, such as VSH [70], ECOH [62], FSB [23], LASH [36], RFSB [38], and SWIFFTX [21], but a major drawback is their efficiency. Additionally, for some of these designs it is hard to guarantee all required security properties. In general, this research direction is not mature enough, and almost all hash functions in literature — most prominently the MDx and SHA-x family of hash functions — are dedicated designs. The basic idea of many dedicated hash functions is that they are designed with the primary focus on efficiency rather than provable security. They are usually constructed to create a strong confidence that the hash function is indeed secure.

The design of a hash function is usually modular. To accommodate the processing of arbitrarily long messages, most hash functions are designed using a compression function in an iterated way. This compression function is an algorithm that can only compress a fixed-length message block, and is in turn frequently based on an underlying block cipher or permutation (either explicitly or implicitly). We refer to the way a function is built on top of underlying primitives as the "mode of operation." Although models exist to achieve security of such functions or primitives against specific types of attacks (such as Markov ciphers [138], the wide trail strategy [74], and the decorrelation theory [217]), security of the hash functions against all types of attacks cannot be guaranteed. Roughly stated, the design is thus deemed secure until someone breaks it. This is a serious risk. As soon as someone breaks a widely used hash function, it needs to be quickly replaced by a proper alternative.

1.3 NIST's SHA-3 Hash Function Competition

For a long time, MD5 and SHA-1 were the most widely used hash functions. In 2004-2005, a series of attacks by Wang et al. [219–221] have exposed security vulnerabilities in the design of (among others) these two designs.[1] As a result, the US National Institute for Standards and Technology (NIST) recommended the replacement of SHA-1 by the SHA-2 hash function family and announced a call for the design of a new SHA-3 hashing algorithm. This competition was planned to be of a similar form as the AES process [179]. The call prescribed that SHA-3 must allow for message digests of length $224, 256, 384$, and 512 bits, it should be efficient, and it should provide an adequate level of security. NIST received 64 submissions, 51 of which advanced to the first round of the competition. In the second round, 14 candidate hash functions were considered, and five of them advanced to the third and final round: BLAKE [25], Grøstl [104], JH [224], Keccak [40], and Skein [93]. On October 2nd, 2012, NIST announced that Keccak has been selected as the winner of the SHA-3 competition. Keccak is about to be standardized and may (or may not) become the most prominent hash function in the next decennia.

1.4 Why Provable Security?

The methods to analyze the security of cryptographic hash functions can be divided into two classes: the *cryptanalytic approach* and the *provable security approach*.

A straightforward way to analyze the security of a hash function is to try to break it. This is known as the cryptanalytic approach, and it forms a considerable research direction in the area of symmetric cryptology. The basic idea is to consider (simplified versions of) a hash function and try to find collisions, preimages, or second preimages faster than expected for a random oracle. A more general way to break a hash function is to present a distinguisher, what essentially comes down to identifying a non-random property of the hash function. Partly boosted by NIST's SHA-3 hash function competition, many popular distinguishing techniques such as the boomerang attack for hash functions [120], the rebound attack [160], and the rotational rebound attack [125] appeared in the last years.

On the other hand, in the provable security approach researchers try to formally *argue* that a hash function is secure. One usually focuses on modes of operation: the way a hash function is constructed from an underlying compression function, and/or a compression function is constructed from an underlying block cipher or permutation. Two major directions to argue security are known. In the *standard*

[1]Collisions for the compression function of MD5 were already published in 1993 by den Boer and Bosselaers [81] and in 1996 by Dobbertin [84].

model, the idea is to relate the security of a scheme to the security of an underlying primitive (also known as a security reduction). In the *ideal model*, one assumes that the underlying primitive of a scheme behaves randomly, and analyzes the probability of an attacker to break the security of that scheme.[2]

Both provable security approaches have their advantages and disadvantages. The standard model does not rely on ideal model assumptions on the underlying primitive, but the available techniques are limited and are mostly restricted to collision, preimage, and second preimage security reductions from a hash function to a compression function. We refer to Andreeva et al. [11, 19] for an extended analysis of standard model security reductions for the most widely adopted hash function designs. Although the ideal model puts stronger assumptions on the underlying primitives, it offers much more possibilities, such as computing the security of a hash or compression function design if the underlying primitive is a block cipher or permutation. Moreover, it can be used for security analyses where the standard model falls short. For instance, Simon [207] showed that no provably collision resistant hash function exists based on just a one-way permutation. This particularly means that the standard security model for block ciphers, PRP (pseudo-random permutation) security, is insufficient to prove security of a block cipher based hash function.

Comparison. Compared to cryptanalytic attacks, provable security results sometimes spark controversy, mostly founded on arguments that (i) it is sometimes obvious whether a mode of operation is secure, no proof is needed for that, and (ii) ideal primitives do not exist.

Considering the first point, while it could be true for simple designs, more complex constructions exist for which no security proof or attack exists (a notable example concerns the security of the block cipher constructions Feistel [208] and Even-Mansour [91], where the number of rounds that can be attacked generically is strictly smaller than the number of rounds needed to prove security [1, 115]) or for which the security and attack bounds leave a significant gap and the actual security is unknown (see, e.g., Table 3.1 on MDC-2 and MDC-4 and Section 8.1 for other open problems). Moreover, constructions exist that appear secure at first sight, but turn out to be insecure after closer inspection (we elaborate on this in Section 6.2). In general, despite the fact that provable security proofs provide security for a *mode of operation* only, and not for the entire design, they guarantee that the design does not suffer from structural flaws. They confirm that the design has a sound mode of operation, and that weaknesses, if any, can only arise from the underlying primitive.

In order to elaborate on the second point, we briefly discuss the latest provable security and cryptanalytic results on the SHA-3 hash function Keccak. In the provable security setting, it is shown that Keccak behaves like an ideal hash function up to 2^n evaluations, where n is the output size of the hash function (we

[2]Without going into further detail, we note that one can also consider a combination of the two models, as is for instance done for the mix-compress-mix approach [151, 190, 193].

refer to Sections 2.5.2 and 7.4 for more details). This result guarantees, among others, optimal collision, preimage, and second preimage security, yet it holds only if the underlying permutation π is modeled as a random permutation. We note that this permutation π consists of 24 rounds. The latest cryptanalytic results on Keccak are round-reduced collision attacks by Dinur et al. [83], where the underlying permutation π is reduced to three rounds (for $n = 512$), four rounds (for $n = 224, 384$), or five rounds (for $n = 256$). Clearly, these results do not contradict each other, as both are in a different model: the former result is in the so-called *ideal permutation model* where π is assumed to behave like a random permutation, and the latter result is where π is reduced to a smaller number of rounds, so where 19 up to 21 rounds are simply ignored.

In both approaches, one thus considers a *simplified* version of the real Keccak hash function, where the permutation is adapted — "idealized," in some sense — to something that makes it analyzable. From this perspective, both approaches consider Keccak from a different point of view, and it is fair to work in an abstract idealized model as compared to the cryptanalytic approach.

Yet, there is a fundamental difference. The cryptanalysts' ultimate goal is to analyze the full design: step by step, they try to increase the number of rounds until they end up with the full function. In the provable security area, technical issues arise when trying to step away from the ideal model. In fact, examples exist of hash functions and compression functions that are secure in the ideal model, but not when the ideal primitive is replaced by a (contrived) secure function [19, 110], or even a practically used function as AES [48]. Nevertheless, in this area the ideal model appears to be the most suitable way to analyze the security of schemes based on block ciphers or permutations.

1.5 Contributions and Outline

This dissertation studies the fundamental security properties of hash functions. By formal modeling, we analyze existing hash functions and introduce new families of designs. We search for the constructive limits within the provable security framework, where minimalism and simplicity of designs are the keywords and the aim is a maximal level of collision and/or preimage security.

In Chapter 2 we introduce basic cryptographic definitions. We discuss the most prominent hash function modes of operation and compression function designs in more detail. Throughout this dissertation, we focus on collision, preimage, and second preimage security of hash and compression functions, as well as indifferentiability; we also present a formal security model for these notions. Additionally, we introduce the notion of chosen-target-forced-midfix preimage resistance. It is a generalization and formalization of security against the Nostradamus or herding attack by Kelsey and Kohno [122]. We prove that the Merkle-Damgård hash function design (and its derivatives) achieve tight security under this notion (up to the known attack bound). The findings on chosen-target-forced-midfix security have been published in [8].

Hash functions are traditionally block cipher based. In Chapter 3 we formally analyze the collision and preimage security of the classical double block length hash function MDC-4. MDC-4 employs a $3n$-to-$2n$-bit compression function making four calls to a block cipher with n-bit message and key space. At the start of this work, the best known security bounds for MDC-4 were the trivial ones, and we significantly improve upon these bounds. This work has been published in [164].

Despite these improved bounds, MDC-4 fails to achieve optimal collision and preimage security, and an open problem is to derive compression functions of similar form ($3n$-to-$2n$-bit using a block cipher with n-bit message and key space) that *do* achieve optimal security. In Chapter 4 we investigate this fundamental problem. In the quest for a minimal design with maximal security, we derive a family of compression functions that make three block cipher calls and achieve optimal collision security and very good preimage security. The contents of this chapter have been published in [163].

In more recent years, a popular approach to build a compression function is to base it on permutations. However, a basic $2n$-to-n-bit function is known to be required to make at least three n-bit permutation calls in order to achieve optimal collision security [199, 209]. Although some designs in this direction are known, in Chapter 5 we investigate this direction and provide a full security classification of all $2n$-to-n-bit compression functions built solely of XOR operators and three permutations. This chapter is based on publication [165].

The popularity of permutation based hashing is also reflected in recent trends in hash function design, with the most prominent example being the sponge hash function design of Bertoni et al. [41] that played a very strong role in NIST's SHA-3 hash function competition; note that the winner Keccak is a sponge function. Sponge functions internally make one permutation evaluation per message block compression, and even if the underlying compression function does not offer a good level security, this does not influence the security of the full sponge hash function which is proven to be indifferentiable from a random oracle. In Chapter 6 we introduce the parazoa family of hash functions as a generalization of the sponge, and we prove that parazoa functions are indifferentiable from a random oracle. Our generalization is crafted towards obtaining secure "sponge-like" hash functions in the indifferentiability framework. The contents of this chapter have been published in [15].

In Chapter 7, we consider NIST's SHA-3 hash function competition from a provable security point of view. We provide a detailed survey of the five SHA-3 finalists, in which we compare their security guarantees. We consider collision, preimage, and second preimage resistance for the $n = 256$ and $n = 512$ variants, as well as their indifferentiability security results. This chapter is an amalgam of five publications [3, 6, 9, 12, 16].

Chapter 8 concludes this dissertation and lists interesting open problems for future work.

2 Foundations

This chapter presents the basic cryptographic tools and concepts relevant to this dissertation. We start with mathematical preliminaries in Section 2.1. In Section 2.2 we discuss symmetric cryptographic primitives at a high level. Next, we formally introduce in Section 2.3 the security model used in this dissertation. In Section 2.4, we present the iterated hashing principle, the standard method to build a hash function from a compression function, and we consider variants and alternatives in Section 2.5. Finally, in Section 2.6, we discuss two fundamental ways to build a compression function.

2.1 Mathematical Preliminaries

For $n \in \mathbb{N}$, we denote by $\mathbb{Z}_2^n$ the set of bit strings of length n, and by $(\mathbb{Z}_2^n)^\infty$ the set of strings of length a positive multiple of n bits. We denote by $\mathbb{Z}_2^\infty$ the set of bit strings of arbitrary length. For two bit strings x, y, $x \| y$ denotes their concatenation and $x \oplus y$ their bitwise XOR. By $|x|$ we denote the length of a bit string x, and for $m, n \in \mathbb{N}$ we denote by $\langle m \rangle_n$ the encoding of m as an n-bit string. The function $\mathsf{left}_n(x)$ takes the n leftmost and $\mathsf{right}_n(x)$ the n rightmost bits of a string x. If x is of even length, we write x^l and x^r to denote its left and right halves where $|x^l| = |x^r|$. We let $y \leftarrow A(x)$ and $y \overset{\$}{\leftarrow} A(x)$ be the assignment to y of the output of a deterministic and randomized algorithm A, respectively, when run on input x. If X is a set, by $x \overset{\$}{\leftarrow} X$ we denote the uniform random sampling of an element from X. For a function f, by $\mathsf{dom}(f)$ and $\mathsf{rng}(f)$ we denote its domain and range, respectively. We denote by $\mathrm{Func}(m, n)$ the set of all functions $f : \mathbb{Z}_2^m \to \mathbb{Z}_2^n$. The set of functions Func may be restricted, for instance to contain block ciphers or permutations only (see Section 2.2).

Let $f(x), g(x)$ be two functions. By $f(x) = O(g(x))$ we denote that there exists an $M > 0$ such that $f(x) \leq M \cdot g(x)$ asymptotically, by $f(x) = \Omega(g(x))$ that there exists an $m > 0$ such that $f(x) \geq m \cdot g(x)$ asymptotically, and by $f(x) = \Theta(g(x))$ that there exist $m, M > 0$ such that $m \cdot g(x) \leq f(x) \leq M \cdot g(x)$ asymptotically.

We regularly use Stirling's approximation, which states that $t! \geq (t/e)^t$ for any t.

2.2 Symmetric Cryptographic Primitives

A *cryptographic hash function* $\mathcal{H} : \mathbb{Z}_2^\infty \to \mathbb{Z}_2^n$ is an algorithm that maps arbitrarily long bit strings to digests of a fixed length. A hash function may be keyed, which means that it has an additional key input $\mathbb{Z}_2^k$. Hash function that are keyed via a secret key are usually referred to as MAC algorithms, but one may obtain a MAC from a keyless hash function too, for instance using the HMAC or NMAC design [28, 30, 96]. The focus in this dissertation is on keyless hash functions. Hash functions are the heart of many (mostly asymmetric) cryptosystems, such as signature schemes and encryption schemes. Security proofs for such schemes usually assume that this underlying hash function is sufficiently secure with respect to some well-defined property.

In the same way hash functions are used as building blocks on their own, they use building blocks themselves too. We discuss some building blocks. We refer to Menezes et al. [161] for a more technical discussion.

A *compression function* $f : \mathbb{Z}_2^m \to \mathbb{Z}_2^n$ for $m \geq n$ is a hash function limited to inputs of a fixed length. More or less the same security definitions hold for both types of functions. Throughout this chapter, we use the common term "compressing function" when our discussion applies to *both* compression and hash functions.

A *block cipher* $E : \mathbb{Z}_2^k \times \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ is a mapping with the property that $E(K, \cdot)$ is a bijection for all $K \in \mathbb{Z}_2^k$. Here, we refer to k as the key size and n as the plaintext size. For a plaintext m, we denote by $c = E(K, m)$ its corresponding ciphertext under key K. Block ciphers are important cryptographic primitives. Popular block cipher design techniques are Feistel networks [208] and generalized Even-Mansour or key-alternating ciphers [56, 91]. In this dissertation, we simply consider block ciphers as mathematical objects used in the design of compressing functions. We denote by $\mathrm{Bloc}(k, n)$ the set of all block ciphers $E : \mathbb{Z}_2^k \times \mathbb{Z}_2^n \to \mathbb{Z}_2^n$. We simply write $\mathrm{Bloc}(n)$ if $k = n$.

A *permutation* $\pi : \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ is a bijective mapping. Permutations have recently become a popular building block for cryptographic hash functions. A way to get a permutation is to use a block cipher E, and fix the key. This, however requires the block cipher to be known-key secure [2, 132]. Similar to block ciphers, we do not analyze the internal structure of permutations, and simply consider them as mathematical objects. By $\mathrm{Perm}(n)$ we denote the set of all permutations operating on n bits.

2.3 Security Definitions for Compressing Functions

As first suggested by Merkle [168], the basic security properties a compressing function is required to satisfy are collision, preimage, and second preimage security. For a hash function $\mathcal{H} : \mathbb{Z}_2^\infty \to \mathbb{Z}_2^n$, these security properties informally mean:

Collision resistance. It is hard to find two distinct messages y, y' such that $\mathcal{H}(y) = \mathcal{H}(y')$;

Preimage resistance. Given an image $z \in \mathbb{Z}_2^n$, it is hard to find a message y such that $\mathcal{H}(y) = z$;

Second preimage resistance. Given a message $y' \in \mathbb{Z}_2^\infty$, it is hard to find a second message y such that $\mathcal{H}(y) = \mathcal{H}(y')$.

However, due to the wide use of hash functions, further security requirements for hash functions appeared over time, such as multicollision security [119], security against the length extension attack [71], and security against the Nostradamus attack [122]. In this section, we first present formal security definitions for the classical notions of collision, preimage, and second preimage resistance. Next, we introduce chosen-target-forced-midfix security, a notion formalizing and generalizing security against the Nostradamus attack. Finally, we discuss the notion of indifferentiability, a model that by design guarantees security against a wide class of attacks.

Although we discuss the security notions in the context of compressing functions, we note that some of these apply to other primitives too. For instance, the indifferentiability approach is used to analyze the security of block ciphers in [1, 71, 72, 85, 115, 202]. This subject, however, falls beyond the scope of this dissertation.

Throughout this dissertation, we consider security of compressing functions under the assumption that the underlying primitive behaves entirely random. For $m, n \in \mathbb{N}$, we define a random m-to-n-bit function to be a function sampled uniformly at random from $\text{Func}(m, n)$. Similar definitions can be derived for random block ciphers (from $\text{Bloc}(k, n)$, $k \in \mathbb{N}$) or permutations (from $\text{Perm}(n)$). A random primitive will also be called "ideal." This idea is known as the random oracle model, suggested by Fiat and Shamir [94] and formalized by Bellare and Rogaway [33]. A random oracle is a public hash function which provides a random output for each new query. Note that we refer to the random oracle model in a broader sense, as we consider it to apply to various types of primitives.

Ideal primitives cannot be constructed in practice, and cryptosystems are known that are secure in the random oracle model but insecure with any instantiation of it [29,49,65,66,178]. Nevertheless, the random oracle methodology is a leading approach in provable security, both in the case a hash function itself is used as underlying primitive (e.g., [33,34,73,186,201]) as in the case a hash function is used *based* on the ideality of some underlying primitives (see also Section 2.6). It guarantees that a certain design does not suffer from structural design flaws, and that any weaknesses come from the underlying primitive.

In the definitions presented in this section, we consider the information-theoretic model that dates back to Shannon [205]. In the information-theoretic setting, we consider an adversary $\mathcal{A}$ that is given black-box oracle access to a randomly sampled primitive $\mathcal{P} \xleftarrow{\$} \text{Prims}$, which is denoted as $\mathcal{A}^{\mathcal{P}}$. The set

Prims depends on the compressing function to be analyzed: it may consist of block ciphers, permutations, random functions, or a combination thereof. This information-theoretic adversary has unbounded computational power, and its complexity is measured solely in the number of queries made to its oracle. The adversary can make queries to $\mathcal{P}$, which are stored in a query history $\mathcal{Q}$ as indexed elements. We assume that $\mathcal{Q}$ always contains the queries required for the attack and that the adversary never makes trivial queries, i.e., queries to which it knows the answer in advance. For $q \geq 0$, by $\mathcal{Q}_q$ we define the query history after q queries.

2.3.1 Classical Security Definitions

The notions of collision, preimage, and second preimage resistance were formalized by Rogaway and Shrimpton [197] in the context of keyed functions $\mathcal{C}$ (but where the key is publicly known, unlike the case of MAC algorithms (cf. Section 2.2)). For preimage security (similarly for second preimage security), three variants were derived, depending on where the randomness comes from: preimage resistance (both the key and the image z random), everywhere preimage (just the key random), and always preimage resistance (just the image z random). Relations among these definitions are derived in [197]. The latter work has been generalized by Reyhanitabar et al. [191], who consider, motivated by the enhanced target collision resistance notion [109], strengthened variants of the definitions where the adversary can freely choose the key of the second $\mathcal{H}$ evaluation. Andreeva and Stam [20] reconsidered the preimage definitions, and introduced (among others) boosted variants to cover cases the adversary is challenged on input of a range value for which no preimage exists.

Regarding the definition of collision security, we remark that it indeed makes no sense to consider collision resistance without random key (something like "always collision resistance"), as was already pointed out by Damgård [76]. The problem lies in the fact that there always exists an efficient adversary that outputs a collision with probability 1: namely one that has hard-coded in it one of the many collisions. This definitional problem was further investigated by Stinson [214] and Rogaway [196] who proposed a different theoretical treatment by using security reductions. These approaches, however, are not relevant for our analysis.

Let $n \in \mathbb{N}$ and $s \in \mathbb{N} \cup \{\infty\}$ with $s \geq n$. Consider a compressing function $\mathcal{C} : \mathbb{Z}_2^s \to \mathbb{Z}_2^n$ instantiated with a randomly chosen primitive $\mathcal{P} \xleftarrow{\$}$ Prims. This dissertation only deals with keyless functions $\mathcal{C}$, but from a formal point of view one can consider them being *keyed through the random primitive* $\mathcal{P}$ (the primitive functions as a key). This means that the always preimage and always second preimage security definitions become irrelevant in this setting (the "key" $\mathcal{P}$ is always random and the adversary has no control over its choice), but the remaining notions carry over to our model. For (second) preimage resistance, we opt for everywhere (second) preimage resistance: this definition is the strongest as it guarantees security on every range (resp. domain) point. For collision resistance, the notion as formalized in [197] is adopted to the current setting.

In the remainder of this section, we present the formal collision, preimage, and second preimage security definitions used in this dissertation, and discuss the generic security bounds.

2.3.1.1 Collision Resistance

A collision-finding adversary $\mathcal{A}$ for $\mathcal{C}$ aims at finding two distinct inputs to $\mathcal{C}$ that compress to the same range value. In more detail, we say that $\mathcal{A}$ succeeds if it finds two distinct inputs y, y' such that $\mathcal{C}(y) = \mathcal{C}(y')$ and $\mathcal{Q}$ contains all queries required for these evaluations of $\mathcal{C}$. Formally, we define the collision security of $\mathcal{C}$ as follows.

Definition 2.3.1. Let $n \in \mathbb{N}$ and $s \in \mathbb{N} \cup \{\infty\}$ with $s \geq n$, and let $\mathcal{C} : \mathbb{Z}_2^s \to \mathbb{Z}_2^n$ be a compressing function using primitive $\mathcal{P} \in$ Prims. The advantage of a collision-finding adversary $\mathcal{A}$ is defined as

$$\mathbf{Adv}_{\mathcal{C}^{\mathcal{P}}}^{\mathrm{col}}(\mathcal{A}) = \mathbf{Pr}\left(\mathcal{P} \xleftarrow{\$} \mathrm{Prims},\ (y, y') \leftarrow \mathcal{A}^{\mathcal{P}} \ :\ y \neq y' \ \wedge\ \mathcal{C}(y) = \mathcal{C}(y')\right).$$

We define by $\mathbf{Adv}_{\mathcal{C}^{\mathcal{P}}}^{\mathrm{col}}(q)$ the maximum advantage of any adversary making q queries to its oracle. We write $\mathbf{Adv}_{\mathcal{C}}^{\mathrm{col}}$ if $\mathcal{P}$ is clear from the context.

Generic Bounds. For a random oracle $\mathcal{R}$ that outputs bit strings of length n, we have $\mathbf{Adv}_{\mathcal{R}}^{\mathrm{col}}(q) \approx \min\left\{1, \binom{q}{2}/2^n\right\}$ for any $q \geq 2$. This bound is due to the birthday attack [218, 226]. It is known as the generic bound, the best possible collision security bound of a hash function.

2.3.1.2 Preimage Resistance

A preimage-finding adversary $\mathcal{A}$ for $\mathcal{C}$ aims at finding an input to $\mathcal{C}$ that hits a given image. Before making any queries to its oracle, a preimage-finding adversary $\mathcal{A}$ decides on a range point z. Then, we say that $\mathcal{A}$ succeeds in finding a preimage if it obtains an input y such that $\mathcal{C}(y) = z$ and $\mathcal{Q}$ contains all queries required for this evaluation of $\mathcal{C}$. Formally, we define the preimage security of $\mathcal{C}$ as follows.

Definition 2.3.2. Let $n \in \mathbb{N}$ and $s \in \mathbb{N} \cup \{\infty\}$ with $s \geq n$, and let $\mathcal{C} : \mathbb{Z}_2^s \to \mathbb{Z}_2^n$ be a compressing function using primitive $\mathcal{P} \in$ Prims. The advantage of an everywhere preimage-finding adversary $\mathcal{A}$ is defined as

$$\mathbf{Adv}_{\mathcal{C}^{\mathcal{P}}}^{\mathrm{epre}}(\mathcal{A}) = \max_{z \in \mathbb{Z}_2^n} \mathbf{Pr}\left(\mathcal{P} \xleftarrow{\$} \mathrm{Prims},\ y \leftarrow \mathcal{A}^{\mathcal{P}}(z) \ :\ \mathcal{C}(y) = z\right).$$

We define by $\mathbf{Adv}_{\mathcal{C}^{\mathcal{P}}}^{\mathrm{epre}}(q)$ the maximum advantage of any adversary making q queries to its oracle. We write $\mathbf{Adv}_{\mathcal{C}}^{\mathrm{epre}}$ if $\mathcal{P}$ is clear from the context.

Generic Bounds. For a random oracle $\mathcal{R}$ that outputs bit strings of length n, we have $\mathbf{Adv}_{\mathcal{R}}^{\mathrm{epre}}(q) \approx \min\{1, q/2^n\}$ for any q. This bound is known as the generic bound, achieved by exhaustive search.

2.3.1.3 Second Preimage Resistance

A second preimage-finding adversary $\mathcal{A}$ for $\mathcal{C}$ aims at finding an input to $\mathcal{C}$ that hits the image of another, given, input. Before making any queries to its oracle, a second preimage-finding adversary $\mathcal{A}$ decides on a domain point y' (for technical reasons, this domain point should be chosen from a finite subset of the domain $\mathbb{Z}_2^s$). Then, we say that $\mathcal{A}$ succeeds in finding a second preimage if it obtains an input y such that $\mathcal{C}(y) = \mathcal{C}(y')$ and $\mathcal{Q}$ contains all queries required for these evaluations of $\mathcal{C}$. Formally, we define the second preimage security of $\mathcal{C}$ as follows.

Definition 2.3.3. Let $n \in \mathbb{N}$ and $s \in \mathbb{N} \cup \{\infty\}$ with $s \geq n$, and let $\mathcal{C} : \mathbb{Z}_2^s \to \mathbb{Z}_2^n$ be a compressing function using primitive $\mathcal{P} \in \text{Prims}$. Let $\lambda \in \mathbb{N}$ with $\lambda \leq s$. The advantage of an everywhere second preimage-finding adversary $\mathcal{A}$ is defined as

$$\mathbf{Adv}_{\mathcal{C}^{\mathcal{P}}}^{\text{esec}[\lambda]}(\mathcal{A}) = \max_{y' \in \mathbb{Z}_2^{\lambda}} \mathbf{Pr}\left(\mathcal{P} \xleftarrow{\$} \text{Prims},\ y \leftarrow \mathcal{A}^{\mathcal{P}}(y')\ :\ y \neq y' \ \wedge\ \mathcal{C}(y) = \mathcal{C}(y')\right).$$

We define by $\mathbf{Adv}_{\mathcal{C}^{\mathcal{P}}}^{\text{esec}[\lambda]}(q)$ the maximum advantage of any adversary making q queries to its oracle. We write $\mathbf{Adv}_{\mathcal{C}}^{\text{esec}[\lambda]}$ if $\mathcal{P}$ is clear from the context.

If $\mathcal{C}$ is a compression function (i.e., $s < \infty$), we usually require $\lambda = s$. Conventionally, we sometimes express the length of the first preimage y' in blocks by 2^L. This translates to $2^L \approx \lambda/m$, where m denotes the length of a message block (message blocks are introduced in detail in Section 2.4).

Generic Bounds. For a random oracle $\mathcal{R}$ that outputs bit strings of length n, we have $\mathbf{Adv}_{\mathcal{R}}^{\text{esec}[\lambda]}(q) \approx \min\{1, q/2^n\}$ for any λ, q. This bound is known as the generic bound, achieved by exhaustive search. Second preimage resistance is implied by collision resistance: $\mathbf{Adv}_{\mathcal{C}^{\mathcal{P}}}^{\text{esec}[\lambda]}(q) \leq \mathbf{Adv}_{\mathcal{C}^{\mathcal{P}}}^{\text{col}}(q)$ for any λ, q [197]. Although the difference with preimage resistance seems subtle, practical hash functions exist that are significantly more preimage than second preimage secure (see, e.g., Section 2.4.1).

2.3.2 Chosen-Target-Forced-Midfix Preimage Resistance

In this section, we generalize and formalize security against the Nostradamus or herding attack by Kelsey and Kohno [122]. Consider a hash function $\mathcal{C} : \mathbb{Z}_2^{\infty} \to \mathbb{Z}_2^n$ built on a primitive $\mathcal{P} \in \text{Func}(n + m, n)$. The herding attack, also known as the chosen-target-forced-prefix (CTFP) attack, considers an adversary that commits to a hash value z for a message that is not entirely under its control. The adversary then demonstrates abilities to incorporate an unknown challenge prefix as part of the original preimage corresponding to the committed value z. While for a random oracle the complexity of such an attack is $\Theta(2^n)$ primitive function calls, Kelsey and Kohno [122] described the herding attack on Merkle-Damgård (see Section 2.4) that takes about $\sqrt{n}2^{2n/3}$ primitive executions and results in a preimage of length $O(n)$. (A more precise attack bound was obtained by Blackburn et al. [54].) The herding attack has been generalized by Mennink in [162], who presented the elongated multidimensional herding attack. In [5], Andreeva et al. showed applications of the herding attack to dither hash functions, and in [4] they generalized the herding attack of [122] to several multi-pass domain extenders, such as the zipper hash (see Section 2.5.3) and the hash twice design. Gauravaram et al. [102, 103] concluded insecurity against herding attacks for the Merkle-Damgård designs where a XOR tweak is used in the final message block.

Neven et al. first formalized the notion of chosen-target-forced-prefix security (as the random-prefix preimage problem) in [177]. We present the new notion of chosen-target-forced-midfix (CTFM) preimage security that allows for challenges that can technically occur at any place in the preimage message. This is achieved by giving the adversary the power to output an "order"-defining mixing function g together with its response, which fixes the challenge message at a selected by the adversary position in the preimage message. This definition has been published by Andreeva and Mennink in [8].

Definition 2.3.4. Let $m, n \in \mathbb{N}$, and let $\mathcal{C} : \mathbb{Z}_2^\infty \to \mathbb{Z}_2^n$ be a compressing function using primitive $\mathcal{P} \in \mathrm{Func}(n+m, n)$. Let $p \in \mathbb{N}$ denote the length of the forced midfix, and $L \geq 1$ be the maximal length of the forged preimage in blocks. The advantage of a chosen-target-forced-midfix (CTFM) preimage-finding adversary $\mathcal{A}$ is defined as

$$\mathbf{Adv}_{\mathcal{C}^{\mathcal{P}}}^{\mathrm{ctfm}}(\mathcal{A}) = \mathbf{Pr}\left(\begin{array}{r} \mathcal{P} \xleftarrow{\$} \mathrm{Func}(n+m, n),\ (z, st) \leftarrow \mathcal{A}^{\mathcal{P}}, \\ P \xleftarrow{\$} \mathbb{Z}_2^p,\ (g, R) \leftarrow \mathcal{A}^{\mathcal{P}}(P, st)\ : \\ \mathcal{C}(g(P, R)) = z\ \wedge\ \left|\mathsf{rng}(g)\right| \leq 2^{Lm} \end{array}\right).$$

We define by $\mathbf{Adv}_{\mathcal{C}^{\mathcal{P}}}^{\mathrm{ctfm}}(q)$ the maximum advantage of any adversary making q queries to its oracle.

The function g can technically be any function as long as the size of its range is at most 2^{Lm}, but for some choices of g the definition becomes irrelevant. For instance, if the mutual information between P and $g(P, R)$ is 0, the CTFM attack is trivial. More generally, the attack becomes easier if the function g is allowed to split P into parts. However, this type of functions does not correspond to any practically relevant attack, and in the remainder we restrict g to satisfy that $g(P, R_1 \| R_2) = R_1 \| P \| R_2$, where R_1, R_2 are of arbitrary length. The chosen-target-forced-prefix attack of Kelsey and Kohno is covered for g restricted to R_1 being the empty string. If $\mathcal{C}$ processes the message blocks in reverse order, the notion covers the "chosen-target-forced-suffix attack," with R_2 being the empty string.[1]

The value p defines the size of the challenge P, and plays an important role in the security results. A smaller value of p allows for higher success probability in

[1]Exactly this attack was performed by Andreeva et al. [4] on the zipper hash function (see Section 2.5.4).

guessing P in the first phase of the attack. A larger value of p limits the number of “free” queries of the adversary, the adversary needs to make at least $\lceil p/m \rceil$ compression function queries for incorporating the challenge P, where m is the message block size.

2.3.3 Indifferentiability

To prove the security of practical systems or schemes, one frequently employs the ideal (random oracle) security model where the hash function is assumed to be an idealized function or a random oracle. While random oracles are monolithic objects, real world hash functions are highly structured. A security notion that formally captures the “distance” between two such objects is indistinguishability. In indistinguishability, we consider an adversary that has access to either of the two objects and its goal is to make a correct guess of the primitive. In this setting, the adversary is usually referred to as a “distinguisher,” as its goal is to distinguish both objects. However, this notion does not always capture the behavior of structured functions, of which the underlying primitive is often publicly available. An improved notion is the indifferentiability framework, introduced by Maurer et al. [159]. Informally, it gives a sufficient condition under which an ideal primitive $\mathcal{R}$ can be replaced by some construction $\mathcal{C}^{\mathcal{P}}$ using an ideal subcomponent $\mathcal{P}$. In other words, indifferentiability guarantees that a mode of operation has no structural design flaws. In this dissertation, we employ the adaption and simplification by Coron et al. [71].

Definition 2.3.5. Let $\mathcal{C}$ be a compressing primitive with oracle access to an ideal primitive $\mathcal{P}$. Let $\mathcal{R}$ be an ideal primitive with the same domain and range as $\mathcal{C}$. Let $\mathcal{S}$ be a simulator with the same domain and range as $\mathcal{P}$ with oracle access to $\mathcal{R}$ and making at most $q_{\mathcal{S}}$ queries, and let $\mathcal{D}$ be a distinguisher making at most $q_{\mathcal{D}}$ queries. The differentiability advantage of $\mathcal{D}$ is defined as

$$\mathbf{Adv}^{\mathrm{iff}}_{\mathcal{C}^{\mathcal{P}},\mathcal{S}}(\mathcal{D}) = \left|\mathbf{Pr}\left(\mathcal{D}^{\mathcal{C}^{\mathcal{P}},\mathcal{P}} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{\mathcal{R},\mathcal{S}^{\mathcal{R}}} = 1\right)\right| .$$

By $\mathbf{Adv}^{\mathrm{iff}}_{\mathcal{C}^{\mathcal{P}},\mathcal{S}}(q_{\mathcal{D}})$ we denote the maximum advantage of any distinguisher making at most $q_{\mathcal{D}}$ queries to its oracles. Primitive $\mathcal{C}$ is said to be $(q_{\mathcal{D}}; q_{\mathcal{S}}; \varepsilon)$ indifferentiable from $\mathcal{R}$ if $\mathbf{Adv}^{\mathrm{iff}}_{\mathcal{C}^{\mathcal{P}},\mathcal{S}}(q_{\mathcal{D}}) < \varepsilon$.

Distinguisher $\mathcal{D}$ can query both its *left oracle* L (either $\mathcal{C}$ or $\mathcal{R}$) and its *right oracle* R (either $\mathcal{P}$ or $\mathcal{S}$). We refer to $\mathcal{C}^{\mathcal{P}}, \mathcal{P}$ as the *real world*, and to $\mathcal{R}, \mathcal{S}^{\mathcal{R}}$ as the *simulated world*; the distinguisher $\mathcal{D}$ converses with either of these worlds and its goal is to tell both worlds apart. The indifferentiability framework is depicted in Figure 2.1. We note that several variants of indifferentiability [88,172,173,225] have appeared in the literature, where random oracles are replaced with other functionalities.

Recent results by Ristenpart et al. [192] show that indifferentiability does not capture all properties of a random oracle, it applies to single stage games only

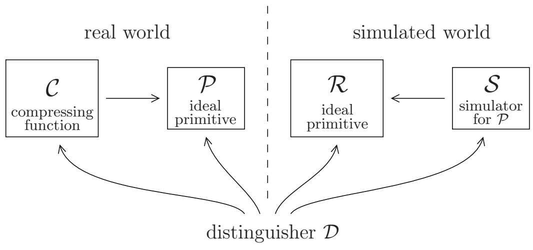


Figure 2.1. A depiction of the indifferentiability framework of Definition 2.3.5.

(we refer to [80, 157] for refinements). Nevertheless, this indifferentiability notion captures pretty many games and remains the best way to prove that a hash function behaves like a random oracle. An indifferentiability bound guarantees that, up to a certain degree, the design is secured against any generic attack, like finding collisions, preimages, multicollisions, fixed-points, etc., that correspond to single stage games. In Theorem 2.3.6, we formally prove a reduction to derive security against specific attacks from the indifferentiability of a hash function. This theorem has been published by Andreeva, Mennink, and Preneel in [12].

Theorem 2.3.6. *Let $\mathcal{C}$ be a compressing primitive, built on underlying primitive $\mathcal{P}$, and $\mathcal{R}$ be an ideal primitive with the same domain and range as $\mathcal{C}$. Let* atk *be a security property of $\mathcal{C}$ (e.g.,* col*), and denote by $\mathbf{Adv}^{\mathrm{atk}}_{\mathcal{C}^{\mathcal{P}}}(q)$ the advantage of breaking $\mathcal{C}^{\mathcal{P}}$ under* atk*, maximized over all adversaries $\mathcal{A}$ making at most q queries. For any simulator $\mathcal{S}$ we have*

$$\mathbf{Adv}^{\mathrm{atk}}_{\mathcal{C}^{\mathcal{P}}}(q) \leq \mathbf{Adv}^{\mathrm{atk}}_{\mathcal{R}}(q) + \mathbf{Adv}^{\mathrm{iff}}_{\mathcal{C}^{\mathcal{P}},\mathcal{S}}(q)\,. \tag{2.1}$$

Proof. Let $\mathcal{S}$ be any simulator. Let $\mathcal{A}$ be any (q, ε) adversary for $\mathcal{C}^{\mathcal{P}}$ under security notion atk. We define a distinguisher $\mathcal{D}$ for the indifferentiability of $\mathcal{C}$ as follows. $\mathcal{D}$ makes the same queries as $\mathcal{A}$, and obtains a query history $\mathcal{Q}$. Next, $\mathcal{D}$ outputs 1 if $\mathcal{Q}$ violates security notion atk (for $\mathcal{C}$ or $\mathcal{R}$), and 0 otherwise. By definition, we have $\left|\mathbf{Pr}\left(\mathcal{D}^{\mathcal{C}^{\mathcal{P}},\mathcal{P}} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{\mathcal{R},\mathcal{S}^{\mathcal{R}}} = 1\right)\right| = \mathbf{Adv}^{\mathrm{iff}}_{\mathcal{C}^{\mathcal{P}},\mathcal{S}}(\mathcal{D}) \leq \mathbf{Adv}^{\mathrm{iff}}_{\mathcal{C}^{\mathcal{P}},\mathcal{S}}(q)$.

By $\mathsf{E}_{\mathcal{C}}$ (resp. $\mathsf{E}_{\mathcal{R}}$), we denote the event that the queries in $\mathcal{Q}$ break $\mathcal{C}$ (resp. $\mathcal{R}$) under security notion atk. The distinguisher outputs 1 if and only if $\mathcal{Q}$ violates security notion atk, and hence we obtain:

$$\begin{aligned}\mathbf{Pr}\left(\mathcal{D}^{\mathcal{C}^{\mathcal{P}},\mathcal{P}} = 1\right) &= \mathbf{Pr}\left(\mathcal{D}^{\mathcal{C}^{\mathcal{P}},\mathcal{P}} = 1 \mid \mathsf{E}_{\mathcal{C}}\right)\mathbf{Pr}\left(\mathsf{E}_{\mathcal{C}}\right) + \\ &\qquad \mathbf{Pr}\left(\mathcal{D}^{\mathcal{C}^{\mathcal{P}},\mathcal{P}} = 1 \mid \neg\, \mathsf{E}_{\mathcal{C}}\right)\mathbf{Pr}\left(\neg\, \mathsf{E}_{\mathcal{C}}\right) \\ &= 1 \cdot \mathbf{Pr}\left(\mathsf{E}_{\mathcal{C}}\right) + 0 = \varepsilon\,.\end{aligned}$$

Similarly, we get $\mathbf{Pr}\left(\mathcal{D}^{\mathcal{R},\mathcal{S}^{\mathcal{R}}} = 1\right) = \mathbf{Pr}\left(\mathsf{E}_{\mathcal{R}}\right) = \mathbf{Adv}^{\mathrm{atk}}_{\mathcal{R}}(q)$. We consequently derive $\varepsilon \leq \mathbf{Adv}^{\mathrm{atk}}_{\mathcal{R}}(q) + \mathbf{Adv}^{\mathrm{iff}}_{\mathcal{C}^{\mathcal{P}},\mathcal{S}}(q)$. This holds for any (q, ε) adversary for $\mathcal{C}$ under security notion atk, which completes the proof. □

2.4 Iterated Hashing Principle

The classical approach of building a hash function $\mathcal{C} : \mathbb{Z}_2^{\infty} \to \mathbb{Z}_2^n$ is the *iterated hash function principle*, which dates back to Rabin [189]: the hash function internally uses a compression function $f : \mathbb{Z}_2^{n+m} \to \mathbb{Z}_2^n$. Now, on input of a message $M \in \mathbb{Z}_2^{\infty}$, the iterated hash function $\mathcal{H}^f$ operates as follows. Firstly, it applies a padding mechanism pad to M, an injective mapping that results in an integral number of m-bit message blocks $(M_1, \ldots, M_k)$ (the padding function is discussed in detail later). Then, it initializes a certain n-bit state with some initialization vector $\mathsf{iv} \in \mathbb{Z}_2^n$, and iteratively compresses each message block with the state. In more detail, $\mathcal{H}^f$ is defined as follows.

$$
\begin{aligned}
&\mathcal{H}^f(M)\\
&\quad (M_1, \ldots, M_k) \leftarrow \mathsf{pad}(M)\,,\\
&\quad h_0 \leftarrow \mathsf{iv}\,,\\
&\quad h_i \leftarrow f(h_{i-1}, M_i) \text{ for } i = 1, \ldots, k\,,\\
&\quad \textbf{return } h_k\,.
\end{aligned}
\tag{2.2}
$$

This principle is also called the plain Merkle-Damgård (MD) design [76,137,167]. It is adopted by most practical hash functions, including the MDx and SHA-x family of hash functions. Often, this basic design is followed by a final transformation g (that may or may not compress a message block) and/or a left-function.[2] This generalized hash function design is displayed in Figure 2.2. If a final chopping is involved, we refer to the construction as a *wide-pipe* design [153], as the iterated state size is larger than the final hash output. In this section, we consider the security of this generalized design, and in Section 2.5 we discuss several variants of this design idea.

The padding function $\mathsf{pad} : \mathbb{Z}_2^{\infty} \to (\mathbb{Z}_2^m)^{\infty}$ is an injective mapping that transforms a message of arbitrary length to a message of length a multiple of m bits (the number of message bits compressed in one compression function iteration). Associated to pad is a function depad, that on input of a bit string M' either outputs $\bot$ if $M' \notin \mathsf{rng}(\mathsf{pad})$, or the unique M such that $M' = \mathsf{pad}(M)$ otherwise. Sometimes, hash function designs compress specific counters or tweaks along with the message blocks (the inputs M_i to f in (2.2)). For the purpose of abstraction, we consider these inputs as part of the padding function, except when explicitly noted.

[2]For technical reasons, the left-function is considered not to be a part of the final transformation: it simply refers to discarding a specified number of bits from the output and involves no underlying randomized primitive.

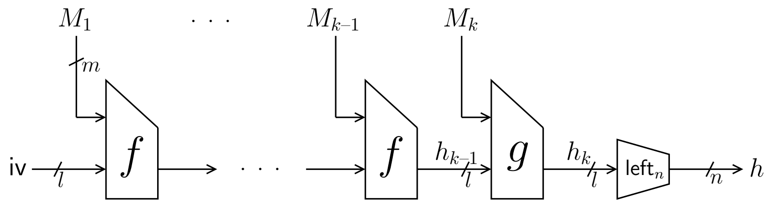


Figure 2.2. Generalized Merkle-Damgård structure. Here, f, g are two compression functions.

We distinguish between "prefix-free" and/or "suffix-free" padding. A padding rule is called suffix-free if for any distinct M, M', there exists no bit string X such that $\mathsf{pad}(M') = X \| \mathsf{pad}(M)$. It is called prefix-free if for any distinct M, M', there exists no bit string X such that $\mathsf{pad}(M') = \mathsf{pad}(M) \| X$.

2.4.1 Classical Security Definitions

2.4.1.1 Collision Resistance

The plain Merkle-Damgård design with fixed iv and any suffix-free padding (also called MD-strengthening [137]) preserves collision resistance [19, 76, 167],[3] which means that it is collision resistant if the underlying compression function f is. We generalize this well-known result by Merkle and Damgård to cover hash functions with a final transformation and a chop function (see Figure 2.2). This theorem has been published by Andreeva, Mennink, and Preneel in [12]. Related work can among others be found in [76, 86, 167, 175].

Theorem 2.4.1. *Let $l, m, n \in \mathbb{N}$ such that $l \geq n$. Let $\mathsf{pad} : \mathbb{Z}_2^\infty \to (\mathbb{Z}_2^m)^\infty$ be a suffix-free padding and let $f, g \in \mathrm{Func}(l + m, l)$ using ideal primitive $\mathcal{P}$. Consider the hash function $\mathcal{H} : \mathbb{Z}_2^\infty \to \mathbb{Z}_2^n$ defined as follows (cf. Figure 2.2).*

```
$\mathcal{H}^{\mathcal{P}}(M)$
    $(M_1, \ldots, M_k) \leftarrow \mathsf{pad}(M)$,
    $h_0 \leftarrow \mathsf{iv}$,
    $h_i \leftarrow f^{\mathcal{P}}(h_{i-1}, M_i)$ for $i = 1, \ldots, k-1$,
    $h_k \leftarrow g^{\mathcal{P}}(h_{k-1}, M_k)$,
    $h \leftarrow \mathsf{left}_n(h_k)$,
    return $h$.
```

Define the function $g' : \mathbb{Z}_2^l \times \mathbb{Z}_2^m \to \mathbb{Z}_2^n$ by $g' = \mathsf{left}_n \circ g$. Then, the advantage of finding a collision for $\mathcal{H}$ is upper bounded by the advantage of finding a collision

[3] In their original proposals, Merkle and Damgård both employed different padding schemes, but the result holds for any suffix-free padding.

for f or g'. Formally,

$$\mathbf{Adv}^{\mathrm{col}}_{\mathcal{H}^{\mathcal{P}}}(q) \leq \mathbf{Adv}^{\mathrm{col}}_{f^{\mathcal{P}}}(q) + \mathbf{Adv}^{\mathrm{col}}_{g'^{\mathcal{P}}}(q)\,,$$

Proof. Suppose $\mathcal{A}$ is a (q, ε) collision-finding adversary for $\mathcal{H}$. We construct collision-finding adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$ for f and g', respectively, using the following observation.

Let M, M' be two distinct messages such that $\mathcal{H}(M) = \mathcal{H}(M')$. Let $(M_1, \dots, M_k)$ be the padded message of M, and $(M'_1, \dots, M'_{k'})$ be the padded message of M'. Define the intermediate state values h_i, h'_i similarly. A collision on M, M' means that $\mathsf{left}_n\big(g(h_{k-1}, M_k)\big) = \mathsf{left}_n\big(g(h'_{k'-1}, M'_{k'})\big)$. Now, if $(h_{k-1}, M_k) \neq (h'_{k'-1}, M'_{k'})$ this results in a collision for g'. Assume the contrary, and let $j \in \{1, \dots, \min\{k, k'\} - 1\}$ be the minimal index such that $(h_{k-j-1}, M_{k-j}) \neq (h'_{k'-j-1}, M'_{k'-j})$. We notice that such index j exists: if $k = k'$ it exists as $M \neq M'$, and if $k \neq k'$ it exists as the padding rule is suffix-free. By definition of the index j, we have $h_{k-j} = h'_{k'-j}$, and in particular we obtain a collision for f:

$$f(h_{k-j-1}, M_{k-j}) = h_{k-j} = h'_{k'-j} = f(h'_{k'-j-1}, M'_{k'-j})\,.$$

Both $\mathcal{B}_1, \mathcal{B}_2$ follow this procedure. If M, M' define a collision for f, $\mathcal{B}_1$ outputs this collision. Similarly for $\mathcal{B}_2$ and g'. Both adversaries make the same amount of queries as $\mathcal{A}$. □

Note that the theorem also covers hash functions whose final transformation does not take a message block. If the design is based on the compression function f only (but it may still include the chopping), the above result can easily be simplified to $\mathbf{Adv}^{\mathrm{col}}_{\mathcal{H}^{\mathcal{P}}}(q) \leq \mathbf{Adv}^{\mathrm{col}}_{f'^{\mathcal{P}}}(q)$, where f' is defined by $f' = \mathsf{left}_n \circ f$. Observe that this result also holds if $l = n$, and in particular, the basic theorems of Merkle and Damgård are covered as well. We note that Theorem 2.4.1 can be generalized arbitrarily, e.g., to more different compression functions, but the current statement suffices for the purpose of this dissertation.

Related to collision resistance of Merkle-Damgård is the attack by Joux [119], who describes a method to find multicollisions for the iterated hash function design. For $t \geq 2$, his algorithm results in 2^t-fold collisions with query complexity approximately $t2^{n/2}$, as opposed to the generic complexity bound $2^{n(t-1)/t}$.

2.4.1.2 Preimage Resistance

The plain Merkle-Damgård design preserves preimage resistance [19], which means that it is preimage resistant if the underlying compression function f is. Note that we do not require a suffix-free padding, as was the case for collision resistance. In a similar fashion as Theorem 2.4.1 it can be generalized and formalized.

Theorem 2.4.2. *Let $l, m, n \in \mathbb{N}$ such that $l \geq n$. Let* $\mathsf{pad} : \mathbb{Z}_2^\infty \to (\mathbb{Z}_2^m)^\infty$ *be a padding and let $f, g \in \mathrm{Func}(l+m, l)$ using ideal primitive $\mathcal{P}$. Consider the hash function $\mathcal{H} : \mathbb{Z}_2^\infty \to \mathbb{Z}_2^n$ defined in Theorem 2.4.1. Define the function $g' : \mathbb{Z}_2^l \times \mathbb{Z}_2^m \to \mathbb{Z}_2^n$ by $g' = \mathsf{left}_n \circ g$. Then, the advantage of finding a preimage for $\mathcal{H}$ is upper bounded by the advantage of finding a preimage g'. Formally,*

$$\mathbf{Adv}^{\mathrm{epre}}_{\mathcal{H}^{\mathcal{P}}}(q) \leq \mathbf{Adv}^{\mathrm{epre}}_{g'^{\mathcal{P}}}(q)\,,$$

Proof. Suppose $\mathcal{A}$ is a (q, ε) preimage-finding adversary for $\mathcal{H}$. We construct a preimage-finding adversary $\mathcal{B}$ for g'. Let $h \in \mathbb{Z}_2^n$ be any range value. $\mathcal{B}$ runs $\mathcal{A}$ on input of h, resulting in a message M such that $\mathcal{H}^{\mathcal{P}}(M) = h$. $\mathcal{B}$ computes the values h_{k-1}, M_k corresponding to the evaluation of $\mathcal{H}^{\mathcal{P}}(M)$, which form the preimage for g'. Adversary $\mathcal{B}$ makes the same amount of queries as $\mathcal{A}$. □

2.4.1.3 Second Preimage Resistance

Everywhere second preimage resistance is not preserved by the Merkle-Damgård design [19]. In fact, Dean [79] and Kelsey and Schneier [123] describe generic second preimage attacks on Merkle-Damgård that require at most approximately 2^{n-L} queries, where the first preimage is of length at most 2^L blocks. It has been proven by Bouillaguet and Fouque that the plain Merkle-Damgård design with suffix-free padding satisfies $\mathbf{Adv}^{\mathrm{esec}[m2^L]}_{\mathcal{H}^f}(q) \leq q2^L/2^n$, where the first preimage is of length 2^L blocks of m bits [59], hence that the attacks of [79, 123] are tight. Enlarging the internal state size increases the complexity of these attacks. In particular, without going into detail, we note that for the hash function $\mathcal{H}$ defined in Theorem 2.4.1 (see Figure 2.2) we obtain $\mathbf{Adv}^{\mathrm{esec}[m2^L]}_{\mathcal{H}^{f,g}}(q) \leq q2^L/2^l + q/2^n$ (we refer to Theorems 7.1.4 and 7.2.2 for similar second preimage resistance proofs).

2.4.2 Chosen-Target-Forced-Midfix Preimage Resistance

We present a CTFM security proof for the Merkle-Damgård domain extender. In more detail, we upper bound the strength of a CTFM adversary in finding in the ideal model a preimage for the Merkle-Damgård design, and show that the herding attack described by Kelsey and Kohno is optimal (asymptotically). We consider the Merkle-Damgård hash function as in (2.2) with the following padding function:

$$\mathsf{pad}(M) = M \| 10^{-|M|-1 \bmod m}\,,$$

split into blocks of length m. We elaborate on generalizations at the end of this section. Throughout, we assume $m = \Theta(n)$.

As demonstrated by Kelsey and Kohno [122] and Blackburn et al. [54], one can obtain a CTFM preimage of length $O(n)$ in about $\sqrt{n}2^{2n/3}$ compression function executions. When larger preimages are allowed, the elongated herding attack of [122] results in faster attacks: for $0 \leq r \leq n/2$, one can find a CTFM preimage of length $L = O(n+2^r)$ in approximately $\sqrt{n}2^{(2n-r)/3}$ queries. As we will demonstrate, this is (asymptotically) the best possible result. In Theorem 2.4.3 we derive an upper bound on $\mathbf{Adv}_{\mathcal{H}^f}^{\mathrm{ctfm}}(q)$ that holds for any q, and we consider the limiting behavior in Corollary 2.4.4, in which we show that at least $2^{2n/3}/L^{1/3}$ queries are needed for an attack to succeed. The findings in this section have been published by Andreeva and Mennink in [8].

Theorem 2.4.3. *Let $m, n \in \mathbb{N}$, and let $\mathcal{C} : \mathbb{Z}_2^\infty \to \mathbb{Z}_2^n$ be a compressing function using ideal primitive $f \in \mathrm{Func}(n+m, n)$. Let $p \in \mathbb{N}$ denote the length of the forced midfix, and $L \geq 1$ be the maximal length of the forged preimage in blocks. Then, for any positive integral value τ,*

$$\mathbf{Adv}_{\mathcal{H}^f}^{\mathrm{ctfm}}(q) \leq \frac{(L-1)\tau q}{2^n} + \frac{m2^{\lceil p/m \rceil}q}{2^p} + \left(\frac{q^2 e}{\tau 2^n}\right)^\tau + \frac{q^3}{2^{2n}}\,.$$

The proof is, for ease of presentation, included in Appendix A.1. The bound includes a parameter τ used to bound multiple events in the security analysis, and the bound holds for any integral τ. Notice that for large values of τ, the first factor of the bound becomes large, while for small values the third term becomes large. Therefore, it is of importance to find a balance for this value τ. Recall that an adversary has a higher success probability if larger preimages are allowed. Consequently, the optimum for τ partially depends on the allowed length L. In Corollary 2.4.4 we analyze the limiting behavior of the bound of Theorem 2.4.3.

Notice that the bound of Theorem 2.4.3 contains a term that does not directly depend on n, the second term. This term essentially represents the "guessing probability" of the adversary: $\mathcal{A}$ may succeed guessing P in advance. If $p = |P|$ is very small, this factor dominates the bound. Therefore, it only makes sense to evaluate this bound for p being "large enough," and we have to put a requirement on p. Although the requirement looks complicated at first sight, it is satisfied by any relevant value of p. In particular, it is satisfied for $p \geq 2n/3$ for $L = O(n)$ preimages and even for lower values of p when L becomes larger.

Corollary 2.4.4. *Let $L = O(2^{n/2})$ and let p be such that $\frac{2^{\lceil p/m \rceil}2^{2n/3}}{L^{1/3}2^p} = O(1)$ for $n \to \infty$. For any $\varepsilon > 0$, we obtain* $\lim_{n\to\infty} \mathbf{Adv}_{\mathcal{H}^f}^{\mathrm{ctfm}}\left(2^{n(2/3-\varepsilon)}/L^{1/3}\right) = 0$.

Proof. The bound of Theorem 2.4.3 holds for any $\tau \geq 1$. As $L = O(2^{n/2})$, there exists a constant c such that $L \leq c2^{n/2}$. We put $\tau = \frac{2^{n/3}}{(L/c)^{2/3}} \geq 1$. Without loss of generality, τ is integral (one can tweak c a little bit to get an integral τ). From Theorem 2.4.3:

$$\mathbf{Adv}_{\mathcal{H}^f}^{\mathrm{ctfm}}(q) \leq \frac{L^{1/3}c^{2/3}q}{2^{2n/3}} + \frac{m2^{\lceil p/m \rceil}q}{2^p} + \left(\frac{(L/c)^{2/3}q^2 e}{2^{4n/3}}\right)^{\frac{2^{n/3}}{(L/c)^{2/3}}} + \frac{q^3}{2^{2n}}\,.$$

For any $\varepsilon > 0$, we obtain:

$$\mathbf{Adv}_{\mathcal{H}^f}^{\mathrm{ctfm}}\left(\frac{2^{n(2/3-\varepsilon)}}{L^{1/3}}\right) \leq \frac{c^{2/3}}{2^{n\varepsilon}} + \frac{m}{2^{n\varepsilon}}\frac{2^{\lceil p/m \rceil}2^{2n/3}}{L^{1/3}2^p} + \left(\frac{e}{c^{2/3}2^{2n\varepsilon}}\right)^{\frac{2^{n/3}}{(L/c)^{2/3}}} + \frac{1}{L2^{3n\varepsilon}}\,.$$

All terms approach 0 for $n \to \infty$ (notice that for the second term we have $m = \Theta(n)$, and for the third term its exponent is ≥ 1). $\square$

Due to their generic nature, Theorem 2.4.3 and Corollary 2.4.4 directly carry over to a broad spectrum of domain extenders derived from Merkle-Damgård, including Merkle-Damgård with length strengthening (Section 2.4.1.1). Indeed, a CTFM attack against strengthened Merkle-Damgård is provably harder than an attack against plain Merkle-Damgård due to the presence of the length encoding at the end. We elaborate on this for other domain extenders in Section 2.5. Our results generalize to any Merkle-Damgård based function with final transformation: one can use security properties of the final transformation to show the adversary knows only a limited amount of state values y' which propagate to the commitment y through the final transformation, and we can analyze the success probability with respect to each of these possible commitments y'.

2.4.3 Indifferentiability

It has been proven that the Merkle-Damgård design, based on ideal compression function or PGV construction with ideal primitive (see Section 2.6.1), with prefix-free padding is indifferentiable from a random oracle [68, 71, 105, 155]. Using a suffix-free padding does not necessarily result in an indifferentiable hash function, as a length extension attack applies [71]. The Merkle-Damgård construction is also indifferentiable if it ends with a function that chops a significant part of the state or with a final transformation [71, 105, 155].

Without going into much detail, we remark another approach to construct indifferentiable hash functions that was demonstrated by Dodis et al. [88]. They show that the Merkle-Damgård design with final transformation is indifferentiable from a random oracle if f is preimage aware, and g ideal. Informally, preimage awareness of f means that if an adversary outputs a range value z for which it later outputs a preimage y, then it must have been aware of this value y prior to outputting z.

2.5 Variants of the Iterated Hashing Principle

We discuss three variants of the iterated hashing principle, the HAIFA design, sponge functions, and hash trees, and briefly elaborate on some other variants.

2.5.1 HAIFA

A design based on the Merkle-Damgård principle is the HAIFA construction by Biham and Dunkelman [46]. In HAIFA the message is padded in a specific way so as to solve some deficiencies of the original Merkle-Damgård construction: in the iteration, each message block is accompanied with a fixed (optional) salt of s bits and a (mandatory) counter T_i of t bits. The counter T_i keeps track of the number of message bits hashed so far and equals 0 if the i^{th} block does not contain any message bits.[4] Partially due to the properties of this counter, the HAIFA padding rule is both suffix- and prefix-free. As a consequence, the construction preserves collision and preimage resistance (Theorems 2.4.1 and 2.4.2) and the indifferentiability results of Coron et al. [71] carry over. For HAIFA, these indifferentiability results have been improved by Bhattacharyya et al. in [44]. Furthermore, the HAIFA construction is proven optimally secure against second preimage attacks (up to bound $2q/2^n$) if the underlying compression function is ideal [59]. The CTFM results of Section 2.4.2 directly generalize to HAIFA.

2.5.2 Sponge Functions

In 2007, Bertoni et al. introduced the sponge hash functions [41] as an alternative to the Merkle-Damgård design. The sponge design idea is to obtain a secure hash function by iterating a compression function that does not necessarily satisfy the main hash function security properties. The sponge hash function design begins with an absorbing phase, in which message blocks of r bits per compression function f call are processed, where the iterated state is of size $r + c$ bits with c being the so-called "capacity." Next, it has an extraction phase, in which an n-bit hash digest is extracted r bits at a time by applying the extraction function g. The sponge function employs a single $(r + c)$-bit permutation π, and the compression function f and extraction function g are defined as $f(h_r \| h_c, M_i) = \pi((h_r \oplus M_i) \| h_c)$, where M_i is an r-bit message block, and $g(h_r \| h_c) = (\pi(h_r \| h_c), h_r)$. It has been proven by Bertoni et al. [42] that sponge functions are indifferentiable from a random oracle under the assumption that π behaves like a random permutation. In particular, a sponge function behaves like a random oracle for up to $O(2^{c/2})$ queries. By Theorem 2.3.6, this result carries over to the collision, preimage, and second preimage resistance of sponge functions.

In Chapter 6, we propose a generalization of the sponge hash function design.

2.5.3 Hash Trees

In contrast to the iterated hashing principle of Section 2.4, tree based hash function constructions allow for parallelism. Tree constructions split the message into blocks which can be processed by independent processors and the final result is combined to produce the hash value. They have the disadvantage of a larger state information that needs to be kept (logarithmic in the message length), but on the other hand allow for parallelism. Tree hashing was first proposed by Merkle [166], and many variants appeared in literature [35,149,150,200]. Merkle's construction is known to preserve collision security [19] but not preimage and second preimage security.

[4]In more detail, $T_i = \langle im \rangle_t$ if $im \le |M|$, $T_i = \langle |M| \rangle_t$ if $(i-1)m < |M| \le im$, and $T_i = \langle 0 \rangle_t$ if $|M| \le (i-1)m$. Here, m denotes the message block size.

Dodis et al. [87] and Bertoni et al. [39] analyzed the required properties of hash trees to obtain indifferentiable designs.

2.5.4 Other Constructions

We briefly mention other interesting constructions, that are not immediately related to the remainder of this dissertation.

The enveloped Merkle-Damgård construction of Bellare and Ristenpart [32] resembles the design of HMAC. It uses two initialization vectors iv and iv'. First, the Merkle-Damgård transform is evaluated on the message input using iv, and then a final compression function is evaluated on iv', the final state value, and remaining message bits. It can be interpreted as a generalized Merkle-Damgård design (Figure 2.2). It is proven to be indifferentiable from a random oracle if the underlying compression function is random. The CTFM results of Section 2.4.2 directly generalize, as the design can be seen as Merkle-Damgård with final transformation.

Merkle-Damgård with permutation of Hirose et al. [113] is similar to the traditional Merkle-Damgård design with the difference that an independent permutation π is evaluated on the state, before the last compression function evaluation. It is proven to be indifferentiable from a random oracle if the underlying compression function and permutation are random. The CTFM results of Section 2.4.2 directly generalize, as the design can be seen as Merkle-Damgård with final transformation.

The zipper hash of Liskov [152] consists of two sequential evaluations of the Merkle-Damgård design, with the difference that the second one is in reverse order. It is proven indifferentiable from a random oracle up to the birthday bound even if the compression functions in use are weak, i.e., they can be inverted and collisions can be found efficiently. Also the zipper hash can be considered as a generalized Merkle-Damgård design (Figure 2.2). The CTFM results of Section 2.4.2 directly generalize, therewith demonstrating the (asymptotic) optimality of the attacks deployed by Andreeva et al. [4].

2.6 Modes of Operation for Compression Functions

The usual approach of building compression functions is to base them on either block ciphers or permutations. In this section, we discuss these two approaches in more detail.

2.6.1 Block Cipher Based

The traditional recipe for the design of a cryptographic hash function is to base it on one or more block ciphers. Since the late 70s, this methodology has become the dominating approach in the area of hash function design and plenty of hash functions have been constructed accordingly (either explicitly or implicitly). The first example of a block cipher based compression function is due to Rabin [189], who suggested the compression function $\mathsf{F}(h,m) = \mathrm{DES}_m(h)$. In [188], Preneel, Govaerts, and Vandewalle (PGV) analyzed and classified the 64 most natural block cipher based $2n$-to-n-bit compression functions, and since, block cipher based hashing has been an active research area. By "PGVx" we denote the x^{th} type compression function of [188]. Black, Rogaway, and Shrimpton [52] formally proved twelve of the PGV-compression functions collision and preimage secure in the ideal cipher model, including the well-known Matyas-Meyer-Oseas (PGV1) [158], Miyaguchi-Preneel (PGV3), and Davies-Meyer (PGV5) [170] compression functions. All functions are shown to be differentiable from a random compression function by Kuwakado and Morii [136]. Simon [207] already demonstrated that no provably collision resistant hash function exists based on just a one-way permutation, and it was demonstrated by Hirose [110] that the security results of [52] do not hold beyond the ideal cipher model. In [48], Biryukov et al. present an attack on the Davies-Meyer compression function when instantiated with the AES [75] block cipher. The results of [52,188] have been generalized by [53,90,210].

The MD4 [195] and MD5 [194] hash functions are Davies-Meyer based, as are SHA-0, SHA-1, and SHA-2. Whirlpool is a hash function that employs the Miyaguchi-Preneel compression function. Two SHA-3 finalist hash functions use a block cipher based compression function: BLAKE [25] and Skein [93]. The BLAKE compression function shows similarities with (but is not equal to) PGV5, and Skein is based on the PGV1 compression function.

Compression functions of this form are known as single (block) length functions, meaning that the output size of the function equals the block size of the cipher. However, this has the disadvantage that when a larger compression function is needed, the underlying block cipher is required to be equally large. Yet, if we consider AES [75], it has a 128-bit message space only and is clearly not scalable.[5] A solution to this issue can be found in double (block) length hashing. Double length hashing is a well-established compression function construction with $2n$-bit output based only on n-bit block ciphers. The idea of double length hashing dates back to the work of Meyer and Schilling [169], with the introduction of the MDC-2 and MDC-4 compression functions in 1988. Double length hash functions have the obvious advantage over classical block cipher based functions that the same type of underlying primitive allows for a larger compression function. Yet, for double length compression functions it is harder to achieve optimal n-bit collision and $2n$-bit preimage security.

In this dissertation, we focus on the simplest and most-studied type of double length compression functions, namely functions that compress $3n$ to $2n$ bits. Those can be classified into two classes: compression functions that internally evaluate a $2n$-bit keyed block cipher $E : \mathbb{Z}_2^{2n} \times \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ (which we will call the DBL^{2n}

[5]We remark, though, that the original Rijndael specification supports larger message spaces too.

class), and ones that employ an n-bit keyed block cipher $E : \mathbb{Z}_2^n \times \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ (the DBL^n class).

The DBL^{2n} class is well understood. It includes the classical compression functions Tandem-DM and Abreast-DM [137], and Hirose's function [112] (see Figure 2.3), as well as Stam's supercharged single call Type-I compression function design [209,210] (reconsidered in [148]) and the generalized designs by Hirose [111] and Özen and Stam [182]. As illustrated in Table 2.1, all of these functions provide optimal collision security guarantees (up to about 2^n queries), and Tandem-DM, Abreast-DM, and Hirose's function are additionally proven optimally preimage resistant (up to about 2^{2n} queries). These bounds also hold in the iteration, when a proper domain extender is applied (see Section 2.4). Lucks [154] introduced a compression function that allows for collisions in about $2^{n/2}$ queries, but achieves optimal collision resistance in the iteration.

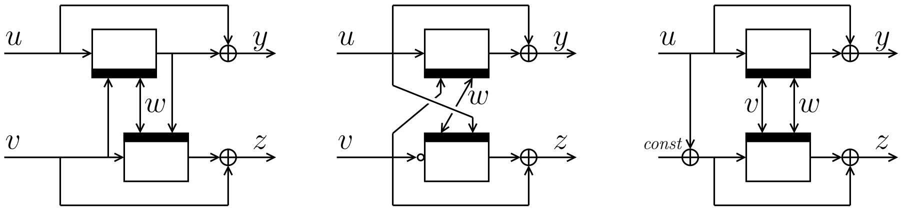


Figure 2.3. Left to right: Tandem-DM, Abreast-DM, and Hirose's compression function [112,137]. All wires carry n bits. For Abreast-DM, the circle $\circ$ denotes bit complementation. For Hirose's function, *const* is any non-zero constant.

Members of the DBL^n class are the MDC-2 and MDC-4 compression functions [169], the MJH construction [144], and a construction by Jetchev et al. [118]. For the MDC-2 and MJH compression functions, collisions and preimages can be found in about $2^{n/2}$ and 2^n queries, respectively.[6] As we prove in Chapter 3, the MDC-4 compression function achieves a higher level of collision and preimage resistance than MDC-2, but contrary to the other functions it makes four block cipher calls. In Chapter 3, we give a more detailed comparison between MDC-2 and MDC-4. Jetchev et al.'s construction makes two block cipher calls and achieves $2^{2n/3}$ collision security. Stam [209] also introduced a design based on two calls, and proved it optimally collision secure in a restricted security model where the adversary must fix its queries in advance. Therefore we did not include this design in the table. In Chapter 4, we propose a new family of double block length compression functions in the DBL^n class that make three calls to their underlying cipher.

[6]In the iteration collision resistance is proven up to $2^{3n/5}$ queries for MDC-2 [211] and $2^{2n/3}$ queries for MJH [144].

Further related results include the work of Nandi et al. [176], who presented a $3n$-to-$2n$-bit compression function making three calls to a $2n$-to-n-bit one-way function, achieving collision security up to $2^{2n/3}$ queries. They extended this result to a $4n$-to-$2n$-bit function using three $2n$-bit keyed block ciphers. Peyrin et al. [185] and Seurin and Peyrin [203] introduced and analyzed double length compression functions based on five compression function calls with n-bit output. Related are also Lucks wide-pipe design [153] and its generalization by Nandi [174]. Several $4n$-to-$2n$-bit compression functions using two block cipher calls appeared in literature, such as PBGV [187], Parallel DM [114], and LOKI DBH [63], all admitting collisions in about $2^{n/2}$ queries [126]. We also mention a compression function design by Knudsen and Preneel [128–130] based on linear error-correcting codes $[r, k, d]$. The idea is to use r cn-to-n-bit random functions (where c is usually small) in parallel to obtain a kcn-to-rn-bit compression function, where the inputs to the underlying primitives are derived via the code. This compression function design has subsequently been analyzed by Watanabe [222], Özen et al. [181, 183], and Lee [139].

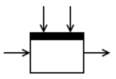
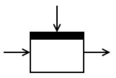
2.6.2 Permutation Based

Block cipher based hash functions are characterized by the fact that the key input to the cipher depends on the input values; this implies that the key schedule has to be strong and that it needs to be executed for every encryption call, which entails a substantial computational cost. An alternative approach is to fix one or more keys, and restrict the hash function design to use the block cipher for these keys only. The usage of fixed-key block ciphers, or alternatively *permutations*, additionally offers the benefit that one does not need to implement an entire block cipher but only a limited number of instantiations of it.

Black, Cochran, and Shrimpton [50, 51] were the first to formally study this approach, demonstrating that a $2n$-to-n-bit compression function F using one n-bit permutation π cannot be secure. Indeed, each input to π corresponds on average to 2^n inputs to F, and two queries to π thus allow for the computation of 2^{n+1} compression function executions, resulting in a collision for F with certainty. This result has been generalized by Rogaway and Steinberger [199], and refined by Stam [209], Steinberger [212], and Steinberger, Sun, and Yang [213]. Consider any mn-to-rn-bit compression function using k n-bit permutations. Then, collisions can be found in at most $(2^n)^{1-(m-r+1)/(k+1)}$ queries to the underlying primitives. Although this bound is proven by Steinberger et al. [212, 213], it is commonly known as "Stam's bound." Collisions and preimages can even be found in at most $(2^n)^{1-(m-r/2)/k}$ and $(2^n)^{1-(m-r)/k}$ queries respectively, provided the compression function satisfies the "uniformity assumption" [199].

The permutation based hashing approach was followed by three finalists of the SHA-3 hash function competition. The hash functions JH [224] and Keccak [40] internally use one permutation as large as the internal state. It is clear that their compression functions allow for collisions and preimages in a constant number

Table 2.1. Asymptotic ideal cipher model security guarantees of double length compression functions in the classes DBL^{2n} (first) and DBL^{n} (second). A more detailed security and efficiency comparison of some of these functions is presented by Bos et al. [58, Appendix A]. The results printed in **bold** are derived in this work (Chapters 3 and 4).

	E-calls	collision security	preimage security	underlying cipher
Lucks	1	$2^{n/2}$	2^{n}	
Stam	1	2^{n} [210]	2^{n} [210]	
Tandem-DM	2	2^{n} [146]	2^{2n} [22,147]	
Abreast-DM	2	2^{n} [100,142]	2^{2n} [22,147]	
Hirose	2	2^{n} [112]	2^{2n} [22,147]	
Hirose's class	2	2^{n} [111]	2^{n} [111]	
Özen-Stam's class	2	2^{n} [182]	2^{n} [182]	
MDC-2	2	$2^{n/2}$	2^{n}	
MJH	2	$2^{n/2}$	2^{n}	
Jetchev et al.	2	$2^{2n/3}$ [118]	2^{n} [118]	
Our class	**3**	$\mathbf{2^{n}}$	$\mathbf{2^{3n/2}}$	
MDC-4	4	$\mathbf{2^{5n/8}}$	$\mathbf{2^{5n/4}}$	

of queries, as also dictated by the generic bounds, but we note that the hash functions do not suffer from this compression function property. Grøstl [104] uses a compression function based on two permutations, and it is proven to achieve tight collision and preimage security up to the generic bounds of [199]. For more details, we refer to Section 7.2.

In this dissertation, we focus on permutation based $2n$-to-n-bit compression functions. Due to Stam's bound, such a construction achieves optimal $2^{n/2}$ collision resistance *only if* it employs at least three permutations. Yet, it cannot achieve optimal preimage resistance if it fulfills the uniformity assumption. These observations apply to the "multi-permutation setting," where each of the permutations is generated independently, as well as the "single-permutation setting" where the permutations are the same.

The construction of $2n$-to-n-bit compression functions (based on three permutations) that provably attain optimal collision security, has turned out to be a very challenging exercise. In [198], Rogaway and Steinberger formally proved a broad class of $2n$-to-n-bit compression functions using three distinct permutations and finite field scalar multiplications optimally collision and preimage secure (with respect to the bounds of [199]), provided the compression function satisfies a so-called "independence criterion" (a similar result for the single-permutation setting has been obtained by Lee and Kwon [141]). Unfortunately, this technical criterion rules out compression functions that are (apart from the three permutations) solely based on XOR operators. As the proof of [198] extensively relies on its independence criterion, it cannot be generalized to compression functions of this type. In [206], Shrimpton and Stam derived a XOR based compression function, using three one-way functions rather than permutations: $\mathsf{F}(x_1, x_2) = f_1(x_1) \oplus f_3(f_1(x_1) \oplus f_2(x_2))$. This function is proven collision resistant up to $2^{n/2}$ queries (asymptotically), but preimages can be found with high probability after $2^{n/2}$ queries [206]. It has been demonstrated by an automated analysis of Rogaway and Steinberger [198] that the same results hold if f_1, f_2, f_3 are Davies-Meyer-like compression functions using permutations π_1, π_2, π_3, i.e., $f_i(x) = x \oplus \pi_i(x)$, but a formal security analysis has never been given. In Chapter 5, we propose a new family of $2n$-to-n-bit compression functions based on three permutations with optimal security up to the bounds of [199].

Regarding compression functions with different values m, k, r, we point out the following results. In [27], Bagheri et al. describe attacks against several compression functions making two permutation calls. Lee and Park derive preimage resistance results for classes of compression functions with $r = m - 1$ [143].

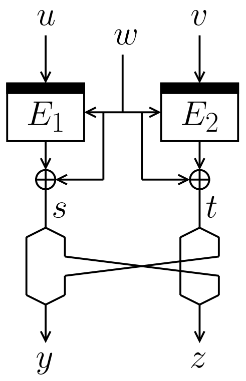
3 Collision and Preimage Security of MDC-4

This chapter studies the classical block cipher based hash function MDC-4. MDC-4 and its related hash function MDC-2 have first been described by Meyer and Schilling in 1988 [169], and have been patented by Brachtl et al. in 1990 [60]. MDC-2 has been standardized in ISO/IEC 10118-2 [117] and is used in numerous applications (see [127, 211] for an exposition), while MDC-4 is part of the IBM CLiC cryptographic module [95, 98]. In their original specification, MDC-2 and MDC-4 are instantiated using the block cipher DES. In this dissertation, we step away from this design criterion and consider the designs based on any block cipher $E : \mathbb{Z}_2^n \times \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ with key and message length n bits (throughout, the first input to the block cipher is the key input).

MDC-2 is a Merkle-Damgård hash function design (Section 2.4) with a $2n$-bit state using a compression function $f_{\text{MDC-2}} : \mathbb{Z}_2^{3n} \to \mathbb{Z}_2^{2n}$ that internally calls the block cipher E twice. It additionally employs two mappings β and γ, applied on the key inputs to the two block cipher calls. As β and γ are originally constructed so as to have a distinct range,[1] we can consider MDC-2 to be based on two block ciphers $E_1(\cdot,\cdot) = E(\beta(\cdot),\cdot)$ and $E_2(\cdot,\cdot) = E(\gamma(\cdot),\cdot)$ [211]. (We note that two such block ciphers can easily be constructed from one block cipher $E : \mathbb{Z}_2^{n+1} \times \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ by domain separation, where we set $E_1(\cdot,\cdot) = E(0\|\cdot,\cdot)$ and $E_2(\cdot,\cdot) = E(1\|\cdot,\cdot)$.) The definition of the compression function $f_{\text{MDC-2}} : \mathbb{Z}_2^{3n} \to \mathbb{Z}_2^{2n}$ is given in Figure 3.1.

Here and throughout this chapter, we assume n is even. $f_{\text{MDC-2}}$ can be considered as a parallel evaluation of two Matyas-Meyer-Oseas (PGV1) constructions followed by a swapping of the right halves of both states. Consequently, $f_{\text{MDC-2}}$ does not achieve the desired level of security: finding a collision or a preimage for $f_{\text{MDC-2}}$ is as hard as finding it for the two MMO constructions independently, hence requires about $2^{n/2}$ or 2^n block cipher calls, respectively. For the full MDC-2 hash function, Knudsen et al. [127] demonstrated that roughly $2^n/n$ block cipher calls suffice for finding a collision, and about 2^n

[1]The original specification (using block cipher DES) defines β and γ as mappings from the ciphertext space to the key space, where the parity bits are omitted, and the second and third bits are set to 10 and 01, respectively.

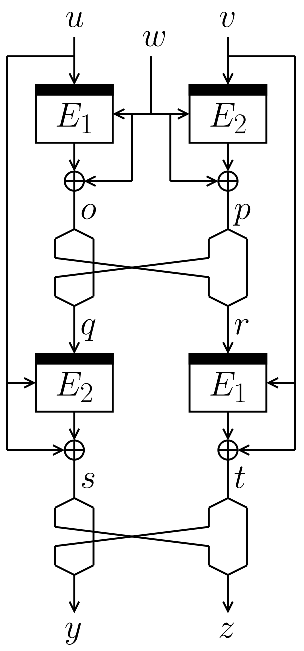


```
$f_{\text{MDC-2}}(u, v, w)$
    $s \leftarrow E_1(u, w) \oplus w$ ,
    $t \leftarrow E_2(v, w) \oplus w$ ,
    $y \leftarrow s^l \| t^r$ ,
    $z \leftarrow t^l \| s^r$ ,
    **return** $(y, z)$ .
```

Figure 3.1. The MDC-2 compression function $f_{\text{MDC-2}}$. For convenience, we swapped the left and right block ciphers of the second $f_{\text{MDC-2}}$ evaluation.

calls for finding a preimage. In 2007, Steinberger derived a non-trivial security lower bound on MDC-2 [211]. Steinberger considers the MDC-2 hash function using one single block cipher E modeled as an ideal cipher, and proves that any adversary with query access to E requires at least $2^{3n/5}$ queries (asymptotically) to find a collision. His proof relies on the observation that a collision for MDC-2 implies a collision for the last two rounds of MDC-2, except for some special cases. These results on MDC-2 are summarized in Table 3.1, all of which hold for the case E_1 and E_2 are different block ciphers as for the case they are the same.

The MDC-4 hash function differs from MDC-2 in the sense that its underlying compression function $f_{\text{MDC-4}}$ makes four calls to the underlying block cipher rather than two. $f_{\text{MDC-4}}$ is defined as two consecutive evaluations of $f_{\text{MDC-2}}$, where the message inputs to the MMO executions in the second evaluation are v for E_1 and u for E_2. The definition of $f_{\text{MDC-4}} : \mathbb{Z}_2^{3n} \to \mathbb{Z}_2^{2n}$ is given in Figure 3.2. Knudsen and Preneel [129] showed that approximately $2^{3n/4}$ block cipher executions suffice to find a collision for $f_{\text{MDC-4}}$. With respect to preimage resistance, the same authors report that $2^{3n/2}$ calls suffice for finding a preimage for $f_{\text{MDC-4}}$ and $2^{7n/4}$ calls result in a preimage for MDC-4. The latter result is recently improved to $2^{3n/2}$ by Hong and Kwon [116]. These results are summarized in Table 3.1. In independent concurrent research, Fleischmann et al. [98, 99] analyzed the collision resistance of MDC-4. They proved that MDC-4 is collision secure up to approximately $2^{3n/5}$ queries. Hence, they prove that the bound of Steinberger for MDC-2 also holds for MDC-4 (in fact, numerically their bound is worse than the bound on MDC-2). A preimage security lower bound has so far never been derived. Since the introduction of the functions in [60, 169], however, the MDC-4 hash function has always been considered the more secure variant of MDC-2. Although mainly a matter of theoretical interest (given that MDC-2 is twice as fast as MDC-4),



```
$f_{\text{MDC-4}}(u, v, w)$
    $o \leftarrow E_1(u, w) \oplus w$ ,
    $p \leftarrow E_2(v, w) \oplus w$ ,
    $q \leftarrow p^l \| o^r$ ,
    $r \leftarrow o^l \| p^r$ ,
    $s \leftarrow E_2(q, u) \oplus u$ ,
    $t \leftarrow E_1(r, v) \oplus v$ ,
    $y \leftarrow t^l \| s^r$ ,
    $z \leftarrow s^l \| t^r$ ,
    **return** $(y, z)$ .
```

Figure 3.2. The MDC-4 compression function $f_{\text{MDC-4}}$. For convenience, we swapped the left and right block ciphers of the second $f_{\text{MDC-2}}$ evaluation.

formal security lower bounds for MDC-4 or $f_{\text{MDC-4}}$ that confirm this longstanding claim have never been derived. In particular, for years the MDC-4 structure has not been thoroughly analyzed, which makes it impossible to classify MDC-4 among others double block length constructions known in literature (see Section 2.6.1).

Contributions of This Chapter

We formally analyze the collision and (everywhere) preimage security of MDC-4 and its underlying $f_{\text{MDC-4}}$ compression function of Figure 3.2. We start with the generalized setting where $f_{\text{MDC-4}}$ is built on one block cipher $E = E_1 = E_2$, which we call the single block cipher setting. Then, we consider the more double block cipher setting, where MDC-4 is based on two independently distributed ciphers E_1 and E_2.

Single block cipher setting. A customary approach for proving collision and preimage resistance of a hash function is to analyze the compression function first and use a preservation result (Section 2.4) to show that the findings also apply to the full hash function. However, when the two block ciphers underlying MDC-4 are

Table 3.1. Security results for the MDC-2 and MDC-4 compression and hash functions. The security bound gives a lower bound and the attack bound gives an upper bound on the number of queries in order to find an attack. By "(triv)" we note that the bound is trivial; these bounds come from the security of the MMO construction [52,158,188], or correspond to generic attacks. The results printed in **bold** are derived in this chapter.

	collision		preimage		ideal primitives
	security	attack	security	attack	
$f_{\text{MDC-2}}$	$2^{n/2}$ (triv)	$2^{n/2}$ (triv)	2^n (triv)	2^n (triv)	E or (E_1, E_2)
MDC-2	$2^{3n/5}$ [211]	$2^n/n$ [127]	2^n (triv)	2^n [127]	
$f_{\text{MDC-4}}$	$2^{n/2}$ (triv)	$2^{n/2}$ (triv)	2^n (triv)	2^n (triv)	E
MDC-4	$\mathbf{2^{5n/8}}$	2^n (triv)	2^n (triv)	$\mathbf{2^n}$	
$f_{\text{MDC-4}}$	$\mathbf{2^{5n/8}}$	$2^{3n/4}$ [129]	$\mathbf{2^{5n/4}}$	$2^{3n/2}$ [129]	(E_1, E_2)
MDC-4	$\mathbf{2^{5n/8}}$	2^n (triv)	$\mathbf{2^{5n/4}}$	$2^{3n/2}$ [116]	

modeled as one single random block cipher E, collisions and preimages for $f_{\text{MDC-4}}$ can be found in at most $2^{n/2}$ and 2^n block cipher calls, respectively: one focuses on state values with the same left and right half (for Figure 3.2 this means $u = v$ and $y = z$, and consequently $o = p$, $q = r$, and $s = t$), and the security boils down to the security of two subsequent MMO evaluations. For the preimage resistance, these type of target images (with $y = z$) can be considered as weak images, these give the adversary significantly more power. In any case, the security preservation approach does not help us out here, and instead we consider the security of the MDC-4 hash function design directly.

Starting with collision resistance, we formally prove that any adversary with query access to E, modeled as an ideal cipher, needs at least $2^{5n/8}$ queries (asymptotically) to find a collision for MDC-4. This is significantly beyond the collision bound $2^{3n/5}$ on MDC-2 by Steinberger [211] and on MDC-4 concurrently derived by Fleischmann et al. [98, 99]. The proof consists of two parts: given that the initial value of MDC-4 consists of two different halves, we prove that all intermediate state values of a MDC-4 evaluation consist of different halves, except with a small probability. Then, it suffices to analyze collision resistance of $f_{\text{MDC-4}}$ restricted to state values with *different* left and right halves, which we prove up to at least $2^{5n/8}$ queries. At the first glance, as $f_{\text{MDC-4}}$ roughly consists of two evaluations of $f_{\text{MDC-2}}$, one is inclined to say the results of Steinberger [211] directly carry over. However, this is not true due to the differences at the second $f_{\text{MDC-2}}$ evaluation where the inputs are swapped and the message inputs differ for the left and right cipher. Instead, we conduct a new collision resistance proof for $f_{\text{MDC-4}}$, and although it shows some similarities with the proof of Steinberger, it differs in many aspects and uses several new ideas to facilitate the analysis.

For (everywhere) preimage resistance, we derive a more surprising result, namely that if the target image (y, z) satisfies $y = z$, a preimage for the MDC-4 hash function can be found in approximately 2^n queries. The attack resembles ideas from the preimage attack on MDC-2 by Knudsen et al. [127] and from above-described preimage attack on $f_{\text{MDC-4}}$ in the single block cipher setting. We stress that the attack does not apply to the original MDC-4 design due to the domain separation by the functions β and γ. On the other hand, if the target image satisfies $y \neq z$, we prove that any adversary requires at least $2^{5n/4}$ queries (asymptotically) to find a preimage for $f_{\text{MDC-4}}$ or MDC-4, hence security beyond the birthday bound is achieved. In order to achieve security beyond 2^n queries, we employ the ideas of free queries and wish lists. These proof tools have been used before by Armknecht et al. and Lee et al. [22, 147] for compression functions based on two block cipher calls (see Section 2.6.1), but because MDC-4 makes four block cipher calls rather than two, and additionally the corresponding wish lists are much harder to bound, the security proof has become considerably more complex. We remark that target images with the same left and right half are rather rare, 2^n out of 2^{2n} target images satisfy this property. If we had opted for preimage resistance where the challenge is randomly generated, we obtain in the single block cipher setting a security bound of approximately $2^{5n/4}$ queries for $f_{\text{MDC-4}}$ and MDC-4, rather than the bound of 2^n queries.

The findings on MDC-4 based on one block cipher E are included in Table 3.1.

Double block cipher setting. As a second part of this chapter, we consider the security of the MDC-4 design where the two block ciphers E_1 and E_2 are modeled as two independent ideal ciphers. We note that, despite the stronger underlying assumption, this model is closer to the original design due to the domain separation by the functions β and γ.

We show that in this model, any adversary with query access to E_1 and E_2, needs at least $2^{5n/8}$ queries (asymptotically) to find a collision for $f_{\text{MDC-4}}$ and at least $2^{5n/4}$ queries (asymptotically) to find a preimage for $f_{\text{MDC-4}}$. These results almost immediately follow from the ones in the single block cipher setting. Note that here we do not make the restriction that the state values should consist of different halves. Because MDC-4 is a Merkle-Damgård transform, it preserves collision and (everywhere) preimage resistance (Section 2.4). Therefore, these results for $f_{\text{MDC-4}}$ directly carry over to the MDC-4 hash function.

Outline

In Section 3.1, we present some mathematical preliminaries complementary to Chapter 2. The security result on the collision resistance of MDC-4 (based on one block cipher $E = E_1 = E_2$) is given in Section 3.2, along with its formal security proof. In Section 3.3, we present our security result on the preimage resistance of MDC-4 (based on E). In Section 3.4, we discuss the implications of these results to the MDC-4 design based on two block ciphers E_1, E_2. The chapter is concluded in Section 3.5.

Bibliographic Notes

The contents of this chapter have been published by Mennink in Designs, Codes and Cryptography [164].

3.1 Security Model

Throughout this chapter, we consider $\mathcal{P} = (E_1, E_2) \xleftarrow{\$} \mathrm{Bloc}(n)^2$ or $\mathcal{P} = E \xleftarrow{\$} \mathrm{Bloc}(n)$, and it is clear from the context which setting we are in. We consider an adversary which attacks MDC-4 or $f_{\text{MDC-4}}$. Throughout, we denote the initial state value of MDC-4 by (g_0, h_0), where $g_0 \neq h_0$. If the adversary attacks MDC-4, its goal is to find either a collision MDC-4(M_1) = MDC-4(M_2) or a preimage MDC-4$(M) = (y, z)$ for a range value (y, z) chosen in advance. If it attacks $f_{\text{MDC-4}}$, its goal is to find either a collision $f_{\text{MDC-4}}(u_1, v_1, w_1) = f_{\text{MDC-4}}(u_2, v_2, w_2)$ or a preimage $f_{\text{MDC-4}}(u, v, w) = (y, z)$ for an in advance chosen range value (y, z). The adversary can make forward and inverse queries to its oracles, and these are stored in query history $\mathcal{Q}$ as indexed tuples of the form (k_i, K_i, m_i, c_i), where i is the index of the query, $k_i \in \{1, 2\}$ indicates the corresponding block cipher, K_i is the key input, and m_i and c_i denote the (plaintext) input and (ciphertext) output of the block cipher. By $m_i \oplus c_i$, we define its XOR-output. The index i and the parameter k are omitted if they are irrelevant or clear from the context.

Regarding the collision resistance definition, by $\mathbf{Adv}^{\mathrm{col}(\neq)}_{f_{\text{MDC-4}}}(q)$ we denote $\mathbf{Adv}^{\mathrm{col}}_{f_{\text{MDC-4}}}(q)$ with the restriction that the state values (u_1, v_1) and (u_2, v_2) should satisfy $u_1 \neq v_1$ and $u_2 \neq v_2$. For preimage resistance, for $\mathcal{C} \in \{\text{MDC-4}, f_{\text{MDC-4}}\}$, by $\mathbf{Adv}^{\mathrm{epre}(\neq)}_{\mathcal{C}}(q)$ we denote $\mathbf{Adv}^{\mathrm{epre}}_{\mathcal{C}}(q)$ restricted to target images (y, z) with $y \neq z$.

3.2 Collision Resistance of MDC-4

We derive a collision security lower bound for MDC-4 in the case the two underlying block ciphers are identical, i.e., $E = E_1 = E_2$. Due to the attack on $f_{\text{MDC-4}}$ (described in the introduction of this chapter), the classical way of proving collision resistance by ways of property preservation does not work here. Therefore, the security analysis is done in a slightly different way. Recall that the attack on $f_{\text{MDC-4}}$ relies on the property that the block cipher evaluations on the left and right sides of Figure 3.2 can be the same. The proof is now roughly divided into two parts: we first prove that, restricted to states with *different* halves, $f_{\text{MDC-4}}$ is collision resistant up to at least approximately $2^{5n/8}$ block cipher queries. Next, we show how this result can be used to prove collision resistance of the full MDC-4.

Theorem 3.2.1. *Let $n \in \mathbb{Z}_2^n$. Then, for any positive integral values τ_1, τ_2, τ_3,*

$$\begin{aligned}\mathbf{Adv}^{\mathrm{col}(\neq)}_{f_{\text{MDC-4}}}(q) \leq \frac{(\tau_1 + 11\tau_1\tau_2 + 3\tau_1\tau_2\tau_3 + 12\tau_2 + 4\tau_2^2)q}{2^n - q} + \frac{q^2}{\tau_1(2^n - q)} + \\ 2 \cdot 2^{n/2}\left(\frac{eq2^{n/2}}{\tau_2(2^n - q)}\right)^{\tau_2} + 2^n\left(\frac{eq}{\tau_3(2^n - q)}\right)^{\tau_3}.\end{aligned} \tag{3.1}$$

The proof of Theorem 3.2.1 is given in Section 3.2.1. It shows similarities with the proof by Steinberger [211] of collision resistance of the MDC-2 hash function, but its structure is entirely different so as to facilitate the proof.

Employing this result, we now obtain the main result for the collision resistance of MDC-4 using a single block cipher E.

Theorem 3.2.2. *Let $n \in \mathbb{Z}_2^n$. Then, for any positive integral values τ_1, τ_2, τ_3,*

$$\begin{aligned}\mathbf{Adv}^{\mathrm{col}}_{\text{MDC-4}}(q) \leq \frac{(2\tau_1 + 11\tau_1\tau_2 + 3\tau_1\tau_2\tau_3 + 14\tau_2 + 5\tau_2^2)q}{2^n - q} + \frac{(\tau_1 + 2\tau_2 + \tau_2^2)q}{2^n - q} + \\ \frac{q^2}{\tau_1(2^n - q)} + 2 \cdot 2^{n/2}\left(\frac{eq2^{n/2}}{\tau_2(2^n - q)}\right)^{\tau_2} + 2^n\left(\frac{eq}{\tau_3(2^n - q)}\right)^{\tau_3}.\end{aligned} \tag{3.2}$$

The proof of Theorem 3.2.2 is given in Section 3.2.2, and we give a brief intuition. Recall that the attack on $f_{\text{MDC-4}}$ relies on the fact that the two halves of the evaluation can be the same. However, for a full MDC-4 iteration, the initial state value consists of two different halves, and in fact all intermediate state values consist of two different halves, except with some small probability. Now, by ways of collision resistance preservation (Theorem 2.4.1), the result of Theorem 3.2.1 carries over with as additional term a bound on the probability that the adversary ends up with a state value with two the same halves.

One can divide the bound of (3.2) into two parts. The first term forms the first part and increases for increasing parameters τ_1, τ_2, τ_3. The remaining three terms form the second part that decreases for increasing τ_1, τ_2, τ_3. For obtaining a sharp bound, we need to fine tune the integral positive parameters τ_1, τ_2, τ_3. The trick is to take parameters τ_1, τ_2, τ_3 minimal so that the second part of (3.2) still goes to 0 for $n \to \infty$. We will show that the advantage of any adversary making slightly less than $2^{5n/8}$ queries approaches 0 when n goes to infinity. To this end, let $\varepsilon > 0$ be any parameter, we consider any adversary making at most $q = 2^{5n/8}/n^{\varepsilon}$ queries

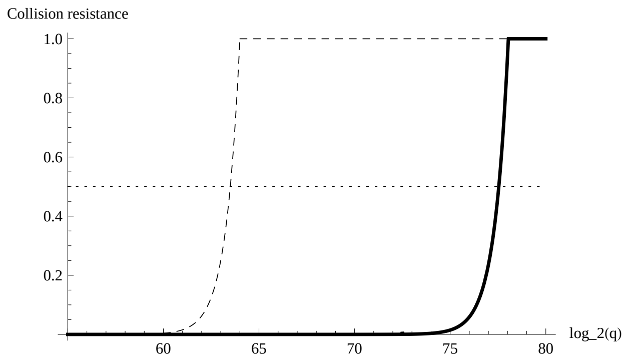


Figure 3.3. The function $\mathbf{Adv}^{\mathrm{col}}_{\text{MDC-4}}(q)$ of (3.2) for $n = 128$ and the particular choice of values τ_1, τ_2, τ_3 (solid line), in comparison with the trivial bound $q(q+1)/2^n$ (dashed line).

to its oracle. We set $\tau_1 = 2^{2n/8}$, $\tau_2 = 2^{n/8}$, and $\tau_3 = 3$. Here, for simplicity we assume τ_1, τ_2 to be integral. If n is no multiple of 8, one sets τ_1, τ_2 to the nearest integers. Note that these parameters satisfy $\tau_1 > \frac{q^2}{2^n}$, $\tau_2 > \frac{q}{2^{n/2}}$ and $\tau_3 > \frac{q}{2^n}$, which are minimal requirements for the second part of (3.2) to be < 1. Now, it is an easy exercise to verify that the bound of (3.2) approaches 0 for $n \to \infty$ when $q = 2^{5n/8}/n^{\varepsilon}$ and τ_1, τ_2, τ_3 are as specified.

Corollary 3.2.3. *For any $\varepsilon > 0$, we obtain* $\lim_{n\to\infty} \mathbf{Adv}^{\mathrm{col}}_{\text{MDC-4}}\left(2^{5n/8}/n^{\varepsilon}\right) = 0$.

The result means that for $n \to \infty$ the function $\mathbf{Adv}^{\mathrm{col}}_{\text{MDC-4}}$ behaves as $q^8/2^{5n}$. A graphical representation of $\mathbf{Adv}^{\mathrm{col}}_{\text{MDC-4}}$ for $n = 128$ is given in Figure 3.3. In this graph, where we have slightly adjusted the parameters τ_1, τ_2 to facilitate the analysis for smaller n and smaller q (the previously chosen values were set to analyze limiting behavior for n, q), we see a significant improvement over the best known bound and the bound independently derived in [98, 99]. For $n = 128$ the collision resistance advantage hits $1/2$ for $\log_2 q \approx 77.5$, which is smaller than the threshold for $q^8/2^{5n}$, 79.9. For larger values of n, by Corollary 3.2.3 the difference goes to 0 for $n \to \infty$.

3.2.1 Proof of Theorem 3.2.1

The collision resistance proof shows some similarities with the proof of Steinberger for MDC-2 [211], but fundamentally differs in various aspects and is as such of independent interest. In particular, due to a different and more structured case distinction we obtain a sharper bound (security up to $2^{5n/8}$ queries) than the bound of Steinberger for MDC-2 (security up to $2^{3n/5}$ queries). Also, our proof significantly improves over the proof by Fleischmann et al. [98, 99], who basically confirmed that the $2^{3n/5}$ bound of MDC-2 applies to MDC-4 too.

We consider any adversary making q queries to its oracle E, which tries to find a collision for $f_{\text{MDC-4}}$. Finding a collision corresponds to obtaining a query history $\mathcal{Q}_q$ of size q that satisfies configuration $\mathsf{col}(\mathcal{Q}_q)$ of Figure 3.4. In other words,

$$\mathbf{Adv}^{\mathrm{col}(\neq)}_{f_{\text{MDC-4}}}(q) = \mathbf{Pr}\left(\mathsf{col}(\mathcal{Q}_q)\right) , \tag{3.3}$$

and we consider the probability of obtaining any query history $\mathcal{Q}_q$ that satisfies configuration $\mathsf{col}(\mathcal{Q}_q)$. Notice that in this configuration, we omit the shifting at the end: as this shifting is bijective, it does not influence the collision-finding advantage. In Figure 3.4, as well as in all subsequent figures in this section, we label the block ciphers as follows to uniquely identify their positions. In the left word of Figure 3.4 (with inputs (u_1, v_1, w_1)) the block ciphers are labeled 1tl, 1tr, 1bl, 1br, for top/bottom left/right. For the right word the block ciphers are identified as 2tl, 2tr, 2bl, 2br. In the remainder, when talking about "a query 1tl," we refer to a query that in a collision occurs at position 1tl (and the same for the other positions). The variables in the figures may take any value, and are simply used to accentuate relations among the two words.

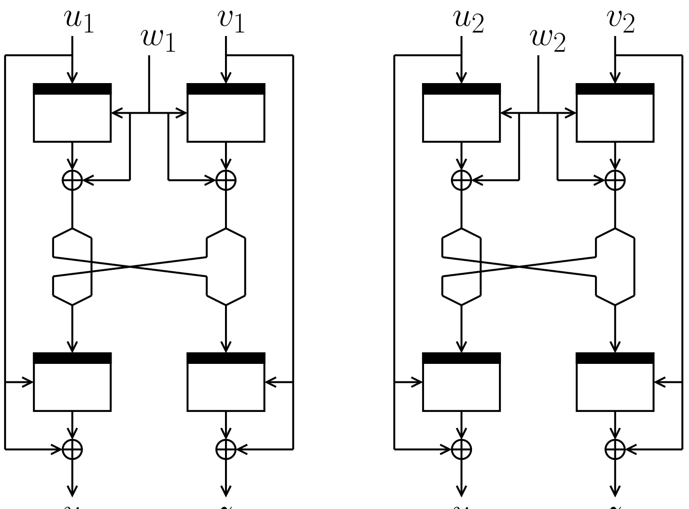


Figure 3.4. Configuration $\mathsf{col}(\mathcal{Q})$. The configuration is satisfied if $\mathcal{Q}$ contains eight (possibly not all different) queries that satisfy this setting. We require $(u_1, v_1, w_1) \neq (u_2, v_2, w_2)$, $u_1 \neq v_1$, and $u_2 \neq v_2$.

We need to evaluate the probability of the adversary finding a query history $\mathcal{Q}_q$ that satisfies configuration $\mathsf{col}(\mathcal{Q}_q)$ of Figure 3.4. For this analysis we introduce a helping event $\mathsf{help}(\mathcal{Q}_q)$. Let $\tau_1, \tau_2, \tau_3 > 0$ be integral. Event $\mathsf{help}(\mathcal{Q}_q)$ is satisfied if either of the following sub-events $\mathsf{help}_k(\mathcal{Q}_q)$ $(k = 1, \ldots, 4)$ occurs:

$$\begin{aligned}
&\mathsf{help}_1(\mathcal{Q}_q): && \left|\left\{(K_i, m_i, c_i), (K_j, m_j, c_j) \in \mathcal{Q}_q \;\middle|\; i \neq j \;\wedge\; m_i \oplus c_i = m_j \oplus c_j\right\}\right| > \tau_1 \,;\\
&\mathsf{help}_2(\mathcal{Q}_q): && \max_{x \in \mathbb{Z}_2^{n/2}} \left|\left\{(K_i, m_i, c_i) \in \mathcal{Q}_q \;\middle|\; (m_i \oplus c_i)^l = x\right\}\right| > \tau_2 \,;\\
&\mathsf{help}_3(\mathcal{Q}_q): && \max_{x \in \mathbb{Z}_2^{n/2}} \left|\left\{(K_i, m_i, c_i) \in \mathcal{Q}_q \;\middle|\; (m_i \oplus c_i)^r = x\right\}\right| > \tau_2 \,;\\
&\mathsf{help}_4(\mathcal{Q}_q): && \max_{x \in \mathbb{Z}_2^{n}} \left|\left\{(K_i, m_i, c_i) \in \mathcal{Q}_q \;\middle|\; m_i \oplus c_i = x\right\}\right| > \tau_3 \,.
\end{aligned}$$

By basic probability theory, we obtain for (3.3):

$$\mathbf{Pr}\left(\mathsf{col}(\mathcal{Q}_q)\right) \leq \mathbf{Pr}\left(\mathsf{col}(\mathcal{Q}_q) \wedge \neg\mathsf{help}(\mathcal{Q}_q)\right) + \mathbf{Pr}\left(\mathsf{help}(\mathcal{Q}_q)\right) \,. \tag{3.4}$$

For the analysis of the event $\mathsf{col}(\mathcal{Q}_q)$, it may be the case that a single query occurs at multiple positions in the configuration. Therefore, we divide $\mathsf{col}(\mathcal{Q}_q)$ into sub-configurations. For two distinct positions $a, b \in \{\mathsf{1tl}, \mathsf{1tr}, \mathsf{1bl}, \mathsf{1br}, \mathsf{2tl}, \mathsf{2tr}, \mathsf{2bl}, \mathsf{2br}\}$ and a binary value $\alpha \in \{0, 1\}$, by $a = b \equiv \alpha$ we say that the same query occurs at both positions a and b if and only if $\alpha = 1$. Now, we define for $\alpha_{\mathsf{tl}}, \alpha_{\mathsf{tr}}, \alpha_{\mathsf{bl}}, \alpha_{\mathsf{br}} \in \{0, 1\}$ the sub-configuration $\mathsf{col}_{\alpha_{\mathsf{tl}}\alpha_{\mathsf{tr}}\alpha_{\mathsf{bl}}\alpha_{\mathsf{br}}}(\mathcal{Q})$ as $\mathsf{col}(\mathcal{Q})$ of Figure 3.4 with the restriction that

$$\mathsf{1tl} = \mathsf{2tl} \equiv \alpha_{\mathsf{tl}}\,, \qquad \mathsf{1tr} = \mathsf{2tr} \equiv \alpha_{\mathsf{tr}}\,, \qquad \mathsf{1bl} = \mathsf{2bl} \equiv \alpha_{\mathsf{bl}}\,, \qquad \mathsf{1br} = \mathsf{2br} \equiv \alpha_{\mathsf{br}}\,.$$

It may be the case that the same query occurs at positions $\mathsf{1tl}$ and $\mathsf{1br}$ or $\mathsf{2br}$, but as becomes clear these cases are included in the analysis. By construction, $\mathsf{col}(\mathcal{Q}_q) \Rightarrow \bigvee_{\alpha_{\mathsf{tl}},\alpha_{\mathsf{tr}},\alpha_{\mathsf{bl}},\alpha_{\mathsf{br}} \in \{0,1\}} \mathsf{col}_{\alpha_{\mathsf{tl}}\alpha_{\mathsf{tr}}\alpha_{\mathsf{bl}}\alpha_{\mathsf{br}}}(\mathcal{Q}_q)$, and from (3.3-3.4) we obtain the following bound on $\mathbf{Adv}^{\mathrm{col}(\neq)}_{f_{\text{MDC-4}}}(q)$:

$$\mathbf{Adv}^{\mathrm{col}(\neq)}_{f_{\text{MDC-4}}}(q) \leq \sum_{\substack{\alpha_{\mathsf{tl}},\alpha_{\mathsf{tr}},\alpha_{\mathsf{bl}},\\ \alpha_{\mathsf{br}} \in \{0,1\}}} \mathbf{Pr}\left(\mathsf{col}_{\alpha_{\mathsf{tl}}\alpha_{\mathsf{tr}}\alpha_{\mathsf{bl}}\alpha_{\mathsf{br}}}(\mathcal{Q}_q) \wedge \neg\mathsf{help}(\mathcal{Q}_q)\right) + \mathbf{Pr}\left(\mathsf{help}(\mathcal{Q}_q)\right) \,. \tag{3.5}$$

The probabilities constituting to the sum of (3.5) are analyzed in Lemmas 3.2.4-3.2.9 as further set forth in Table 3.2. Probability $\mathbf{Pr}\left(\mathsf{help}(\mathcal{Q}_q)\right)$ is analyzed in Lemma 3.2.10.

Lemma 3.2.4. $\mathbf{Pr}\left(\mathsf{col}_{0000}(\mathcal{Q}_q) \wedge \neg\mathsf{help}(\mathcal{Q}_q)\right) \leq \frac{(\tau_1\tau_2\tau_3 + \tau_1\tau_2 + 2\tau_2)q}{2^n - q}$.

Proof. A visualization of configuration $\mathsf{col}_{0000}(\mathcal{Q}_q)$ can be found in Figure 3.5. In this figure, the queries corresponding to locations a and $!a$ are required to be different, and the same for the queries at positions $(b, !b)$, $(c, !c)$, and $(d, !d)$. For the analysis of $\mathsf{col}_{0000}(\mathcal{Q}_q) \wedge \neg\mathsf{help}(\mathcal{Q}_q)$, we say that the i^{th} query $(i \in \{1, \ldots, q\})$ is successful if it makes configuration $\mathsf{col}_{0000}(\mathcal{Q}_i)$ satisfied and $\neg\mathsf{help}(\mathcal{Q}_i)$ holds. Now, by basic probability theory, we can analyze the probability of the i^{th} query being successful, and sum over $i = 1, \ldots, q$.

Table 3.2. For $\alpha_{\mathsf{tl}}, \alpha_{\mathsf{tr}}, \alpha_{\mathsf{bl}}, \alpha_{\mathsf{br}} \in \{0,1\}$, the probability bound on $\mathsf{col}_{\alpha_{\mathsf{tl}}\alpha_{\mathsf{tr}}\alpha_{\mathsf{bl}}\alpha_{\mathsf{br}}}(\mathcal{Q}_q) \wedge \neg\mathsf{help}(\mathcal{Q}_q)$ (cf. (3.5)) is analyzed in the corresponding lemma.

$\alpha_{\mathsf{tl}}\alpha_{\mathsf{tr}}\alpha_{\mathsf{bl}}\alpha_{\mathsf{br}}$	0000	0001	0010	0011	0100	0101	0110	0111
Lemma	3.2.4	3.2.5	3.2.5	3.2.6	3.2.7	3.2.8	3.2.9	3.2.9
$\alpha_{\mathsf{tl}}\alpha_{\mathsf{tr}}\alpha_{\mathsf{bl}}\alpha_{\mathsf{br}}$	1000	1001	1010	1011	1100	1101	1110	1111
Lemma	3.2.7	3.2.9	3.2.8	3.2.9	3.2.9	3.2.9	3.2.9	3.2.9

Let $i \in \{1, \ldots, q\}$. We will analyze the probability of the i^{th} query to be successful, i.e., to satisfy $\mathsf{col}_{0000}(\mathcal{Q}_i) \wedge \neg\mathsf{help}(\mathcal{Q}_i)$. If $\mathsf{help}(\mathcal{Q}_i)$ holds, the i^{th} query can certainly not be successful, so we assume $\neg\mathsf{help}(\mathcal{Q}_i)$ and analyze the probability the i^{th} query makes $\mathsf{col}_{0000}(\mathcal{Q}_i)$ satisfied.

Without loss of generality (by symmetry), the i^{th} query occurs in the left word. It may be the case that the i^{th} query also occurs in the right word (e.g., at $2\mathsf{br}$), but as becomes clear from the proof, this case is automatically included. Note that, as $u_1 \neq v_1$, it can impossibly occur at $(1\mathsf{tl}, 1\mathsf{tr})$ or $(1\mathsf{bl}, 1\mathsf{br})$. Therefore, without loss of generality (by symmetry) it suffices to analyze the cases the query occurs at the following positions: $1\mathsf{tl}$ only, $1\mathsf{br}$ only, $(1\mathsf{tl}, 1\mathsf{br})$ only, or $(1\mathsf{tl}, 1\mathsf{bl})$ only. We distinguish among these four cases.

Query occurs at 1tl only. By $\neg\mathsf{help}_1(\mathcal{Q}_i)$, there are $\leq \tau_1$ choices for queries at positions $(1\mathsf{br}, 2\mathsf{br})$ (we note that the query at $2\mathsf{br}$ may equal the i^{th} query, but this does not invalidate the ongoing analysis). For any of these $\leq \tau_1$ choices, let $K_{1\mathsf{br}}$ and $K_{2\mathsf{br}}$ be the key inputs corresponding to positions $1\mathsf{br}$ and $2\mathsf{br}$. By $\neg\mathsf{help}_2(\mathcal{Q}_i)$, there are $\leq \tau_2$ choices for $2\mathsf{tl}$. For any of these $\leq \tau_1\tau_2$ choices $2\mathsf{tl}$, the query at position $2\mathsf{tr}$ and consequently the query at position $2\mathsf{bl}$ is uniquely determined (if they exist at all), and so is the XOR-output y of $2\mathsf{bl}$. By $\neg\mathsf{help}_4(\mathcal{Q}_i)$, there are $\leq \tau_3$ choices for $1\mathsf{bl}$. For any of these $\leq \tau_1\tau_2\tau_3$ choices $1\mathsf{bl}$, let $K_{1\mathsf{bl}}$ be the key input corresponding to position $1\mathsf{bl}$. The i^{th} query is successful only if its XOR-output equals $K_{1\mathsf{br}}^l \| K_{1\mathsf{bl}}^r$, which happens with probability at most $\frac{1}{2^n - q}$. The total probability is at most $\frac{\tau_1\tau_2\tau_3}{2^n - q}$.

Query occurs at 1br only. By $\neg\mathsf{help}_1(\mathcal{Q}_i)$, there are $\leq \tau_1$ choices for $(1\mathsf{bl}, 2\mathsf{bl})$. For any of these $\leq \tau_1$ choices, let $K_{2\mathsf{bl}}$ be the key input corresponding to position $2\mathsf{bl}$. By $\neg\mathsf{help}_2(\mathcal{Q}_i)$, there are $\leq \tau_2$ choices for $2\mathsf{tr}$. For any of these $\leq \tau_1\tau_2$ choices $2\mathsf{tr}$, the query at position $2\mathsf{tl}$ and consequently the query at position $2\mathsf{br}$ is uniquely determined, and so is the XOR-output z of $2\mathsf{br}$. The i^{th} query is successful only if its XOR-output equals this value z, which happens with probability at most $\frac{1}{2^n - q}$. The total success probability is at most $\frac{\tau_1\tau_2}{2^n - q}$.

Query occurs at 1tl and 1br only. Let K be the key input for the i^{th} query. As the query occurs at both positions, we require $K^l = z^l$, which fixes z^l. By

$\neg\mathsf{help}_2(\mathcal{Q}_i)$, there are $\leq \tau_2$ choices for $2\mathsf{br}$. For any of these $\leq \tau_2$ choices $2\mathsf{br}$, we obtain a different z. The i^{th} query is successful only if its XOR-output equals this value z, which happens with probability at most $\frac{1}{2^n-q}$. The total success probability is at most $\frac{\tau_2}{2^n-q}$.

Query occurs at 1tl and 1bl only. Let K be the key input for the i^{th} query. As the query occurs at both positions, we require $K^r = y^r$, which fixes y^r. By $\neg\mathsf{help}_3(\mathcal{Q}_i)$, there are $\leq \tau_2$ choices for $2\mathsf{bl}$. For any of these $\leq \tau_2$ choices $2\mathsf{bl}$, we obtain a different y. The i^{th} query is successful only if its XOR-output equals this value y, which happens with probability at most $\frac{1}{2^n-q}$. The total success probability is at most $\frac{\tau_2}{2^n-q}$.

The i^{th} query is successful with probability at most $\frac{\tau_1\tau_2\tau_3+\tau_1\tau_2+2\tau_2}{2^n-q}$. The claimed bound is obtained by summing over $i = 1, \ldots, q$. □

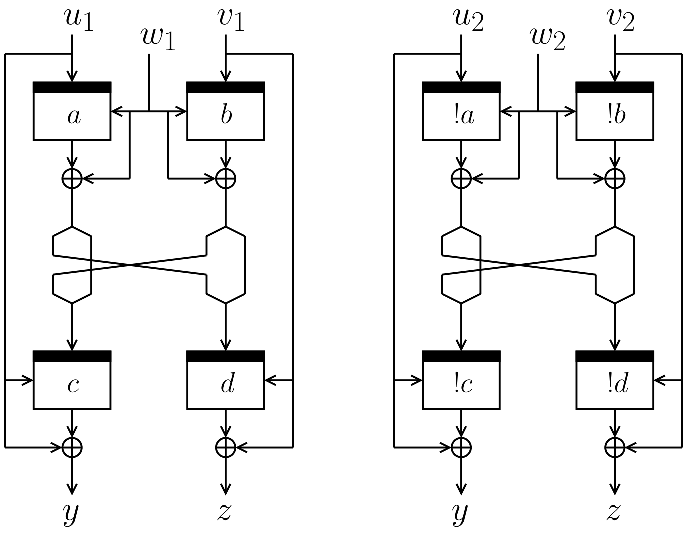


Figure 3.5. Configuration $\mathsf{col}_{0000}(\mathcal{Q})$. We require $(u_1, v_1, w_1) \neq (u_2, v_2, w_2)$, $u_1 \neq v_1$, and $u_2 \neq v_2$.

Lemma 3.2.5. $\mathbf{Pr}\left(\mathsf{col}_{\alpha_{\mathsf{tl}}\alpha_{\mathsf{tr}}\alpha_{\mathsf{bl}}\alpha_{\mathsf{br}}}(\mathcal{Q}_q) \wedge \neg\mathsf{help}(\mathcal{Q}_q)\right) \leq \frac{(2\tau_1\tau_2+2\tau_2+\tau_2^2)q}{2^n-q}$ *for* $\alpha_{\mathsf{tl}}\alpha_{\mathsf{tr}}\alpha_{\mathsf{bl}}\alpha_{\mathsf{br}} \in \{0001, 0010\}$.

Proof. The cases are equivalent by symmetry, and we consider $\mathsf{col}_{0001}(\mathcal{Q}_q)$ only. A visualization of configuration $\mathsf{col}_{0001}(\mathcal{Q}_q)$ can be found in Figure 3.6. For the basic proof idea, we refer to the proof of Lemma 3.2.4. Let $i \in \{1, \ldots, q\}$. As in Lemma 3.2.4, we assume $\neg\mathsf{help}(\mathcal{Q}_i)$ and analyze the probability the i^{th} query makes $\mathsf{col}_{0001}(\mathcal{Q}_i)$ satisfied.

Without loss of generality (by symmetry), the i^{th} query occurs at the following positions: $1\mathsf{tl}$ only, $1\mathsf{tr}$ only, $1\mathsf{bl}$ only, $(1\mathsf{tr}, 1\mathsf{bl})$ only, or $(1\mathsf{tl}, 1\mathsf{bl})$ only. We distinguish among these five cases. Note that, as $u_1 \neq v$, it can impossibly occur at $(1\mathsf{tl}, 1\mathsf{tr})$. It may be the case that the i^{th} query also occurs in the right word, but this case is automatically included.

Query occurs at 1tl only. By $\neg\mathsf{help}_1(\mathcal{Q}_i)$, there are $\leq \tau_1$ choices for $(1\mathsf{bl}, 2\mathsf{bl})$. For any of these $\leq \tau_1$ choices, let $K_{1\mathsf{bl}}$ and $K_{2\mathsf{bl}}$ be the key inputs corresponding to positions $1\mathsf{bl}$ and $2\mathsf{bl}$. By $\neg\mathsf{help}_3(\mathcal{Q}_i)$, there are $\leq \tau_2$ choices for $2\mathsf{tl}$. For any of these $\leq \tau_1\tau_2$ choices $2\mathsf{tl}$, we obtain a different t. The i^{th} query is successful only if its XOR-output equals $t\|K^r_{1\mathsf{bl}}$, which happens with probability at most $\frac{1}{2^n-q}$. The total success probability is at most $\frac{\tau_1\tau_2}{2^n-q}$.

Query occurs at 1tr only. By $\neg\mathsf{help}_1(\mathcal{Q}_i)$, there are $\leq \tau_1$ choices for $(1\mathsf{bl}, 2\mathsf{bl})$. For any of these $\leq \tau_1$ choices, let $K_{1\mathsf{bl}}$ and $K_{2\mathsf{bl}}$ be the key inputs corresponding to positions $1\mathsf{bl}$ and $2\mathsf{bl}$. By $\neg\mathsf{help}_2(\mathcal{Q}_i)$, there are $\leq \tau_2$ choices for $2\mathsf{tr}$. For any of these $\leq \tau_1\tau_2$ choices $2\mathsf{tr}$, we obtain a different y. The i^{th} query is successful only if its XOR-output equals $K^l_{1\mathsf{bl}}\|y$, which happens with probability at most $\frac{1}{2^n-q}$. The total success probability is at most $\frac{\tau_1\tau_2}{2^n-q}$.

Query occurs at 1bl only: inverse query $\boldsymbol{m \leftarrow E^{-1}(K, c)}$. By $\neg\mathsf{help}_3(\mathcal{Q}_i)$, there are $\leq \tau_2$ choices for $1\mathsf{tl}$. For any of these $\leq \tau_2$ choices, let $K_{1\mathsf{tl}}$ be the key input corresponding to position $1\mathsf{tl}$. The i^{th} query is successful only if $m = K_{1\mathsf{tl}}$, which happens with probability at most $\frac{1}{2^n-q}$. The total success probability is at most $\frac{\tau_2}{2^n-q}$.

Query occurs at 1bl only: forward query $\boldsymbol{c \leftarrow E(K, m)}$. By $\neg\mathsf{help}_2(\mathcal{Q}_i)$, there are $\leq \tau_2$ choices for $1\mathsf{tr}$. For any of these $\leq \tau_2$ choices $1\mathsf{tr}$, the query at position $1\mathsf{tl}$ is uniquely determined (it requires key input m and message input w_1 defined by query $1\mathsf{tr}$), and so are the strings (v, t). By $\neg\mathsf{help}_2(\mathcal{Q}_i)$, there are $\leq \tau_2$ choices for $2\mathsf{tl}$. For any of these $\leq \tau_2^2$ choices $2\mathsf{tl}$, the query at position $2\mathsf{tr}$ is uniquely determined (it requires key input v and message input w_2 defined by query $2\mathsf{tl}$). Consequently, the query at position $2\mathsf{br}$ is uniquely determined, and so is the XOR-output z of $2\mathsf{br}$. The i^{th} query is successful only if its XOR-output equals this value z, which happens with probability at most $\frac{1}{2^n-q}$. The total success probability is at most $\frac{\tau_2^2}{2^n-q}$.

Query occurs at 1tr and 1bl only. Let K be the key input for the i^{th} query. As the query occurs at both positions, we require $K^l = z^l$, which fixes z^l. By $\neg\mathsf{help}_2(\mathcal{Q}_i)$, there are $\leq \tau_2$ choices for $2\mathsf{bl}$. For any of these $\leq \tau_2$ choices $2\mathsf{bl}$, we obtain a different z. The i^{th} query is successful only if its XOR-output equals this value z, which happens with probability at most $\frac{1}{2^n-q}$. The total success probability is at most $\frac{\tau_2}{2^n-q}$.

Query occurs at 1tl and 1bl only. Let K be the key input for the i^{th} query. As the query occurs at both positions, we require $K^r = z^r$, which fixes z^r. By $\neg\mathsf{help}_3(\mathcal{Q}_i)$, there are $\leq \tau_2$ choices for $2\mathsf{bl}$. For any of these $\leq \tau_2$ choices $2\mathsf{bl}$, we obtain a different z. The i^{th} query is successful only if its XOR-output equals this value z, which happens with probability at most $\frac{1}{2^n-q}$. The total success probability is at most $\frac{\tau_2}{2^n-q}$.

The i^{th} query is successful with probability at most $\frac{2\tau_1\tau_2+2\tau_2+\tau_2^2}{2^n-q}$. The claimed bound is obtained by summing over $i = 1, \dots, q$. $\square$

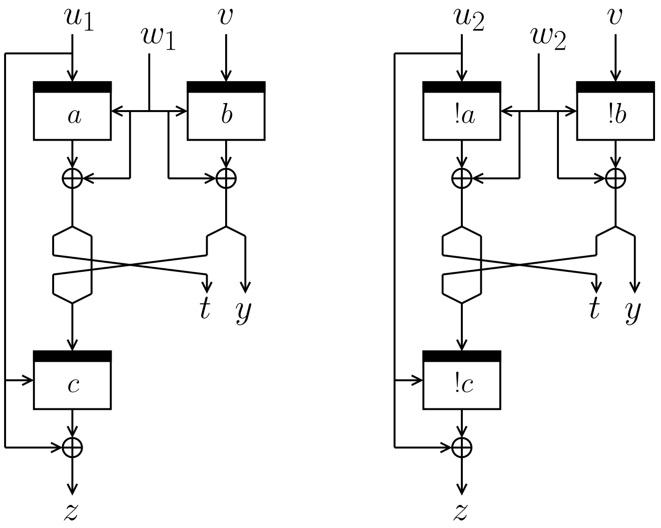


Figure 3.6. Configuration $\mathsf{col}_{0001}(\mathcal{Q})$. We require $(u_1, w_1) \neq (u_2, w_2)$, $u_1 \neq v$, and $u_2 \neq v$.

Lemma 3.2.6. $\mathbf{Pr}\left(\mathsf{col}_{0011}(\mathcal{Q}_q) \wedge \neg\mathsf{help}(\mathcal{Q}_q)\right) \leq \frac{\tau_1 q}{2^n-q}$.

Proof. A visualization of configuration $\mathsf{col}_{0011}(\mathcal{Q}_q)$ can be found in Figure 3.7. For the basic proof idea, we refer to the proof of Lemma 3.2.4. Let $i \in \{1, \dots, q\}$. As in Lemma 3.2.4, we assume $\neg\mathsf{help}(\mathcal{Q}_i)$ and analyze the probability the i^{th} query makes $\mathsf{col}_{0011}(\mathcal{Q}_i)$ satisfied.

Without loss of generality (by symmetry), the i^{th} query occurs at the position $1\mathsf{tl}$ only. Note that, as $u \neq v$, it can impossibly occur at $(1\mathsf{tl}, 1\mathsf{tr})$. It may be the case that the i^{th} query also occurs in the right word, but this case is automatically included.

Query occurs at 1tl. By $\neg\mathsf{help}_1(\mathcal{Q}_i)$, there are $\leq \tau_1$ choices for $(1\mathsf{tr}, 2\mathsf{tr})$. For any of these $\leq \tau_1$ choices, as u is fixed (it equals the key input for the i^{th} query) the query at position $2\mathsf{tl}$ is uniquely determined, and so is the XOR-output y of $2\mathsf{tl}$. The i^{th} query is successful only if its XOR-output equals this value y, which happens with probability at most $\frac{1}{2^n-q}$. The total success probability is at most $\frac{\tau_1}{2^n-q}$.

The i^{th} query is successful with probability at most $\frac{\tau_1}{2^n-q}$. The claimed bound is obtained by summing over $i = 1, \ldots, q$. $\square$

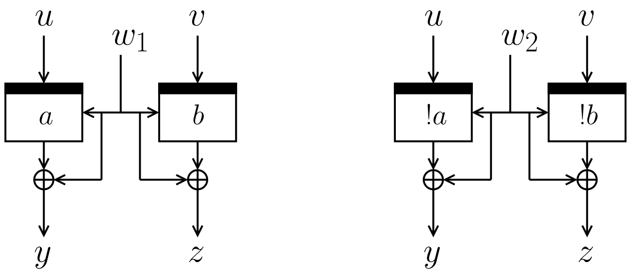


Figure 3.7. Configuration $\mathsf{col}_{0011}(\mathcal{Q})$. We require $w_1 \neq w_2$ and $u \neq v$.

Lemma 3.2.7. $\mathbf{Pr}\left(\mathsf{col}_{\alpha_{\mathsf{tl}}\alpha_{\mathsf{tr}}\alpha_{\mathsf{bl}}\alpha_{\mathsf{br}}}(\mathcal{Q}_q) \wedge \neg\mathsf{help}(\mathcal{Q}_q)\right) \leq \frac{(\tau_1\tau_2\tau_3+2\tau_1\tau_2+2\tau_2)q}{2^n-q}$ *for* $\alpha_{\mathsf{tl}}\alpha_{\mathsf{tr}}\alpha_{\mathsf{bl}}\alpha_{\mathsf{br}} \in \{0100, 1000\}$.

Proof. The cases are equivalent by symmetry, and we consider $\mathsf{col}_{0100}(\mathcal{Q}_q)$ only. A visualization of configuration $\mathsf{col}_{0100}(\mathcal{Q}_q)$ can be found in Figure 3.8. For the basic proof idea, we refer to the proof of Lemma 3.2.4. Let $i \in \{1, \ldots, q\}$. As in Lemma 3.2.4, we assume $\neg\mathsf{help}(\mathcal{Q}_i)$ and analyze the probability the i^{th} query makes $\mathsf{col}_{0100}(\mathcal{Q}_i)$ satisfied.

Without loss of generality (by symmetry), the i^{th} query occurs at the following positions: $1\mathsf{tl}$ only, $1\mathsf{bl}$ only, $1\mathsf{br}$ only, $(1\mathsf{tl}, 1\mathsf{br})$ only, or $(1\mathsf{tl}, 1\mathsf{bl})$ only. We distinguish among these five cases. Note that, as $u_1 \neq v$, it can impossibly occur at $(1\mathsf{bl}, 1\mathsf{br})$. It may be the case that the i^{th} query also occurs in the right word, but this case is automatically included.

Query occurs at $1\mathsf{tl}$ only. We note that the query at position $1\mathsf{tr} = 2\mathsf{tr}$ is not depicted in Figure 3.8 but is defined as a query $(2, v, w, s\|t \oplus w)$. By $\neg\mathsf{help}_1(\mathcal{Q}_i)$, there are $\leq \tau_1$ choices for queries at positions $(1\mathsf{br}, 2\mathsf{br})$. For any of these $\leq \tau_1$ choices, let $K_{1\mathsf{br}}$ and $K_{2\mathsf{br}}$ be the key inputs corresponding to positions $1\mathsf{br}$ and $2\mathsf{br}$. By $\neg\mathsf{help}_2(\mathcal{Q}_i)$, there are $\leq \tau_2$ choices for $2\mathsf{tl}$. For any of these $\leq \tau_1\tau_2$ choices $2\mathsf{tl}$, the query at position $1\mathsf{tr} = 2\mathsf{tr}$ and consequently the query at position $2\mathsf{bl}$ is uniquely determined, and so is the XOR-output y of $2\mathsf{bl}$. By $\neg\mathsf{help}_4(\mathcal{Q}_i)$, there are $\leq \tau_3$ choices for $1\mathsf{bl}$. For any of these $\leq \tau_1\tau_2\tau_3$ choices $1\mathsf{bl}$, let $K_{1\mathsf{bl}}$ be the key input corresponding to position $1\mathsf{bl}$. The i^{th} query is successful only if its XOR-output equals $K^l_{1\mathsf{br}}\|K^r_{1\mathsf{bl}}$, which happens with probability at most $\frac{1}{2^n-q}$. The total probability is at most $\frac{\tau_1\tau_2\tau_3}{2^n-q}$.

Query occurs at $1\mathsf{bl}$ only. By $\neg\mathsf{help}_1(\mathcal{Q}_i)$, there are $\leq \tau_1$ choices for $(1\mathsf{br}, 2\mathsf{br})$. For any of these $\leq \tau_1$ choices, let $K_{2\mathsf{br}}$ be the key input corresponding to position

2br. By $\neg\mathsf{help}_2(\mathcal{Q}_i)$, there are $\leq \tau_2$ choices for 2tl. For any of these $\leq \tau_1\tau_2$ choices 2tl, the query at position 2bl is uniquely determined, and so is the XOR-output y of 2bl. The i^{th} query is successful only if its XOR-output equals this value y, which happens with probability at most $\frac{1}{2^n-q}$. The total success probability is at most $\frac{\tau_1\tau_2}{2^n-q}$.

Query occurs at 1br only. By $\neg\mathsf{help}_1(\mathcal{Q}_i)$, there are $\leq \tau_1$ choices for (1bl, 2bl). For any of these $\leq \tau_1$ choices, let K_{2bl} be the key input corresponding to position 2bl. By $\neg\mathsf{help}_3(\mathcal{Q}_i)$, there are $\leq \tau_2$ choices for 2tl. For any of these $\leq \tau_1\tau_2$ choices 2tl, the key input to the query at position 2br, say K_{2br}, is uniquely determined. Suppose the i^{th} query is a forward query $c \leftarrow E(K, m)$ (exactly the same reasoning applies to inverse queries). If $K_{\mathsf{2br}} = K$, the queries at positions 1br and 2br must be the same and the collision is invalid. Therefore, we assume $K_{\mathsf{2br}} \neq K$. For the key K_{2br}, let $(K_{\mathsf{2br}}, m_{\mathsf{2br}}, c_{\mathsf{2br}})$ be any query in the query history. The i^{th} query makes the configuration satisfied if $m_{\mathsf{2br}} = m$ and $m_{\mathsf{2br}} \oplus c_{\mathsf{2br}} = m \oplus c$, or more concretely if

$$m_{\mathsf{2br}} = m \text{ and } c_{\mathsf{2br}} = c\,. \tag{3.6}$$

This means that, irrespectively of whether the i^{th} query is a forward or inverse query, the query at position 2br is uniquely determined. The i^{th} query is successful only if it satisfies (3.6), which happens with probability at most $\frac{1}{2^n-q}$. The total probability is at most $\frac{\tau_1\tau_2}{2^n-q}$.

Query occurs at 1tl and 1br only. Let K be the key input for the i^{th} query. As the query occurs at both positions, we require $K^l = z^l$, which fixes z^l. By $\neg\mathsf{help}_2(\mathcal{Q}_i)$, there are $\leq \tau_2$ choices for 2br. For any of these $\leq \tau_2$ choices 2br, we obtain a different z. The i^{th} query is successful only if its XOR-output equals this value z, which happens with probability at most $\frac{1}{2^n-q}$. The total success probability is at most $\frac{\tau_2}{2^n-q}$.

Query occurs at 1tl and 1bl only. Let K be the key input for the i^{th} query. As the query occurs at both positions, we require $K^r = y^r$, which fixes y^r. By $\neg\mathsf{help}_3(\mathcal{Q}_i)$, there are $\leq \tau_2$ choices for 2bl. For any of these $\leq \tau_2$ choices 2bl, we obtain a different y. The i^{th} query is successful only if its XOR-output equals this value y, which happens with probability at most $\frac{1}{2^n-q}$. The total success probability is at most $\frac{\tau_2}{2^n-q}$.

The i^{th} query is successful with probability at most $\frac{\tau_1\tau_2\tau_3+2\tau_1\tau_2+2\tau_2}{2^n-q}$. The claimed bound is obtained by summing over $i = 1, \ldots, q$. □

Lemma 3.2.8. $\mathbf{Pr}\left(\mathsf{col}_{\alpha_{\mathsf{tl}}\alpha_{\mathsf{tr}}\alpha_{\mathsf{bl}}\alpha_{\mathsf{br}}}(\mathcal{Q}_q) \wedge \neg\mathsf{help}(\mathcal{Q}_q)\right) \leq \frac{(\tau_1\tau_2+\tau_2+\tau_2^2)q}{2^n-q}$ *for* $\alpha_{\mathsf{tl}}\alpha_{\mathsf{tr}}\alpha_{\mathsf{bl}}\alpha_{\mathsf{br}} \in \{0101, 1010\}$.

Proof. The cases are equivalent by symmetry, and we consider $\mathsf{col}_{0101}(\mathcal{Q}_q)$ only. A visualization of configuration $\mathsf{col}_{0101}(\mathcal{Q}_q)$ can be found in Figure 3.9. For the

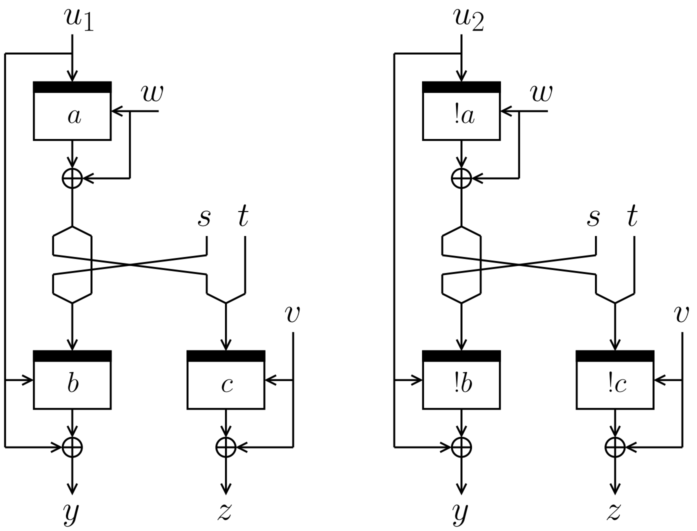


Figure 3.8. Configuration $\mathsf{col}_{0100}(\mathcal{Q})$. We require $u_1 \neq u_2$, $u_1 \neq v$, and $u_2 \neq v$.

basic proof idea, we refer to the proof of Lemma 3.2.4. Let $i \in \{1, \ldots, q\}$. As in Lemma 3.2.4, we assume $\neg\mathsf{help}(\mathcal{Q}_i)$ and analyze the probability the i^{th} query makes $\mathsf{col}_{0101}(\mathcal{Q}_i)$ satisfied.

Without loss of generality (by symmetry), the i^{th} query occurs at the following positions: $1\mathsf{tl}$ only, $1\mathsf{bl}$ only, or $(1\mathsf{tl}, 1\mathsf{bl})$ only. We distinguish among these three cases. It may be the case that the i^{th} query also occurs in the right word, but this case is automatically included.

Query occurs at 1tl only. By $\neg\mathsf{help}_1(\mathcal{Q}_i)$, there are $\leq \tau_1$ choices for $(1\mathsf{bl}, 2\mathsf{bl})$. For any of these $\leq \tau_1$ choices, let $K_{1\mathsf{bl}}$ and $K_{2\mathsf{bl}}$ be the key inputs corresponding to positions $1\mathsf{bl}$ and $2\mathsf{bl}$. By $\neg\mathsf{help}_3(\mathcal{Q}_i)$, there are $\leq \tau_2$ choices for $2\mathsf{tl}$. For any of these $\leq \tau_1\tau_2$ choices $2\mathsf{tl}$, we obtain a different y. The i^{th} query is successful only if its XOR-output equals $y \| K^r_{1\mathsf{bl}}$, which happens with probability at most $\frac{1}{2^n - q}$. The total success probability is at most $\frac{\tau_1\tau_2}{2^n - q}$.

Query occurs at 1bl only. Let K be the key input for the i^{th} query. By $\neg\mathsf{help}_3(\mathcal{Q}_i)$, there are $\leq \tau_2$ choices for $1\mathsf{tl}$. For any of these $\leq \tau_2$ choices, we obtain a different y. By $\neg\mathsf{help}_2(\mathcal{Q}_i)$, there are $\leq \tau_2$ choices for $2\mathsf{tl}$. For any of these $\leq \tau_2^2$ choices $2\mathsf{tl}$, the query at position $2\mathsf{bl}$ is uniquely determined, and so is the XOR-output z of $2\mathsf{bl}$. The i^{th} query is successful only if its XOR-output equals this value z, which happens with probability at most $\frac{1}{2^n - q}$. The total success probability is at most $\frac{\tau_2^2}{2^n - q}$.

Query occurs at 1tl and 1bl only. Let K be the key input for the i^{th} query. As the query occurs at both positions, we require $K^r = z^r$, which fixes z^r. By $\neg\mathsf{help}_3(\mathcal{Q}_i)$, there are $\leq \tau_2$ choices for 2bl. For any of these $\leq \tau_2$ choices 2bl, we obtain a different z. The i^{th} query is successful only if its XOR-output equals this value z, which happens with probability at most $\frac{1}{2^n-q}$. The total success probability is at most $\frac{\tau_2}{2^n-q}$.

The i^{th} query is successful with probability at most $\frac{\tau_1\tau_2+\tau_2+\tau_2^2}{2^n-q}$. The claimed bound is obtained by summing over $i = 1, \ldots, q$. $\square$

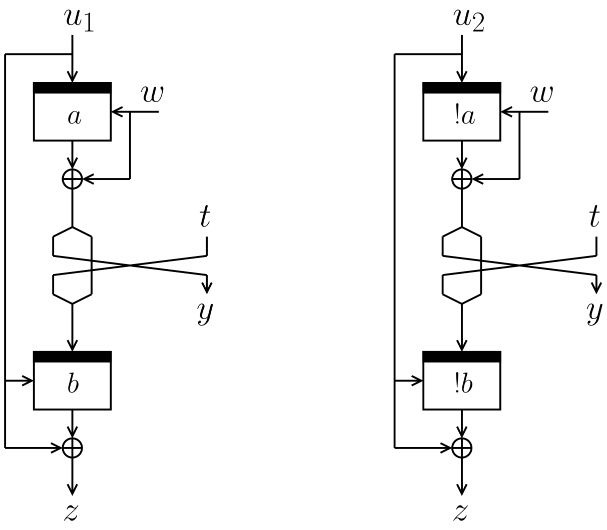


Figure 3.9. Configuration $\mathsf{col}_{0101}(\mathcal{Q})$. We require $u_1 \neq u_2$.

Lemma 3.2.9. *$\mathbf{Pr}\left(\mathsf{col}_{\alpha_{\mathsf{tl}}\alpha_{\mathsf{tr}}\alpha_{\mathsf{bl}}\alpha_{\mathsf{br}}}(\mathcal{Q}_q) \wedge \neg\mathsf{help}(\mathcal{Q}_q)\right) = 0$ for $\alpha_{\mathsf{tl}}\alpha_{\mathsf{tr}}\alpha_{\mathsf{bl}}\alpha_{\mathsf{br}} \in \{11**, 1**1, *11*\}$.*

Proof. If 1tl = 2tl and 1tr = 2tr, we obtain $(u_1, v_1, w_1) = (u_2, v_2, w_2)$ in the configuration of Figure 3.4, and the collision is invalid. The same observation applies if $(1\mathsf{tl}, 1\mathsf{br}) = (2\mathsf{tl}, 2\mathsf{br})$ or $(1\mathsf{tr}, 1\mathsf{bl}) = (2\mathsf{tr}, 2\mathsf{bl})$. $\square$

Lemma 3.2.10. $\mathbf{Pr}\left(\mathsf{help}(\mathcal{Q}_q)\right) \leq \frac{q^2}{\tau_1(2^n-q)} + 2\cdot 2^{n/2}\left(\frac{eq2^{n/2}}{\tau_2(2^n-q)}\right)^{\tau_2} + 2^n\left(\frac{eq}{\tau_3(2^n-q)}\right)^{\tau_3}$.

Proof. It suffices to consider the events $\mathbf{Pr}\left(\mathsf{help}_k(\mathcal{Q}_q)\right)$ $(k = 1, \ldots, 4)$ separately.
$\mathbf{help_1(\mathcal{Q}_q)}$. We copy the approach of Steinberger [211]. For $i \neq j$, the two queries (K_i, m_i, c_i) and (K_j, m_j, c_i) have the same XOR-output with probability at most

$\frac{1}{2^n-q}$. Hence, the expected value $\mathsf{E}(m_i \oplus c_i = m_j \oplus c_j)$ is at most $\frac{1}{2^n-q}$, and consequently

$$\mathsf{E}\left(\left|\left\{(K_i, m_i, c_i), (K_j, m_j, c_j) \in \mathcal{Q}_q \;\middle|\; i \neq j \;\wedge\; m_i \oplus c_i = m_j \oplus c_j\right\}\right|\right) \leq \sum_{i \neq j} \frac{1}{2^n - q} \leq \frac{q^2}{2^n - q}\,.$$

By Markov's inequality, we obtain

$$\mathbf{Pr}\left(\mathsf{help}_1(\mathcal{Q}_q)\right) \leq \frac{q^2}{\tau_1(2^n - q)}\,. \tag{3.7}$$

$\mathsf{help}_k(\mathcal{Q}_q)$ for $k \in \{2,3\}$. The cases are equivalent by symmetry, and we consider $\mathsf{help}_2(\mathcal{Q}_q)$ only. Let $x \in \mathbb{Z}_2^{n/2}$. Consider the i^{th} query (K_i, m_i, c_i). This query makes equation $(m_i \oplus c_i)^l = x$ satisfied with probability at most $\frac{2^{n/2}}{2^n-q}$. More than τ_2 queries result in a solution with probability at most $\binom{q}{\tau_2}\left(\frac{2^{n/2}}{2^n-q}\right)^{\tau_2} \leq \left(\frac{eq2^{n/2}}{\tau_2(2^n-q)}\right)^{\tau_2}$, where we use Stirling's approximation. Considering any possible choice for x, we obtain for $k = 2, 3$:

$$\mathbf{Pr}\left(\mathsf{help}_k(\mathcal{Q}_q)\right) \leq 2^{n/2}\left(\frac{eq2^{n/2}}{\tau_2(2^n - q)}\right)^{\tau_2}. \tag{3.8}$$

$\mathsf{help}_4(\mathcal{Q}_q)$. A similar analysis as for $\mathsf{help}_2(\mathcal{Q}_q)$ results in the following bound:

$$\mathbf{Pr}\left(\mathsf{help}_4(\mathcal{Q}_q)\right) \leq 2^{n}\left(\frac{eq}{\tau_3(2^n - q)}\right)^{\tau_3}. \tag{3.9}$$

The claim is obtained by adding (3.7-3.9). □

We are ready to finish the proof of Theorem 3.2.1. Lemmas 3.2.4-3.2.10 imply for $\mathbf{Adv}_{f_{\text{MDC-4}}}^{\text{col}(\neq)}(q)$ of (3.5):

$$\mathbf{Adv}_{f_{\text{MDC-4}}}^{\text{col}(\neq)}(q) \leq \frac{(\tau_1 + 11\tau_1\tau_2 + 3\tau_1\tau_2\tau_3 + 12\tau_2 + 4\tau_2^2)q}{2^n - q} + \frac{q^2}{\tau_1(2^n - q)} + 2 \cdot 2^{n/2}\left(\frac{eq2^{n/2}}{\tau_2(2^n - q)}\right)^{\tau_2} + 2^n\left(\frac{eq}{\tau_3(2^n - q)}\right)^{\tau_3},$$

This completes the proof of Theorem 3.2.1.

3.2.2 Proof of Theorem 3.2.2

As explained in Section 3.2 we essentially only need to consider the probability that an adversary finds an $f_{\text{MDC-4}}$ evaluation where the state consists of two different halves and the output state consists of two the same halves. The formal treatment of this is more elaborate.

We consider any adversary making q queries to its oracle E, which tries to find a collision for MDC-4. Suppose the adversary finds a collision, i.e., two lists

$$(g_0, h_0) \xrightarrow{M_1} (g_1, h_1) \xrightarrow{M_2} \cdots \xrightarrow{M_k} (g_k, h_k)\,,$$
$$(g_0, h_0) \xrightarrow{M_1'} (g_1', h_1') \xrightarrow{M_2'} \cdots \xrightarrow{M_k'} (g_{k'}', h_{k'}')$$

of internal state values of the two evaluations, where $k, k' \geq 1$ and $(g_k, h_k) = (g_{k'}', h_{k'}')$. The collision is non-trivial if $k \neq k'$ or if $(g_i, h_i, M_i) \neq (g_i', h_i', M_i')$ for some $i = 1, \ldots, k = k'$, and we consider non-trivial collisions only. If the adversary finds a collision of this form, we can distinguish between the following two cases:

(i) $g_i \neq h_i$ for all $i \in \{1, \ldots, k\}$ and $g_i' \neq h_i'$ for all $i \in \{1, \ldots, k'\}$;
(ii) $g_i = h_i$ for some $i \in \{1, \ldots, k\}$ or $g_i' = h_i'$ for some $i \in \{1, \ldots, k'\}$.

Suppose the adversary finds a collision in case (i). This implies (by Theorem 2.4.1) the adversary necessarily needs to obtain a query history $\mathcal{Q}_q$ of size q that satisfies configuration $\mathsf{col}(\mathcal{Q}_q)$ of Figure 3.4.

On the other hand, suppose the adversary finds a collision in case (ii). Without loss of generality, a state-half collision occurs in the first word. As $g_0 \neq h_0$, there exists an i such that $g_i = h_i$ but $g_{i-1} \neq h_{i-1}$. This means that in this case the adversary necessarily needs to obtain a query history $\mathcal{Q}_q$ of size q that satisfies configuration $\mathsf{statecol}(\mathcal{Q}_q)$ of Figure 3.10. Here, z represents $g_i = h_i$ and (u, v) represents (g_{i-1}, h_{i-1}). As in Section 3.2.1, in this configuration we have omitted the shifting at the end. We stress that this does not harm the security analysis. We label the block ciphers $\mathsf{tl}, \ldots, \mathsf{br}$.

Concluding, we find

$$\mathbf{Adv}^{\mathrm{col}}_{\mathrm{MDC\text{-}4}}(q) \leq \mathbf{Pr}\left(\mathsf{col}(\mathcal{Q}_q) \vee \mathsf{statecol}(\mathcal{Q}_q)\right)\,. \tag{3.10}$$

We employ the helping event $\mathsf{help}(\mathcal{Q}_q)$ from Section 3.2.1, and obtain for (3.10):

$$\begin{aligned}\mathbf{Pr}\left(\mathsf{col}(\mathcal{Q}_q) \vee \mathsf{statecol}(\mathcal{Q}_q)\right) \leq{} & \mathbf{Pr}\left(\mathsf{col}(\mathcal{Q}_q) \wedge \neg\mathsf{help}(\mathcal{Q}_q)\right) + \mathbf{Pr}\left(\mathsf{help}(\mathcal{Q}_q)\right) + \\ & \mathbf{Pr}\left(\mathsf{statecol}(\mathcal{Q}_q) \wedge \neg\mathsf{help}(\mathcal{Q}_q)\right)\,. \end{aligned} \tag{3.11}$$

For the first two probabilities, the proof of Theorem 3.2.1 applies. The probability bound on $\mathsf{statecol}(\mathcal{Q}_q) \wedge \neg\mathsf{help}(\mathcal{Q}_q)$ is analyzed in Lemma 3.2.11.

Lemma 3.2.11. $\mathbf{Pr}\left(\mathsf{statecol}(\mathcal{Q}_q) \wedge \neg\mathsf{help}(\mathcal{Q}_q)\right) \leq \frac{(\tau_1 + 2\tau_2 + \tau_2^2)q}{2^n - q}$.

Proof. We consider configuration $\mathsf{statecol}(\mathcal{Q}_q)$ of Figure 3.10. The proof idea is the same as the proof of Lemma 3.2.4. Let $i \in \{1, \ldots, q\}$. As in Lemma 3.2.4, we assume $\neg\mathsf{help}(\mathcal{Q}_i)$ and analyze the probability the ith query makes $\mathsf{statecol}(\mathcal{Q}_i)$ satisfied.

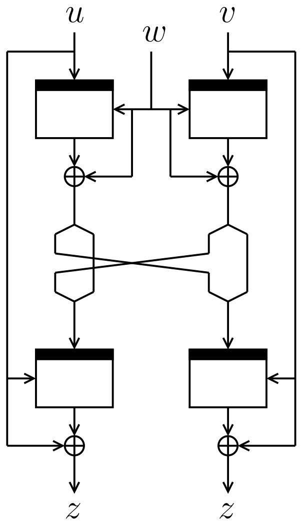


Figure 3.10. Configuration $\mathsf{statecol}(\mathcal{Q})$. We require $u \neq v$.

Recall that the positions in Figure 3.10 are simply referred to as $\mathsf{tl}, \mathsf{tr}, \mathsf{bl}, \mathsf{br}$, without a leading 1. Without loss of generality (by symmetry), the i^{th} query occurs at the following positions: tl only, bl only, $(\mathsf{tl}, \mathsf{br})$ only, or $(\mathsf{tl}, \mathsf{bl})$ only. Note that, as $u \neq v$, it can impossibly occur at $(\mathsf{tl}, \mathsf{tr})$ or $(\mathsf{bl}, \mathsf{br})$.

Query occurs at tl only. By $\neg\mathsf{help}_1(\mathcal{Q}_i)$, there are $\leq \tau_1$ choices for $(\mathsf{bl}, \mathsf{br})$. For any of these $\leq \tau_1$ choices, let K_{bl} and K_{br} be the key inputs corresponding to positions bl and br. The i^{th} query is successful only if its XOR-output equals $K^l_{\mathsf{br}} \| K^r_{\mathsf{bl}}$, which happens with probability at most $\frac{1}{2^n - q}$. The total success probability is at most $\frac{\tau_1}{2^n - q}$.

Query occurs at bl only. Let K be the key input for the i^{th} query. By $\neg\mathsf{help}_2(\mathcal{Q}_i)$ and $\neg\mathsf{help}_3(\mathcal{Q}_i)$, there are $\leq \tau_2$ choices for tl and $\leq \tau_2$ choices for tr. For any of these $\leq \tau_2^2$ choices, the query at position br is uniquely determined, and so is the XOR-output z of br. The i^{th} query is successful only if its XOR-output equals this value z, which happens with probability at most $\frac{1}{2^n - q}$. The total success probability is at most $\frac{\tau_2^2}{2^n - q}$.

Query occurs at tl and br only. Let K be the key input for the i^{th} query. As the query occurs at both positions, we require $K^l = z^l$, which fixes z^l. By $\neg\mathsf{help}_2(\mathcal{Q}_i)$, there are $\leq \tau_2$ choices for bl. For any of these $\leq \tau_2$ choices bl, we obtain a different z. The i^{th} query is successful only if its XOR-output equals this value z, which happens with probability at most $\frac{1}{2^n - q}$. The total success probability is at most $\frac{\tau_2}{2^n - q}$.

Query occurs at tl and bl only. Let K be the key input for the i^{th} query. As the query occurs at both positions, we require $K^r = z^r$, which fixes z^r. By

$\neg\mathsf{help}_3(\mathcal{Q}_i)$, there are $\leq \tau_2$ choices for br. For any of these $\leq \tau_2$ choices br, we obtain a different z. The i^{th} query is successful only if its XOR-output equals this value z, which happens with probability at most $\frac{1}{2^n-q}$. The total success probability is at most $\frac{\tau_2}{2^n-q}$.

The i^{th} query is successful with probability at most $\frac{\tau_1+2\tau_2+\tau_2^2}{2^n-q}$. The claimed bound is obtained by summing over $i = 1, \dots, q$. □

We are ready to finish the proof of Theorem 3.2.2. The findings of Section 3.2.1 and Lemma 3.2.11 imply for (3.11):

$$\mathbf{Adv}^{\text{col}}_{\text{MDC-4}}(q) \leq \frac{(2\tau_1 + 11\tau_1\tau_2 + 3\tau_1\tau_2\tau_3 + 14\tau_2 + 5\tau_2^2)q}{2^n-q} + \frac{(\tau_1+2\tau_2+\tau_2^2)q}{2^n-q} + \frac{q^2}{\tau_1(2^n-q)} + 2\cdot 2^{n/2}\left(\frac{eq2^{n/2}}{\tau_2(2^n-q)}\right)^{\tau_2} + 2^n\left(\frac{eq}{\tau_3(2^n-q)}\right)^{\tau_3},$$

This completes the proof of Theorem 3.2.2.

3.3 Preimage Resistance of MDC-4

We analyze the preimage security of MDC-4 in the case the two underlying block ciphers are identical, i.e., $E = E_1 = E_2$. Let (y, z) be the target image. We distinguish between $y = z$ and $y \neq z$.

$\boldsymbol{y = z}$**.** If the two halves of the image are the same, a preimage for the compression function $f_{\text{MDC-4}}$ can be found in about 2^n queries (cf. the introduction of this chapter): one focuses on preimages with the same left and right halves $u = v$, in which case it suffices to find u, w such that

$$E(E(u,w)\oplus w, u)\oplus u = y = z\,.$$

We demonstrate that this weakness propagates through the iteration of the MDC-4 hash function, resulting in an everywhere preimage attack for the MDC-4 hash function in 2^n queries (on average). We recall that everywhere preimage resistance is defined as the maximum advantage over all images, thus including the weak images consisting of two identical halves. If we had opted for preimage resistance where the challenge is randomly generated, this preimage attack succeeds only with small probability as 2^n out of 2^{2n} target images are weak.

The attack uses ideas from Knudsen et al. [127] to find preimages for MDC-2. It is a meet-in-the-middle attack and at a high level works as follows. First, one constructs a tree with 2^n leaves with root (y, z). The edges in this tree correspond to evaluations of $f_{\text{MDC-4}}$. In the general case, the construction of this tree requires the adversary to find approximately 2^{n+1} preimages, but as turns out for $f_{\text{MDC-4}}$ the workload is significantly lower. Then, starting from the initial value (g_0, h_0), one varies the message input w to hit any of the 2^n leaves. In more detail, the attack works as follows.

(i) Fix any $M_0, M_1 \in \mathbb{Z}_2^n$ such that $t\|M_b$ is a correct padding for any $b \in \{0,1\}$ and $t \in \mathbb{Z}_2^{n \cdot n}$;[2]

(ii) For $b = 0,1$ operate as follows. For any $u \in \mathbb{Z}_2^n$ query $r \leftarrow E(u, M_b)$ and $s \leftarrow E(r \oplus M_b, u)$. These queries correspond to the evaluation $f_{\text{MDC-4}}(u, u, M_b) = (s \oplus u, s \oplus u)$. Add the input-output tuple $((u,u); (s \oplus u, s \oplus u))$ to a list L_b;

(iii) Let (z,z) be the target image. On average, this item occurs once in each list L_0, L_1, which results in two $f_{\text{MDC-4}}$ preimages for (z,z). It may result in more than two $f_{\text{MDC-4}}$ preimages if (z,z) occurs multiple times in one of the lists. The same procedure can be iteratively executed for all resulting preimages, until a tree of approximately 2^n leaves is formed, with from each leave a path of n edges to (z,z);[3]

(iv) Starting from initial value (g_0, h_0), vary M until $f_{\text{MDC-4}}(g_0, h_0, M)$ hits any of the 2^n leaves.

Step (i) requires 2^{n+2} block cipher queries. Step (iv) is a brute force attack and requires approximately $4 \cdot (2^{2n}/2^n)$ evaluations of E. In total, this attack requires approximately 2^{n+3} queries. The attack has time and space complexity $O(2^n)$.

$y \neq z$. For the everywhere preimage resistance of the $f_{\text{MDC-4}}$ compression function of Figure 3.2 with the restriction that $y \neq z$, we derive the following results. The findings directly carry over to MDC-4 as it is a Merkle-Damgård transform, which preserves everywhere preimage resistance (Theorem 2.4.2).

Theorem 3.3.1. *Let $n \in \mathbb{Z}_2^n$. Suppose the provided the image (y,z) satisfies $y \neq z$. Then, for any positive integral values τ_1, τ_2, provided $\tau_1 \leq q$,*

$$\begin{aligned}\mathbf{Adv}_{f_{\text{MDC-4}}}^{\text{epre}(\neq)}(q) \leq \frac{4\tau_2^3 + 4\tau_1\tau_2 + 20 + 4 \cdot 2^{n/2}}{2^n} + \frac{16\tau_1\tau_2 + 24\tau_1\tau_2 2^{n/2}}{2^{2n}} + \frac{64q}{2^{3n}} + \\ 2 \cdot 2^{n/2}\left(\frac{4eq}{\tau_1 2^{n/2}}\right)^{\tau_1/2} + \frac{4q}{2^{n/2}}\left(\frac{8eq}{\tau_1 2^{n/2}}\right)^{\frac{\tau_1 2^n}{4q}} + \\ 2^n\left(\frac{4eq}{\tau_2 2^n}\right)^{\tau_2/2} + 2q\left(\frac{8eq}{\tau_2 2^n}\right)^{\frac{\tau_2 2^n}{4q}}. \qquad (3.12)\end{aligned}$$

The proof of Theorem 3.3.1 is given in Section 3.3.1. It employs ideas of the preimage resistance proof by Armknecht et al. [22] and Lee et al. [145, 147] for double block length compression functions, namely the issuance of free queries and the usage of wish lists. However, the analysis has become considerably more

[2] We assume the padding takes less than n bits.

[3] Due to collisions in the lists L_0, L_1, the amount of 2^n will usually not be reached. Elaborate statistical analysis shows that the average number of leaves at distance n from the root (z,z) varies between 2^{n-1} and 2^n.

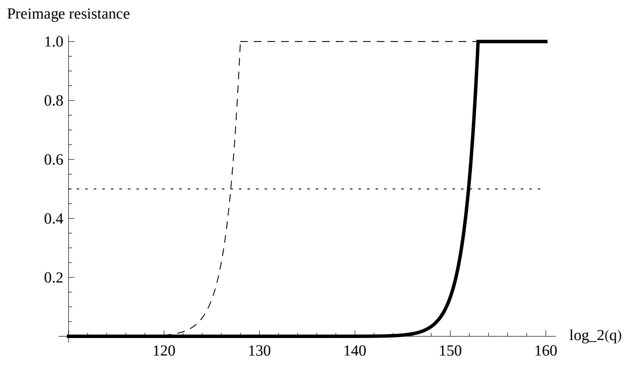


Figure 3.11. The function $\mathbf{Adv}^{\text{epre}(\neq)}_{f_{\text{MDC-4}}}(q)$ of (3.12) for $n = 128$ and the particular choice of values τ_1, τ_2 (solid line), in comparison with the best known bound $q/2^n$ (dashed line).

complex because the MDC-4 compression function uses four block ciphers rather than two, and consequently the derivation of bounds on the sizes of the wish lists has become more elaborate.

The bound of (3.12) can be analyzed in a similar manner as is done in Section 3.2, and we skip the details. Let $\varepsilon > 0$ be any parameter, we consider any adversary making at most $q = 2^{5n/4}/n^{\varepsilon}$ queries to its oracle. We set $\tau_1 = 2^{3n/4}$ and $\tau_2 = 2^{n/4}/n^{\varepsilon/2}$. Again, τ_1, τ_2 are assumed to be integral. Note that for interesting values of ε, we have $\tau_1 \leq q$ as desired. As before, it immediately follows that the bound of (3.12) approaches 0 for $n \to \infty$ when $q = 2^{5n/4}/n^{\varepsilon}$ and τ_1, τ_2 are as specified.

Corollary 3.3.2. *For any $\varepsilon > 0$, we obtain $\lim_{n\to\infty} \mathbf{Adv}^{\text{epre}(\neq)}_{f_{\text{MDC-4}}}\left(2^{5n/4}/n^{\varepsilon}\right) = 0$.*

The result means that for $n \to \infty$ the function $\mathbf{Adv}^{\text{epre}(\neq)}_{f_{\text{MDC-4}}}$ (as well as $\mathbf{Adv}^{\text{epre}(\neq)}_{\text{MDC-4}}$ by preimage resistance preservation) behaves as $q^4/2^{5n}$. a graphical representation of $\mathbf{Adv}^{\text{epre}(\neq)}_{f_{\text{MDC-4}}}$ for $n = 128$ is given in Figure 3.11. As in the case of Section 3.2, we have slightly adjusted the parameters τ_1, τ_2 to facilitate the analysis for smaller n and smaller q. For $n = 128$ the preimage resistance advantage hits 1/2 for $\log_2 q \approx 151.9$. Also in this case, the gap between this value and threshold for $q^4/2^{5n}$, 159.75, is caused by the choice for small n. By Corollary 3.3.2 the difference goes to 0 for $n \to \infty$.

3.3.1 Proof of Theorem 3.3.1

We consider any adversary making q queries to its oracle E, which tries to find a preimage for $f_{\text{MDC-4}}$. Let $(y, z) \in \mathbb{Z}_2^{2n}$ be the point to invert, chosen by the adversary prior to making any query. Finding a preimage for (y, z) corresponds to obtaining a query history $\mathcal{Q}_q$ of size q that satisfies configuration $\mathsf{pre}(\mathcal{Q}_q)$ of Figure 3.12. In other words,

$$\mathbf{Adv}^{\mathrm{epre}(\neq)}_{f_{\mathrm{MDC\text{-}4}}}(q) = \mathbf{Pr}\left(\mathsf{pre}(\mathcal{Q}_q)\right) , \tag{3.13}$$

and we consider the probability of obtaining any query history $\mathcal{Q}_q$ that satisfies configuration $\mathsf{pre}(\mathcal{Q}_q)$. As is done in Section 3.2.1, we again omit the bijective shifting at the end as it does not influence the preimage security. We use the same convention for the figures as is used in Section 3.2.1, with the difference that in Figure 3.12 the variables y, z are underlined to denote that these are fixed. As we only consider one word (rather than two, in Section 3.2.1), we label the block ciphers simply as tl, tr, bl, br for top/bottom left/right.

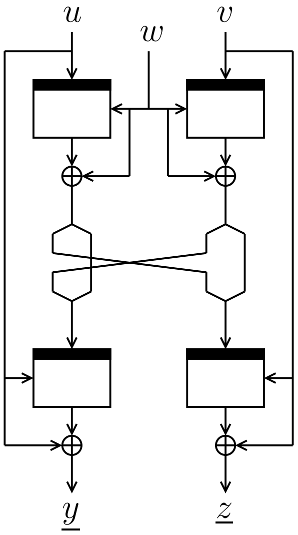


Figure 3.12. Configuration $\mathsf{pre}(\mathcal{Q})$. The configuration is satisfied if $\mathcal{Q}$ contains four (possibly not all different) queries that satisfy this setting. We have $y \neq z$.

The analysis in this section relies on the issuance of free super queries [22, 145, 147]. If the adversary has made 2^{n-1} queries to E under the same key, it will receive the remaining 2^{n-1} queries for this key for free. As in [22, 147], we call this query a super query. Formally, these free queries can be modeled as queries the adversary is forced to make, but at no charge. For convenience, we use $\mathcal{Q}_q$ to denote the query history after q normal queries. This query history thus contains all normal queries plus all super queries made so far. A super query is a set of 2^{n-1} single queries, and any query in the query history is either a normal query or a part of a super query, but not both. Notice that the adversary needs 2^{n-1} queries as preparatory work to enforce a super query. As the adversary makes at most q queries, at most $q/2^{n-1}$ super queries will occur.

For the analysis of $\mathsf{pre}(\mathcal{Q}_q)$, we introduce a helping event $\mathsf{help}(\mathcal{Q}_q)$. Let $\tau_1, \tau_2 > 0$ be integral. Event $\mathsf{help}(\mathcal{Q}_q)$ is satisfied if either of the following sub-events $\mathsf{help}_k(\mathcal{Q}_q)$ $(k = 1, 2, 3)$ occurs:

$$\begin{aligned}
\mathsf{help}_1(\mathcal{Q}_q):\ & \max\nolimits_{x\in\mathbb{Z}_2^{n/2}} \left|\left\{(K_i, m_i, c_i) \in \mathcal{Q}_q \mid (m_i \oplus c_i)^l = x\right\}\right| > \tau_1\,;\\
\mathsf{help}_2(\mathcal{Q}_q):\ & \max\nolimits_{x\in\mathbb{Z}_2^{n/2}} \left|\left\{(K_i, m_i, c_i) \in \mathcal{Q}_q \mid (m_i \oplus c_i)^r = x\right\}\right| > \tau_1\,;\\
\mathsf{help}_3(\mathcal{Q}_q):\ & \max\nolimits_{x\in\mathbb{Z}_2^{n}} \left|\left\{(K_i, m_i, c_i) \in \mathcal{Q}_q \mid m_i \oplus c_i = x\right\}\right| > \tau_2\,.
\end{aligned}$$

These helping events are the same as $\mathsf{help}_k(\mathcal{Q}_q)$ $(k = 2, 3, 4)$ used in the proof of collision resistance in Section 3.2.1, but are reintroduced for simplicity. Note that $\mathsf{help}_3(\mathcal{Q}_q)$ particularly covers the values y, z as XOR-outputs. By basic probability theory, we obtain for (3.13):

$$\mathbf{Pr}\left(\mathsf{pre}(\mathcal{Q}_q)\right) \le \mathbf{Pr}\left(\mathsf{pre}(\mathcal{Q}_q) \wedge \neg\mathsf{help}(\mathcal{Q}_q)\right) + \mathbf{Pr}\left(\mathsf{help}(\mathcal{Q}_q)\right)\,. \tag{3.14}$$

In Lemma 3.3.3, we bound $\mathbf{Pr}\left(\mathsf{pre}(\mathcal{Q}_q) \wedge \neg\mathsf{help}(\mathcal{Q}_q)\right)$. Regarding $\mathbf{Pr}\left(\mathsf{help}(\mathcal{Q}_q)\right)$, we note that, as we aim for preimage security beyond the birthday bound, the proof of Lemma 3.2.10 does not directly apply. It is analyzed in Lemma 3.3.4.

Lemma 3.3.3. $\mathbf{Pr}\left(\mathsf{pre}(\mathcal{Q}_q) \wedge \neg\mathsf{help}(\mathcal{Q}_q)\right) \le \frac{4\tau_2^3+4\tau_1\tau_2+20+4\cdot 2^{n/2}}{2^n} + \frac{16\tau_1\tau_2+24\tau_1\tau_2 2^{n/2}}{2^{2n}} + \frac{64q}{2^{3n}}$.

Proof. We consider the probability of the adversary finding a solution to configuration $\mathsf{pre}(\mathcal{Q}_q)$ of Figure 3.12, in such a way that $\mathcal{Q}_q$ satisfies $\neg\mathsf{help}(\mathcal{Q}_q)$. For a set of solutions complying with configuration $\mathsf{pre}(\mathcal{Q}_q)$, it may be the case that two queries are the same or belong to the same super query. We call a (normal or super) query winning if it makes the configuration satisfied for any other queries in the query history *strictly before* this winning query is made. We make the following distinction regarding the winning query:

(i) It contributes to exactly one position of configuration $\mathsf{pre}(\mathcal{Q}_q)$;
(ii) It contributes to exactly two positions of configuration $\mathsf{pre}(\mathcal{Q}_q)$;
(iii) It contributes to exactly three positions of configuration $\mathsf{pre}(\mathcal{Q}_q)$.

Note that a winning query cannot occur at all four positions: if this would be the case, we would have $u = v$ and thus $y = z$. In particular, in the remainder of the proof we will use that a winning *normal* query cannot occur at positions $(\mathsf{bl}, \mathsf{br})$, and a winning query (*normal or super*) cannot contribute at positions $(\mathsf{tl}, \mathsf{tr})$.

Case (i). In this case, the winning query may be a normal query or a super query. As in [145, 147], we make use of *wish lists* for the analysis of this case. Intuitively, a wish list is a continuously updated sequence of query tuples that would make configuration $\mathsf{pre}(\mathcal{Q})$ satisfied. During the attack of the adversary, we maintain four initially empty wish lists $\mathcal{W}_{\mathsf{tl}}$, $\mathcal{W}_{\mathsf{tr}}$, $\mathcal{W}_{\mathsf{bl}}$, $\mathcal{W}_{\mathsf{br}}$, corresponding to the four positions of configuration $\mathsf{pre}(\mathcal{Q})$. If a query is made, the wish lists are updated according to the following requirements:

- If the query fits $\mathsf{pre}_{\mathsf{tl}}(\mathcal{Q})$ of Figure 3.13 for any two other queries in the query history, the corresponding tuple $(u, w, (s\|t) \oplus w)$ is added to $\mathcal{W}_{\mathsf{tl}}$;
- If the query fits $\mathsf{pre}_{\mathsf{tr}}(\mathcal{Q})$ of Figure 3.13 for any two other queries in the query history, the corresponding tuple $(v, w, (s\|t) \oplus w)$ is added to $\mathcal{W}_{\mathsf{tr}}$;
- If the query fits $\mathsf{pre}_{\mathsf{bl}}(\mathcal{Q})$ of Figure 3.13 for any two other queries in the query history, the corresponding tuple $(s\|t, u, y \oplus u)$ is added to $\mathcal{W}_{\mathsf{bl}}$;
- If the query fits $\mathsf{pre}_{\mathsf{br}}(\mathcal{Q})$ of Figure 3.13 for any two other queries in the query history, the corresponding tuple $(s\|t, v, z \oplus v)$ is added to $\mathcal{W}_{\mathsf{br}}$.

As in this case we consider the winning query to be different from all other queries made before, we can assume a query never adds itself to a wish list. It is clear that the adversary finds a preimage for MDC-4 (in this case) only if it makes a query that is already a member of any of the wish lists. Suppose the adversary makes a query $E(K, m)$ (either as a normal query or as a part of a super query), and suppose $(K, m, c) \in \mathcal{W}_{\mathsf{tl}} \cup \mathcal{W}_{\mathsf{br}}$ for some c. Then, we say that (K, m, c) is wished for, and the wish is granted if the response of the block cipher is c. Similar naming is used for inverse queries to E. Notice that the adversary may wish for multiple queries at the same time, but this does not invalidate the analysis. Additionally, each wish list element can be wished for only once. In order to find a preimage, the adversary needs at least a wish to be granted. Let (K, m, c) be an element in any of the wish lists, and suppose the adversary makes a query $E(K, m)$ or $E^{-1}(K, c)$. In case of normal queries, the answer is generated from a set of size at least 2^{n-1}, and the wish is granted with probability at most $1/2^{n-1}$. In case the query is a part of a super query, the answer is generated from a set of size exactly 2^{n-1} and the wish is also granted with probability at most $1/2^{n-1}$. Because each element of the wish lists can be wished for only once, the adversary finds a preimage with probability at most

$$\frac{|\mathcal{W}_{\mathsf{tl}}| + |\mathcal{W}_{\mathsf{tr}}| + |\mathcal{W}_{\mathsf{bl}}| + |\mathcal{W}_{\mathsf{br}}|}{2^{n-1}} .$$

It remains to bound the sizes of the wish lists after q queries. Configuration $\mathsf{pre}_{\mathsf{tl}}(\mathcal{Q}_q)$ of Figure 3.13 has $\leq \tau_2$ solutions for each bl and br (by $\neg\mathsf{help}_3(\mathcal{Q}_q)$), and consequently $\leq \tau_2$ solutions for tr (by $\neg\mathsf{help}_3(\mathcal{Q}_q)$). Thus $|\mathcal{W}_{\mathsf{tl}}| \leq \tau_2^3$, and similarly we obtain $|\mathcal{W}_{\mathsf{tr}}| \leq \tau_2^3$. Configuration $\mathsf{pre}_{\mathsf{bl}}(\mathcal{Q}_q)$ of Figure 3.13 has $\leq \tau_2$ solutions for br (by $\neg\mathsf{help}_3(\mathcal{Q}_q)$), and consequently $\leq \tau_1$ solutions for tl (by $\neg\mathsf{help}_1(\mathcal{Q}_q)$). For any of these $\leq \tau_1\tau_2$ choices, the query at position tr is uniquely determined (if it exists at all). Thus $|\mathcal{W}_{\mathsf{bl}}| \leq \tau_1\tau_2$, and similarly $|\mathcal{W}_{\mathsf{br}}| \leq \tau_1\tau_2$ (using $\neg\mathsf{help}_2(\mathcal{Q}_q)$). Hence, in this case a preimage is found with probability at most $\frac{4\tau_2^3+4\tau_1\tau_2}{2^n}$.

Case (ii). We make the following distinction, and consider the two sub-cases separately:

(a) The contributed queries are different for both positions;
(b) The contributed queries are the same for both positions.

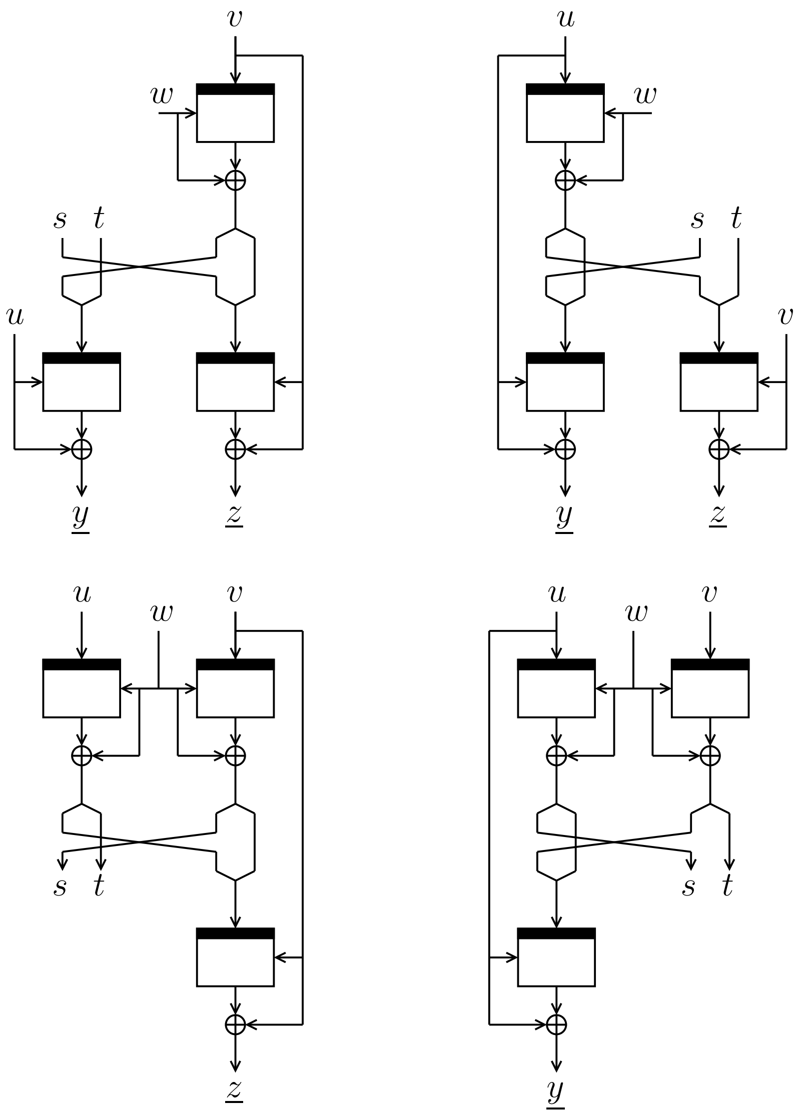


Figure 3.13. Left to right, top to bottom: configurations $\mathsf{pre}_{\mathsf{tl}}(\mathcal{Q})$, $\mathsf{pre}_{\mathsf{tr}}(\mathcal{Q})$, $\mathsf{pre}_{\mathsf{bl}}(\mathcal{Q})$, and $\mathsf{pre}_{\mathsf{br}}(\mathcal{Q})$.

Case (iia). In this particular case, the winning query must be a super query. Similar to case (i), we make use of wish lists, but now for the specific case that a super query contributes two queries to a configuration. As a super query cannot contribute to $(\mathsf{tl}, \mathsf{tr})$, it can only contribute to positions $(\mathsf{tl}, \mathsf{br})$, $(\mathsf{tr}, \mathsf{bl})$, $(\mathsf{tl}, \mathsf{bl})$,

$(\mathsf{tr},\mathsf{br})$, or $(\mathsf{bl},\mathsf{br})$. Note that if a super query contributes to positions $(\mathsf{tl},\mathsf{br})$, the left half of the XOR-output of tl should equal the left half of the key input to br, which is the same as the key input to tl (similar for super queries contributing to $(\mathsf{tr},\mathsf{bl})$). Note that if a super query contributes to positions $(\mathsf{tl},\mathsf{bl})$, the key input to bl equals the key input to tl which equals the message input to bl (similar for super queries contributing to $(\mathsf{tr},\mathsf{br})$). Also, note that if a super query contributes to positions $(\mathsf{bl},\mathsf{br})$, the XOR-outputs of tl and tr must be the same. During the attack of the adversary, we maintain five initially empty wish lists $\mathcal{W}_1,\dots,\mathcal{W}_5$, corresponding to above cases. If a query is made by the adversary, the wish lists are updated according to the following requirements:

- If the query fits $\mathsf{pre}_1(\mathcal{Q})$ of Figure 3.14 for any query in the query history, the corresponding tuple $(r\|t, w, (r\|s)\oplus w, v, z\oplus v)$ is added to $\mathcal{W}_1$;
- If the query fits $\mathsf{pre}_2(\mathcal{Q})$ of Figure 3.14 for any query in the query history, the corresponding tuple $(r\|t, w, (r\|s)\oplus w, u, y\oplus u)$ is added to $\mathcal{W}_2$;
- If the query fits $\mathsf{pre}_3(\mathcal{Q})$ of Figure 3.14 for any query in the query history, the corresponding tuple $(t\|s, w, (r\|s)\oplus w, t\|s, y\oplus(t\|s))$ is added to $\mathcal{W}_3$;
- If the query fits $\mathsf{pre}_4(\mathcal{Q})$ of Figure 3.14 for any query in the query history, the corresponding tuple $(t\|s, w, (r\|s)\oplus w, t\|s, z\oplus(t\|s))$ is added to $\mathcal{W}_4$;
- If the query fits $\mathsf{pre}_5(\mathcal{Q})$ of Figure 3.15 for any query in the query history, the corresponding tuple $(s\|t, u, y\oplus u, v, z\oplus v)$ is added to $\mathcal{W}_5$.

Of these tuples, the first element identifies the key for which the super query is made, the second and third element define the input and output of the cipher in the top row (either left or right), and the fourth and fifth element define the input and output of the cipher in the bottom row (either right or left). For $\mathcal{W}_5$, the second and third element correspond to bl and the fourth and fifth element to br. A query trivially does not add itself to the wish list (as these cases would be covered by Case (iii)). Suppose the adversary makes a super query to E for key K, and suppose $(K, m_{\mathsf{tl}}, c_{\mathsf{tl}}, m_{\mathsf{br}}, c_{\mathsf{br}})\in\mathcal{W}_1$ for some $m_{\mathsf{tl}}, c_{\mathsf{tl}}, m_{\mathsf{br}}, c_{\mathsf{br}}$. This wish is then granted if the response satisfies $c_{\mathsf{tl}} = E(K, m_{\mathsf{tl}})$ and $c_{\mathsf{br}} = E(K, m_{\mathsf{br}})$. In order to find a preimage, the adversary needs at least a wish to be granted. As the answers are generated from a set of size exactly 2^{n-1}, a wish is granted with probability at most $\frac{1}{2^{n-1}(2^{n-1}-1)}$. Because each element of the wish lists can be wished for only once, the adversary finds a preimage with probability at most

$$\frac{|\mathcal{W}_1|+|\mathcal{W}_2|+|\mathcal{W}_3|+|\mathcal{W}_4|+|\mathcal{W}_5|}{2^{n-1}(2^{n-1}-1)}.$$

It remains to bound the sizes of the wish lists after q queries. Configuration $\mathsf{pre}_1(\mathcal{Q}_q)$ has $\le\tau_2$ solutions for bl (by $\neg\mathsf{help}_3(\mathcal{Q}_q)$), and at most $\le\tau_1$ solutions for tr (by $\neg\mathsf{help}_1(\mathcal{Q}_q)$). Thus, $|\mathcal{W}_1|\le\tau_1\tau_2$ and similarly $|\mathcal{W}_2|\le\tau_1\tau_2$. Configuration $\mathsf{pre}_3(\mathcal{Q}_q)$ has $\le\tau_2$ solutions for br (by $\neg\mathsf{help}_3(\mathcal{Q}_q)$), and at most $\le\tau_1$ solutions for tr (by $\neg\mathsf{help}_2(\mathcal{Q}_q)$). Additionally, there are $2^{n/2}$ possibilities for s. Thus $|\mathcal{W}_3|\le\tau_1\tau_2 2^{n/2}$ and similarly $|\mathcal{W}_4|\le\tau_1\tau_2 2^{n/2}$. Configuration $\mathsf{pre}_5(\mathcal{Q}_q)$ has $2^{n/2}$

choices for s, for any of these choices it has $\leq \tau_1$ solutions for tr (by $\neg\mathsf{help}_1(\mathcal{Q}_q)$), and consequently $\leq \tau_2$ solutions for tl (by $\neg\mathsf{help}_3(\mathcal{Q}_q)$). Thus also $|\mathcal{W}_5| \leq \tau_1\tau_2 2^{n/2}$. Hence, in this case a preimage is found with probability at most $\frac{16\tau_1\tau_2+24\tau_1\tau_2 2^{n/2}}{2^{2n}}$.

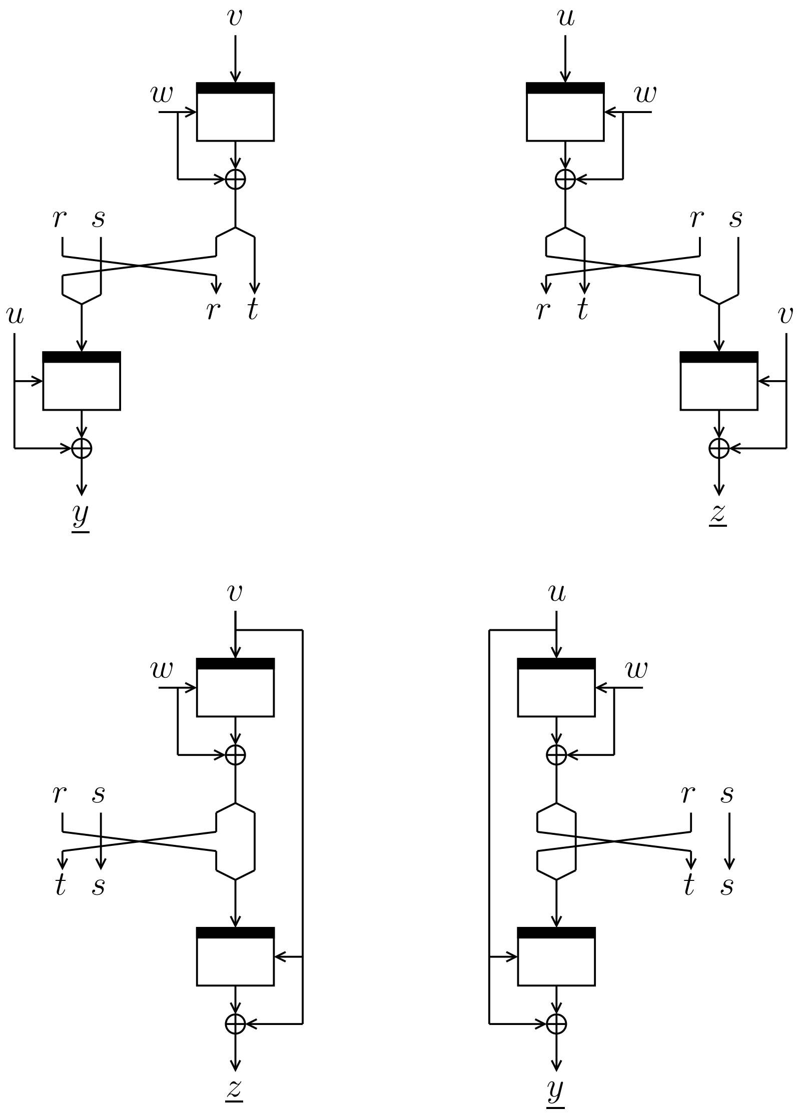


Figure 3.14. Left to right, top to bottom: configurations $\mathsf{pre}_1(\mathcal{Q}), \ldots, \mathsf{pre}_4(\mathcal{Q})$.

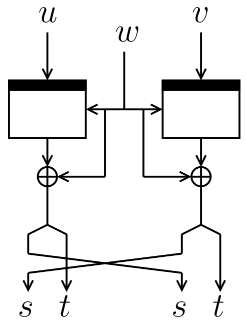


Figure 3.15. Configuration $\mathsf{pre}_5(\mathcal{Q})$.

Case (iib). In this case, the winning query may be a normal query or a super query. The winning query can only contribute to positions $(\mathsf{tl} = \mathsf{br})$, $(\mathsf{tr} = \mathsf{bl})$, $(\mathsf{tl} = \mathsf{bl})$, or $(\mathsf{tr} = \mathsf{br})$.

We first consider the winning query to contribute to $\mathsf{tl} = \mathsf{br}$. Suppose the adversary makes a query $c \leftarrow E(K, m)$ (either as a normal query or as a part of a super query). As it occurs at position br we require $m \oplus c = z$ (because of this, the analysis for inverse queries is equivalent). Additionally, the query at position tr should have key input as well as message input equal to m. This particularly means that the key input K to $\mathsf{tl} = \mathsf{br}$ must satisfy $K = z^l \| (E(m,m) \oplus m)^r$. By construction of the queries at positions bl, br, the adversary can only succeed if it ever finds an $m \in \mathbb{Z}_2^n$ that satisfies

$$E((E(m,m) \oplus m)^l \| z^r, z^l \| (E(m,m) \oplus m)^r) \oplus z^l \| (E(m,m) \oplus m)^r = y \text{ and}$$
$$E(z^l \| (E(m,m) \oplus m)^r, m) \oplus m = z\,.$$

As y and z are fixed, the adversary finds such m with probability at most $\frac{2^n}{2^{n-1}2^{n-1}} = \frac{4}{2^n}$. The same probability bound is obtained for winning queries to appear at $(\mathsf{tr}, \mathsf{bl})$.

Consider a query contributing to $\mathsf{tl} = \mathsf{bl}$. By construction, this query must be of the form $E(K, K) = c$ where $K \oplus c = y$ and $K^r = y^r$. As y is fixed, the adversary finds such query with probability at most $\frac{2^{n/2}}{2^{n-1}} = \frac{2 \cdot 2^{n/2}}{2^n}$ (either in case of forward or inverse query). The same probability bound is obtained for winning queries to appear at $(\mathsf{tr}, \mathsf{br})$.

Consequently, a preimage is found in this case with probability at most $\frac{8+4\cdot 2^{n/2}}{2^n}$.

Case (iii). Recall that a query can never contribute to $(\mathsf{tl}, \mathsf{tr})$ at the same time. Therefore, we only need to consider queries contributing at positions $(\mathsf{tl}, \mathsf{bl}, \mathsf{br})$ or $(\mathsf{tr}, \mathsf{bl}, \mathsf{br})$. We make the following distinction, and consider the two sub-cases separately:

(a) The contributed queries are different for all positions;
(b) The contributed queries are the same at two positions.

Note that as the queries at $(\mathsf{bl}, \mathsf{br})$ cannot be the same, there is no need to consider the case all three queries are the same.
Case (iiia). As before, we consider two wish lists $\mathcal{W}_{\mathsf{tl}}$, $\mathcal{W}_{\mathsf{tr}}$, corresponding to position to which the winning query does not contribute. Note that if a super query contributes to positions $(\mathsf{tr}, \mathsf{bl}, \mathsf{br})$, the XOR-outputs of tl and tr must be the same, and equal to the key input to tr. In order words, any query to tl fixes exactly one wish list tuple in $\mathcal{W}_{\mathsf{tl}}$. In more detail, if a query $E(K, m) = c$ is made the wish lists are updated as follows.

- The tuple $(m \oplus c, m, c, K, y \oplus K, m \oplus c, z \oplus m \oplus c)$ is added to $\mathcal{W}_{\mathsf{tl}}$;
- The tuple $(m \oplus c, m, c, m \oplus c, y \oplus m \oplus c, K, z \oplus K)$ is added to $\mathcal{W}_{\mathsf{tr}}$.

Of these tuples, the first element identifies the key, the second and third element define the input and output of the cipher in the top row (either left or right), the fourth and fifth element define the input and output of the cipher at bl and the sixth and seventh element the input and output of the cipher at br. Because each element of the wish lists can be wished for only once, the adversary finds a preimage with probability at most

$$\frac{|\mathcal{W}_{\mathsf{tl}}| + |\mathcal{W}_{\mathsf{tr}}|}{2^{n-1}(2^{n-1} - 1)(2^{n-1} - 2)} .$$

Clearly, $|\mathcal{W}_{\mathsf{tl}}|, |\mathcal{W}_{\mathsf{tr}}| \leq q$. Hence, in this case a preimage is found with probability at most $\frac{64q}{2^{3n}}$.
Case (iiib). Note that the same query can only occur at positions $(\mathsf{tl} = \mathsf{br})$, $(\mathsf{tr} = \mathsf{bl})$, $(\mathsf{tl} = \mathsf{bl})$, or $(\mathsf{tr} = \mathsf{br})$. The analysis of case (iib) carries over directly, but for the latter two scenarios we can do better. Consider a query contributing to $\mathsf{tl} = \mathsf{bl}$. Note that, as the super query also contributes to position br, the key inputs to $\mathsf{tl}, \mathsf{bl}, \mathsf{br}$ are the same. By construction, the query at $\mathsf{tl} = \mathsf{bl}$ must be of the form $E(K, K) = c$ where $K \oplus c = y$ and $K = y$. As y is fixed, the adversary finds such query with probability at most $\frac{1}{2^{n-1}} = \frac{2}{2^n}$ (either in case of forward or inverse query). The same probability bound is obtained for winning queries to appear at $(\mathsf{tr} = \mathsf{br})$. Consequently, a preimage is found in this case with probability at most $\frac{12}{2^n}$.

The claim is obtained by summing the bounds obtained for the three cases. $\square$

Lemma 3.3.4. *Provided $\tau_1 \leq q$, we have* $\mathbf{Pr}\left(\mathsf{help}(\mathcal{Q}_q)\right) \leq 2 \cdot 2^{n/2} \left(\frac{4eq}{\tau_1 2^{n/2}}\right)^{\tau_1/2} + \frac{4q}{2^{n/2}} \left(\frac{8eq}{\tau_1 2^{n/2}}\right)^{\frac{\tau_1 2^n}{4q}} + 2^n \left(\frac{4eq}{\tau_2 2^n}\right)^{\tau_2/2} + 2q \left(\frac{8eq}{\tau_2 2^n}\right)^{\frac{\tau_2 2^n}{4q}}$.

Proof. It suffices to consider the events $\mathbf{Pr}\left(\mathsf{help}_k(\mathcal{Q}_q)\right)$ $(k = 1, 2, 3)$ separately.
$\mathsf{help}_k(\mathcal{Q}_q)$ for $k \in \{1, 2\}$. The cases are equivalent by symmetry, and we consider $\mathsf{help}_1(\mathcal{Q}_q)$ only. Let $x \in \mathbb{Z}_2^{n/2}$. Denote by $\mathcal{Q}_q^{(n)}$ the restriction of $\mathcal{Q}_q$ to normal queries, and by $\mathcal{Q}_q^{(s)}$ the restriction of $\mathcal{Q}_q$ to queries that belong to super queries. In order for $\mathcal{Q}_q$ to have more than τ_1 solutions to $(m_i \oplus c_i)^l = x$, at least one of the following criteria needs to hold:

(i) $\mathcal{Q}_q^{(n)}$ has more than $\tau_1/2$ solutions;
(ii) $\mathcal{Q}_q^{(s)}$ has more than $\tau_1/2$ solutions.

We consider these two scenarios separately. In case of normal queries, each query (K_i, m_i, c_i) is answered with a value generated at random from a set of size at least 2^{n-1}, and hence it satisfies $(m_i \oplus c_i)^l = x$ with probability at most $\frac{2^{n/2}}{2^{n-1}} = \frac{2}{2^{n/2}}$. More than $\tau_1/2$ queries result in a solution with probability at most $\binom{q}{\tau_1/2}\left(\frac{2}{2^{n/2}}\right)^{\tau_1/2} \leq \left(\frac{4eq}{\tau_1 2^{n/2}}\right)^{\tau_1/2}$.

The analysis for super queries is more elaborate. In order for $\mathcal{Q}_q^{(s)}$ to have more than $\tau_1/2$ solutions, as at most $q/2^{n-1}$ super queries occur, at least one of the super queries needs to provide more than $\tau_1' := \frac{\tau_1}{2q/2^{n-1}} = \frac{\tau_1 2^n}{4q}$ solutions. Consider any super query, consisting of 2^{n-1} queries. It provides more than τ_1' solutions with probability at most

$$\binom{2^{n-1}}{\tau_1'} \prod_{j=0}^{\tau_1'-1} \frac{2^{n/2}}{2^{n-1} - j} \leq \binom{2^{n-1}}{\tau_1'} \left(\frac{2^{n/2}}{2^{n-1} - \tau_1'}\right)^{\tau_1'} \leq \left(\frac{e 2^{n-1} 2^{n/2}}{\tau_1'(2^{n-1} - \tau_1')}\right)^{\tau_1'}.$$

Provided $\tau_1 \leq q$, we have $\tau_1' = \frac{\tau_1 2^n}{4q} \leq 2^{n-2}$, and thus $\frac{1}{2^{n-1} - \tau_1'} \leq \frac{1}{2^{n-2}}$. Consequently, this super query adds more than $\frac{\tau_1 2^n}{4q}$ solutions with probability at most $\left(\frac{8eq}{\tau_1 2^{n/2}}\right)^{\frac{\tau_1 2^n}{4q}}$. In order to cover any super query, we need to multiply this probability with $q/2^{n-1}$.

Considering any possibly choice for x, we obtain for $k = 1, 2$:

$$\mathbf{Pr}\left(\mathsf{help}_k(\mathcal{Q}_q)\right) \leq 2^{n/2} \left(\frac{4eq}{\tau_1 2^{n/2}}\right)^{\tau_1/2} + 2^{n/2} \cdot \frac{q}{2^{n-1}} \left(\frac{8eq}{\tau_1 2^{n/2}}\right)^{\frac{\tau_1 2^n}{4q}}. \tag{3.15}$$

$\mathsf{help}_3(\mathcal{Q}_q)$. A similar analysis as for $\mathsf{help}_1(\mathcal{Q}_q)$ results in the following bound:

$$\mathbf{Pr}\left(\mathsf{help}_3(\mathcal{Q}_q)\right) \leq 2^n \left(\frac{4eq}{\tau_2 2^n}\right)^{\tau_2/2} + 2^n \cdot \frac{q}{2^{n-1}} \left(\frac{8eq}{\tau_2 2^n}\right)^{\frac{\tau_2 2^n}{4q}}. \tag{3.16}$$

The claim is obtained by adding (3.15) (twice) and (3.16). □

With respect to Lemma 3.3.4, we note that $\mathsf{help}_3(\mathcal{Q})$ is similar to the event $\mathsf{Lucky}(\mathcal{Q})$ analyzed by Armknecht et al. [22] and Lee et al. [147]; the only difference is that $\mathsf{help}_3(\mathcal{Q})$ is required to hold for any $x \in \mathbb{Z}_2^n$. In their analysis of $\mathsf{Lucky}(\mathcal{Q})$, [22, 147] make a distinction between the normal and super queries (just as we do in the proof of Lemma 3.3.4) but for the super queries their analysis is based on Markov's inequality and is consequently much simpler. However, because our helping events are required to hold for any $x \in \mathbb{Z}_2^n$ (event $\mathsf{help}_3(\mathcal{Q})$) and for any $x \in \mathbb{Z}_2^{n/2}$ (event $\mathsf{help}_{1\vee 2}(\mathcal{Q})$), a similar approach using Markov's inequality would result in a trivial bound and a more elaborate treatment was required.

From (3.13-3.14) and Lemmas 3.3.3-3.3.4 we conclude for $\mathbf{Adv}^{\mathrm{epre}(\neq)}_{f_{\text{MDC-4}}}(q)$:

$$\begin{aligned}\mathbf{Adv}^{\mathrm{epre}(\neq)}_{f_{\text{MDC-4}}}(q) \leq{}& \frac{4\tau_2^3 + 4\tau_1\tau_2 + 20 + 4\cdot 2^{n/2}}{2^n} + \frac{16\tau_1\tau_2 + 24\tau_1\tau_2 2^{n/2}}{2^{2n}} + \frac{64q}{2^{3n}} + \\ & 2\cdot 2^{n/2}\left(\frac{4eq}{\tau_1 2^{n/2}}\right)^{\tau_1/2} + \frac{4q}{2^{n/2}}\left(\frac{8eq}{\tau_1 2^{n/2}}\right)^{\frac{\tau_1 2^n}{4q}} + \\ & 2^n\left(\frac{4eq}{\tau_2 2^n}\right)^{\tau_2/2} + 2q\left(\frac{8eq}{\tau_2 2^n}\right)^{\frac{\tau_2 2^n}{4q}} .\end{aligned}$$

This completes the proof of Theorem 3.3.1.

3.4 Security of MDC-4 with Two Distinct Block Ciphers E_1, E_2

We consider the collision and preimage resistance of MDC-4 in the setting where the two block ciphers E_1, E_2 are independently distributed. Although the analysis of MDC-4 with one block cipher is more general, the MDC-4 mode of operation with two independent block cipher is much closer to the original design due to the domain separation by the functions β and γ (see the introduction of this chapter). In this setting, the following results can be obtained by straightforward simplifications of the proofs of Theorems 3.2.1 and 3.3.1.

Starting with collision resistance, we obtain the following results. They directly carry over to MDC-4 as it is a Merkle-Damgård transform, which preserves collision resistance (Theorem 2.4.1). We stress that we pose no limitation on the state values.

Theorem 3.4.1. *Let $n \in \mathbb{Z}_2^n$. Then, for any positive integral values τ_1, τ_2, τ_3,*

$$\begin{aligned}\mathbf{Adv}^{\mathrm{col}}_{f_{\text{MDC-4}}}(q) \leq{}& \frac{(\tau_1 + 7\tau_1\tau_2 + 3\tau_1\tau_2\tau_3 + 5\tau_2 + 4\tau_2^2)q}{2^n - q} + \frac{q^2}{\tau_1(2^n - q)} + \\ & 2\cdot 2^{n/2}\left(\frac{eq2^{n/2}}{\tau_2(2^n - q)}\right)^{\tau_2} + 2^n\left(\frac{eq}{\tau_3(2^n - q)}\right)^{\tau_3} .\end{aligned} \tag{3.17}$$

In the proof of Theorem 3.2.1, the restrictions $u_1 \neq v_1$ and $u_2 \neq v_2$ are essentially used to guarantee that the queries occurring at positions $(1\mathsf{tl}, 1\mathsf{tr})$ are distinct, and so are the ones at $(1\mathsf{bl}, 1\mathsf{br})$. Regarding Theorem 3.4.1, E_1, E_2 are independently distributed this is directly guaranteed. The bound of Theorem 3.4.1 immediately follows by leaving out some cases in the proof of Theorem 3.2.1 (such as for $\mathsf{col}_{0000}(\mathcal{Q}_q)$ the case that the last query appears at $(1\mathsf{tl}, 1\mathsf{bl})$). By a similar reasoning as before, Corollary 3.2.3 applies to $\mathbf{Adv}^{\mathrm{col}}_{f_{\text{MDC-4}}}(q)$ too.

By the same arguments, we find the following bound on the preimage resistance. Corollary 3.3.2 applies to $\mathbf{Adv}^{\mathrm{epre}}_{f_{\text{MDC-4}}}(q)$ too. The results directly carry over to MDC-4 as it is a Merkle-Damgård transform, which preserves everywhere preimage resistance (Theorem 2.4.2).

Theorem 3.4.2. *Let $n \in \mathbb{Z}_2^n$. Then, for any positive integral values τ_1, τ_2, provided $\tau_1 \leq q$,*

$$\begin{aligned}\mathbf{Adv}^{\mathrm{epre}}_{f_{\text{MDC-4}}}(q) \leq \frac{4\tau_2^3 + 4\tau_1\tau_2}{2^n} + \frac{16\tau_1\tau_2}{2^{2n}} + \frac{8}{2^n} + 2 \cdot 2^{n/2}\left(\frac{4eq}{\tau_1 2^{n/2}}\right)^{\tau_1/2} + \\ \frac{4q}{2^{n/2}}\left(\frac{8eq}{\tau_1 2^{n/2}}\right)^{\frac{\tau_1 2^n}{4q}} + 2^n\left(\frac{4eq}{\tau_2 2^n}\right)^{\tau_2/2} + 2q\left(\frac{8eq}{\tau_2 2^n}\right)^{\frac{\tau_2 2^n}{4q}}.\end{aligned} \tag{3.18}$$

3.5 Conclusions

With the $2^{5n/8}$ collision and $2^{5n/4}$ preimage security bounds we have formally confirmed the widespread belief that MDC-4 offers a higher level of security compared to MDC-2. Despite that this security gain is obtained at the price of efficiency loss, it is an interesting and important result that allows us to make a fairer comparison among the double block length hash functions and that gives us more insight in the possibilities and impossibilities of block cipher based hashing (see Section 2.6.1). In particular, to our knowledge this is the first time the preimage resistance of a double block length compression function design with more than two block cipher evaluations is analyzed. Given that MDC-4 is originally not constructed from a provable security point of view (but rather an efficiency point of view), more elaborate designs with more than two block cipher calls may likely offer a higher level of security; we consider this goal in Chapter 4. Although our findings improve the existing bounds on MDC-4 significantly, large gaps between the security bounds and the best known attacks remain. A more technical and elaborate analysis may result in better bounds, and it remains an interesting open problem to improve the security bounds or the generic attacks for MDC-2 and MDC-4. Additionally, we only considered MDC-2 and MDC-4 in a Merkle-Damgård mode of operation, and it is of interest to investigate if the compression functions render a higher level of security in other modes of operation (see Section 2.5).

4 Optimal Collision Security in Double-Length Hashing with Single-Length Key

This chapter studies double length compression functions that compress $3n$ to $2n$ bits. Recall from Section 2.6 that those can be classified into two classes: compression functions that internally evaluate a $2n$-bit keyed block cipher $E : \mathbb{Z}_2^{2n} \times \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ (the DBL^{2n} class), and ones that employ an n-bit keyed block cipher $E : \mathbb{Z}_2^n \times \mathbb{Z}_2^n \to \mathbb{Z}_2^n$ (the DBL^n class). The DBL^{2n} class is well understood, but on the other hand, for the DBL^n class no optimally secure compression function is known. The situation is the same for the iteration, where none of these designs has been proven to achieve optimal security. Determinative to this gap is the difference in the underlying primitive: in the DBL^{2n} class, the underlying primitive maps $3n$ bits to n bits and thus allows for more compression. In particular, if we look at Tandem-DM, Abreast-DM, and Hirose's function (Figure 2.3), the first cipher call already compresses the entire input (u, v, w) to the compression function, and the second cipher call is simply used to assure a $2n$-bit output. In fact, these designs achieve their level of security merely due to this property, for their proofs crucially rely on this (see also Section 4.3).

From a more theoretical point of view, one can consider the strength of the extra n bits of key input as follows. If in the Tandem-DM, Abreast-DM, and Hirose's compression function one leaves out the w input, one obtains an optimally collision and preimage secure $2n$-to-$2n$-bit random function and the existing proofs apply. Now, the extra n bits of key input are simply used to compress w and the security proofs carry over at no extra cost. Similarly, one can easily construct an optimally collision and preimage secure $100n$-to-$2n$-bit compression function from an n-bit block cipher, provided the key input to this block cipher is of size $99n$ bits.[1]

[1]We stress that this is no attempt to ridicule the well-established double length hashing designs: we just want to point out the influence of a larger key space to the security proofs.

Contributions of This Chapter

From a theoretical point of view it is unreasonable to compare DBL^{2n} and DBL^{n}. But the gap between the two classes leaves us with an interesting open problem: starting from a single block cipher $E : \mathbb{Z}_2^n \times \mathbb{Z}_2^n \to \mathbb{Z}_2^n$, is it possible to construct a double length compression function that achieves optimal collision and preimage security? In this chapter, we analyze this problem. Note that Stam's bound [209] (see Section 2.6.2) does not help us here: it claims collisions can be found in at most $(2^n)^{(2k-1)/(k+1)}$ queries, where k denotes the number of block cipher calls, which results in trivial bound for $k \geq 2$. For $k \geq 2$, denote by $\mathsf{F}^k : \mathbb{Z}_2^{3n} \to \mathbb{Z}_2^{2n}$ a compression function that makes k calls to its primitive E.

As a first contribution, we consider F^2, and prove that for a very large class of functions of this form one expects collisions in approximately $2^{n/2}$ queries. Covered by the attack are among others designs with linear finalization function (the function that produces the $2n$-bit output given the $3n$-bit input and the block cipher responses). We note that the compression function by Jetchev et al. [118] is not vulnerable to the attack due to its non-linear finalization function. Nevertheless, this result strengthens the claim that no practical optimally collision secure F^2 function exists. Motivated by this, we increase the number of calls to E, and consider F^3. In this setting, we derive a family of compression functions which we prove asymptotically optimally collision resistant up to 2^n queries and preimage resistant up to $2^{3n/2}$ queries, asymptotically. Our compression function family, thus, achieves the same level of collision security as the well-established Tandem-DM, Abreast-DM, and Hirose's function, albeit based on a much weaker assumption. In the DBL^n class, our design clearly compares favorably to MDC-4 that makes four block cipher evaluations, and from a provable security point of view it beats MDC-2 and MJH, still, an extra E evaluation has to be made which results in an efficiency loss. The introduced class of compression functions is simple and easy to understand: they are defined by 4×4 matrices over the field $GF(2^n)$ which are required to comply with easily satisfied conditions. Two example compression functions in this class are given in Figure 4.1.

The security proofs of our compression function family rely on basic principles from previous proofs, but in order to accomplish optimal collision security (and as our designs use n-bit keyed block ciphers) our proofs have become significantly more complex. The collision and preimage security proofs of all known DBL^{2n} functions (see Table 2.1) crucially rely on the property that one block cipher evaluation defines the input to the second one. For F^3 this cannot be achieved as each primitive call fixes at most $2n$ bits of the function input. Although one may expect this to cause an optimal proof to become unlikely, this is not the case. Using a recent proof approach — we smartly apply the methodology of *wish lists* (by Armknecht et al. and Lee et al. [22, 147]) to collision resistance — we manage to achieve asymptotically close to 2^n collision security for our family of functions.

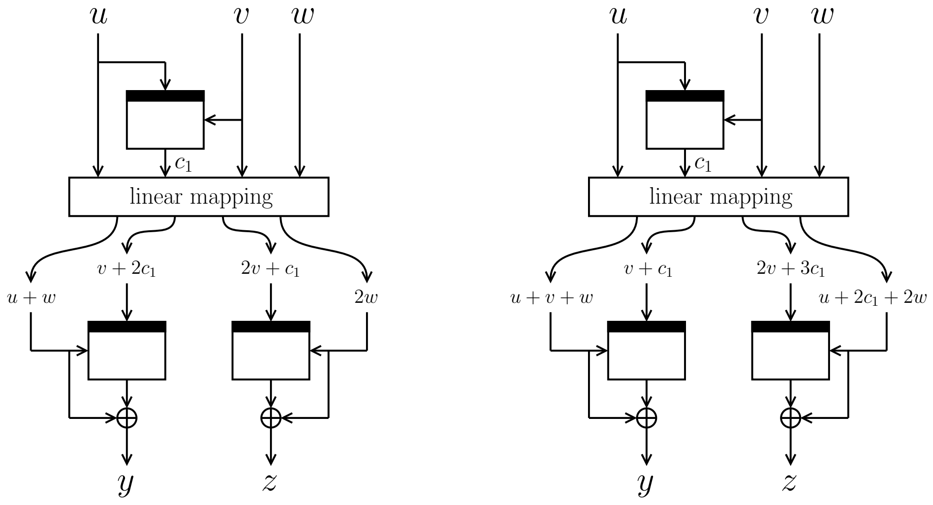


Figure 4.1. Two example compression functions from the family of functions introduced and evaluated in this work. For these constructions, all wires carry $n = 128$ bits, and the arithmetic is done over $GF(2^{128})$. We further elaborate on these designs and their derivations in Section 4.3.

Nonetheless, the bound on preimage resistance does not reach the optimal level of 2^{2n} queries. One can see this as the price we pay for using single key length rather than double key length block ciphers: a rather straightforward generalization of the pigeonhole-birthday attack of Rogaway and Steinberger [199] shows that, when the compression function behaves "sufficiently random," one may expect a preimage in approximately $2^{5n/3}$ queries (cf. Section 4.1). The asymptotic preimage bound of $2^{3n/2}$ found in this work closely approaches this generic bound.

Outline

In Section 4.1, we present some mathematical preliminaries complementary to Chapter 2, and formally describe the set of block cipher based compression functions we analyze. Then, in Section 4.2 we derive our impossibility result on F^2. We propose our family of compression functions in Section 4.3. The chapter is concluded in Section 4.4.

Bibliographic Notes

An extended abstract of this chapter has been published by Mennink at ASIACRYPT 2012 [163].

4.1 Security Model

Let $E \in \mathrm{Bloc}(n)$. For $k \geq 1$, let $\mathsf{F}^k : \mathbb{Z}_2^{3n} \to \mathbb{Z}_2^{2n}$ be a double length compression function making k calls to its block cipher E. We can represent F^k by mappings $f_i : \mathbb{Z}_2^{(i+2)n} \to \mathbb{Z}_2^{2n}$ for $i = 1, \ldots, k+1$ as follows.

```
$\mathsf{F}^k(u, v, w)$
  for $i = 1, \ldots, k$:
    $(K_i, m_i) \leftarrow f_i(u, v, w; c_1, \ldots, c_{i-1})$ ,
    $c_i \leftarrow E(K_i, m_i)$ ,
  $(y, z) \leftarrow f_{k+1}(u, v, w; c_1, \ldots, c_r)$ ,
  **return** $(y, z)$ .
```

For $k = 3$, the F^k compression function design is depicted in Figure 4.2. This generic design is a generalization of the permutation based hash function construction described by Rogaway and Steinberger [199]. In fact, it is straightforward to generalize the main findings of [199] to our F^k design and we state them as preliminary results. If the collision- and preimage-degeneracies are sufficiently small (these values intuitively capture the degree of non-randomness of the design with respect to the occurrence of collisions and preimages), one can expect collisions after approximately $2^{n(2-2/k)}$ queries and preimages after approximately $2^{n(2-1/k)}$ queries. We refer to [199] for the details. First of all, these findings confirm that at least two cipher calls are required to get 2^n collision resistance. More importantly, from these results we can conclude that F^k can impossibly achieve optimal 2^{2n} preimage resistance. Yet, it may still be possible to construct a function that achieves optimal collision resistance and almost-optimal preimage resistance.

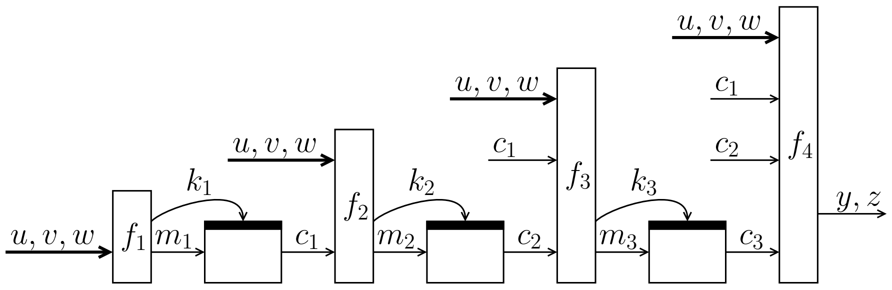


Figure 4.2. $\mathsf{F}^3 : \mathbb{Z}_2^{3n} \to \mathbb{Z}_2^{2n}$ making three block cipher evaluations.

4.1.1 Security Notions

Throughout this chapter, we consider $\mathcal{P} = E \xleftarrow{\$} \text{Bloc}(n)$. We consider an adversary which attacks F^k and whose goal it is to find either a collision $\mathsf{F}^k(u,v,w) = \mathsf{F}^k(u',v',w')$ or a preimage $\mathsf{F}^k(u,v,w) = (y,z)$ for an in advance chosen range value (y,z). The adversary can make forward and inverse queries to E, and these are stored in a query history $\mathcal{Q}$ as indexed tuples of the form (K_i, m_i, c_i), where K_i denotes the key input, and (m_i, c_i) the plaintext/ciphertext pair.

4.2 Impossibility Result for Hashing with Two E-calls

We present an attack on a wide class of double block length compression functions with two calls to their underlying block cipher $E : \mathbb{Z}_2^n \times \mathbb{Z}_2^n \to \mathbb{Z}_2^n$. Let F^2 be a compression function of this form. We pose a condition on the finalization function f_3, such that if this condition is satisfied, collisions for F^2 can be found in about $2^{n/2}$ queries. Although we are not considering all possible compression functions, we cover the most interesting and intuitive ones, such as compression functions with linear finalization function f_3. Compression functions with non-linear f_3 are covered up to some degree (but we note that the attack does not apply to the compression function of Jetchev et al. [118], for which collision security up to $2^{2n/3}$ queries is proven).

Jetchev et al. also consider the impossibility of a variant of F^2, based on two parallel evaluations of random functions, and argue several requirements needed for the pre- and postprocessing functions. However, their approach is less general, most importantly, our attack works on any possible functions f_1, f_2.

We first state the attack. Then, by ways of examples, we illustrate its generality.

Theorem 4.2.1. *Let* $\mathsf{F}^2 : \mathbb{Z}_2^{3n} \to \mathbb{Z}_2^{2n}$ *be a compression function as described in Section 4.1. Suppose there exists a bijective function* L *such that for any* $u, v, w, c_1, c_2 \in \mathbb{Z}_2^n$ *we have*

$$\mathsf{left}_n \circ L \circ f_3(u,v,w;c_1,c_2) = \mathsf{left}_n \circ L \circ f_3(u,v,w;c_1,0)\,. \tag{4.1}$$

Then, one can expect collisions for F^2 *after* $2^{n/2}$ *queries.*

Proof. Let F^2 be a compression function and let L be a bijection such that (4.1) holds. First, we consider the case of L being the identity function, and next we show how this attack extends to the case L is an arbitrary bijection.

Suppose (4.1) holds with L the identity function. This means that the first n bits of $f_3(u,v,w;c_1,c_2)$ do not depend on c_2 and we can write f_3 as a concatenation of two functions $g_1 : \mathbb{Z}_2^{4n} \to \mathbb{Z}_2^n$ and $g_2 : \mathbb{Z}_2^{5n} \to \mathbb{Z}_2^n$ as follows.

$$f_3(u,v,w;c_1,c_2) = g_1(u,v,w;c_1) \| g_2(u,v,w;c_1,c_2)\,.$$

Let $\alpha \in \mathbb{N}$. We present an adversary $\mathcal{A}$ for F^2. The first part of the attack is derived from [199].

(i) Make α queries $(K_1, m_1) \to c_1$ that maximize the number of tuples (u, v, w) with $f_1(u, v, w)$ hitting any of these values (K_1, m_1). By the balls-and-bins principle,[2] the adversary obtains at least $\alpha \cdot 2^{3n}/2^{2n} = \alpha 2^n$ tuples $(u, v, w; c_1)$ for which it knows the first block cipher evaluation;
(ii) Again by the balls-and-bins principle, there exists a value y such that at least α tuples satisfy $g_1(u, v, w; c_1) = y$;
(iii) Varying over these α tuples, compute $(K_2, m_2) = f_2(u, v, w; c_1)$ and query (K_2, m_2) to the cipher to obtain a c_2. $\mathcal{A}$ finds a collision for F^2 if it obtains two tuples $(u, v, w; c_1, c_2)$, $(u', v', w'; c_1', c_2')$ that satisfy $g_2(u, v, w; c_1, c_2) = g_2(u', v', w'; c_1', c_2')$.

In step (iii) one expects to find a collision if $\alpha^2/2^n = 1$, or equivalently if $\alpha = 2^{n/2}$. In total, the attack is done in approximately $2 \cdot 2^{n/2}$ queries.

It remains to consider the case of L being an arbitrary bijection. Define $\overline{\mathsf{F}}^2$ as F^2 with f_3 replaced by $\overline{f_3} = L \circ f_3$. Using the idea of equivalence classes on compression functions (we extensively discuss this in Chapter 5) we prove that F^2 and $\overline{\mathsf{F}}^2$ are equally secure with respect to collisions. Let $\overline{\mathcal{A}}$ be a collision-finding adversary for $\overline{\mathsf{F}}^2$. We construct a collision-finding adversary $\mathcal{A}$ for F^2, with oracle access to E, that uses $\overline{\mathcal{A}}$ to output a collision for F^2. Adversary $\mathcal{A}$ proceeds as follows. It forwards all queries made by $\overline{\mathcal{A}}$ to its own oracle. Eventually, $\overline{\mathcal{A}}$ outputs two tuples $(u, v, w), (u', v', w')$ such that $\overline{\mathsf{F}}^2(u, v, w) = \overline{\mathsf{F}}^2(u', v', w')$. Denote by c_1 the block cipher outcome on input of $f_1(u, v, w)$ and by c_2 the outcome on input of $f_2(u, v, w; c_1)$. Define c_1' and c_2' similarly. By construction, as (u, v, w) and (u', v', w') form a collision for $\overline{\mathsf{F}}^2$, we have

$$L \circ f_3(u, v, w; c_1, c_2) = L \circ f_3(u', v', w'; c_1', c_2') \,.$$

Now, bijectivity of L implies that $f_3(u, v, w; c_1, c_2) = f_3(u', v', w'; c_1', c_2')$, and hence (u, v, w) and (u', v', w') form a collision for F^2. (Recall that F^2 and $\overline{\mathsf{F}}^2$ only differ in the finalization function f_3, the functions f_1 and f_2 are the same.) We thus obtain $\mathbf{Adv}^{\text{col}}_{\overline{\mathsf{F}}^2}(q) \leq \mathbf{Adv}^{\text{col}}_{\mathsf{F}^2}(q)$. The derivation in reverse order is the same by symmetry. But $\overline{\mathsf{F}}^2$ satisfies (4.1) for L the identity function. Therefore, the attack described in the first part of the proof applies to $\overline{\mathsf{F}}^2$, and thus to F^2. $\square$

We demonstrate the impact of the attack by giving several example functions that fall in the categorization. We stress that the requirement of Theorem 4.2.1 is in fact solely a requirement on f_3; f_1 and f_2 can be any function.

[2]If k balls are thrown in l bins, the α fullest bins in total contain at least $\alpha k/l$ balls.

Suppose F^2 uses a linear finalization function f_3. Say, f_3 is defined as follows.

$$\begin{pmatrix} \mathsf{a}_{11} & \mathsf{a}_{12} & \mathsf{a}_{13} & \mathsf{a}_{14} & \mathsf{a}_{15} \\ \mathsf{a}_{21} & \mathsf{a}_{22} & \mathsf{a}_{23} & \mathsf{a}_{24} & \mathsf{a}_{25} \end{pmatrix} (u, v, w, c_1, c_2)^\top = (y, z)^\top,$$

where addition and multiplication is done over the field $GF(2^n)$. Now, if $\mathsf{a}_{25} = 0$ we set $L = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$ which corresponds to swapping y and z. If $\mathsf{a}_{25} \neq 0$, we set $L = \left(\begin{smallmatrix} 1 & -\mathsf{a}_{15}\mathsf{a}_{25}^{-1} \\ 0 & 1 \end{smallmatrix}\right)$, which corresponds to subtracting the second equation $\mathsf{a}_{15}\mathsf{a}_{25}^{-1}$ times from the first one.

The attack also covers designs whose finalization function f_3 rotates or shuffles its inputs, where one defines L so that the rotation gets undone. For instance, for MDC-2 (Figure 3.1) f_3 is defined over $n/2$-bit words as $f_3(u^l, u^r, v^l, v^r, w^l, w^r; c_1^l, c_1^r, c_2^l, c_2^r) = (c_1^l \oplus w^l, c_2^r \oplus w^r, c_2^l \oplus w^l, c_1^r \oplus w^r)$, where u^l and u^r denote the left and right half of u, and it satisfies (4.1) for

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

Re-shuffling the output of f_3 can even be defined at bit level, in which case L is a $2^{2n} \times 2^{2n}$ permutation matrix.

In general, if f_3 is a sufficiently simple add-rotate-XOR function, it is possible to derive a bijective L that makes (4.1) satisfied. This is possible by composing multiple bijective mappings L. Up to a degree, the attack also covers general non-linear finalization functions. However, it clearly does not cover all functions and it remains an open problem to either close this gap or to come with a (possibly impractical) F^2 compression function that provably achieves optimal collision resistance. One direction may be to start from the compression function with non-linear finalization f_3 by Jetchev et al. [118], for which collision resistance up to $2^{2n/3}$ queries is proven.

4.3 Double-Length Hashing with Three E-calls

Motivated by the negative result of Section 4.2, we target the existence of double length hashing with three block cipher calls. We introduce a family of double length compression functions making three cipher calls that achieve asymptotically optimal 2^n collision resistance and preimage resistance significantly beyond the birthday bound (up to $2^{3n/2}$ queries). We note that, although the preimage bound is non-optimal, it closely approaches the generic bound dictated by the pigeonhole-birthday attack (Section 4.1).

Let $GF(2^n)$ be the field of order 2^n. We identify bit strings from $\mathbb{Z}_2^n$ and finite field elements in $GF(2^n)$ to define addition and scalar multiplication over

$\mathbb{Z}_2^n$. In the family of double block length functions we propose in this section, the functions f_1, f_2, f_3, f_4 of Figure 4.2 will be linear functions over $GF(2^n)$. For two tuples $x = (x_1, \ldots, x_l)$ and $y = (y_1, \ldots, y_l)$ of elements from $\mathbb{Z}_2^n$, we define by $x \cdot y$ their inner product $\sum_{i=1}^{l} x_i y_i \in \mathbb{Z}_2^n$.

Before introducing the design, we first explain the fundamental consideration upon which the family is based. The security proofs of all DBL^{2n} functions known in the literature (cf. Table 2.1) crucially rely on the property that one block cipher evaluation defines the input to the other one. For DBL^{2n} functions this can easily be achieved: any block cipher evaluation can take as input the full $3n$-bit input state (u, v, w). Considering the class of functions DBL^n, and F^k of Figure 4.2 in particular, this can impossibly be achieved: one block cipher "processes" at most $2n$ out of $3n$ input bits. In our design, we slightly relax this requirement, by requiring that any *two* block cipher evaluations define the input to the third one. Although from a technical point of view one may expect that this change causes optimal collision resistance to be harder or even impossible to be achieved, we will demonstrate that this is not the case due to new proof techniques employed to analyze the collision resistance.

Based on this key observation we propose the compression function design F_A^3 of Figure 4.3. Here,

$$\mathsf{A} = \begin{pmatrix} \mathsf{a}_1 \\ \mathsf{a}_2 \\ \mathsf{a}_3 \\ \mathsf{a}_4 \end{pmatrix} = \begin{pmatrix} \mathsf{a}_{11} & \mathsf{a}_{12} & \mathsf{a}_{13} & 0 \\ \mathsf{a}_{21} & \mathsf{a}_{22} & \mathsf{a}_{23} & \mathsf{a}_{24} \\ \mathsf{a}_{31} & \mathsf{a}_{32} & \mathsf{a}_{33} & 0 \\ \mathsf{a}_{41} & \mathsf{a}_{42} & \mathsf{a}_{43} & \mathsf{a}_{44} \end{pmatrix}$$

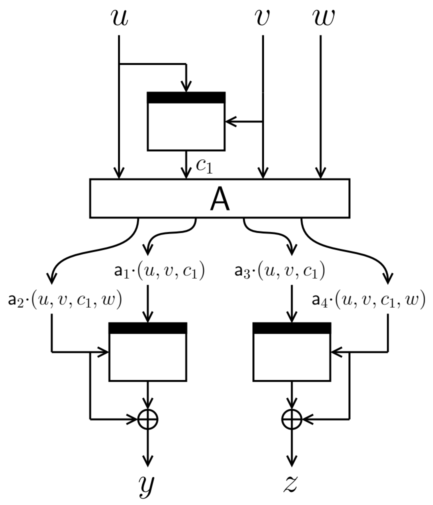
is a 4×4 matrix over $GF(2^n)$. Note that, provided A is invertible and $\mathsf{a}_{24}, \mathsf{a}_{44} \neq 0$, any two block cipher evaluations of F_A^3 define (the inputs of) the third one. For instance, evaluations of the second and third block cipher fix the vector $\mathsf{A}(u, v, c_1, w)^\top$, which by invertibility of A fixes (u, v, c_1, w) and thus the first block cipher evaluation. Evaluations of the first and second block cipher fix the inputs of the third block cipher as $\mathsf{a}_{24} \neq 0$. For the proofs of collision and preimage resistance, however, we will need to posit additional requirements on A. As we will explain, these requirements are easily satisfied.

In the remainder of this section, we state our results on the collision resistance of F_A^3 in Section 4.3.1 and on the preimage resistance in Section 4.3.2.

4.3.1 Collision Resistance of F_A^3

We prove that, provided its underlying matrix A satisfies some simple conditions, F_A^3 satisfies optimal collision resistance. In more detail, we pose the following requirements on A:

- A is invertible;
- $\mathsf{a}_{12}, \mathsf{a}_{13}, \mathsf{a}_{24}, \mathsf{a}_{32}, \mathsf{a}_{33}, \mathsf{a}_{44} \neq 0$;
- $\mathsf{a}_{12} \neq \mathsf{a}_{32}$ and $\mathsf{a}_{13} \neq \mathsf{a}_{33}$.


u
v
w
c_1
A
$\mathsf{a}_1\cdot(u,v,c_1)$
$\mathsf{a}_3\cdot(u,v,c_1)$
$\mathsf{a}_2\cdot(u,v,c_1,w)$
$\mathsf{a}_4\cdot(u,v,c_1,w)$
y
z

```
$\mathsf{F}^3_{\mathsf{A}}(u,v,w)$
    $c_1 \leftarrow E(u,v)$ ,
    $K_2 \leftarrow \mathsf{a}_1\cdot(u,v,c_1)$ ,
    $m_2 \leftarrow \mathsf{a}_2\cdot(u,v,c_1,w)$ ,
    $y \leftarrow E(K_2,m_2)+m_2$ ,
    $K_3 \leftarrow \mathsf{a}_3\cdot(u,v,c_1)$ ,
    $m_3 \leftarrow \mathsf{a}_4\cdot(u,v,c_1,w)$ ,
    $z \leftarrow E(K_3,m_3)+m_3$ ,
    **return** $(y,z)$ .
```

Figure 4.3. The family of compression functions $\mathsf{F}^3_{\mathsf{A}}$ where A is a 4×4 matrix as specified in the text. Arithmetics is done over $GF(2^n)$.

We refer to the logical AND of these requirements as `colreq`.

Theorem 4.3.1. *Let $n \in \mathbb{Z}_2^n$. Suppose* A *satisfies* `colreq`*. Then, for any positive integral values τ_1, τ_2,*

$$\mathbf{Adv}^{\mathrm{col}}_{\mathsf{F}^3_{\mathsf{A}}}(q) \le \frac{2\tau_2^2 q + 3\tau_2 q + 11q + 3\tau_1\tau_2^2 + 7\tau_1\tau_2}{2^n - q} + \frac{q^2}{\tau_1(2^n-q)} + 3\cdot 2^n \left(\frac{eq}{\tau_2(2^n-q)}\right)^{\tau_2} . \tag{4.2}$$

The proof is given in Section 4.3.1.1. The basic proof idea is similar to the proof in Section 3.2.1 and is based on the usage of thresholds τ_1, τ_2. For increasing values of τ_1, τ_2 the first two terms of the bound increase, while the second two terms decrease. Although the proof derives basic proof principles from literature, for the technical part we deviate from existing proof techniques in order to get a bound that is "as tight as possible." In particular, we introduce the usage of wish lists in the context of collisions, an approach that allows for significantly better bounds. Wish lists have been introduced by Armknecht et al. [22] and Lee et al. [145, 147] for the preimage resistance analysis of DBL^{2n} functions, but they have never been used for collision resistance as there never was a need to do so. Our analysis relies on this proof methodology, but as for collisions more block cipher evaluations are involved (one collision needs six block cipher calls while a preimage requires three) this makes the analysis more technical and delicate.

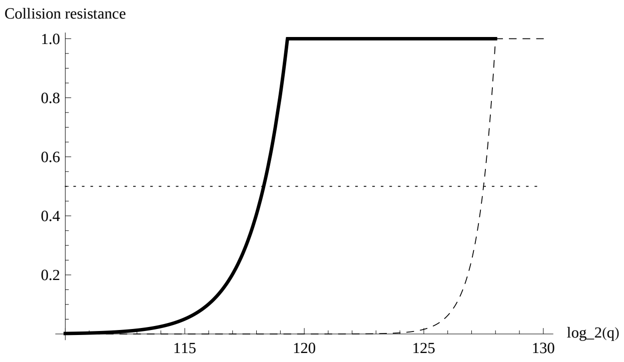


Figure 4.4. The function $\mathbf{Adv}^{\mathrm{col}}_{\mathsf{F}^3_\mathsf{A}}(q)$ of (4.2) for $n = 128$, $\varepsilon = 1/35$, and the particular choice of values τ_1, τ_2 (solid line), in comparison with the optimal bound of $q(q+1)/2^{2n}$ (dashed line).

The goal now is to find a good threshold between the first term and the latter two terms of (4.2). To this end, let $\varepsilon > 0$ be any parameter. We put $\tau_1 = q$ and $\tau_2 = 2^{n\varepsilon}$ (we can assume τ_2 to be integral). Then, the bound simplifies to

$$\mathbf{Adv}^{\mathrm{col}}_{\mathsf{F}^3_\mathsf{A}}(q) \leq \frac{5 \cdot 2^{2n\varepsilon}q + 10 \cdot 2^{n\varepsilon}q + 11q}{2^n - q} + \frac{q}{2^n - q} + 3 \cdot 2^n \left(\frac{eq}{2^{n\varepsilon}(2^n - q)}\right)^{2^{n\varepsilon}} .$$

We find that for any $\varepsilon > 0$, this term goes to 0 for $n \to \infty$ when $q = 2^n/2^{3n\varepsilon}$.

Corollary 4.3.2. *For any $\varepsilon > 0$, we obtain* $\lim_{n\to\infty} \mathbf{Adv}^{\mathrm{col}}_{\mathsf{F}^3_\mathsf{A}}\left(2^n/2^{3n\varepsilon}\right) = 0$.

Hence, the F^3_A compression function achieves close to optimal 2^n collision security for $n \to \infty$. For $n = 128$, the bound on $\mathbf{Adv}^{\mathrm{col}}_{\mathsf{F}^3_\mathsf{A}}$ is depicted in Figure 4.4, where we take a slightly different value for τ_1 to achieve a better bound (to be precise, $\tau_1 = q/(3\tau_2^2 + 7\tau_2)^{1/2}$). The collision advantage hits 1/2 for $\log_2 q \approx 118.3$, relatively close to the threshold 127.5 for $q(q+1)/2^{2n}$. For larger values of n this gap approaches 0.

4.3.1.1 Proof of Theorem 4.3.1

The proof of collision resistance of F^3_A follows the basic spirit of the proof in Section 3.2.1, but crucially differs in the way the probability bounds are computed. A new approach here is the usage of wish lists. While the idea of wish lists is not new — it has been introduced by Armknecht et al. [22] and Lee et al. [145, 147] for double block length compression functions, and used in Section 3.3.1 for the analysis of MDC-4 — in these works wish lists are solely used for the analysis of preimage resistance rather than collision resistance. Given that in a collision more block cipher evaluations are involved, the analysis becomes more complex.

At a high level, wish lists rely on the idea that in order to find a collision, the adversary must at some point make a query that "completes this collision" together with some other queries already in the query history. Wish lists keep track of such query tuples, and the adversary's goal is to ever obtain a query tuple that is in such wish list. A more technical treatment can be found in the proof of Lemma 4.3.3.

We consider any adversary that has query access to its oracle E and makes q queries stored in a query history $\mathcal{Q}_q$. Its goal is to find a collision for F^3_A, in which it by definition only succeeds if it obtains a query history $\mathcal{Q}_q$ that satisfies configuration $\mathsf{col}(\mathcal{Q}_q)$ of Figure 4.5. This means,

$$\mathbf{Adv}^{\mathrm{col}}_{\mathsf{F}^3_\mathsf{A}}(q) = \mathbf{Pr}\left(\mathsf{col}(\mathcal{Q}_q)\right) . \tag{4.3}$$

For the sake of readability of the proof, we label the block cipher positions in Figure 4.5 as follows. In the left F^3_A evaluation (on input (u,v,w)), the block ciphers are labeled 1t (the one on input (u,v)), 1bl (the bottom left one), and 1br (the bottom right one). The block ciphers for the right F^3_A evaluation are labeled 2t, 2bl, 2br in a similar way. When we say "a query 1t," we refer to a query that in a collision occurs at position 1t.

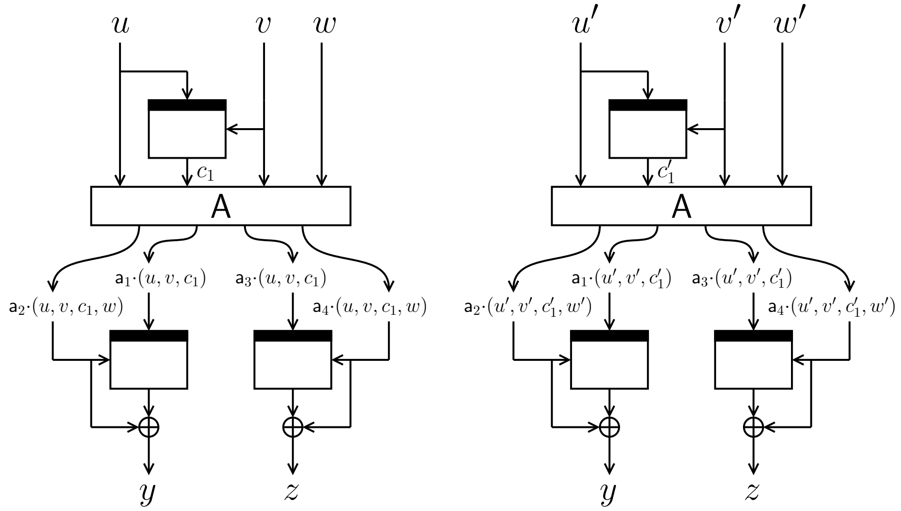

u
v
w
c_1
A
$\mathsf{a}_1\cdot(u,v,c_1)$
$\mathsf{a}_3\cdot(u,v,c_1)$
$\mathsf{a}_2\cdot(u,v,c_1,w)$
$\mathsf{a}_4\cdot(u,v,c_1,w)$
y
z
u'
v'
w'
c_1'
A
$\mathsf{a}_1\cdot(u',v',c_1')$
$\mathsf{a}_3\cdot(u',v',c_1')$
$\mathsf{a}_2\cdot(u',v',c_1',w')$
$\mathsf{a}_4\cdot(u',v',c_1',w')$
y
z

Figure 4.5. Configuration $\mathsf{col}(\mathcal{Q})$. The configuration is satisfied if $\mathcal{Q}$ contains six (possibly not all different) queries that satisfy this setting. We require $(u,v,w) \neq (u',v',w')$.

For the analysis of $\mathbf{Pr}\left(\mathsf{col}(\mathcal{Q}_q)\right)$ we introduce an auxiliary event $\mathsf{aux}(\mathcal{Q}_q)$. Let $\tau_1, \tau_2 > 0$ be any integral values. We define $\mathsf{aux}(\mathcal{Q}_q) = \mathsf{aux}_1(\mathcal{Q}_q) \vee \cdots \vee \mathsf{aux}_4(\mathcal{Q}_q)$, where

$$\begin{aligned}
&\mathsf{aux}_1(\mathcal{Q}_q): && \left|\left\{(K_i,m_i,c_i),(K_j,m_j,c_j)\in\mathcal{Q}_q \;\middle|\; i\neq j \;\wedge\; m_i+c_i=m_j+c_j\right\}\right|>\tau_1\,;\\
&\mathsf{aux}_2(\mathcal{Q}_q): && \max\nolimits_{z\in\mathbb{Z}_2^n}\left|\left\{(K_i,m_i,c_i)\in\mathcal{Q}_q \;\middle|\; \mathsf{a}_1\cdot(K_i,m_i,c_i)=z\right\}\right|>\tau_2\,;\\
&\mathsf{aux}_3(\mathcal{Q}_q): && \max\nolimits_{z\in\mathbb{Z}_2^n}\left|\left\{(K_i,m_i,c_i)\in\mathcal{Q}_q \;\middle|\; \mathsf{a}_3\cdot(K_i,m_i,c_i)=z\right\}\right|>\tau_2\,;\\
&\mathsf{aux}_4(\mathcal{Q}_q): && \max\nolimits_{z\in\mathbb{Z}_2^n}\left|\left\{(K_i,m_i,c_i)\in\mathcal{Q}_q \;\middle|\; m_i+c_i=z\right\}\right|>\tau_2\,.
\end{aligned}$$

By basic probability theory, we obtain for (4.3):

$$\mathbf{Pr}\left(\mathsf{col}(\mathcal{Q}_q)\right)\leq\mathbf{Pr}\left(\mathsf{col}(\mathcal{Q}_q)\wedge\neg\mathsf{aux}(\mathcal{Q}_q)\right)+\mathbf{Pr}\left(\mathsf{aux}(\mathcal{Q}_q)\right)\,. \tag{4.4}$$

We start with the analysis of $\mathbf{Pr}\left(\mathsf{col}(\mathcal{Q}_q)\wedge\neg\mathsf{aux}(\mathcal{Q}_q)\right)$. For obtaining a query history that fulfills configuration $\mathsf{col}(\mathcal{Q}_q)$, it may be the case that a query appears at multiple positions. For instance, the queries at positions $1\mathsf{t}$ and $2\mathsf{bl}$ are the same. We split the analysis of $\mathsf{col}(\mathcal{Q}_q)$ into essentially all different possible cases, but we do this in two steps. In the first step, we distinct among the cases a query occurs in both words at the same position. We define for binary $\alpha_1,\alpha_2,\alpha_3$ by $\mathsf{col}_{\alpha_1\alpha_2\alpha_3}(\mathcal{Q})$ the configuration $\mathsf{col}(\mathcal{Q})$ of Figure 4.5 restricted to

$$1\mathsf{t}=2\mathsf{t}\equiv\alpha_1\,,\qquad 1\mathsf{bl}=2\mathsf{bl}\equiv\alpha_2\,,\qquad 1\mathsf{br}=2\mathsf{br}\equiv\alpha_3\,.$$

By construction, $\mathsf{col}(\mathcal{Q}_q)\Rightarrow\bigvee_{\alpha_1,\alpha_2,\alpha_3\in\{0,1\}}\mathsf{col}_{\alpha_1\alpha_2\alpha_3}(\mathcal{Q}_q)$, and from (4.3-4.4) we obtain the following bound on $\mathbf{Adv}^{\mathrm{col}}_{\mathsf{F}^3_\mathsf{A}}(q)$:

$$\mathbf{Adv}^{\mathrm{col}}_{\mathsf{F}^3_\mathsf{A}}(q)\leq\sum_{\substack{\alpha_1,\alpha_2,\\ \alpha_3\in\{0,1\}}}\mathbf{Pr}\left(\mathsf{col}_{\alpha_1\alpha_2\alpha_3}(\mathcal{Q}_q)\wedge\neg\mathsf{aux}(\mathcal{Q}_q)\right)+\mathbf{Pr}\left(\mathsf{aux}(\mathcal{Q}_q)\right)\,. \tag{4.5}$$

Note that we did not make a distinction yet whether or not a query occurs at two *different* positions (e.g., at positions $1\mathsf{t}$ and $2\mathsf{bl}$). These cases are analyzed for each of the sub-configurations separately, as becomes clear later. Probabilities $\mathbf{Pr}\left(\mathsf{col}_{\alpha_1\alpha_2\alpha_3}(\mathcal{Q}_q)\wedge\neg\mathsf{aux}(\mathcal{Q}_q)\right)$ for the different choices of $\alpha_1,\alpha_2,\alpha_3$ are bounded in Lemmas 4.3.3-4.3.6. The proofs are rather similar, and we only bound the probability on $\mathsf{col}_{000}(\mathcal{Q}_q)$ in full detail (Lemma 4.3.3). A bound on $\mathbf{Pr}\left(\mathsf{aux}(\mathcal{Q}_q)\right)$ is derived in Lemma 4.3.7.

Lemma 4.3.3. $\mathbf{Pr}\left(\mathsf{col}_{000}(\mathcal{Q}_q)\wedge\neg\mathsf{aux}(\mathcal{Q}_q)\right)\leq\frac{\tau_2q+7q+3\tau_1\tau_2^2+3\tau_1\tau_2}{2^n-q}$.

Proof. Sub-configuration $\mathsf{col}_{000}(\mathcal{Q}_q)$ is given in Figure 4.6. The block cipher queries at positions a and $!a$ are required to be different, and so are the ones are positions $b,!b$ and $c,!c$.

We consider the probability of the adversary finding a solution to configuration $\mathsf{col}_{000}(\mathcal{Q}_q)$ such that $\mathcal{Q}_q$ satisfies $\neg\mathsf{aux}(\mathcal{Q}_q)$. Consider the i^{th} query, for $i\in\{1,\ldots,q\}$. We say this query is a winning query if it makes $\mathsf{col}_{000}(\mathcal{Q}_i)\wedge\neg\mathsf{aux}(\mathcal{Q}_i)$ satisfied for any set of other queries in the query history $\mathcal{Q}_{i-1}$. We can assume

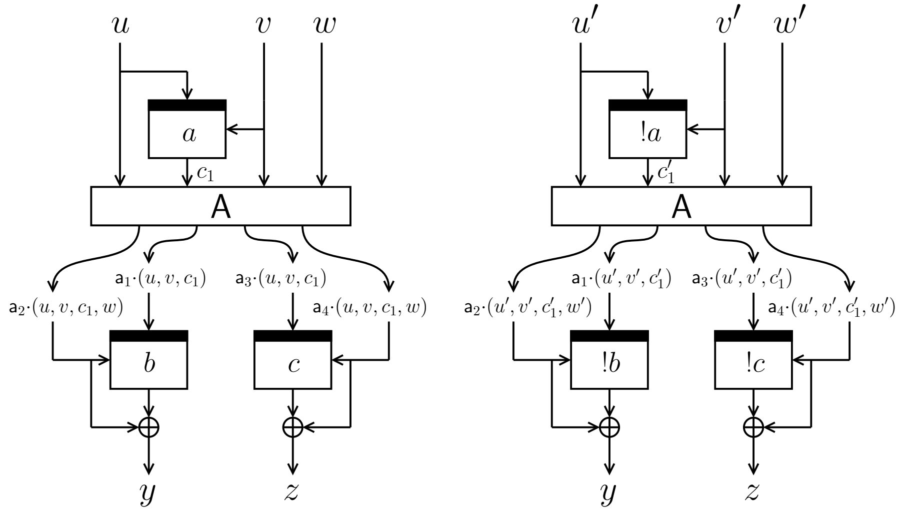


Figure 4.6. Configuration $\mathsf{col}_{000}(\mathcal{Q})$. We require $(u,v,w) \neq (u',v',w')$.

the i^{th} query does not make $\mathsf{aux}(\mathcal{Q}_i)$ satisfied: if it would, by definition it cannot be a winning query.

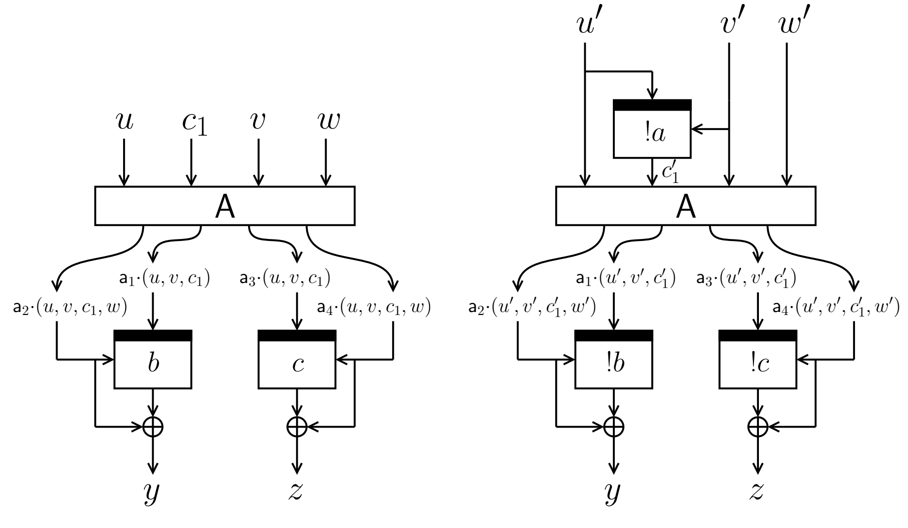
Recall that, although we narrowed down the number of possible positions for a winning query to occur (in $\mathsf{col}_{000}(\mathcal{Q}_q)$ it cannot occur at both 1t and 2t, at both 1bl and 2bl, or at both 1br and 2br), it may still be the case that such a query contributes to multiple *different* positions, e.g., 1t and 2bl. Note that by construction, a winning query can contribute to at most three block cipher positions of Figure 4.6. In total, there are 26 sets of positions at which the winning query can contribute at the same time. Discarding symmetric cases caused by swapping (u,v,w) and (u',v',w'), one identifies the following 13 sets of positions:

$$\begin{aligned}
&\mathcal{S}_1 = \{\mathsf{1t}\}\,, && \mathcal{S}_4 = \{\mathsf{1t},\mathsf{1bl}\}\,, && \mathcal{S}_7 = \{\mathsf{1t},\mathsf{2bl}\}\,, && \mathcal{S}_{10} = \{\mathsf{1t},\mathsf{1bl},\mathsf{1br}\}\,,\\
&\mathcal{S}_2 = \{\mathsf{1bl}\}\,, && \mathcal{S}_5 = \{\mathsf{1t},\mathsf{1br}\}\,, && \mathcal{S}_8 = \{\mathsf{1t},\mathsf{2br}\}\,, && \mathcal{S}_{11} = \{\mathsf{1t},\mathsf{1bl},\mathsf{2br}\}\,,\\
&\mathcal{S}_3 = \{\mathsf{1br}\}\,, && \mathcal{S}_6 = \{\mathsf{1bl},\mathsf{1br}\}\,, && \mathcal{S}_9 = \{\mathsf{1bl},\mathsf{2br}\}\,, && \mathcal{S}_{12} = \{\mathsf{1t},\mathsf{2bl},\mathsf{1br}\}\,,\\
& && && && \mathcal{S}_{13} = \{\mathsf{1t},\mathsf{2bl},\mathsf{2br}\}\,.
\end{aligned}$$

Note that there are many more symmetric cases among these, but we are not allowed to discard those as these may result in effectively different collisions. For $j = 1,\ldots,13$ we denote by $\mathsf{col}_{000:\mathcal{S}_j}(\mathcal{Q})$ configuration $\mathsf{col}_{000}(\mathcal{Q})$ with the restriction that the winning query *must* appear at the positions in $\mathcal{S}_j$. By basic probability theory,

$$\mathbf{Pr}\left(\mathsf{col}_{000}(\mathcal{Q}_q) \wedge \neg\mathsf{aux}(\mathcal{Q}_q)\right) \leq \sum_{j=1}^{13} \mathbf{Pr}\left(\mathsf{col}_{000:\mathcal{S}_j}(\mathcal{Q}_q) \wedge \neg\mathsf{aux}(\mathcal{Q}_q)\right) . \tag{4.6}$$

$\mathsf{col}_{000:\mathcal{S}_1}(\mathcal{Q}_q)$. Rather than considering the success probability of the i^{th} query, and then sum over $i = 1, \ldots, q$ (as is done in the analysis of [100,111,112,118,142, 146,182,210] and Section 3.3.1, hence all collision security proofs of Table 2.1), the approach in this proof is to focus on *wish lists*. Intuitively, a wish list is a continuously updated sequence of query tuples that would make configuration $\mathsf{col}_{000:\mathcal{S}_j}(\mathcal{Q}_q)$ satisfied. During the attack of the adversary, we maintain an initially empty wish list $\mathcal{W}_{\mathcal{S}_1}$. Consider configuration $\mathsf{col}_{000}(\mathcal{Q})$ with the query at position $\mathcal{S}_1 = \{\mathsf{1t}\}$ left out (see Figure 4.7). If a new query is made, suppose it fits this configuration for some other queries in the query history (the new query appearing at least once), jointly representing queries at positions $\{\mathsf{1bl}, \mathsf{1br}, \mathsf{2t}, \mathsf{2bl}, \mathsf{2br}\}$. Then the corresponding tuple (u, v, c_1) is added to $\mathcal{W}_{\mathcal{S}_1}$. Note that this tuple is uniquely determined by the queries at $\mathsf{1bl}$ and $\mathsf{1br}$ by invertibility of A, but different combinations of queries may define the same wish. The latter does, however, not invalidate the analysis: this is covered by the upper bound on $\mathcal{W}_{\mathcal{S}_1}$ that will be computed later in the proof, and will simply render a slightly worse bound.



u
c_1
v
w
A
$\mathsf{a}_1\cdot(u,v,c_1)$
$\mathsf{a}_3\cdot(u,v,c_1)$
$\mathsf{a}_2\cdot(u,v,c_1,w)$
$\mathsf{a}_4\cdot(u,v,c_1,w)$
b
c
y
z
u'
v'
w'
$!a$
c_1'
A
$\mathsf{a}_1\cdot(u',v',c_1')$
$\mathsf{a}_3\cdot(u',v',c_1')$
$\mathsf{a}_2\cdot(u',v',c_1',w')$
$\mathsf{a}_4\cdot(u',v',c_1',w')$
$!b$
$!c$
y
z

Figure 4.7. Configuration $\mathsf{col}_{000:\mathcal{S}_1}(\mathcal{Q})$. We require $(u, v, w) \neq (u', v', w')$.

As we have restricted to the case the winning query only occurring at the position of $\mathcal{S}_1$, we can assume a query never adds itself to a wish list.[3] Clearly,

[3]A winning query that would appear at multiple positions is counted in $\mathsf{col}_{000:\mathcal{S}_j}(\mathcal{Q}_q)$ for some other set $\mathcal{S}_j$.

in order to find a collision for $\mathsf{F}^3_{\mathsf{A}}$ in this sub-configuration, the adversary needs to wish for a query at least once. Suppose the adversary makes a query $E(k,m)$ where $(k,m,c) \in \mathcal{W}_{\mathcal{S}_1}$ for some c. We say that (k,m,c) is wished for, and the wish is granted if the query response equals c. As the adversary makes at most q queries, such wish is granted with probability at most $\frac{1}{2^n-q}$, and the same for inverse queries. By construction, each element from $\mathcal{W}_{\mathcal{S}_1}$ can be wished for only once, and we find that the adversary finds a collision with probability at most $\frac{|\mathcal{W}_{\mathcal{S}_1}|}{2^n-q}$.

Now, it suffices to upper bound the size of the wish list $\mathcal{W}_{\mathcal{S}_1}$ after q queries, and to this end we bound the number of solutions to the configuration of Figure 4.7. By $\neg\mathsf{aux}_1(\mathcal{Q}_q)$, the configuration has at most τ_1 choices for $1\mathsf{bl}, 2\mathsf{bl}$. For any such choice, by $\neg\mathsf{aux}_2(\mathcal{Q}_q)$ we have at most τ_2 choices for $2\mathsf{t}$. Any such choice fixes w' (as $\mathsf{a}_{24} \neq 0$), and thus the query at position $2\mathsf{br}$, and consequently z. By $\neg\mathsf{aux}_4(\mathcal{Q}_q)$, we have at most τ_2 choices for $1\mathsf{br}$. The queries at positions $1\mathsf{bl}$ and $1\mathsf{br}$ uniquely fix (u,v,c_1) by invertibility of A. We find $|\mathcal{W}_{\mathcal{S}_1}| \leq \tau_1\tau_2^2$, and hence in this setting a collision is found with probability at most $\frac{\tau_1\tau_2^2}{2^n-q}$.

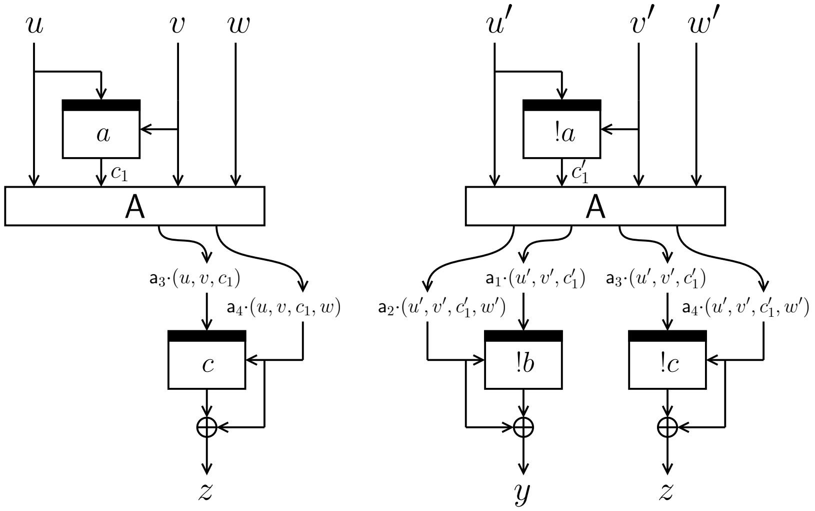
$\mathsf{col}_{000:\mathcal{S}_j}(\mathcal{Q}_q)$ for $j = 2, 3$. Both cases are the same by symmetry, and we consider $\mathcal{S}_2$ only.

The analysis is similar to the one for $\mathcal{S}_1$, and we only present the computation of the bound on the wish list $\mathcal{W}_{\mathcal{S}_2}$ after q queries. Consider configuration $\mathsf{col}_{000}(\mathcal{Q})$ with the query at position $\mathcal{S}_2 = \{1\mathsf{bl}\}$ left out (see Figure 4.8). By $\neg\mathsf{aux}_1(\mathcal{Q}_q)$, the configuration has at most τ_1 choices for $1\mathsf{br}, 2\mathsf{br}$. For any such choice, by $\neg\mathsf{aux}_3(\mathcal{Q}_q)$ we have at most τ_2 choices for $1\mathsf{t}$ and at most τ_2 choices for $2\mathsf{t}$. Any such choice fixes w' (as $\mathsf{a}_{44} \neq 0$), and thus the query at position $2\mathsf{bl}$, and consequently y. The query at position $1\mathsf{t}$ fixes (u,v,c_1) and together with query $1\mathsf{br}$ this fixes w. Any choice of queries thus uniquely fixes $(\mathsf{a}_1{\cdot}(u,v,c_1), \mathsf{a}_1{\cdot}(u,v,c_1,w), y-\mathsf{a}_1{\cdot}(u,v,c_1,w))$. We find $|\mathcal{W}_{\mathcal{S}_2}| \leq \tau_1\tau_2^2$, and hence in this setting a collision is found with probability at most $\frac{\tau_1\tau_2^2}{2^n-q}$.

$\mathsf{col}_{000:\mathcal{S}_j}(\mathcal{Q}_q)$ for $j = 4, 5$. Both cases are the same by symmetry, and we consider $\mathcal{S}_4$ only.

The analysis differs from the ones before, because in this setting the success probability can be analyzed more easily. As the winning query (k,m,c) should appear at positions $1\mathsf{t}$ and $1\mathsf{bl}$, we require it to satisfy $k = \mathsf{a}_1 \cdot (k,m,c)$. Any query satisfies this equation with probability at most $\frac{1}{2^n-q}$ (as $\mathsf{a}_{12}, \mathsf{a}_{13} \neq 0$). As the adversary makes at most q queries, in this setting a collision is found with probability at most $\frac{q}{2^n-q}$.

$\mathsf{col}_{000:\mathcal{S}_6}(\mathcal{Q}_q)$. By construction, there must be a query (k,m,c) in the query history (corresponding to position $1\mathsf{t}$) that satisfies $\mathsf{a}_1 \cdot (k,m,c) = \mathsf{a}_3 \cdot (k,m,c)$. So the problem shifts to bounding the probability that the adversary ever finds a query (k,m,c) that satisfies this equation. As $\mathsf{a}_{12} \neq \mathsf{a}_{32}$ and $\mathsf{a}_{13} \neq \mathsf{a}_{33}$, the



u
v
w
a
c_1
A
$\mathsf{a}_3{\cdot}(u,v,c_1)$
$\mathsf{a}_4{\cdot}(u,v,c_1,w)$
c
z
u'
v'
w'
$!a$
c_1'
A
$\mathsf{a}_2{\cdot}(u',v',c_1',w')$
$\mathsf{a}_1{\cdot}(u',v',c_1')$
$\mathsf{a}_3{\cdot}(u',v',c_1')$
$\mathsf{a}_4{\cdot}(u',v',c_1',w')$
$!b$
$!c$
y
z

Figure 4.8. Configuration $\mathsf{col}_{000:\mathcal{S}_2}(\mathcal{Q})$. We require $(u,v,w) \neq (u',v',w')$.

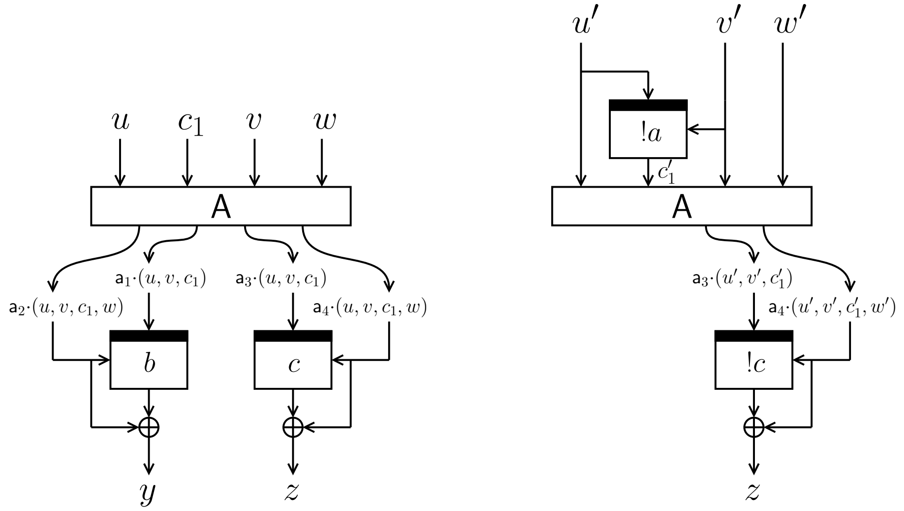
adversary never obtains such query except with probability at most $\frac{q}{2^n-q}$ (for the same reasoning as for $\mathcal{S}_4$).

$\mathsf{col}_{000:\mathcal{S}_j}(\mathcal{Q}_q)$ for $j = 7, 8$. Both cases are the same by symmetry, and we consider $\mathcal{S}_7$ only.

The analysis is similar to the one for $\mathcal{S}_1$, and we only present the definition of the wish list $\mathcal{W}_{\mathcal{S}_7}$ and the computation of the bound on $\mathcal{W}_{\mathcal{S}_7}$. Consider configuration $\mathsf{col}_{000}(\mathcal{Q})$ with the queries at positions $\mathcal{S}_7 = \{1\mathsf{t}, 2\mathsf{bl}\}$ left out (see Figure 4.9). For any set of queries that fits this configuration at positions $\{1\mathsf{bl}, 1\mathsf{br}, 2\mathsf{t}, 2\mathsf{br}\}$, the tuple (u, v, c_1) is added to $\mathcal{W}_{\mathcal{S}_7}$. By construction, this tuple is required to satisfy $(u,v,c_1) = (\mathsf{a}_1{\cdot}(u',v',c_1'), \mathsf{a}_1{\cdot}(u',v',c_1',w'), y{-}\mathsf{a}_1{\cdot}(u',v',c_1',w'))$.

We upper bound the number of solutions to the configuration of Figure 4.9 after q queries. By $\neg\mathsf{aux}_1(\mathcal{Q}_q)$, the configuration has at most τ_1 choices for $1\mathsf{br}, 2\mathsf{br}$. For any such choice, by $\neg\mathsf{aux}_3(\mathcal{Q}_q)$ we have at most τ_2 choices for $2\mathsf{t}$. The queries at positions $2\mathsf{t}$ and $2\mathsf{br}$ uniquely fix (u',v',c_1',w') (as $\mathsf{a}_{44} \neq 0$), and thus the values $u = \mathsf{a}_1 \cdot (u',v',c_1')$ and $v = \mathsf{a}_1 \cdot (u',v',c_1',w')$. Together with the query $1\mathsf{br}$ this fixes c_1 (as $\mathsf{a}_{33} \neq 0$). Any choice of queries thus uniquely fixes (u,v,c_1). We find $|\mathcal{W}_{\mathcal{S}_7}| \leq \tau_1\tau_2$, and hence in this setting a collision is found with probability at most $\frac{\tau_1\tau_2}{2^n-q}$.

$\mathsf{col}_{000:\mathcal{S}_9}(\mathcal{Q}_q)$. The analysis is similar to the one for $\mathcal{S}_1$, and we only present the definition of the wish list $\mathcal{W}_{\mathcal{S}_9}$ and the computation of the bound on $\mathcal{W}_{\mathcal{S}_9}$.



u
c_1
v
w
A
$\mathsf{a}_1\cdot(u,v,c_1)$
$\mathsf{a}_3\cdot(u,v,c_1)$
$\mathsf{a}_2\cdot(u,v,c_1,w)$
$\mathsf{a}_4\cdot(u,v,c_1,w)$
b
c
y
z
u'
v'
w'
$!a$
c_1'
A
$\mathsf{a}_3\cdot(u',v',c_1')$
$\mathsf{a}_4\cdot(u',v',c_1',w')$
$!c$
z

Figure 4.9. Configuration $\mathsf{col}_{000:\mathcal{S}_7}(\mathcal{Q})$. We require $(u,v,w)\neq(u',v',w')$ and $(u,v,c_1)=(\mathsf{a}_1\cdot(u',v',c_1'),\mathsf{a}_1\cdot(u',v',c_1',w'),y-\mathsf{a}_1\cdot(u',v',c_1',w'))$.

Consider configuration $\mathsf{col}_{000}(\mathcal{Q})$ with the queries at positions $\mathcal{S}_9=\{\mathsf{1bl},\mathsf{2br}\}$ left out (see Figure 4.10). For any set of queries that fits this configuration at positions $\{\mathsf{1t},\mathsf{1br},\mathsf{2t},\mathsf{2bl}\}$, the tuple

$$\begin{aligned}&(\mathsf{a}_1\cdot(u,v,c_1),\mathsf{a}_1\cdot(u,v,c_1,w),y-\mathsf{a}_1\cdot(u,v,c_1,w))\\ &=(\mathsf{a}_3\cdot(u',v',c_1'),\mathsf{a}_4\cdot(u',v',c_1',w'),z-\mathsf{a}_4\cdot(u',v',c_1',w'))\end{aligned}$$

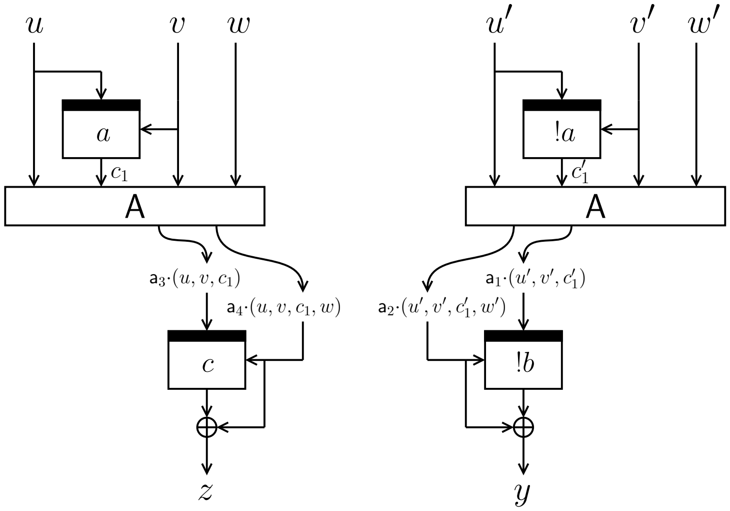
is added to $\mathcal{W}_{\mathcal{S}_9}$. Note that we particularly require $y=z$.

We split this wish list up into two sets: $\mathcal{W}_{\mathcal{S}_9}^{=}$ contains only wishes of which the corresponding queries at positions $\mathsf{1br},\mathsf{2bl}$ are the same, and $\mathcal{W}_{\mathcal{S}_9}^{\neq}$ contains only wishes of which the corresponding queries at positions $\mathsf{1br},\mathsf{2bl}$ are different. Note that by construction, a wish may occur in both sets, but this does not invalidate the security analysis: it only results in a slightly worse bound. As before, each element from $\mathcal{W}_{\mathcal{S}_9}$ can be wished for only once, and we find that the adversary finds a collision with probability at most

$$\frac{|\mathcal{W}_{\mathcal{S}_9}^{=}|+|\mathcal{W}_{\mathcal{S}_9}^{\neq}|}{2^n-q}\,.$$

We upper bound the number of solutions to the configuration of Figure 4.10, restricted to either $\mathsf{1br}=\mathsf{2bl}$ and $\mathsf{1br}\neq\mathsf{2bl}$, after q queries.

$\mathsf{1br}=\mathsf{2bl}$. We have at most q choices for $\mathsf{1br}=\mathsf{2bl}$. For any such choice, by $\neg\mathsf{aux}_3(\mathcal{Q}_q)$ we have at most τ_2 choices for $\mathsf{1t}$. The queries at positions



u
v
w
a
c_1
A
$\mathsf{a}_3\cdot(u,v,c_1)$
$\mathsf{a}_4\cdot(u,v,c_1,w)$
c
z
u'
v'
w'
$!a$
c_1'
A
$\mathsf{a}_1\cdot(u',v',c_1')$
$\mathsf{a}_2\cdot(u',v',c_1',w')$
$!b$
y

Figure 4.10. Configuration $\mathsf{col}_{000:\mathcal{S}_9}(\mathcal{Q})$. We require $(u,v,w) \neq (u',v',w')$ and $(\mathsf{a}_1\cdot(u,v,c_1), \mathsf{a}_1\cdot(u,v,c_1,w), y-\mathsf{a}_1\cdot(u,v,c_1,w)) = (\mathsf{a}_3\cdot(u',v',c_1'), \mathsf{a}_4\cdot(u',v',c_1',w'), z-\mathsf{a}_4\cdot(u',v',c_1',w'))$.

1t and $1\mathsf{br}$ uniquely fix (u,v,c_1,w). The query $1\mathsf{br} = 2\mathsf{bl}$ fixes $y = z$. Any choice of queries thus uniquely fixes $(\mathsf{a}_1\cdot(u,v,c_1), \mathsf{a}_1\cdot(u,v,c_1,w), y-\mathsf{a}_1\cdot(u,v,c_1,w))$. We find $|\mathcal{W}^{=}_{\mathcal{S}_9}| \leq \tau_2 q$;

$1\mathsf{br} \neq 2\mathsf{bl}$. As we require $y = z$, by $\neg\mathsf{aux}_1(\mathcal{Q}_q)$ the configuration has at most τ_1 choices for $1\mathsf{br}, 2\mathsf{bl}$. The remainder is the same, and we find $|\mathcal{W}^{\neq}_{\mathcal{S}_9}| \leq \tau_1\tau_2$.

Hence, in this setting a collision is found with probability at most $\frac{\tau_1\tau_2+\tau_2 q}{2^n-q}$.

$\mathsf{col}_{000:\mathcal{S}_j}(\mathcal{Q}_q)$ for $j = 10, 11, 12$. For these cases, the analysis for $\mathcal{S}_4$ directly applies.

$\mathsf{col}_{000:\mathcal{S}_{13}}(\mathcal{Q}_q)$. For this case, the analysis for $\mathcal{S}_6$ directly applies.

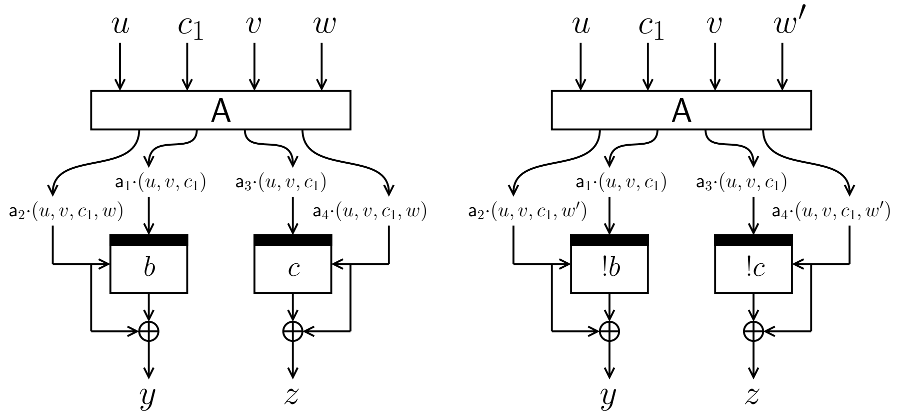
The proof is now completed by adding all bounds in accordance with (4.6). □

Lemma 4.3.4. $\mathbf{Pr}\left(\mathsf{col}_{100}(\mathcal{Q}_q) \wedge \neg\mathsf{aux}(\mathcal{Q}_q)\right) \leq \frac{2q+2\tau_1\tau_2}{2^n-q}$.

Proof. Sub-configuration $\mathsf{col}_{100}(\mathcal{Q}_q)$ is given in Figure 4.11. Note that here we in particular have $(u,v,c_1) = (u',v',c_1')$ as $1\mathsf{t} = 2\mathsf{t}$.

The approach is similar to the one for Lemma 4.3.3 and we only highlight the structural differences. Discarding symmetric cases caused by swapping w and w', one identifies 4 sets of positions at which the winning query can contribute:

$$\mathcal{S}_1 = \{1\mathsf{bl}\}, \qquad \mathcal{S}_2 = \{1\mathsf{br}\}, \qquad \mathcal{S}_3 = \{1\mathsf{bl}, 1\mathsf{br}\}, \qquad \mathcal{S}_4 = \{1\mathsf{bl}, 2\mathsf{br}\}.$$



u c_1 v w
A
$\mathsf{a}_1{\cdot}(u,v,c_1)$ $\mathsf{a}_3{\cdot}(u,v,c_1)$
$\mathsf{a}_2{\cdot}(u,v,c_1,w)$ $\mathsf{a}_4{\cdot}(u,v,c_1,w)$
b c
y z
u c_1 v w'
A
$\mathsf{a}_1{\cdot}(u,v,c_1)$ $\mathsf{a}_3{\cdot}(u,v,c_1)$
$\mathsf{a}_2{\cdot}(u,v,c_1,w')$ $\mathsf{a}_4{\cdot}(u,v,c_1,w')$
$!b$ $!c$
y z

Figure 4.11. Configuration $\mathsf{col}_{100}(\mathcal{Q})$. We require $w \neq w'$.

As before, we find

$$\mathbf{Pr}\left(\mathsf{col}_{100}(\mathcal{Q}_q) \wedge \neg\mathsf{aux}(\mathcal{Q}_q)\right) \leq \sum_{j=1}^{4} \mathbf{Pr}\left(\mathsf{col}_{100:\mathcal{S}_j}(\mathcal{Q}_q) \wedge \neg\mathsf{aux}(\mathcal{Q}_q)\right) . \qquad (4.7)$$

$\mathsf{col}_{100:\mathcal{S}_j}(\mathcal{Q}_q)$ for $j = 1, 2$. Both cases are the same by symmetry, and we consider $\mathcal{S}_1$ only.

Consider configuration $\mathsf{col}_{100}(\mathcal{Q})$ with the query at position $\mathcal{S}_1 = \{1\mathsf{bl}\}$ left out. By $\neg\mathsf{aux}_1(\mathcal{Q}_q)$, the configuration has at most τ_1 choices for $1\mathsf{br}, 2\mathsf{br}$. For any such choice, by $\neg\mathsf{aux}_3(\mathcal{Q}_q)$ we have at most τ_2 choices for $1\mathsf{t} = 2\mathsf{t}$. (Note that the query at $1\mathsf{t} = 2\mathsf{t}$ may be made *after* the winning query. This is because in this case "winning query" refers to a winning query for configuration $\mathsf{col}_{100}(\mathcal{Q}_q)$. We stress that this does not invalidate the security analysis.) Any such choice fixes u, v, c_1, w' (as $\mathsf{a}_{44} \neq 0$), and thus the query at position $2\mathsf{bl}$, and consequently y. The query at position $1\mathsf{t}$ fixes (u, v, c_1) and together with query $1\mathsf{br}$ this fixes w. Any choice of queries thus uniquely fixes $(\mathsf{a}_1{\cdot}(u,v,c_1), \mathsf{a}_1{\cdot}(u,v,c_1,w), y - \mathsf{a}_1{\cdot}(u,v,c_1,w))$. We find $|\mathcal{W}_{\mathcal{S}_1}| \leq \tau_1\tau_2$, and hence in this setting a collision is found with probability at most $\frac{\tau_1\tau_2}{2^n - q}$.

$\mathsf{col}_{100:\mathcal{S}_3}(\mathcal{Q}_q)$. By construction, there must be a query (k, m, c) in the query history (corresponding to position $1\mathsf{t}$) that satisfies $\mathsf{a}_1 \cdot (k, m, c) = \mathsf{a}_3 \cdot (k, m, c)$. So the problem shifts to bounding the probability that the adversary ever finds a query (k, m, c) that satisfies this equation. As $\mathsf{a}_{12} \neq \mathsf{a}_{32}$ and $\mathsf{a}_{13} \neq \mathsf{a}_{33}$, the adversary never obtains such query except with probability at most $\frac{q}{2^n - q}$.

$\mathsf{col}_{100:\mathcal{S}_4}(\mathcal{Q}_q)$. As in the current case we have $(u, v, c_1) = (u', v', c_1')$, the approach for $\mathcal{S}_3$ applies.

The proof is now completed by adding all bounds in accordance with (4.7). □

Lemma 4.3.5. $\mathbf{Pr}\left(\mathsf{col}_{\alpha_1\alpha_2\alpha_3}(\mathcal{Q}_q) \wedge \neg\mathsf{aux}(\mathcal{Q}_q)\right) \leq \frac{\tau_2^2 q+\tau_2 q+q+\tau_1\tau_2}{2^n-q}$ *for* $\alpha_1\alpha_2\alpha_3 \in \{010, 001\}$.

Proof. Both cases are the same by symmetry, and we consider $\alpha_1\alpha_2\alpha_3 = 010$ only. Sub-configuration $\mathsf{col}_{010}(\mathcal{Q})$ is given in Figure 4.12. Note that here we in particular have $\mathsf{a}_1\cdot(u,v,c_1) = \mathsf{a}_1\cdot(u',v',c_1')$ and $\mathsf{a}_3\cdot(u,v,c_1,w) = \mathsf{a}_3\cdot(u',v',c_1',w')$.

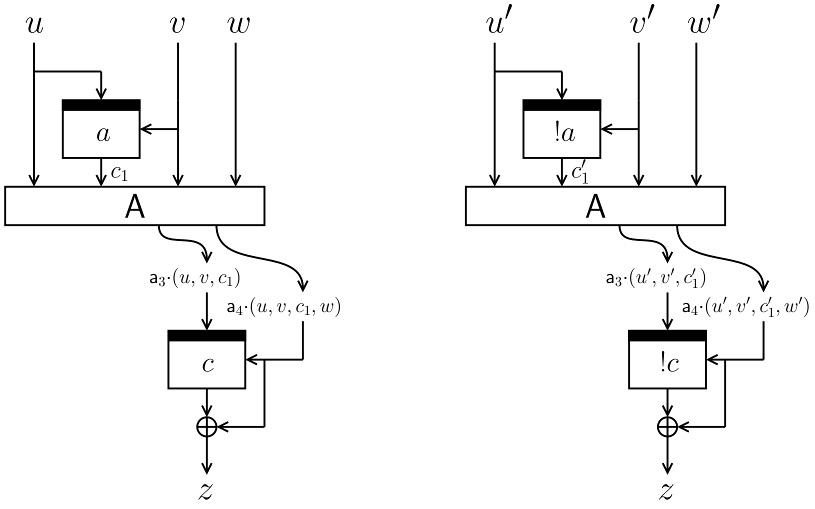


Figure 4.12. Configuration $\mathsf{col}_{010}(\mathcal{Q})$. We require $(u,v,w) \neq (u',v',w')$, $\mathsf{a}_1 \cdot (u,v,c_1) = \mathsf{a}_1 \cdot (u',v',c_1')$, and $\mathsf{a}_3 \cdot (u,v,c_1,w) = \mathsf{a}_3\cdot(u',v',c_1',w')$.

The approach is similar to the one for Lemma 4.3.3 and we only highlight the structural differences. Discarding symmetric cases caused by swapping (u,v,w) and (u',v',w'), one identifies 4 sets of positions at which the winning query can contribute:

$$\mathcal{S}_1 = \{1\mathsf{t}\}\,, \qquad \mathcal{S}_2 = \{1\mathsf{br}\}\,, \qquad \mathcal{S}_3 = \{1\mathsf{t}, 1\mathsf{br}\}\,, \qquad \mathcal{S}_4 = \{1\mathsf{t}, 2\mathsf{br}\}\,.$$

As before, we find

$$\mathbf{Pr}\left(\mathsf{col}_{010}(\mathcal{Q}_q) \wedge \neg\mathsf{aux}(\mathcal{Q}_q)\right) \leq \sum_{j=1}^{4} \mathbf{Pr}\left(\mathsf{col}_{010:\mathcal{S}_j}(\mathcal{Q}_q) \wedge \neg\mathsf{aux}(\mathcal{Q}_q)\right)\,. \tag{4.8}$$

$\boldsymbol{\mathsf{col}_{010:\mathcal{S}_1}(\mathcal{Q}_q)}$**.** Consider configuration $\mathsf{col}_{010}(\mathcal{Q})$ with the query at position $\mathcal{S}_1 = \{1\mathsf{t}\}$ left out. By $\neg\mathsf{aux}_1(\mathcal{Q}_q)$, the configuration has at most τ_1 choices for $1\mathsf{br}, 2\mathsf{br}$. For any such choice, by $\neg\mathsf{aux}_3(\mathcal{Q}_q)$ we have at most τ_2 choices for $2\mathsf{t}$. Any such choice fixes u', v', c_1', w' (as $\mathsf{a}_{44} \neq 0$), and thus the values $\mathsf{a}_1{\cdot}(u, v, c_1) = \mathsf{a}_1{\cdot}(u', v', c_1')$ and $\mathsf{a}_3 \cdot (u, v, c_1, w) = \mathsf{a}_3 \cdot (u', v', c_1', w')$. Together with the query $1\mathsf{br}$ this fixes (u, v, c_1) by invertibility of A. We find $|\mathcal{W}_{\mathcal{S}_1}| \leq \tau_1\tau_2$, and hence in this setting a collision is found with probability at most $\frac{\tau_1\tau_2}{2^n-q}$.

$\mathsf{col}_{010:\mathcal{S}_2}(\mathcal{Q}_q)$. Consider configuration $\mathsf{col}_{010}(\mathcal{Q})$ with the query at position $\mathcal{S}_2 = \{1\mathsf{br}\}$ left out. We have at most q choices for $1\mathsf{bl} = 2\mathsf{bl}$. For any such choice, by $\neg\mathsf{aux}_2(\mathcal{Q}_q)$ we have at most τ_2 choices for $1\mathsf{t}$ and at most τ_2 choices for $2\mathsf{t}$. Any such choice fixes the query at position $2\mathsf{br}$, and thus z. The query at position $1\mathsf{t}$ fixes (u, v, c_1) and together with query $1\mathsf{bl}$ this fixes w. Any choice of queries thus uniquely fixes $(\mathsf{a}_3{\cdot}(u, v, c_1), \mathsf{a}_4{\cdot}(u, v, c_1, w), z - \mathsf{a}_4{\cdot}(u, v, c_1, w))$. We find $|\mathcal{W}_{\mathcal{S}_2}| \leq \tau_2^2 q$, and hence in this setting a collision is found with probability at most $\frac{\tau_2^2 q}{2^n-q}$.

$\mathsf{col}_{010:\mathcal{S}_3}(\mathcal{Q}_q)$. As the winning query (k, m, c) should appear at positions $1\mathsf{t}$ and $1\mathsf{br}$, we require it to satisfy $k = \mathsf{a}_3 \cdot (k, m, c)$. Any query satisfies this equation with probability at most $\frac{1}{2^n-q}$ (as $\mathsf{a}_{32}, \mathsf{a}_{33} \neq 0$). As the adversary makes at most q queries, in this setting a collision is found with probability at most $\frac{q}{2^n-q}$.

$\mathsf{col}_{010:\mathcal{S}_4}(\mathcal{Q}_q)$. Consider any query, without loss of generality a forward query on input (k, m). Note that, as the query appears at positions $1\mathsf{t}$ and $2\mathsf{br}$, we have $k = u = \mathsf{a}_3 \cdot (u', v', c_1')$ and $m = v = \mathsf{a}_4 \cdot (u', v', c_1', w')$. By $\neg\mathsf{aux}_3(\mathcal{Q}_q)$, the configuration has at most τ_2 choices for $2\mathsf{t}$. Any such query fixes (u', v', c_1'). Recall that, as $1\mathsf{bl} = 2\mathsf{bl}$, we require $\mathsf{a}_1 \cdot (u, v, c_1) = \mathsf{a}_1 \cdot (u', v', c_1')$. Now, the query succeeds only if c_1 satisfies this equation, hence with probability at most $\frac{\tau_2}{2^n-q}$ (as $\mathsf{a}_{13} \neq 0$). Exactly the same statement holds for inverse queries (as $\mathsf{a}_{12} \neq 0$). As the adversary makes at most q queries, in this setting a collision is found with probability at most $\frac{\tau_2 q}{2^n-q}$.

The proof is now completed by adding all bounds in accordance with (4.8). $\square$

Lemma 4.3.6. $\mathbf{Pr}\left(\mathsf{col}_{\alpha_1\alpha_2\alpha_3}(\mathcal{Q}_q) \wedge \neg\mathsf{aux}(\mathcal{Q}_q)\right) = 0$ *when* $\alpha_1 + \alpha_2 + \alpha_3 \geq 2$.

Proof. First suppose $\alpha_2 = \alpha_3 = 1$. By design of $\mathsf{F}^3_{\mathsf{A}}$ we require $\mathsf{A}(u, v, c_1, w)^\top = \mathsf{A}(u', v', c_1', w')^\top$. By invertibility of A this gives $(u, v, w) = (u', v', w')$ and this implies that the collision is trivial. Now, suppose $\alpha_1 = \alpha_2 = 1$ (the case of $\alpha_1 = \alpha_3 = 1$ is the same by symmetry). In this case, by design of $\mathsf{F}^3_{\mathsf{A}}$ we have $(u, v, c_1) = (u', v', c_1')$ (by $\alpha_1 = 1$) and $\mathsf{a}_1 \cdot (u, v, c_1, w) = \mathsf{a}_1 \cdot (u', v', c_1', w')$ (by $\alpha_2 = 1$). As $\mathsf{a}_{24} \neq 0$ this implies $w = w'$ and thus that the collision is trivial. $\square$

Lemma 4.3.7. $\mathbf{Pr}\left(\mathsf{aux}(\mathcal{Q}_q)\right) \leq \frac{q^2}{\tau_1(2^n-q)} + 3 \cdot 2^n \left(\frac{eq}{\tau_2(2^n-q)}\right)^{\tau_2}$.

Proof. Note that $\mathsf{aux}_1(\mathcal{Q}_q)$ essentially equals $\mathsf{help}_1(\mathcal{Q}_q)$ of Section 3.2.1, and the proof and bound directly carry over. The analysis for $\mathsf{aux}_2(\mathcal{Q}_q)$, $\mathsf{aux}_3(\mathcal{Q}_q)$, and

$\mathsf{aux}_4(\mathcal{Q}_q)$ essentially equals the one for $\mathsf{help}_4(\mathcal{Q}_q)$ of Section 3.2.1. We include the proof for completeness.

It suffices to consider the events $\mathbf{Pr}\left(\mathsf{aux}_k(\mathcal{Q}_q)\right)$ $(k = 1, \ldots, 4)$ separately.
$\mathsf{aux}_1(\mathcal{Q}_q)$. For $i \neq j$, the two queries (K_i, m_i, c_i) and (K_j, m_j, c_i) satisfy $m_i + c_i = m_j + c_j$ with probability at most $\frac{1}{2^n-q}$. Hence, the expected value $\mathsf{E}(m_i + c_i = m_j + c_j)$ is at most $\frac{1}{2^n-q}$, and consequently

$$\begin{aligned}\mathsf{E}\left(\left|\left\{(K_i, m_i, c_i), (K_j, m_j, c_j) \in \mathcal{Q}_q \;\middle|\; i \neq j \;\wedge\; m_i + c_i = m_j + c_j\right\}\right|\right) &\leq \sum_{i \neq j} \frac{1}{2^n - q} \\ &\leq \frac{q^2}{2^n - q}\,.\end{aligned}$$

By Markov's inequality, we obtain

$$\mathbf{Pr}\left(\mathsf{aux}_1(\mathcal{Q}_q)\right) \leq \frac{q^2}{\tau_1(2^n - q)}\,. \tag{4.9}$$

$\mathsf{aux}_k(\mathcal{Q}_q)$ for $k \in \{2, 3, 4\}$. For the proof to go through we use $\mathsf{a}_{12}, \mathsf{a}_{13} \neq 0$ (for $\mathsf{aux}_2(\mathcal{Q}_q)$) and $\mathsf{a}_{32}, \mathsf{a}_{33} \neq 0$ (for $\mathsf{aux}_3(\mathcal{Q}_q)$). The cases are equivalent by symmetry, and we consider $\mathsf{aux}_2(\mathcal{Q}_q)$ only. Let $z \in \mathbb{Z}_2^n$. Consider the i^{th} query (K_i, m_i, c_i). This query makes equation $\mathsf{a}_1 \cdot (K_i, m_i, c_i) = z$ satisfied with probability at most $\frac{1}{2^n-q}$. More than τ_2 queries result in a solution with probability at most $\binom{q}{\tau_2}\left(\frac{1}{2^n-q}\right)^{\tau_2} \leq \left(\frac{eq}{\tau_2(2^n-q)}\right)^{\tau_2}$, where we use Stirling's approximation. Considering any possible choice for z, we obtain for $k = 2, 3, 4$:

$$\mathbf{Pr}\left(\mathsf{aux}_k(\mathcal{Q}_q)\right) \leq 2^n \left(\frac{eq}{\tau_2(2^n - q)}\right)^{\tau_2}. \tag{4.10}$$

The claim is obtained by adding (4.9-4.10). □

From (4.5) and the results of Lemmas 4.3.3-4.3.7 we conclude for $\mathbf{Adv}^{\mathrm{col}}_{\mathsf{F}^3_{\mathsf{A}}}(q)$:

$$\begin{aligned}\mathbf{Adv}^{\mathrm{col}}_{\mathsf{F}^3_{\mathsf{A}}}(q) \leq \frac{2\tau_2^2 q + 3\tau_2 q + 11q + 3\tau_1\tau_2^2 + 7\tau_1\tau_2}{2^n - q} + \frac{q^2}{\tau_1(2^n - q)} + \\ 3 \cdot 2^n \left(\frac{eq}{\tau_2(2^n - q)}\right)^{\tau_2}.\end{aligned}$$

This completes the proof of Theorem 4.3.1.

4.3.2 Preimage Resistance of $\mathsf{F}^3_{\mathsf{A}}$

In this section we consider the preimage resistance of $\mathsf{F}^3_{\mathsf{A}}$. Though we do not obtain optimal preimage resistance — which is impossible to achieve after all, due to the generic bounds of the pigeonhole-birthday attack (Section 4.1) — we achieve preimage resistance up to $2^{3n/2}$ queries, much better than the preimage bounds on MDC-2 and MDC-4 (see Chapter 3), relatively close to the generic bound. Yet, for the proof to hold we need to put slightly stronger requirements on A.

- $\mathsf{A} - \left(\begin{array}{cc} \mathsf{B}_1 & \begin{array}{cc} 0 & 0 \\ 0 & 0 \end{array} \\ \mathsf{B}_2 & \begin{array}{cc} 0 & 0 \\ 0 & 0 \end{array} \end{array}\right)$ is invertible for any $\mathsf{B}_1, \mathsf{B}_2 \in \left\{\left(\begin{smallmatrix} 0 & 0 \\ 0 & 0 \end{smallmatrix}\right), \left(\begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix}\right), \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)\right\}$. In the remainder, we denote the subtracted matrix by $\left[\mathsf{B}_1/\mathsf{B}_2\right]$;
- $\mathsf{a}_{12}, \mathsf{a}_{13}, \mathsf{a}_{24}, \mathsf{a}_{32}, \mathsf{a}_{33}, \mathsf{a}_{44} \neq 0$;
- $\mathsf{a}_{12} \neq \mathsf{a}_{32}$, $\mathsf{a}_{13} \neq \mathsf{a}_{33}$, and $\mathsf{a}_{24} \neq \mathsf{a}_{44}$.

We refer to the logical AND of these requirements as `prereq`. We remark that `prereq` $\Rightarrow$ `colreq`, and that matrices satisfying `prereq` are easily found. Simple matrices complying with these conditions over the field $GF(2^{128})$ are

$$\begin{pmatrix} 0 & 1 & 2 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix}, \qquad \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 2 & 3 & 0 \\ 1 & 0 & 2 & 2 \end{pmatrix}. \tag{4.11}$$

These are the matrices corresponding to the compression functions of Figure 4.1. Here, we use $x^{128} + x^{127} + x^{126} + x^{121} + 1$ as our irreducible polynomial and we represent bit strings as polynomials in the obvious way ($1 = 1$, $2 = x$, $3 = 1 + x$). Note that the choice of matrix A influences the efficiency of the construction. The first matrix of (4.11) has as many zeroes as possible, which reduces the amount of computation.

Theorem 4.3.8. *Let $n \in \mathbb{Z}_2^n$. Suppose A satisfies `prereq`. Then, for any positive integral value τ, provided $\tau \leq q$,*

$$\mathbf{Adv}_{\mathsf{F}_\mathsf{A}^3}^{\mathrm{epre}}(q) \leq \frac{6\tau^2 + 18\tau + 26}{2^n - 2} + 4 \cdot 2^n \left(\frac{4eq}{\tau 2^n}\right)^{\tau/2} + 8q \left(\frac{8eq}{\tau 2^n}\right)^{\frac{\tau 2^n}{4q}}. \tag{4.12}$$

The proof is given in Section 4.3.2.1. As for the bound on the collision resistance (Theorem 4.3.1), the idea is to make a smart choice of τ to minimize this bound. Let $\varepsilon > 0$ be any parameter. Then, for $\tau = q^{1/3}$, the bound simplifies to

$$\mathbf{Adv}_{\mathsf{F}_\mathsf{A}^3}^{\mathrm{epre}}(q) \leq \frac{6q^{2/3} + 18q^{1/3} + 26}{2^n - 2} + 4 \cdot 2^n \left(\frac{4eq^{2/3}}{2^n}\right)^{q^{1/3}/2} + 8q \left(\frac{8eq^{2/3}}{2^n}\right)^{\frac{2^n}{4q^{2/3}}}.$$

We find that for any $\varepsilon > 0$, this term goes to 0 for $n \to \infty$ when $q = 2^{3n/2}/2^{n\varepsilon}$.

Corollary 4.3.9. *For any $\varepsilon > 0$, we obtain $\lim_{n\to\infty} \mathbf{Adv}_{\mathsf{F}_\mathsf{A}^3}^{\mathrm{epre}}\left(2^{3n/2}/2^{n\varepsilon}\right) = 0$.*

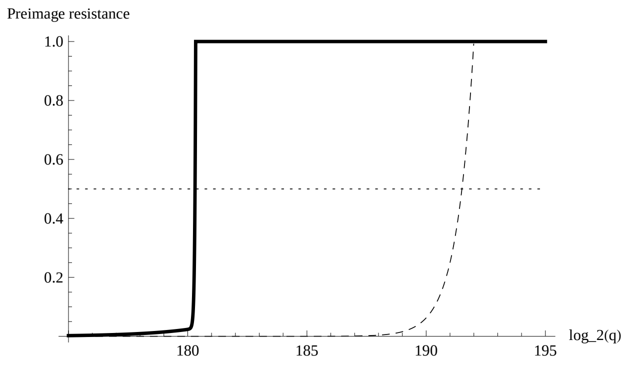


Figure 4.13. The function $\mathbf{Adv}^{\text{epre}}_{\mathsf{F}^3_{\mathsf{A}}}(q)$ of (4.12) for $n = 128$ and the particular choice of value τ (solid line), in comparison with the asymptotic bound of $q^2/2^{3n}$ (dashed line). The steepness of our bound is caused by the last term of (4.12) which explodes for q approaching $\tau 2^n$ due to its decreasing exponent.

Hence, the $\mathsf{F}^3_{\mathsf{A}}$ compression function achieves close to $2^{3n/2}$ preimage security for $n \to \infty$. For $n = 128$, the bound on $\mathbf{Adv}^{\text{epre}}_{\mathsf{F}^3_{\mathsf{A}}}$ is depicted in Figure 4.13. The preimage advantage hits $1/2$ for $\log_2 q \approx 180.3$, relatively close to the threshold 191.5 for $q^2/2^{3n}$. For larger values of n this gap approaches 0.

The result shows that $\mathsf{F}^3_{\mathsf{A}}$ with A compliant to `prereq` satisfies preimage resistance up to about $2^{3n/2}$ queries. We note that our proof is the best possible for this design, by demonstrating a preimage-finding adversary that with high probability succeeds in at most $O(2^{3n/2})$ queries. Let $\alpha \in \mathbb{N}$. The adversary proceeds as follows.

(i) Make $\alpha 2^n$ queries to the block cipher corresponding to the bottom-left position of Figure 4.3. One expects to find α tuples (K_2, m_2, c_2) that satisfy $m_2 + c_2 = y$;
(ii) Repeat the first step for the bottom-right position. One expects to find α tuples (K_3, m_3, c_3) satisfying $m_3 + c_3 = z$;
(iii) By invertibility of A, any choice of (K_2, m_2, c_2) and (K_3, m_3, c_3) uniquely defines a tuple (u, v, c_1, w) for the $\mathsf{F}^3_{\mathsf{A}}$ evaluation. Likely, the emerged tuples (u, v, c_1) are all different, and we find about α^2 such tuples;
(iv) Varying over all α^2 tuples (u, v, c_1), query (u, v) to the block cipher. If it responds c_1, we have obtained a preimage for $\mathsf{F}^3_{\mathsf{A}}$.

In step (iv) one expects to find a preimage if $\alpha^2/2^n = 1$, or equivalently if $\alpha = 2^{n/2}$. Steps (i) and (ii) both require approximately $2^{3n/2}$ queries, and step (iv) takes 2^n queries. In total, the attack is done in approximately $2 \cdot 2^{3n/2} + 2^n$ queries.

4.3.2.1 Proof of Theorem 4.3.8

The proof of preimage resistance of $\mathsf{F}^3_{\mathsf{A}}$ follows the basic spirit of Section 3.3.1. We consider any adversary that has query access to its oracle E and makes q queries stored in a query history $\mathcal{Q}_q$. Its goal is to find a preimage for $\mathsf{F}^3_{\mathsf{A}}$, in which it by definition only succeeds if it obtains a query history $\mathcal{Q}_q$ that satisfies configuration $\mathsf{pre}(\mathcal{Q}_q)$ of Figure 4.14. This means,

$$\mathbf{Adv}^{\mathrm{epre}}_{\mathsf{F}^3_{\mathsf{A}}}(q) = \mathbf{Pr}\left(\mathsf{pre}(\mathcal{Q}_q)\right) . \tag{4.13}$$

We inherit the notation of Section 4.3.1.1. The underlining of y and z means that these are fixed (by the adversary) from the start. We name the block ciphers $1\mathsf{t}, 1\mathsf{bl}, 1\mathsf{br}$ similarly.

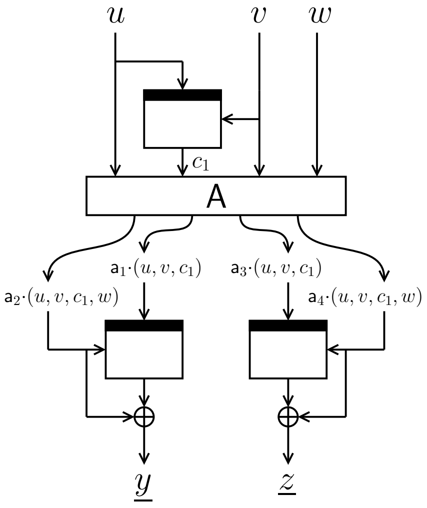

u
v
w
c_1
A
$\mathsf{a}_1\cdot(u,v,c_1)$
$\mathsf{a}_3\cdot(u,v,c_1)$
$\mathsf{a}_2\cdot(u,v,c_1,w)$
$\mathsf{a}_4\cdot(u,v,c_1,w)$
$\underline{y}$
$\underline{z}$

Figure 4.14. Configuration $\mathsf{pre}(\mathcal{Q})$. The configuration is satisfied if $\mathcal{Q}$ contains three (possibly not all different) queries that satisfy this setting.

For the analysis of the preimage resistance, we use the idea of free super queries (see [22,145,147] and Section 3.3.1). The issuance of free super queries is a well-established proof trick for achieving preimage resistance beyond the birthday bound. If under some key the adversary has made 2^{n-1} queries to E, it receives the remaining 2^{n-1} queries for this key for free. As in [22,147], we call this query a super query. Free queries can be formalized as queries the adversary is forced to make, but for which it will not be charged. For convenience, we use $\mathcal{Q}_q$ to denote the query history after q normal queries: it thus contains all normal queries plus all super queries made so far. A super query is a set of 2^{n-1} single queries, and any query in the query history is either a normal query or a part of a super query, but not both. Notice that at most $q/2^{n-1}$ super queries will occur: the adversary makes q queries, and needs 2^{n-1} queries as preparatory work to enforce one super query.

For the analysis of $\mathbf{Pr}\left(\mathsf{pre}(\mathcal{Q}_q)\right)$ we introduce an auxiliary event $\mathsf{aux}(\mathcal{Q}_q)$. Let $\tau > 0$ be any integral value. We define $\mathsf{aux}(\mathcal{Q}_q) = \mathsf{aux}_2(\mathcal{Q}_q) \vee \cdots \vee \mathsf{aux}_5(\mathcal{Q}_q)$, where

$$\begin{aligned}
\mathsf{aux}_2(\mathcal{Q}_q):&\quad \max_{z\in\mathbb{Z}_2^n} \left|\left\{(K_i,m_i,c_i)\in\mathcal{Q}_q \mid \mathsf{a}_1\cdot(K_i,m_i,c_i)=z\right\}\right| > \tau\,;\\
\mathsf{aux}_3(\mathcal{Q}_q):&\quad \max_{z\in\mathbb{Z}_2^n} \left|\left\{(K_i,m_i,c_i)\in\mathcal{Q}_q \mid \mathsf{a}_3\cdot(K_i,m_i,c_i)=z\right\}\right| > \tau\,;\\
\mathsf{aux}_4(\mathcal{Q}_q):&\quad \max_{z\in\mathbb{Z}_2^n} \left|\left\{(K_i,m_i,c_i)\in\mathcal{Q}_q \mid m_i+c_i=z\right\}\right| > \tau\,;\\
\mathsf{aux}_5(\mathcal{Q}_q):&\quad \max_{z\in\mathbb{Z}_2^n} \left|\left\{(K_i,m_i,c_i)\in\mathcal{Q}_q \mid (\mathsf{a}_1-\mathsf{a}_3)\cdot(K_i,m_i,c_i)=z\right\}\right| > \tau\,.
\end{aligned}$$

Note that $\mathsf{aux}_2(\mathcal{Q}_q), \mathsf{aux}_3(\mathcal{Q}_q), \mathsf{aux}_4(\mathcal{Q}_q)$ equal the ones of Section 4.3.1.1, but we reintroduce them for convenience. By basic probability theory, we obtain for (4.13):

$$\mathbf{Pr}\left(\mathsf{pre}(\mathcal{Q}_q)\right) \le \mathbf{Pr}\left(\mathsf{pre}(\mathcal{Q}_q) \wedge \neg\mathsf{aux}(\mathcal{Q}_q)\right) + \mathbf{Pr}\left(\mathsf{aux}(\mathcal{Q}_q)\right)\,. \tag{4.14}$$

Probability $\mathbf{Pr}\left(\mathsf{pre}(\mathcal{Q}_q) \wedge \neg\mathsf{aux}(\mathcal{Q}_q)\right)$ is bounded in Lemma 4.3.10. A bound on $\mathbf{Pr}\left(\mathsf{aux}(\mathcal{Q}_q)\right)$ is derived in Lemma 4.3.11.

Lemma 4.3.10. $\mathbf{Pr}\left(\mathsf{pre}(\mathcal{Q}_q) \wedge \neg\mathsf{aux}(\mathcal{Q}_q)\right) \le \frac{6\tau^2+18\tau+26}{2^n-2}$.

Proof. We consider the probability of the adversary finding a solution to configuration $\mathsf{pre}(\mathcal{Q}_q)$ of Figure 4.14 such that $\mathcal{Q}_q$ satisfies $\neg\mathsf{aux}(\mathcal{Q}_q)$. The proof shows similarities with the proof of Lemma 4.3.3, We call a (normal or super) query winning if it makes the configuration satisfied for any other queries in the query history *strictly before* this winning query is made. It may be the case that a winning query contributes to two or three positions in the configuration. In more detail, one can identify the following 7 sets of positions at which the winning query can contribute:

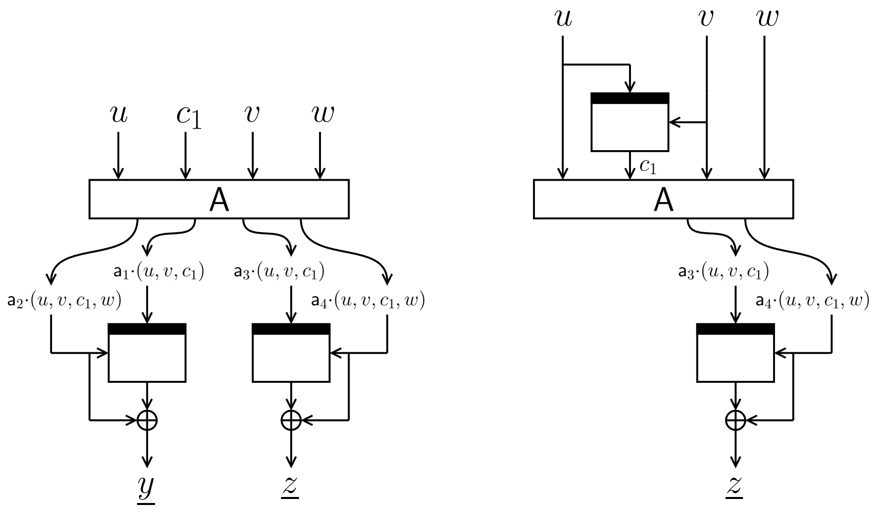
$$\begin{aligned}
\mathcal{S}_1 &= \{\mathsf{1t}\}\,, & \mathcal{S}_4 &= \{\mathsf{1t},\mathsf{1bl}\}\,, & \mathcal{S}_7 &= \{\mathsf{1t},\mathsf{1bl},\mathsf{1br}\}\,,\\
\mathcal{S}_2 &= \{\mathsf{1bl}\}\,, & \mathcal{S}_5 &= \{\mathsf{1t},\mathsf{1br}\}\,, & &\\
\mathcal{S}_3 &= \{\mathsf{1br}\}\,, & \mathcal{S}_6 &= \{\mathsf{1bl},\mathsf{1br}\}\,. & &
\end{aligned}$$

For $j = 1,\ldots,7$ we denote by $\mathsf{pre}_{\mathcal{S}_j}(\mathcal{Q}_q)$ configuration $\mathsf{pre}(\mathcal{Q}_q)$ with the restriction that the winning query *must* contribute to the positions in $\mathcal{S}_j$. Recall that a winning query may consist of different queries if it is a super query. By basic probability theory,

$$\mathbf{Pr}\left(\mathsf{pre}(\mathcal{Q}_q) \wedge \neg\mathsf{aux}(\mathcal{Q}_q)\right) \le \sum_{j=1}^{7} \mathbf{Pr}\left(\mathsf{pre}_{\mathcal{S}_j}(\mathcal{Q}_q) \wedge \neg\mathsf{aux}(\mathcal{Q}_q)\right)\,. \tag{4.15}$$

$\boldsymbol{\mathsf{pre}_{\mathcal{S}_1}(\mathcal{Q}_q)}$**.** In this case, the winning query may be a normal query or a super query. As is done in the proof of Lemma 4.3.3, we use wish lists for the analysis.

Consider configuration $\mathsf{pre}(\mathcal{Q}_q)$ with the query at position $\mathcal{S}_1 = \{1\mathsf{t}\}$ left out (see Figure 4.15). For any pair of queries that fits this configuration at positions $\{1\mathsf{bl}, 1\mathsf{br}\}$, the tuple (u, v, c_1) is added to $\mathcal{W}_{\mathcal{S}_1}$. Note that this tuple is uniquely determined by the queries at $1\mathsf{bl}$ and $1\mathsf{br}$ by invertibility of A.



u
c_1
v
w
A
$\mathsf{a}_1\cdot(u,v,c_1)$
$\mathsf{a}_3\cdot(u,v,c_1)$
$\mathsf{a}_2\cdot(u,v,c_1,w)$
$\mathsf{a}_4\cdot(u,v,c_1,w)$
$\underline{y}$
$\underline{z}$
u
v
w
c_1
A
$\mathsf{a}_3\cdot(u,v,c_1)$
$\mathsf{a}_4\cdot(u,v,c_1,w)$
$\underline{z}$

Figure 4.15. Configurations $\mathsf{pre}_{\mathcal{S}_1}(\mathcal{Q})$ (left) and $\mathsf{pre}_{\mathcal{S}_2}(\mathcal{Q})$ (right).

As before, as the winning query only occurs at $\mathcal{S}_1$, we can assume a query never adds itself to a wish list. In order to find a preimage for $\mathsf{F}^3_{\mathsf{A}}$ in this sub-configuration the adversary needs to get a wish granted at least once. The adversary can make each wish at most once. Note that it can make multiple wishes at the same time (in case of super queries), but this does not invalidate the analysis. Suppose the adversary makes a query $E(k, m)$ where $(k, m, c) \in \mathcal{W}_{\mathcal{S}_1}$ for some c. If the query is a normal query, the answer is drawn uniformly at random from a set of size at least 2^{n-1}. If, on the other hand, this wish is a part of a super query, the answer is generated from a set of size 2^{n-1}. In both cases, the wish is granted with probability at most $1/2^{n-1}$ (and the same for inverse queries). Thus, by construction, in this setting the adversary finds a preimage with probability at most $\frac{|\mathcal{W}_{\mathcal{S}_1}|}{2^{n-1}}$.

Now, it suffices to upper bound the size of the wish list $\mathcal{W}_{\mathcal{S}_1}$ after q queries, and to this end we bound the number of solutions to the configuration of Figure 4.15. By $\neg\mathsf{aux}_4(\mathcal{Q}_q)$, the configuration has at most τ choices for $1\mathsf{bl}$ and at most τ choices for $1\mathsf{br}$. For any such choice, the queries at positions $1\mathsf{bl}$ and $1\mathsf{br}$ uniquely fix (u, v, c_1). We find $|\mathcal{W}_{\mathcal{S}_1}| \le \tau^2$, and hence in this setting a preimage is found with probability at most $\tau^2/2^{n-1}$.

$\mathsf{pre}_{\mathcal{S}_j}(\mathcal{Q}_q)$ for $j = 2, 3$. Both cases are the same by symmetry, and we consider $\mathcal{S}_2$ only.

The analysis is similar to the one for $\mathcal{S}_1$, and we only present the computation of the bound on the wish list $\mathcal{W}_{\mathcal{S}_2}$ after q queries. Consider configuration $\mathsf{pre}(\mathcal{Q}_q)$ with the query at position $\mathcal{S}_2 = \{1\mathsf{bl}\}$ left out (see Figure 4.15). By $\neg\mathsf{aux}_4(\mathcal{Q}_q)$, the configuration has at most τ choices for $1\mathsf{br}$. For any such choice, by $\neg\mathsf{aux}_3(\mathcal{Q}_q)$ we have at most τ choices for $1\mathsf{t}$. The query at position $1\mathsf{t}$ fixes (u, v, c_1) and together with query $1\mathsf{br}$ this fixes w (as $\mathsf{a}_{44} \neq 0$). Any choice of queries thus uniquely fixes $(\mathsf{a}_1{\cdot}(u, v, c_1), \mathsf{a}_1{\cdot}(u, v, c_1, w), y - \mathsf{a}_1{\cdot}(u, v, c_1, w))$. We find $|\mathcal{W}_{\mathcal{S}_2}| \leq \tau^2$, and hence in this setting a preimage is found with probability at most $\tau^2/2^{n-1}$.

$\mathsf{pre}_{\mathcal{S}_j}(\mathcal{Q}_q)$ for $j = 4, 5$. Both cases are the same by symmetry, and we consider $\mathcal{S}_4$ only.

We make the distinction between whether or not the two queries at positions $\mathcal{S}_4 = \{1\mathsf{t}, 1\mathsf{bl}\}$ are the same (normal or super query), or are different (super query).

$1\mathsf{t} = 1\mathsf{bl}$. In this case, the wish list contains tuples of the form (k, m, c) that by construction are required to satisfy $k = \mathsf{a}_1 \cdot (k, m, c)$ and $m = \mathsf{a}_1 \cdot (k, m, c, w)$ for some w. As was the case with $\mathcal{S}_1$, each wish is granted with probability at most $1/2^{n-1}$. By $\neg\mathsf{aux}_4(\mathcal{Q}_q)$, the configuration has at most τ choices for $1\mathsf{br}$. For any such choice, this query fixes values $\mathsf{a}_3 \cdot (k, m, c)$ and $\mathsf{a}_4 \cdot (k, m, c, w)$. Together with the equations on a_1 and a_2 this uniquely fixes (k, m, c) by invertibility of $\mathsf{A} - \left[\left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)/\left(\begin{smallmatrix} 0 & 0 \\ 0 & 0 \end{smallmatrix}\right)\right]$. We find $|\mathcal{W}_{\mathcal{S}_4}| \leq \tau$, and hence in this setting a preimage is found with probability at most $\tau/2^{n-1}$;

$1\mathsf{t} \neq 1\mathsf{bl}$. In this case, the wish list contains tuples of the form (k, m_1, c_1, m_2, c_2), where (k, m_1, c_1) is the wished query at $1\mathsf{t}$ and (k, m_2, c_2) is the wished query at $1\mathsf{bl}$. By construction, these tuples are required to satisfy $k = \mathsf{a}_1{\cdot}(k, m_1, c_1)$ and $m_2 = \mathsf{a}_1 \cdot (k, m_1, c_1, w)$ for some w. Additionally the wish is required to satisfy $m_2 + c_2 = y$. As in a super query the answers are generated from a set of size 2^{n-1}, a wish is granted with probability at most $\frac{1}{2^{n-1}(2^{n-1}-1)}$. Thus, by construction, in this setting the adversary finds a preimage with probability at most

$$\frac{|\mathcal{W}_{\mathcal{S}_4}|}{2^{n-1}(2^{n-1}-1)}\,.$$

By $\neg\mathsf{aux}_4(\mathcal{Q}_q)$, the configuration has at most τ choices for $1\mathsf{br}$. For any such choice, this query fixes values $\mathsf{a}_3{\cdot}(k, m_1, c_1)$ and $\mathsf{a}_4{\cdot}(k, m_1, c_1, w)$. We have 2^n choices for c_2. This uniquely fixes m_2. Now, this uniquely fixes (k, m_1, c_1) by invertibility of $\mathsf{A} - \left[\left(\begin{smallmatrix} 1 & 0 \\ 0 & 0 \end{smallmatrix}\right)/\left(\begin{smallmatrix} 0 & 0 \\ 0 & 0 \end{smallmatrix}\right)\right]$. We find $|\mathcal{W}_{\mathcal{S}_4}| \leq \tau 2^n$, and hence in this setting a preimage is found with probability at most $\frac{\tau 2^n}{2^{n-1}(2^{n-1}-1)}$.

$\mathsf{pre}_{\mathcal{S}_6}(\mathcal{Q}_q)$. We make the distinction between whether or not the two queries at positions $\mathcal{S}_6 = \{1\mathsf{bl}, 1\mathsf{br}\}$ are the same (normal or super query), or are different (super query).

$\mathsf{1bl} = \mathsf{1br}$. In this case, the wish list contains tuples of the form (k, m, c) that by construction are required to satisfy $k = \mathsf{a}_1 \cdot (u, v, c_1) = \mathsf{a}_3 \cdot (u, v, c_1)$, and $m = \mathsf{a}_1 \cdot (u, v, c_1, w) = \mathsf{a}_4 \cdot (u, v, c_1, w)$ for some u, v, c_1, w. Additionally the wish is required to satisfy $m + c = y = z$. As was the case with $\mathcal{S}_1$, each wish is granted with probability at most $1/2^{n-1}$.

By $\neg\mathsf{aux}_5(\mathcal{Q}_q)$, noting that $\mathsf{a}_1 \cdot (u, v, c_1) = \mathsf{a}_3 \cdot (u, v, c_1)$, the configuration has at most τ choices for $\mathsf{1t}$. For any such choice, this query fixes values (u, v, c_1), and thus k. Equation $\mathsf{a}_1 \cdot (u, v, c_1, w) = \mathsf{a}_4 \cdot (u, v, c_1, w)$ fixes w (as $\mathsf{a}_{24} \neq \mathsf{a}_{44}$), and thus m. Using $m + c = y$ this uniquely fixes (k, m, c). We find $|\mathcal{W}_{\mathcal{S}_6}| \leq \tau$, and hence in this setting a preimage is found with probability at most $\tau/2^{n-1}$;

$\mathsf{1bl} \neq \mathsf{1br}$. In this case, the wish list contains tuples of the form (k, m_2, c_2, m_3, c_3), where (k, m_2, c_2) is the wished query at $\mathsf{1bl}$ and (k, m_3, c_3) is the wished query at $\mathsf{1br}$. By construction, these tuples are required to satisfy $k = \mathsf{a}_1 \cdot (u, v, c_1) = \mathsf{a}_3 \cdot (u, v, c_1)$, $m_2 = \mathsf{a}_1 \cdot (u, v, c_1, w)$, and $m_3 = \mathsf{a}_4 \cdot (u, v, c_1, w)$ for some u, v, c_1, w. Additionally the wish is required to satisfy $m_2 + c_2 = y$ and $m_3 + c_3 = z$. As before, in this setting the adversary finds a preimage with probability at most

$$\frac{|\mathcal{W}_{\mathcal{S}_6}|}{2^{n-1}(2^{n-1} - 1)} .$$

By $\neg\mathsf{aux}_5(\mathcal{Q}_q)$, noting that $\mathsf{a}_1 \cdot (u, v, c_1) = \mathsf{a}_3 \cdot (u, v, c_1)$, the configuration has at most τ choices for $\mathsf{1t}$. For any such choice, this query fixes values (u, v, c_1) and thus k. We have 2^n choices for c_3. This uniquely fixes m_3. This uniquely fixes w, and subsequently m_2 and c_2. We find $|\mathcal{W}_{\mathcal{S}_6}| \leq \tau 2^n$, and hence in this setting a preimage is found with probability at most $\frac{\tau 2^n}{2^{n-1}(2^{n-1}-1)}$.

$\mathsf{pre}_{\mathcal{S}_7}(\mathcal{Q}_q)$. We make the following distinction: $\mathsf{1t} = \mathsf{1bl} = \mathsf{1br}$, $\mathsf{1t} = \mathsf{1bl} \neq \mathsf{1br}$, $\mathsf{1t} = \mathsf{1br} \neq \mathsf{1bl}$, $\mathsf{1bl} = \mathsf{1br} \neq \mathsf{1t}$, and $\{\mathsf{1t}, \mathsf{1bl}, \mathsf{1br}\}$ all different.

$\mathsf{1t} = \mathsf{1bl} = \mathsf{1br}$. In this case, the wish list contains tuples of the form (k, m, c) that by construction are required to satisfy $k = \mathsf{a}_1 \cdot (k, m, c) = \mathsf{a}_3 \cdot (k, m, c)$ and $m = \mathsf{a}_1 \cdot (k, m, c, w) = \mathsf{a}_4 \cdot (k, m, c, w)$ for some w. These equations uniquely determine (k, m, c, w) by invertibility of $\mathsf{A} - \left[\left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right) / \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)\right]$, and we find $|\mathcal{W}_{\mathcal{S}_1}| = 1$. Hence, in this setting a preimage is found with probability at most $1/2^{n-1}$;

$\mathsf{1t} = \mathsf{1bl} \neq \mathsf{1br}$ or $\mathsf{1t} = \mathsf{1br} \neq \mathsf{1bl}$. Both cases are the same by symmetry, and we consider $\mathsf{1t} = \mathsf{1bl} \neq \mathsf{1br}$ only.

In this case, the wish list contains tuples of the form (k, m, c, m_3, c_3), where (k, m, c) is the wished query at $\mathsf{1t} = \mathsf{1bl}$ and (k, m_3, c_3) is the wished query at $\mathsf{1br}$. By construction, these tuples are required to satisfy $k = \mathsf{a}_1 \cdot (k, m, c) = \mathsf{a}_3 \cdot (k, m, c)$, $m = \mathsf{a}_1 \cdot (k, m, c, w)$, and $m_3 = \mathsf{a}_4 \cdot (k, m, c, w)$ for some w. Additionally the wish is required to satisfy $m + c = y$ and $m_3 + c_3 = z$.

As before, in this setting the adversary finds a preimage with probability at most $\frac{|\mathcal{W}_{\mathcal{S}_7}|}{2^{n-1}(2^{n-1}-1)}$.
We have 2^n choices for c_3. This uniquely fixes m_3. This uniquely fixes (k,m,c) by invertibility of $\mathsf{A}-\left[\left(\begin{smallmatrix}1&0\\0&1\end{smallmatrix}\right)/\left(\begin{smallmatrix}1&0\\0&0\end{smallmatrix}\right)\right]$. We find $|\mathcal{W}_{\mathcal{S}_7}|\leq 2^n$, and hence in this setting a preimage is found with probability at most $\frac{2^n}{2^{n-1}(2^{n-1}-1)}$;

$\mathsf{1bl}=\mathsf{1br}\neq\mathsf{1t}$. In this case, the wish list contains tuples of the form (k,m_1,c_1,m,c), where (k,m_1,c_1) is the wished query at $\mathsf{1t}$ and (k,m,c) is the wished query at $\mathsf{1bl}=\mathsf{1br}$. By construction, these tuples are required to satisfy $k=\mathsf{a}_1\cdot(k,m_1,c_1)=\mathsf{a}_3\cdot(k,m_1,c_1)$ and $m=\mathsf{a}_1\cdot(k,m_1,c_1,w)=\mathsf{a}_4\cdot(k,m_1,c_1,w)$ for some w. Additionally the wish is required to satisfy $m+c=y=z$. As before, in this setting the adversary finds a preimage with probability at most $\frac{|\mathcal{W}_{\mathcal{S}_7}|}{2^{n-1}(2^{n-1}-1)}$.
We have 2^n choices for c. This uniquely fixes m. This uniquely fixes (k,m_1,c_1) by invertibility of $\mathsf{A}-\left[\left(\begin{smallmatrix}1&0\\0&0\end{smallmatrix}\right)/\left(\begin{smallmatrix}1&0\\0&0\end{smallmatrix}\right)\right]$. We find $|\mathcal{W}_{\mathcal{S}_7}|\leq 2^n$, and hence in this setting a preimage is found with probability at most $\frac{2^n}{2^{n-1}(2^{n-1}-1)}$;

$\{\mathsf{1t},\mathsf{1bl},\mathsf{1br}\}$ all different. In this case, the wish list contains tuples of the form $(k,m_1,c_1,m_2,c_2,m_3,c_3)$, where (k,m_1,c_1) is the wished query at $\mathsf{1t}$, (k,m_2,c_2) is the wished query at $\mathsf{1bl}$, and (k,m_3,c_3) is the wished query at $\mathsf{1br}$. By construction, these tuples are required to satisfy $k=\mathsf{a}_1\cdot(k,m_1,c_1)=\mathsf{a}_3\cdot(k,m_1,c_1)$, $m_2=\mathsf{a}_1\cdot(k,m_1,c_1,w)$, and $m_3=\mathsf{a}_4\cdot(k,m_1,c_1,w)$ for some w. Additionally the wish is required to satisfy $m_2+c_2=y$ and $m_3+c_3=z$. As before, in this setting the adversary finds a preimage with probability at most $\frac{|\mathcal{W}_{\mathcal{S}_7}|}{2^{n-1}(2^{n-1}-1)(2^{n-1}-2)}$.
We have 2^n choices for both c_2 and c_3. These uniquely fix m_2 and m_3. Any such choice uniquely fixes (k,m_1,c_1) by invertibility of $\mathsf{A}-\left[\left(\begin{smallmatrix}1&0\\0&0\end{smallmatrix}\right)/\left(\begin{smallmatrix}1&0\\0&0\end{smallmatrix}\right)\right]$. We find $|\mathcal{W}_{\mathcal{S}_7}|\leq 2^n(2^n-1)$, and hence in this setting a preimage is found with probability at most $\frac{2^{2n}}{2^{n-1}(2^{n-1}-1)(2^{n-1}-2)}$.

The proof is now completed by adding and simplifying all bounds in accordance with (4.15):

$$\begin{aligned}\mathbf{Pr}\left(\mathsf{pre}(\mathcal{Q}_q)\wedge\neg\mathsf{aux}(\mathcal{Q}_q)\right)&\leq\frac{3\tau^2+3\tau+1}{2^{n-1}}+\frac{(3\tau+3)2^n}{2^{n-1}(2^{n-1}-1)}+\\&\qquad\frac{2^{2n}}{2^{n-1}(2^{n-1}-1)(2^{n-1}-2)}\\&\leq\frac{6\tau^2+18\tau+26}{2^n-2},\end{aligned}$$

where we use that $\frac{1}{2^{n-1}-2}\leq\frac{3}{2^n}$ for $n\geq 4$. □

Lemma 4.3.11. *Provided $\tau \leq q$, we have* $\mathbf{Pr}\left(\mathsf{aux}(\mathcal{Q}_q)\right) \leq 4 \cdot 2^n \left(\frac{4eq}{\tau 2^n}\right)^{\tau/2} + 4 \cdot 2q \left(\frac{8eq}{\tau 2^n}\right)^{\frac{\tau 2^n}{4q}}$.

Proof. Note that $\mathsf{aux}_2(\mathcal{Q}_q), \ldots, \mathsf{aux}_5(\mathcal{Q}_q)$ essentially equal $\mathsf{help}_3(\mathcal{Q}_q)$ of Section 3.3.1, and the proof and bound directly carries over. We include the proof for completeness.

It suffices to consider the events $\mathbf{Pr}\left(\mathsf{aux}_k(\mathcal{Q}_q)\right)$ ($k = 2, \ldots, 5$) separately. For the proof to go through we use $\mathsf{a}_{12}, \mathsf{a}_{13} \neq 0$ (for $\mathsf{aux}_2(\mathcal{Q}_q)$), $\mathsf{a}_{32}, \mathsf{a}_{33} \neq 0$ (for $\mathsf{aux}_3(\mathcal{Q}_q)$), and $\mathsf{a}_{12} \neq \mathsf{a}_{32}$ and $\mathsf{a}_{13} \neq \mathsf{a}_{33}$ (for $\mathsf{aux}_5(\mathcal{Q}_q)$). The cases are equivalent by symmetry, and we consider $\mathsf{aux}_2(\mathcal{Q}_q)$ only.

Let $z \in \mathbb{Z}_2^n$. Denote by $\mathcal{Q}_q^{(n)}$ the restriction of $\mathcal{Q}_q$ to normal queries, and by $\mathcal{Q}_q^{(s)}$ the restriction of $\mathcal{Q}_q$ to queries that belong to super queries. In order for $\mathcal{Q}_q$ to have more than τ solutions to $\mathsf{a}_1 \cdot (K_i, m_i, c_i) = z$, at least one of the following criteria needs to hold:

1. $\mathcal{Q}_q^{(n)}$ has more than $\tau/2$ solutions;
2. $\mathcal{Q}_q^{(s)}$ has more than $\tau/2$ solutions.

We consider these two scenarios separately. In case of normal queries, each query (K_i, m_i, c_i) is answered with a value generated at random from a set of size at least 2^{n-1}, and hence it satisfies $\mathsf{a}_1 \cdot (K_i, m_i, c_i) = z$ with probability at most $1/2^{n-1} = 2/2^n$. More than $\tau/2$ queries result in a solution with probability at most $\binom{q}{\tau/2}\left(\frac{2}{2^n}\right)^{\tau/2} \leq \left(\frac{4eq}{\tau 2^n}\right)^{\tau/2}$.

The analysis for super queries is more elaborate. In order for $\mathcal{Q}_q^{(s)}$ to have more than $\tau/2$ solutions, as at most $q/2^{n-1}$ super queries occur, at least one of the super queries needs to provide more than $\tau' := \frac{\tau}{2q/2^{n-1}} = \frac{\tau 2^n}{4q}$ solutions. Consider any super query, consisting of 2^{n-1} queries. It provides more than τ' solutions with probability at most

$$\binom{2^{n-1}}{\tau'} \prod_{j=0}^{\tau'-1} \frac{1}{2^{n-1} - j} \leq \binom{2^{n-1}}{\tau'} \left(\frac{1}{2^{n-1} - \tau'}\right)^{\tau'} \leq \left(\frac{e 2^{n-1}}{\tau'(2^{n-1} - \tau')}\right)^{\tau'}.$$

Provided $\tau \leq q$, we have $\tau' = \frac{\tau 2^n}{4q} \leq 2^{n-2}$, and thus $\frac{1}{2^{n-1} - \tau'} \leq \frac{1}{2^{n-2}}$. Consequently, this super query adds more than $\frac{\tau 2^n}{4q}$ solutions with probability at most $\left(\frac{8eq}{\tau 2^n}\right)^{\frac{\tau 2^n}{4q}}$. In order to cover any super query, we need to multiply this probability with $q/2^{n-1}$.

Considering any possibly choice for z, we obtain for $k = 2, \ldots, 5$:

$$\mathbf{Pr}\left(\mathsf{aux}_k(\mathcal{Q}_q)\right) \leq 2^n \left(\frac{4eq}{\tau 2^n}\right)^{\tau/2} + 2^n \cdot \frac{q}{2^{n-1}} \left(\frac{8eq}{\tau 2^n}\right)^{\frac{\tau 2^n}{4q}}.$$

The claim is obtained by multiplying this equation with 4. □

From (4.13-4.14) and Lemmas 4.3.10-4.3.11 we conclude for $\mathbf{Adv}_{\mathsf{F}^3_{\mathsf{A}}}^{\mathrm{epre}}(q)$:

$$\mathbf{Adv}_{\mathsf{F}^3_{\mathsf{A}}}^{\mathrm{epre}}(q) \leq \frac{6\tau^2 + 18\tau + 26}{2^n - 2} + 4 \cdot 2^n \left(\frac{4eq}{\tau 2^n}\right)^{\tau/2} + 8q \left(\frac{8eq}{\tau 2^n}\right)^{\frac{\tau 2^n}{4q}} .$$

This completes the proof of Theorem 4.3.8.

4.4 Conclusions

In the area of double block length hashing, where a $3n$-to-$2n$-bit compression function is constructed from n-bit block ciphers, all optimally secure constructions known in the literature employ a block cipher with $2n$-bit key space. We have reconsidered the principle of double length hashing, focusing on double length hashing from a block cipher with *n-bit message and key space*. Unlike in the DBL^{2n} class, we demonstrate that there does not exist any optimally secure design with reasonably simple finalization function that makes two cipher calls. By allowing one extra call, optimal collision resistance can nevertheless be achieved, as we have proven by introducing our family of designs $\mathsf{F}^3_{\mathsf{A}}$.

In our quest for optimal collision secure compression function designs, we had to resort to designs with three block cipher calls rather than two, which moreover are not parallelizable. This entails an efficiency loss compared to MDC-2, MJH, and Jetchev et al.'s construction. On the other hand, our family of functions is based on simple arithmetic in the finite field: unlike constructions by Stam [209, 210], Lee and Steinberger [148], and Jetchev et al. [118], our design does not make use of full field multiplications. The example matrices A given in (4.11) are designed to use a minimal amount of non-zero elements. We note that specific choices of A may be more suited for this construction to be used in an iterated design.

This work provides new insights in double length hashing, but also results in interesting research questions. Most importantly, is it possible to construct other collision secure F^3 constructions (beyond our family of functions $\mathsf{F}^3_{\mathsf{A}}$), that achieve optimal $2^{5n/3}$ preimage resistance? Jetchev et al.'s construction is a good starting point for this question. Given the negative collision resistance result for a wide class of compression functions F^2, is it possible to achieve optimal collision security *in the iteration* anyhow? This question is beyond the scope of this work. On the other hand, is it possible to achieve an impossibility result for F^3 restricted to the XOR-only design (where $f_1, \dots, f_4$ only XOR their parameters)? A similar question is asked in Chapter 5 in the context of permutation based hashing.

5 Hash Functions Based on Three Permutations

In this chapter, we focus on $2n$-to-n-bit compression functions based on three permutations. We refer to Section 2.6.2 for a detailed discussion of the state of the art. In [198], Rogaway and Steinberger formally proved a wide class of $2n$-to-n-bit compression functions using three distinct permutations and finite field scalar multiplications optimally collision and preimage secure, provided the compression function satisfies a so-called "independence criterion." Unfortunately, this technical criterion rules out the most intuitive and elegant type of designs, namely compression functions that are (apart from the three permutations) solely based on XOR operators. In [206], Shrimpton and Stam derived a XOR based compression function, using three one-way functions rather than permutations: $\mathsf{F}(x_1, x_2) = f_1(x_1) \oplus f_3(f_1(x_1) \oplus f_2(x_2))$. This function is proven collision resistant up to $2^{n/2}$ queries (asymptotically), but preimages can be found with high probability after $2^{n/2}$ queries [206]. It has been demonstrated by an automated analysis of Rogaway and Steinberger [198] that the same results hold if f_1, f_2, f_3 are Davies-Meyer-like compression functions using permutations π_1, π_2, π_3, i.e., $f_i(x) = x \oplus \pi_i(x)$, but a formal security analysis has never been given. Since these works, a synthetic formal collision and preimage security analysis of XOR based compression functions has remained an interesting and important theoretical open problem, because of their elegance and simplicity (the functions only employ XOR operators) as well as their slight efficiency improvement (XOR operators are slightly cheaper than finite field multiplications).

Contributions of This Chapter

In this chapter, we focus on the entire family of $2n$-to-n-bit compression functions constructed only of three isolated permutations and of XOR operators, and analyze the security of these functions against information-theoretic adversaries. For each of the functions, we either provide a proof of optimal collision resistance or a collision attack faster than the birthday bound. We also analyze the preimage resistance of the schemes that have optimal collision security.

The approach followed in this dissertation is based on defining an equivalence class on the set of compression functions, and is of independent interest: informally, two compression functions are equivalent if there exists a tight bi-directional collision and preimage security reduction (cf. Definition 5.2.1). Consequently, security results of one compression function hold for the entire class, and it suffices to analyze the security of one function per class. In this dissertation we restrict to equivalence reductions that are easy to verify, such as interchanging the inputs to the compression function.

For the *multi-permutation* setting, where the three permutations π_1, π_2, π_3 are assumed to be selected independently and uniformly at random, the results are as follows. A compression function F is optimally collision secure (asymptotically) *if and only if* it is equivalent to one of the four compression functions $\mathsf{F}_1, \ldots, \mathsf{F}_4$:

$$\begin{aligned}
\mathsf{F}_1(x_1, x_2) &= x_2 \oplus \pi_2(x_2) \oplus \pi_3(x_1 \oplus x_2 \oplus \pi_1(x_1))\,,\\
\mathsf{F}_2(x_1, x_2) &= x_1 \oplus \pi_1(x_1) \oplus \pi_2(x_2) \oplus \pi_3(x_1 \oplus x_2 \oplus \pi_1(x_1))\,,\\
\mathsf{F}_3(x_1, x_2) &= x_1 \oplus \pi_1(x_1) \oplus \pi_3(x_1 \oplus x_2 \oplus \pi_1(x_1) \oplus \pi_2(x_2))\,,\\
\mathsf{F}_4(x_1, x_2) &= x_1 \oplus x_2 \oplus \pi_1(x_1) \oplus \pi_3(x_1 \oplus x_2 \oplus \pi_1(x_1) \oplus \pi_2(x_2))\,.
\end{aligned} \tag{5.1}$$

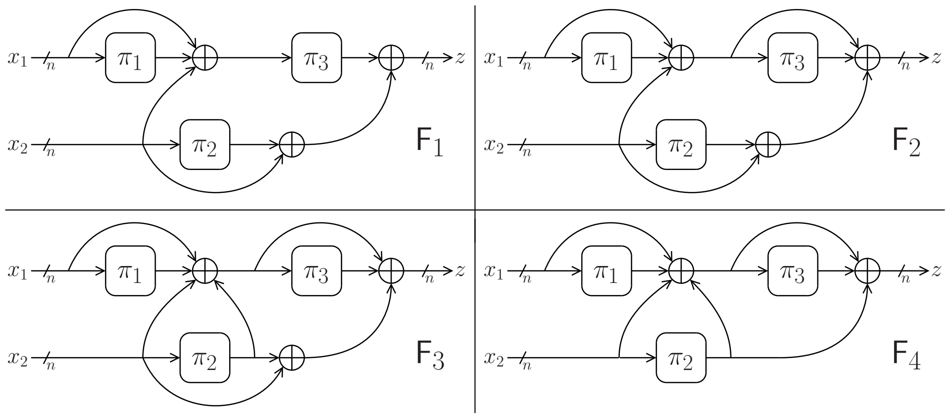


Figure 5.1. Compression functions $\mathsf{F}_1, \ldots, \mathsf{F}_4$ of (5.1).

These compression functions are depicted in Figure 5.1. Not surprisingly, the permutation based variant of the Shrimpton-Stam compression function [206] is included, it equals F_3. For compression functions non-equivalent to any of $\mathsf{F}_1, \mathsf{F}_2, \mathsf{F}_3, \mathsf{F}_4$, collisions can be found faster than the birthday bound, namely in at most $2^{2n/5}$ queries. Compression functions equivalent to F_2 are proven optimally preimage secure up to $2^{2n/3}$ queries, and compression functions equivalent to $\mathsf{F}_1, \mathsf{F}_3$, or F_4 are additionally shown to achieve tight $2^{n/2}$ preimage security. Therefore, a compression function achieves optimal collision and preimage resistance (with respect to the bounds of [199]) if and only if it is equivalent to F_2. Particularly, this class of functions beats the Shrimpton-Stam

Table 5.1. Security results for the multi-permutation setting derived in this chapter. The functions $\mathsf{F}_1, \ldots, \mathsf{F}_4$ are given in (5.1) and Figure 5.1. The equivalence relation is defined in Definition 5.2.1. For F_2, the obtained security results are optimal with respect to the bounds of Rogaway and Steinberger [199]. The proofs of the results with appended "[c]" fall back on Conjecture 5.3.2.

	collision		preimage	
F equivalent to:	security	attack	security	attack
$\mathsf{F}_1, \mathsf{F}_4$	$2^{n/2}$ [c]	$2^{n/2}$	$2^{n/2}$	$2^{n/2}$
F_2	$2^{n/2}$ [c]	$2^{n/2}$	$2^{2n/3}$ [c]	$2^{2n/3}$
F_3	$2^{n/2}$	$2^{n/2}$	$2^{n/2}$	$2^{n/2}$
none of these	?	$2^{2n/5}$	?	?

compression function [206] with respect to preimage resistance. These results are summarized in Table 5.1.

A minor part of the results in the multi-permutation setting, more concretely the collision resistance of $\mathsf{F}_1, \mathsf{F}_2$, and F_4, and the preimage resistance of F_2, are based on an extremal graph theory based conjecture. Informally, this conjecture bounds the number of solutions $(x_1, x_2, x_3) \in X_1 \times X_2 \times X_3$ such that $x_2 \oplus x_3 = x_1 \oplus \pi_1(x_1)$, where X_1, X_2, X_3 are three sets of q elements. This conjecture is similar to (but more complex than) a problem posed by Zarankiewicz in 1951 (cf. [57, Ch. 6.2]), and is of independent interest. In Appendix B.4, we analyze our conjecture in more detail, provide it with a heuristic argument, and compare it with the conjecture of Zarankiewicz.

In the *single-permutation* setting, where the compression function makes three calls to the same random permutation π, *there does not exist any* compression function in the given model that achieves optimal collision resistance. In particular, for any possible function, collisions can be found in at most $2^{2n/5}$ queries, beating the desired birthday bound. This negative result is surprising, given the fair amount of secure functions we have found in the multi-permutation setting. The attacks mainly rely on the fact that the adversary can misuse the single-permutation property by introducing dependencies between the two input values x_1 and x_2. For instance, the function F_2 of (5.1) satisfies $\mathsf{F}_2(x_1, x_2) = \mathsf{F}_2(x_1, x_2 \oplus x_1 \oplus \pi(x_1))$ in the single-permutation setting. This result raises the interesting question whether (larger) compression functions exist based only on XOR operators and (more than three invocations of) one single permutation.

Outline

In Section 5.1, we present some mathematical preliminaries complementary to Chapter 2, and formally describe the set of permutation based compression functions we analyze. In Section 5.2, the equivalence relation on the set of compression functions is formally defined. The main results are given in Section 5.3 for the multi-permutation setting and in Section 5.4 for the single-permutation setting. The chapter is concluded in Section 5.5.

Bibliographic Notes

An extended abstract of this chapter has been published by Mennink and Preneel at CRYPTO 2012 [165].

5.1 Security Model

By log we denote the logarithm function with respect to base 2. Vectors are denoted as a, and by $\|\mathsf{a}\| = \sum_i |a_i|$ we denote the 1-norm of a. For a matrix A, by $a_{i,j}$ we denote its coefficient at the i^{th} row and j^{th} column. By $\mathsf{a}_{i,*}$ we denote the i^{th} row of A, and by $\mathsf{a}_{*,j}$ its j^{th} column.

5.1.1 Permutation Based Compression Functions

We consider the following type of $2n$-to-n-bit compression functions. Let $\pi_1, \pi_2, \pi_3 \in \text{Perm}(n)$ be three permutations. For a binary 4×5 matrix A of the form

$$\mathsf{A} = \left(\begin{array}{cc|ccc} a_{11} & a_{12} & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 0 \\ a_{31} & a_{32} & a_{33} & a_{34} & 0 \\ \hline a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \end{array}\right), \tag{5.2}$$

the compression function $\mathsf{F}_\mathsf{A} : \mathbb{Z}_2^{2n} \to \mathbb{Z}_2^n$ is defined as follows.

$$\begin{aligned} &\mathsf{F}_\mathsf{A}(x_1, x_2) \\ &\quad y_1 \leftarrow \pi_1(a_{11}x_1 \oplus a_{12}x_2)\,, \\ &\quad y_2 \leftarrow \pi_2(a_{21}x_1 \oplus a_{22}x_2 \oplus a_{23}y_1)\,, \\ &\quad y_3 \leftarrow \pi_3(a_{31}x_1 \oplus a_{32}x_2 \oplus a_{33}y_1 \oplus a_{34}y_2)\,, \\ &\quad z \leftarrow a_{41}x_1 \oplus a_{42}x_2 \oplus a_{43}y_1 \oplus a_{44}y_2 \oplus a_{45}y_3\,, \\ &\quad \textbf{return } z\,. \end{aligned} \tag{5.3}$$

The function F_A is depicted in Figure 5.2. If the three permutations are all different, we refer to it as the *multi-permutation* setting. If π_1, π_2, π_3 are equal to one permutation π, we are in the *single-permutation* setting. In total, we thus analyze $2 \cdot 2^{14}$ compression functions. Many of these, however, are trivially weak (cf. Section 5.1.3).

For the single-permutation setting, it is of interest to also consider the case where n-bit constants are added to the inputs to the permutations (e.g., $y_1 \leftarrow \pi_1(a_{11}x_1 \oplus a_{12}x_2 \oplus b_1)$ for $b_1 \in \mathbb{Z}_2^n$). This results in many more schemes, but requires a more complex analysis. Therefore, we present our main results on F_A of (5.3), and in Appendix B.3 we generalize our findings on the single-permutation setting to cover any F_A where additional affine transformations on the permutation inputs are taken into account.

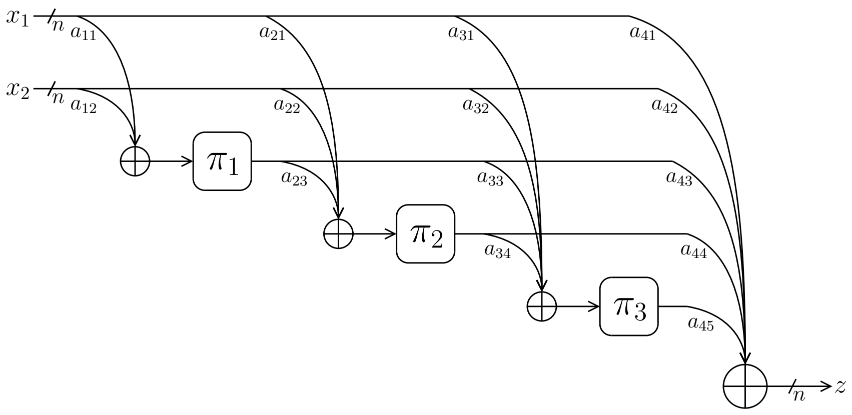


Figure 5.2. The permutation based compression function F_A of (5.3).

5.1.2 Security Notions

Throughout this chapter, we consider $\mathcal{P} = (\pi_1, \pi_2, \pi_3) \xleftarrow{\$} \mathrm{Perm}(n)^3$, unless explicitly stated otherwise. We consider an adversary which attacks F_A and whose goal it is to find either a collision or a preimage. The adversary can make forward and inverse queries to its oracles, and these are stored in query history $\mathcal{Q}$ as indexed tuples of the form (x_k, y_k), where for $k \in \{1, 2, 3\}$, we denote that $y_k = \pi_k(x_k)$.

For the single-permutation setting (Section 5.4), we consider similar definitions where $\mathcal{P} = \pi \xleftarrow{\$} \mathrm{Perm}(n)$.

5.1.3 Invalid Matrices

We will classify the set of optimally collision secure compression functions F_A of the form described in Section 5.1.1, but for some matrices A the induced compression function will clearly not fulfill the desired security requirements. For instance, if a compression function does not use one or more permutations, attacks faster than the birthday bound can easily be constructed. We introduce the notion of "valid" matrices, in order to rule out compression functions that trivially fail to achieve optimal collision resistance. A matrix A is called valid if it satisfies the following properties:

(i) For the j^{th} column ($j = 1, 2$), we have $a_{1j} + a_{2j} + a_{3j} \geq 1$. This requirement ensures that input x_j is used in the computation of at least one permutation. If this would not be the case, collisions can easily be constructed;

(ii) For the j^{th} column ($j = 3, 4, 5$), we have $\|\mathsf{a}_{*,j}\| \geq 1$, and for the i^{th} row ($i = 1, 2, 3$), we have $\|\mathsf{a}_{i,*}\| \geq 1$. Notice that if the i^{th} row (resp. j^{th} column) would consist of zeroes only, it means that permutation π_i (resp. π_{j-2}) is not used in the computation, and collisions can be found in at most $2^{n/3}$ queries by Stam's bound (see Section 2.6.2).

In the remainder, we will consider valid matrices A only. By an extensive computation one can show that $2796 < 2^{12}$ out of 2^{14} matrices are valid (for both the single- and multi-permutation setting).

5.2 Equivalence Classes of Permutation Based Compression Functions

We define an equivalence relation on the set of compression functions F_A. This equivalence relation intuitively describes classes of "equally secure" compression functions, and can be used to reduce the number of compression functions to be analyzed. Indeed, security properties of one compression function naturally convey to all compression functions in the same equivalence class. The equivalence relation is defined in Definition 5.2.1, and in Propositions 5.2.2-5.2.5 we describe the four equivalence reductions that will be used in this chapter.

Definition 5.2.1. Two compression functions F_A and $\mathsf{F}_{\mathsf{A}'}$ are *equivalent* if for both collision and preimage security there exists a tight reduction from F_A to $\mathsf{F}_{\mathsf{A}'}$, and vice versa.

Proposition 5.2.2 (x-reduction). *Consider two matrices* $\mathsf{A} = \big(\mathsf{a}_{*,1}\,;\,\mathsf{a}_{*,2}\,;\,\mathsf{a}_{*,3}\,;\,\mathsf{a}_{*,4}\,;\,\mathsf{a}_{*,5}\big)$ *and* $\mathsf{A}' = \big(\mathsf{a}_{*,2}\,;\,\mathsf{a}_{*,1}\,;\,\mathsf{a}_{*,3}\,;\,\mathsf{a}_{*,4}\,;\,\mathsf{a}_{*,5}\big)$. *Then, the compression functions* F_A *and* $\mathsf{F}_{\mathsf{A}'}$ *are equivalent. Intuitively, this reduction corresponds to swapping* x_1 *and* x_2.

Proposition 5.2.3 (XOR-reduction). *Consider a matrix* $\mathsf{A} = \big(\mathsf{a}_{*,1}\,;\,\mathsf{a}_{*,2}\,;\,\mathsf{a}_{*,3}\,;\,\mathsf{a}_{*,4}\,;\,\mathsf{a}_{*,5}\big)$, *and let* $k = \min\{\ i \mid a_{i,2} \neq 0\ \}$ *(notice that* $k \in \{1, 2, 3\}$ *as* A *is valid). Let* $c_0, \ldots, c_2 \in \{0, 1\}$. *Consider the matrix* $\mathsf{A}' = \mathsf{A} \oplus \big(c_0\mathsf{a}_{*,2}\,;\,0\,;\,[k \geq 2]c_1\mathsf{a}_{*,2}\,;\,[k \geq 3]c_2\mathsf{a}_{*,2}\,;\,0\big)$, *where* $[X] = 1$ *if* X *holds and* 0 *otherwise. Then, the compression functions* F_A *and* $\mathsf{F}_{\mathsf{A}'}$ *are equivalent. Intuitively,* π_k *is the first permutation that incorporates* x_2, *and this reduction represents replacing* x_2 *by* $x_2 \oplus c_0 x_1 \oplus \sum_{i=1}^{k-1} c_i y_i$, *where y_i is the outcome of the i^{th} permutation. Using Proposition 5.2.2, the same reduction holds for x_1.*

Proposition 5.2.4 (π-swap-reduction)**.** *Let $i \in \{1, 2\}$, and consider a matrix A with $a_{i+1,i+2} = 0$. Consider the matrix A' obtained from A by swapping rows $\mathsf{a}_{i,*}$ and $\mathsf{a}_{i+1,*}$ and consequently swapping columns $\mathsf{a}_{*,i+2}$ and $\mathsf{a}_{*,i+3}$. Then, the compression functions F_A and $\mathsf{F}_{\mathsf{A}'}$ are equivalent. Intuitively, this reduction corresponds to swapping π_i and π_{i+1}, which is only possible if the outcome of π_i is not used as input of π_{i+1} (i.e., if $a_{i+1,i+2} = 0$).*

Proposition 5.2.5 (π-inverse-reduction)**.** *Consider a matrix A with $(a_{11}, a_{12}) = (1, 0)$. Consider the matrix A' obtained from A by swapping (a_{21}, a_{31}, a_{41}) and (a_{23}, a_{33}, a_{43}). Then, the compression functions F_A and $\mathsf{F}_{\mathsf{A}'}$ are equivalent. Intuitively, this reduction corresponds to replacing π_1 by π_1^{-1}. Using Proposition 5.2.2 and Proposition 5.2.4 on $i = 1$, the same reduction holds for π_2.*

Proof of Propositions 5.2.2-5.2.5. Let F_A and $\mathsf{F}_{\mathsf{A}'}$ be two compression functions defined as in either of the propositions. For simplicity, in case of Proposition 5.2.3 we only consider $k = 2$ (so $a_{12} = 0$, $a_{22} = 1$ and $c_2 = 0$), for Proposition 5.2.4 we only consider $i = 1$ (so $a_{23} = 0$). By construction, the compression functions F_A and $\mathsf{F}_{\mathsf{A}'}$ satisfy the following properties:

$$\mathsf{F}_\mathsf{A}^{\pi_1,\pi_2,\pi_3}(x_1, x_2) = \begin{cases} \mathsf{F}_{\mathsf{A}'}^{\pi_1,\pi_2,\pi_3}(x_2, x_1) & \text{for Proposition 5.2.2,} \\ \mathsf{F}_{\mathsf{A}'}^{\pi_1,\pi_2,\pi_3}(x_1, x_2 \oplus c_0 x_1 \oplus c_1 \pi_1(a_{11}x_1)) & \text{for Proposition 5.2.3,} \\ \mathsf{F}_{\mathsf{A}'}^{\pi_2,\pi_1,\pi_3}(x_1, x_2) & \text{for Proposition 5.2.4,} \\ \mathsf{F}_{\mathsf{A}'}^{\pi_1^{-1},\pi_2,\pi_3}(\pi_1(x_1), x_2) & \text{for Proposition 5.2.5.} \end{cases} \tag{5.4}$$

We need to provide a bi-directional collision and preimage security reduction. For conciseness, we will provide only the collision security reduction; the case of preimage resistance is similar and is therefore omitted. Let $\mathcal{A}$ be a collision-finding adversary for the compression function F_A, that on input of $\pi_1, \pi_2, \pi_3 \xleftarrow{\$} \mathrm{Perm}(n)$, outputs two tuples (x_1, x_2), (x_1', x_2') such that $\mathsf{F}_\mathsf{A}^{\pi_i}(x_1, x_2) = \mathsf{F}_\mathsf{A}^{\pi_i}(x_1', x_2')$. We construct a collision-finding adversary $\mathcal{A}'$ for $\mathsf{F}_{\mathsf{A}'}$ that uses $\mathcal{A}$ as a subroutine and on input of $\pi_1', \pi_2', \pi_3' \xleftarrow{\$} \mathrm{Perm}(n)$ outputs a collision for $\mathsf{F}_{\mathsf{A}'}^{\pi_i'}$. Adversary $\mathcal{A}'$ operates as follows.

(i) In Propositions 5.2.2 and 5.2.3, the adversary $\mathcal{A}'$ sends $(\pi_1, \pi_2, \pi_3) \leftarrow (\pi_1', \pi_2', \pi_3')$ to $\mathcal{A}$. In Proposition 5.2.4, $\mathcal{A}'$ sends $(\pi_1, \pi_2, \pi_3) \leftarrow (\pi_2', \pi_1', \pi_3')$ to $\mathcal{A}$. In Proposition 5.2.5, $\mathcal{A}'$ sends $(\pi_1, \pi_2, \pi_3) \leftarrow ((\pi_1')^{-1}, \pi_2', \pi_3')$ to $\mathcal{A}$;

(ii) $\mathcal{A}$ outputs two tuples (x_1, x_2), (x_1', x_2') such that $\mathsf{F}_\mathsf{A}^{\pi_i}(x_1, x_2) = \mathsf{F}_\mathsf{A}^{\pi_i}(x_1', x_2')$;

(iii) In Proposition 5.2.2, $\mathcal{A}'$ outputs collision (x_2, x_1) and (x_2', x_1'). In Proposition 5.2.3, $\mathcal{A}'$ outputs $(x_1, x_2 \oplus c_0 x_1 \oplus c_1 \pi_1(a_{11}x_1))$ and $(x_1', x_2' \oplus c_0 x_1' \oplus c_1 \pi_1(a_{11}x_1'))$. In Proposition 5.2.4, $\mathcal{A}'$ outputs (x_1, x_2) and (x_1', x_2'). In Proposition 5.2.5, $\mathcal{A}'$ outputs $((\pi_1')^{-1}(x_1), x_2)$ and $((\pi_1')^{-1}(x_1'), x_2')$.

Notice that in step (i), the permutations (π_1, π_2, π_3) are clearly randomly and independently distributed as (π_1', π_2', π_3') are, and therefore $\mathcal{A}$ can output (x_1, x_2), (x_1', x_2') such that $\mathsf{F}_{\mathsf{A}}^{\pi_1,\pi_2,\pi_3}(x_1, x_2) = \mathsf{F}_{\mathsf{A}}^{\pi_1,\pi_2,\pi_3}(x_1', x_2')$ with probability $\mathbf{Adv}_{\mathsf{F}_\mathsf{A}}^{\mathrm{col}}(\mathcal{A})$. For $\mathsf{F}_{\mathsf{A}'}$ of Proposition 5.2.4, these tuples indeed render a collision as given in step (iii):

$$\begin{aligned}\mathsf{F}_{\mathsf{A}'}^{\pi_1',\pi_2',\pi_3'}(x_1, x_2) &= \mathsf{F}_{\mathsf{A}}^{\pi_2',\pi_1',\pi_3'}(x_1, x_2), \text{ by (5.4)},\\ &= \mathsf{F}_{\mathsf{A}}^{\pi_2',\pi_1',\pi_3'}(x_1', x_2'), \text{ by collision for } \mathsf{F}_\mathsf{A},\\ &= \mathsf{F}_{\mathsf{A}'}^{\pi_1',\pi_2',\pi_3'}(x_1', x_2'), \text{ by (5.4)}.\end{aligned}$$

The same argument applies to the other propositions. In any case, $\mathcal{A}'$ needs at most four queries more than $\mathcal{A}$, and thus we obtain $\mathbf{Adv}_{\mathsf{F}_\mathsf{A}}^{\mathrm{col}}(q) \leq \mathbf{Adv}_{\mathsf{F}_{\mathsf{A}'}}^{\mathrm{col}}(q + 4)$. The reductions in the other direction (from $\mathsf{F}_{\mathsf{A}'}$ to F_A) are identical due to symmetry. □

Except for Proposition 5.2.5, the reductions also hold in the single-permutation setting. We remark that these reductions are not only restricted to binary matrices, but apply to general matrices A. In particular, the independence criterion of [198] can be derived using the given reductions (by generalizing the upcoming Lemma 5.3.3). Also, we note that the reductions can easily be represented by linear matrix operations.

5.3 Main Result for Multi-Permutation Setting

We classify the set of permutation based compression functions of the form (5.3) that achieve optimal collision resistance. Theorem 5.3.1 shows that the set of (asymptotically) secure functions is fully covered by four equivalence classes; for any other compression function collisions can be found faster than the birthday bound. One of these four classes — defined by $\mathsf{F}_{\mathsf{A}_2}$ below — provides optimal (asymptotic) $2^{2n/3}$ preimage security, for the other three classes preimages can be found significantly faster.

Theorem 5.3.1. *Consider the multi-permutation setting. Let* F_A *be any compression function defined by a binary matrix* A *of the form (5.2). Let* $\mathsf{F}_{\mathsf{A}_k}$ *for* $k = 1, 2, 3, 4$ *be the compression functions defined by matrices*

$$\mathsf{A}_1 = \left(\begin{array}{cc|ccc} 1 & 0 & 0 & 0 & 0\\ 0 & 1 & 0 & 0 & 0\\ 1 & 1 & 1 & 0 & 0\\ \hline 0 & 1 & 0 & 1 & 1 \end{array}\right), \quad \mathsf{A}_2 = \left(\begin{array}{cc|ccc} 1 & 0 & 0 & 0 & 0\\ 0 & 1 & 0 & 0 & 0\\ 1 & 1 & 1 & 0 & 0\\ \hline 1 & 0 & 1 & 1 & 1 \end{array}\right),$$
$$\mathsf{A}_3 = \left(\begin{array}{cc|ccc} 1 & 0 & 0 & 0 & 0\\ 0 & 1 & 0 & 0 & 0\\ 1 & 1 & 1 & 1 & 0\\ \hline 1 & 0 & 1 & 0 & 1 \end{array}\right), \quad \mathsf{A}_4 = \left(\begin{array}{cc|ccc} 1 & 0 & 0 & 0 & 0\\ 0 & 1 & 0 & 0 & 0\\ 1 & 1 & 1 & 1 & 0\\ \hline 1 & 1 & 1 & 0 & 1 \end{array}\right). \tag{5.5}$$

Let $\varepsilon > 0$.

(i) If $\mathsf{F_A}$ is equivalent to $\mathsf{F_{A_k}}$ for $k \in \{1,2,3,4\}$, it satisfies $\lim_{n\to\infty} \mathbf{Adv}^{\mathrm{col}}_{\mathsf{F_A}}(2^{n/2(1-\varepsilon)}) = 0$. Otherwise, it satisfies $\mathbf{Adv}^{\mathrm{col}}_{\mathsf{F_A}}(q) = \Omega(q^5/2^{2n})$;

(ii) If $\mathsf{F_A}$ is equivalent to $\mathsf{F_{A_2}}$, it satisfies $\lim_{n\to\infty} \mathbf{Adv}^{\mathrm{epre}}_{\mathsf{F_A}}(2^{2n/3(1-\varepsilon)}) = 0$;

(iii) If $\mathsf{F_A}$ is equivalent to $\mathsf{F_{A_k}}$ for $k \in \{1,3,4\}$, it satisfies $\mathbf{Adv}^{\mathrm{epre}}_{\mathsf{F_A}}(q) = \Theta(q^2/2^n)$.

In other words, a compression function offers optimal collision resistance *if and only if* it is equivalent to either of $\mathsf{F_{A_1}}, \mathsf{F_{A_2}}, \mathsf{F_{A_3}}, \mathsf{F_{A_4}}$, and additionally achieves optimal preimage resistance (with respect to the bounds of [199]) *if and only if* it is equivalent to $\mathsf{F_{A_2}}$.

In order to prove Theorem 5.3.1, more specifically part (i) for $k = 1, 2, 4$ and part (ii), we pose the following conjecture. This conjecture relates to the area of extremal graph theory and is of independent interest. In particular, it can be shown to be similar to (but more complex than) a longstanding problem of Zarankiewicz from 1951 [57, Ch. 6.2].

Conjecture 5.3.2. Let $q \le 2^n$, and let Z be a set of q elements taken uniformly at random from $\mathbb{Z}_2^n$. Let β denote the maximum number of tuples $(x_1, x_2, z) \in X_1 \times X_2 \times Z$ such that $x_1 \oplus x_2 = z$, where X_1, X_2 are any two subsets of $\mathbb{Z}_2^n$ of size q. Formally:

$$\beta := \max_{\substack{X_1, X_2 \subseteq \mathbb{Z}_2^n \\ |X_1| = |X_2| = q}} \left|\{(x_1, x_2, z) \in X_1 \times X_2 \times Z \mid x_1 \oplus x_2 = z\}\right|.$$

There exists a constant d_1 such that $\mathbf{Pr}(\beta > d_1 q \log q) \to 0$ for $n \to \infty$ and $q < 2^{n/2}$. Similarly, there exists a constant d_2 such that $\mathbf{Pr}\left(\beta > d_2 q^{3/2}\right) \to 0$ for $n \to \infty$ and $q < 2^{2n/3}$.

The first bound is used in the proof Theorem 5.3.1(i) for $k = 1, 2, 4$, and the second bound in the proof Theorem 5.3.1(ii). A detailed heuristic for Conjecture 5.3.2 is given in Appendix B.4, together with a comparison with Zarankiewicz's conjecture, but we leave a full proof of Conjecture 5.3.2 as an open problem.

5.3.1 Proof of Theorem 5.3.1

The proof of Theorem 5.3.1 is structured as follows. Firstly, in Lemma 5.3.3 we show that any compression function $\mathsf{F_A}$ can be reduced either to an invalid compression function or to a compression function $\mathsf{F_{A'}}$ defined by a matrix $\mathsf{A'}$ with first two rows $10000, 01000$. By construction (see Section 5.2), the security properties of one compression function are valid for the whole equivalence class. Secondly, in Lemma 5.3.4 several collision attacks are described that invalidate the security of each of the remaining compression functions, except for the classes defined by $\mathsf{F_{A_k}}$ ($k \in \{1,2,3,4\}$) for A_k as in (5.5). Thirdly, the collision and preimage resistance of the remaining four compression functions are analyzed in Lemma 5.3.5, which completes the proof of Theorem 5.3.1.

Lemma 5.3.3. *Any compression function* F_A*, for valid* A*, is equivalent to a compression function* $\mathsf{F}_{\mathsf{A}'}$*, where either* A' *is invalid or the first two rows of* A' *equal* $10000, 01000$.

Proof. The proof is constructive. Several reductions are used, but for ease of notation apostrophes are omitted. Let F_A be a compression function defined by some valid matrix A. As A is valid, we have $a_{11} + a_{12} \geq 1$. If $a_{11} + a_{12} = 2$, we can apply Proposition 5.2.3 on $c_0 = 1$ to obtain $a_{11} + a_{12} = 1$. Now, by Proposition 5.2.2 we can assume that $(a_{11}, a_{12}) = (1, 0)$.

Considering the second row of A, we distinguish between $a_{22} = 1$ and $a_{22} = 0$. In the former case, a XOR-reduction (Proposition 5.2.3) on $(c_0, c_1) = (a_{21}, a_{23})$ reduces the scheme to the required form. In the latter case, where $a_{22} = 0$, we proceed as follows. If $a_{32} = 0$, A is equivalent to an invalid matrix. Otherwise, by applying Proposition 5.2.3 with $(c_0, c_1, c_2) = (a_{31}, a_{33}, a_{34})$ we obtain that F_A is equivalent to a compression function $\mathsf{F}_{\mathsf{A}'}$, for some matrix A' with rows $(10000, a'_{21}0a'_{23}00, 01000, a'_{41}a'_{42}a'_{43}a'_{44}a'_{45})$. The result is now obtained by swapping π_2 and π_3 (Proposition 5.2.4 for $i = 2$). $\square$

As a direct consequence of Lemma 5.3.3, it suffices to consider compression functions F_A, where

$$\mathsf{A} = \left(\begin{array}{cc|ccc} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ a_{31} & a_{32} & a_{33} & a_{34} & 0 \\ \hline a_{41} & a_{42} & a_{43} & a_{44} & 1 \end{array}\right) \tag{5.6}$$

for some binary values $a_{31}, \ldots, a_{44}$. Notice that $a_{45} = 1$ because of the validity of the matrix. We describe a couple of collision attacks that apply to compression functions of this form. We note that similar results also hold for preimage resistance.

Lemma 5.3.4. *Let* F_A *be a compression function defined by a valid matrix* A *of the form (5.6).*

(i) If A *satisfies* $(a_{31} + a_{33})(a_{32} + a_{34}) = 0$*, then* $\mathbf{Adv}^{\mathrm{col}}_{\mathsf{F}_\mathsf{A}}(q) = \Omega(q^4/2^n)$*;*

(ii) If A *satisfies* $\bigvee_{j=1}^{4} a_{3j} = a_{4j} = 0$*, then* $\mathbf{Adv}^{\mathrm{col}}_{\mathsf{F}_\mathsf{A}}(q) = \Omega(q^3/2^n)$*;*

(iii) If A *satisfies* $\bigwedge_{j=1}^{2} a_{3j}a_{4,j+2} \neq a_{3,j+2}a_{4j}$*, then* $\mathbf{Adv}^{\mathrm{col}}_{\mathsf{F}_\mathsf{A}}(q) = \Omega(q^3/2^n)$*;*

(iv) If A *satisfies* $a_{41} + a_{42} + a_{43} + a_{44} = 1$*, then* $\mathbf{Adv}^{\mathrm{col}}_{\mathsf{F}_\mathsf{A}}(q) = \Omega(q^5/2^{2n})$*.*

For clarity, the proofs of results (i), (ii), (iii), and (iv) will be given separately.

Proof of Lemma 5.3.4(i). Without loss of generality, we assume $a_{32} + a_{34} = 0$, i.e., $a_{32} = a_{34} = 0$. Hence, we consider matrices A with $\left(\begin{smallmatrix} a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{smallmatrix}\right) =$

$\left(\begin{smallmatrix} a_{31} & 0 & a_{33} & 0 \\ a_{41} & a_{42} & a_{43} & 1 \end{smallmatrix}\right)$, where $a_{31} + a_{33} \geq 1$, by validity of A. This matrix defines the compression function:

$$\mathsf{F}_\mathsf{A}(x_1, x_2) = a_{41}x_1 \oplus a_{42}x_2 \oplus a_{43}\pi_1(x_1) \oplus \pi_2(x_2) \oplus \pi_3(a_{31}x_1 \oplus a_{33}\pi_1(x_1))\,.$$

Define the functions $f_1(x) = a_{41}x \oplus a_{43}\pi_1(x) \oplus \pi_3(a_{31}x \oplus a_{33}\pi_1(x))$ and $f_2(x) = a_{42}x \oplus \pi_2(x)$. Notice that $\mathsf{F}_\mathsf{A}(x_1, x_2) = f_1(x_1) \oplus f_2(x_2)$. A collision-finding adversary $\mathcal{A}$ for F_A proceeds as follows. It sets up two lists of q random elements $X_1 := \{x_1^{(1)}, \ldots, x_1^{(q)}\}$ and $X_2 := \{x_2^{(1)}, \ldots, x_2^{(q)}\}$, and computes the corresponding values $f_1(x_1^{(k)})$ and $f_2(x_2^{(k)})$ (for $k = 1, \ldots, q$). Thus, in total $\mathcal{A}$ makes q queries to each of its random oracles. Given one of the $\binom{q}{2}^2$ combinations $x_1, x_1' \in X_1$, $x_2, x_2' \in X_2$, this combination yields a collision for F_A with probability $\Theta(2^{-n})$. Concluding, $\mathbf{Adv}_{\mathsf{F}_\mathsf{A}}^{\mathrm{col}}(q) = \Omega(q^4/2^n)$. □

Proof of Lemma 5.3.4(ii). For the cases $j \in \{3, 4\}$ as explained in Section 5.1.3 (these cases are in fact redundant due to the validity of A), collisions can be found in at most $2^{n/3}$ queries due to Stam's bound (see Section 2.6.2). We consider a matrix A with $a_{32} = a_{42} = 0$ (the case $j = 2$), a similar analysis holds for $j = 1$. Note that F_A satisfies $\mathsf{F}_\mathsf{A}(x_1, x_2) = \mathsf{F}_{\mathsf{A}'}(x_1, \pi_2(x_2))$, where A' has third and fourth rows $(a_{31}a_{34}a_{33}00, a_{41}a_{44}a_{43}01)$. The compression function $\mathsf{F}_{\mathsf{A}'}$ satisfies the condition of this lemma for $j = 4$, and invertibility of π_2 guarantees a collision for F_A in the same amount of queries plus 2. We note that the result also follows from Proposition 5.2.5, but as we will use Lemma 5.3.4(ii) in the single-permutation setting as well, we here consider a more robust reduction. □

Proof of Lemma 5.3.4(iii). The idea of the attack is to focus on collisions $(x_1, x_2) \neq (x_1', x_2')$ for which the input to the third permutation π_3 is the same. We first consider the case of matrices A with $\left(\begin{smallmatrix} a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{smallmatrix}\right) = \left(\begin{smallmatrix} 1 & 1 & 0 & 0 \\ a_{41} & a_{42} & 1 & 1 \end{smallmatrix}\right)$, the general case is discussed afterwards. The matrix defines compression function

$$\mathsf{F}_\mathsf{A}(x_1, x_2) = a_{41}x_1 \oplus a_{42}x_2 \oplus \pi_1(x_1) \oplus \pi_2(x_2) \oplus \pi_3(x_1 \oplus x_2)\,.$$

We construct an adversary $\mathcal{A}$ that aims at finding a collision $(x_1, x_2) \neq (x_1', x_2')$ such that

$$x_1 \oplus x_2 = x_1' \oplus x_2'\,, \tag{5.7a}$$

$$a_{41}x_1 \oplus a_{42}x_2 \oplus \pi_1(x_1) \oplus \pi_2(x_2) = a_{41}x_1' \oplus a_{42}x_2' \oplus \pi_1(x_1') \oplus \pi_2(x_2')\,. \tag{5.7b}$$

The adversary sets up two lists of $q = 2^\alpha$ elements $X_1 := \{x_1^{(1)}, \ldots, x_1^{(q)}\}$ and $X_2 := \{x_2^{(1)}, \ldots, x_2^{(q)}\}$, where $x_1^{(k)} = x_2^{(k)} = 0^{n-\alpha} \| \langle k-1 \rangle_\alpha$ for $k = 1, \ldots, q$. It computes the corresponding values $\pi_1(x_1^{(k)})$ and $\pi_2(x_2^{(k)})$ (for $k = 1, \ldots, q$). Fix any x_1, x_2, x_1' such that $x_1 \neq x_1'$. Then, there is exactly one x_2' such that (5.7a) is satisfied. For any of these $q\binom{q}{2}$ options, (5.7b) is satisfied with probability $\Theta(2^{-n})$.

For any of such succeeding tuples, the adversary additionally queries $\pi_3(x_1 \oplus x_2) = \pi_3(x_1' \oplus x_2')$ in order to get a collision. Concluding, $\mathbf{Adv}^{\mathrm{col}}_{\mathsf{F_A}}(q) = \Omega(q^3/2^n)$.

The described attack relies on the key property that the set of equations

$$\begin{pmatrix} a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \begin{pmatrix} x_1 \oplus x_1' \\ x_2 \oplus x_2' \\ \pi_1(x_1) \oplus \pi_1(x_1') \\ \pi_2(x_2) \oplus \pi_2(x_2') \end{pmatrix} = 0$$

contains an equation in which x_1, x_2, x_1', x_2' occur exactly once. By the requirement of A, $\left(\begin{smallmatrix} a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{smallmatrix}\right)$ contains at least two zeroes. If two zeroes are located in the same row, this key property is satisfied and the attack succeeds. On the other hand, if both rows contain exactly one zero, one can XOR the first equation to the second one to return to the first case. $\square$

Proof of Lemma 5.3.4(iv). Without loss of generality, we assume $a_{41} = 1$. By Lemma 5.3.4(ii), we can consider $a_{32} = a_{33} = a_{34} = 1$. The matrix defines compression function

$$\mathsf{F_A}(x_1, x_2) = x_1 \oplus \pi_3(a_{31}x_1 \oplus x_2 \oplus \pi_1(x_1) \oplus \pi_2(x_2)) \,.$$

We construct a collision adversary $\mathcal{A}$ for $\mathsf{F_A}$. The adversary sets up a list of $q = 2^\alpha$ random elements $X_2 := \{x_2^{(1)}, \ldots, x_2^{(q)}\}$, and computes the corresponding values $y_2^{(k)} = \pi_2(x_2^{(k)})$ (for $k = 1, \ldots, q$). Additionally, the adversary sets up two lists $X_1 := \{x_1^{(1)}, \ldots, x_1^{(q)}\}$ and $Y_3 := \{y_3^{(1)}, \ldots, y_3^{(q)}\}$, where $x_1^{(k)} = y_3^{(k)} = 0^{n-\alpha}\|\langle k-1\rangle_\alpha$ for $k = 1, \ldots, q$. It computes the corresponding values $y_1^{(k)} = \pi_1(x_1^{(k)})$ and $x_3^{(k)} = \pi_3^{-1}(y_3^{(k)})$ (for $k = 1, \ldots, q$). Fix any x_1, y_3, x_1' such that $x_1 \neq x_1'$. Then, there is exactly one y_3' such that $x_1 \oplus y_3 = x_1' \oplus y_3'$. The adversary obtains a collision for $\mathsf{F_A}$ if X_2 contains two elements x_2, x_2' such that $x_2 \oplus y_2 = a_{31}x_1 \oplus y_1 \oplus x_3$ and $x_2' \oplus y_2' = a_{31}x_1' \oplus y_1' \oplus x_3'$. Two such x_2, x_2' exist with probability $\Omega(\binom{q}{2}/2^{2n})$. As the adversary needs to succeed for only one of the $q\binom{q}{2}$ choices of x_1, y_3, x_1', it finds a collision for $\mathsf{F_A}$ with probability $\Omega(q^5/2^{2n})$. $\square$

Next, the compression functions evolved from Lemma 5.3.3 are analyzed with respect to the attacks of Lemma 5.3.4. Before proceeding, we remark that for the multi-permutation setting, the following reductions apply to the compression function classes evolved from Lemma 5.3.3. We refer to these reductions as the "M- and N-reduction."

M-reduction. Applying Proposition 5.2.2, and Proposition 5.2.4 on $i = 1$ corresponds to mutually swapping $\left(\begin{smallmatrix} a_{31} \\ a_{41} \end{smallmatrix}\right) \leftrightarrow \left(\begin{smallmatrix} a_{32} \\ a_{42} \end{smallmatrix}\right)$ and $\left(\begin{smallmatrix} a_{33} \\ a_{43} \end{smallmatrix}\right) \leftrightarrow \left(\begin{smallmatrix} a_{34} \\ a_{44} \end{smallmatrix}\right)$;

N-reduction. Proposition 5.2.5 reduces to swapping $\left(\begin{smallmatrix} a_{3j} \\ a_{4j} \end{smallmatrix}\right) \leftrightarrow \left(\begin{smallmatrix} a_{3,j+2} \\ a_{4,j+2} \end{smallmatrix}\right)$ for $j \in \{1, 2\}$.

We now continue evaluating the matrices A of the form (5.6), and consider the different values of $\|\mathsf{a}_{3,*}\|$.

$\|\mathsf{a}_{3,*}\| = \mathbf{0}$. The matrix is invalid and excluded by definition;

$\|\mathsf{a}_{3,*}\| = \mathbf{1}$. The matrix is vulnerable to the attack of Lemma 5.3.4(i);

$\|\mathsf{a}_{3,*}\| = \mathbf{2}$. The matrix contradicts either one of the requirements of Lemma 5.3.4. Technically, if $(a_{31} + a_{33})(a_{32} + a_{34}) = 0$ it violates Lemma 5.3.4(i), and otherwise the values $a_{41}, \ldots, a_{44}$ will violate either the requirement of Lemma 5.3.4(ii) or of Lemma 5.3.4(iii);

$\|\mathsf{a}_{3,*}\| = \mathbf{3}$. By M- and N-reductions, it suffices to consider $a_{31}a_{32}a_{33}a_{34} = 1110$, and consequently $a_{44} = 1$ by Lemma 5.3.4(ii). Lemma 5.3.4(iii) now states that we require $a_{41} = a_{43}$, which gives the following four options for $a_{41}a_{42}a_{43}$: 000, 010, 101 and 111. The first one is vulnerable to the attack of Lemma 5.3.4(iv), and the fourth matrix is equivalent to the second (by consequently applying Proposition 5.2.3 on $(c_0, c_1) = (1, 1)$, and Proposition 5.2.4 for $i = 2$). We are left with A_1 and A_2 of (5.5):

$$\mathsf{A}_1 = \left(\begin{array}{cc|ccc} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ \hline 0 & 1 & 0 & 1 & 1 \end{array}\right), \qquad \mathsf{A}_2 = \left(\begin{array}{cc|ccc} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ \hline 1 & 0 & 1 & 1 & 1 \end{array}\right);$$

$\|\mathsf{a}_{3,*}\| = \mathbf{4}$. By M- and N-reductions, it suffices to consider $a_{41}a_{42}a_{43}a_{44} \in \{0000, 1000, 1010, 1100, 1110, 1111\}$. The cases 1000 and 1100 are vulnerable to the attacks of Lemmas 5.3.4(iv) and 5.3.4(iii), respectively. For the cases 0000 and 1111, finding collisions is as hard as finding collisions for $\mathsf{F}(x_1, x_2) = x_1 \oplus x_2 \oplus \pi_1(x_1) \oplus \pi_2(x_2)$ (for which collisions are found in at most $2^{n/4}$ queries). We are left with A_3 and A_4 of (5.5):

$$\mathsf{A}_3 = \left(\begin{array}{cc|ccc} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ \hline 1 & 0 & 1 & 0 & 1 \end{array}\right), \qquad \mathsf{A}_4 = \left(\begin{array}{cc|ccc} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ \hline 1 & 1 & 1 & 0 & 1 \end{array}\right).$$

It remains to analyze collision and preimage security of the four compression functions defined by the matrices of (5.5), which is done in the following lemma. Particularly, Lemma 5.3.5 completes the proof of Theorem 5.3.1.

Lemma 5.3.5. *Let $\varepsilon > 0$. Then:*

(i) $\lim_{n\to\infty} \mathbf{Adv}^{\mathrm{col}}_{\mathsf{F}_{\mathsf{A}_k}}(2^{n/2(1-\varepsilon)}) = 0$ *for* $k = 1, 2, 3, 4$*;*

(ii) $\lim_{n\to\infty} \mathbf{Adv}^{\mathrm{epre}}_{\mathsf{F}_{\mathsf{A}_2}}(2^{2n/3(1-\varepsilon)}) = 0$*, and* $\mathbf{Adv}^{\mathrm{epre}}_{\mathsf{F}_{\mathsf{A}_k}}(q) = \Theta(q^2/2^n)$ *for* $k = 1, 3, 4$.

Proof. Part (i) is proven in Appendix B.1, part (ii) in Appendix B.2. □

5.4 Main Result for Single-Permutation Setting

In a similar fashion as in Section 5.3, we analyze the security of compression functions based on three calls to the same permutations, the single-permutation setting. It turns out that *there does not exist any* compression function of the form (5.3) that achieves optimal collision resistance. We note that this result does not rely on Conjecture 5.3.2. In Appendix B.3 we show how the results of this section can be generalized to cover any single-permutation compression function where additional affine transformations on the permutation inputs are taken into account.

Theorem 5.4.1. *Consider the single-permutation setting, where* $\mathcal{P} = \pi \xleftarrow{\$} \mathrm{Perm}(n)$ *and* $\pi_1 = \pi_2 = \pi_3 = \pi$. *Any compression function* F_A *defined by a binary matrix* A *of the form (5.2) satisfies* $\mathbf{Adv}^{\mathrm{col}}_{\mathsf{F}_\mathsf{A}}(q) = \Omega(q^5/2^{2n})$.

Proof. The proof of Theorem 5.4.1 is similar to the proof of Theorem 5.3.1, and we highlight the differences. Lemmas 5.3.3 and 5.3.4 still apply, and additionally the M-reduction also holds in the single-permutation setting. Notice that the N-reduction *does not hold* as it incorporates Proposition 5.2.5. Similar to before, we will evaluate the matrices A of the form (5.6). The case $\|\mathsf{a}_{3,*}\| \leq 2$ is the same as before.

$\|\mathsf{a}_{3,*}\| = 3$. By M-reductions, it suffices to consider $a_{31}a_{32}a_{33}a_{34} \in \{1110, 0111\}$.

- $a_{31}a_{32}a_{33}a_{34} = 1110$. The same analysis as in Section 5.3.1 applies, leaving the matrices A_1 and A_2 of (5.5). In the single-permutation setting, the two corresponding compression functions satisfy $\mathsf{F}_{\mathsf{A}_1}(x_1, \pi(x_1)) = \pi^2(x_1)$ and $\mathsf{F}_{\mathsf{A}_2}(x_1, x_2) = \mathsf{F}_{\mathsf{A}_2}(x_1, x_1 \oplus x_2 \oplus \pi(x_1))$ for any x_1, x_2. Collisions can thus be trivially found;
- $a_{31}a_{32}a_{33}a_{34} = 0111$. By Lemma 5.3.4(ii), we have $a_{41} = 1$. Lemma 5.3.4(iii) now states that we require $a_{42} = a_{44}$, which gives the following four options for $a_{42}a_{43}a_{44}$: 000, 010, 101, and 111. The first one is vulnerable to the attack of Lemma 5.3.4(iv), the second, third, and fourth matrix satisfy $\mathsf{F}_\mathsf{A}(x_1, x_1) = x_1$, $\mathsf{F}_\mathsf{A}(x_1, x_1) = 0$, and $\mathsf{F}_\mathsf{A}(x_1, x_1) = \pi(x_1)$, respectively, for any x_1. Collisions can thus be trivially found;

$\|\mathsf{a}_{3,*}\| = 4$. Except for $a_{41}a_{42}a_{43}a_{44} \in \{1010, 1001, 0110, 0101\}$, all induced compression functions satisfy $\mathsf{F}_\mathsf{A}(x_1, x_1) \oplus \pi(0) \in \{0, x_1, \pi(x_1)\}$ for any x_1, for which collisions can be trivially found. The cases $1001, 0110$ are vulnerable to Lemma 5.3.4(iii). The remaining two cases, which are equivalent by M-reduction, allow for trivial collisions as well: the compression function induced by $(a_{41}a_{42}a_{43}a_{44}) = (1010)$ satisfies $\mathsf{F}_\mathsf{A}(x_1, \pi^{-1}(x_1 \oplus \pi(x_1))) = 0$ for any x_1 (cf. [198]).

Hence, the analyzed compression functions either allow for trivial collision or are vulnerable to Lemma 5.3.4, therewith allowing for collisions in at most $2^{2n/5}$ queries. □

Concluding, for any compression function F_A of the form (5.3), where the three permutations are equal to one single permutation π, collisions can be found in at most $2^{2n/5}$ queries, hence considerably faster than in $2^{n/2}$ queries.

5.5 Conclusions

We provided a full security classification of $2n$-to-n-bit compression functions that are solely built of XOR operators and of three permutations. Therewith, we have analyzed compression functions that are not included in the analysis of Rogaway and Steinberger [198], but yet are interesting because of their elegance (they only employ XOR operators) and efficiency (XOR operators are slightly cheaper than finite field multiplications by constants). For any of the 2^{15} compression functions of the described form, we either provide a formal collision and preimage security proof or a collision attack more efficient than the birthday bound.

For the multi-permutation setting, where the three permutations are different, there are exactly four equivalence classes of functions that allow for optimal collision resistance, one class of which the compression functions achieve optimal preimage resistance with respect to the bounds of [199]. A summary of these results is given in Table 5.1. Regarding the absolute number of collision/preimage secure compression functions, by ways of an extensive computation one finds 96 functions equivalent to $\mathsf{F}_{\mathsf{A}_1}$ (including the $\mathsf{F}_{\mathsf{A}_1}$ itself), 48 functions in each of the classes defined by $\mathsf{F}_{\mathsf{A}_2}$ and $\mathsf{F}_{\mathsf{A}_4}$, and 24 functions equivalent to $\mathsf{F}_{\mathsf{A}_3}$. In total, we have thus proven 216 compression functions optimally collision secure, 48 of which we have proven optimally preimage secure. A small part of the results for the multi-permutation setting relies on an extremal graph theory based conjecture, Conjecture 5.3.2, which we supported by an extensive and detailed heuristic. We leave the full analysis of Conjecture 5.3.2 as an open problem.

For the single-permutation setting, where the three permutations are the same, we show that it is not possible to construct a $2n$-to-n-bit compression function that achieves optimal collision resistance. In light of the amount of optimally secure compression functions we have found in the multi-permutation setting, this observation is not as expected. This negative result casts doubts over the existence of any (larger) permutation based XOR based compression function built on (multiple invocations of) one single permutation. We leave this question as an open problem.

The results in this chapter are derived in the multi- and single-permutation setting. It is of interest to investigate the functions where only two primitives are the same; it is likely that secure functions in this setting exist (Shrimpton and Stam already make this observation for their compression function [206]). Different results may be obtained if we consider three underlying primitives to be one-way functions: in particular, the π-inverse-reduction (Proposition 5.2.5) and Lemma 5.3.4 rely on the invertibility of these primitives. Further research questions include the applicability of the approach followed in this chapter to different classes of compression functions, for instance with larger domain and range, with more permutations or random functions instead, or defined over different fields.

6 The Parazoa Hash Function Family

In 2007, Bertoni et al. introduced the sponge hash functions [41] as an alternative of the Merkle-Damgård design (see also Section 2.5.2). Since the introduction of sponge functions, it has been a standard practice in the cryptographic community to call hash functions "sponge-like" if they bear resemblances with the original sponge design in terms of iterating a wide state and employing underlying permutations in an extraction and absorbing phases. Despite the similarities, the indifferentiability results of the sponge hash function do not carry over in a straightforward manner to the sponge-like constructions and hence, an independent security analysis is required. At times even a small adjustment to the sponge design may render it insecure (cf. Section 6.2). It is thus an interesting research problem to come up with a secure class of hash functions generalizing the original sponge construction, and the results of which could simply be carried over to its members.

Contributions of This Chapter

We introduce the parazoa family of hash functions[1] as a generalization of the sponge hash function. Our generalization is crafted towards obtaining secure sponge-like hash functions in the indifferentiability framework. The parazoa hash function family allows for a wider class of compression and extraction functions that satisfy a set of simple conditions. These conditions facilitate the indifferentiability proof, but we note that these are easily satisfied and realistic for practical purposes. Similar to the original sponge design, parazoa functions allow for variable length outputs. In [210, Section 4.2], Stam analyzes permutation based compression functions satisfying certain criteria, "overloaded single call Type-I compression functions," that are similar to the compression functions employed in the parazoa design (albeit the requirements posited in [210, Definition 17] are stronger). The major difference is that in the parazoa design the compression function is *not* necessarily preimage/collision resistant. In particular, Stam leaves

[1]Parazoa is the name of the subkingdom of animals to which the sponges belong [223].

it as an open problem to analyze security of overloaded single call compression functions in the iteration.

We prove that the maximum advantage of any distinguisher in differentiating a parazoa hash function, based on ideal primitive π, from a random oracle is upper bounded by $O((Kq)^2/2^{s-p-d})$, where the distinguisher makes at most q queries of length at most K blocks. Here, s denotes the iterated state size, p denotes the number of bits extracted in one execution of the extraction function, and d is called the capacity loss, a quantity inherent to the specific parazoa design (cf. Table 6.1). Even though the indifferentiability proof focuses on parazoa designs where both the compression and extraction function are based on one single permutation, the result easily extends to designs where multiple random permutations and/or random functions are employed.

Naturally, the sponge function design (Section 2.5.2) falls within the categorization of parazoa functions, and our indifferentiability result confirms the bound of [42]. Additional hash function designs covered by the parazoa specification are Grindahl [131], and second round SHA-3 candidate hash functions CubeHash [37], Fugue [107], JH [224], Keccak [40], and (a restricted variant of) Luffa [78], two of which advanced to the final round of NIST's hash function competition. The implications of our indifferentiability results on these functions are summarized in Table 6.1, and we elaborate on it in Section 6.5. We note that not all obtained bounds are as expected. In particular, our indifferentiability bound on JH is worse than the indifferentiability bound proven by Moody et al. [171]. The difference may be a price to pay in return for generality. For the generic parazoa design, we note that our indifferentiability bound is optimal: for the original sponge design, the best generic attack meets the derived security bound [41]. Still, for concrete instantiations of the parazoa hash function a design-specific proof may result in a better bound.

Outline

In Section 6.1, we present some mathematical preliminaries complementary to Chapter 2. In Section 6.2, we present a "sponge-like" function that is insecure. Parazoa functions are introduced and formalized in Section 6.3, and an indifferentiability result for parazoa functions is given in Section 6.4. We finish the chapter with concluding remarks in Section 6.5.

Bibliographic Notes

The contents of this chapter have been published by Andreeva, Mennink, and Preneel in the International Journal of Information Security [15]. The results were also presented at the ECRYPT II Hash Workshop 2011 [14].

Table 6.1. Implications of the parazoa indifferentiability result for various hash functions. Here, s is the internal state size, m the message block size, p the extraction block size, n the number of output bits, and the capacity loss d is further explained in Section 6.4. The distinguisher makes at most q queries of maximal length K blocks. For Fugue, an indifferentiability result in a different model has been derived in [108]. See Section 6.5 for more details.

	(s, m, p)	d	existing bound	our bound
Sponge	$(r+c, r, r)$	0	$O((Kq)^2/2^c)$ [42]	$O((Kq)^2/2^c)$
Grindahl	(s, m, n)	m	—	$O((Kq)^2/2^{s-n-m})$
Quark	$(r+c, r, r)$	0	$O((Kq)^2/2^c)$ [42]	$O((Kq)^2/2^c)$
PHOTON ($r' \leq r$)	$(r+c, r, r')$	$r-r'$	$O((Kq)^2/2^c)$ [42]	$O((Kq)^2/2^c)$
PHOTON ($r' \geq r$)	$(r+c, r, r')$	0	—	$O\left((Kq)^2/2^{c+r-r'}\right)$
SPONGENT	$(r+c, r, r)$	0	$O((Kq)^2/2^c)$ [42]	$O((Kq)^2/2^c)$
CubeHash-n	$(1024, 257, n)$	1	—	$O((Kq)^2/2^{1023-n})$
Fugue-n ($n \leq 256$)	$(960, 32, n)$	m	[108]	$O((Kq)^2/2^{928-n})$
Fugue-n ($n > 256$)	$(1152, 32, n)$	m	[108]	$O((Kq)^2/2^{1120-n})$
JH-n	$(1024, 512, n)$	m	$O((Kq)^2/2^{512})$ [171]	$O((Kq)^2/2^{512-n})$
Keccak-n	$(1600, s-2n, n)$	$s-3n$	$O((Kq)^2/2^{2n})$ [42]	$O((Kq)^2/2^{2n})$
Luffa-n ($n \leq 256$)	$(768, 256, 256)$	0	—	$O((Kq)^2/2^{512})$
Luffa-384	$(1024, 256, 256)$	0	—	$O((Kq)^2/2^{768})$
Luffa-512	$(1280, 256, 256)$	0	—	$O((Kq)^2/2^{1024})$

6.1 Security Model

A function $f : \mathbb{Z}_2^m \to \mathbb{Z}_2^n$ for $m \geq n$ is called "balanced" if any $y \in \mathbb{Z}_2^n$ has exactly 2^{m-n} preimages under f. We define its inverse function by $f^{-1} : y \mapsto \{x \in \mathbb{Z}_2^m \mid f(x) = y\}$. We use the indifferentiability notion of Definition 2.3.5. Throughout this chapter, $\mathcal{R}$ will be a random oracle, and we consider $\mathcal{P} = \pi \xleftarrow{\$} \text{Perm}(s)$. The right oracle R has two interfaces, as the distinguisher can make forward as well as inverse queries to the permutation π.

6.2 Differentiability of a "Sponge-Like" Function

In this section we show that a simple modification of the original sponge design can render it insecure with respect to indifferentiability. To this end, we construct the following sponge-like design.

Consider the following hash function $\mathcal{H} : \mathbb{Z}_2^\infty \to \mathbb{Z}_2^n$, that has a state size $s = 2n$, and processes message blocks of n bits (see also Figure 6.1). It is based on a $2n$-bit permutation π, and uses a simple injective padding function pad to process messages of arbitrary length: $\mathsf{pad}(M) = M\|1\|0^{-|M|-1 \bmod n}$, parsed into message blocks of n bits. For an initial value $\mathsf{iv} = \mathsf{iv}^l\|\mathsf{iv}^r$, the hash function $\mathcal{H}$ processes a message M as follows. $\mathcal{H}(M)$ outputs the n rightmost bits of h_k, where $(M_1, \dots, M_k) \leftarrow \mathsf{pad}(M)$, $h_0 \leftarrow \mathsf{iv}^l\|\mathsf{iv}^r$, and $h_i \leftarrow \pi(h_{i-1} \oplus (M\|0^n))$ for $i = 1, \dots, k$. Notice that this design follows the original sponge design, with a small modification that the message digest is defined by the other half of the state (but we stress that this observation does not invalidate the security of the sponge function design). We construct a distinguisher $\mathcal{D}$ that can distinguish $(\mathcal{H}^\pi, \pi)$ from $(\mathcal{R}, \mathcal{S}^\mathcal{R})$, for any simulator $\mathcal{S}$.

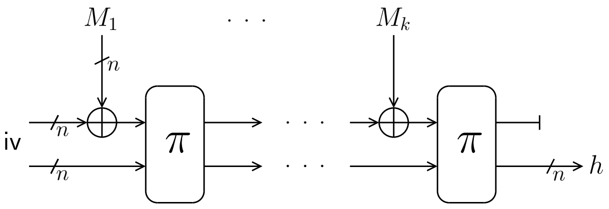


Figure 6.1. The sponge-like hash function $\mathcal{H}$ of Section 6.2.

- First, $\mathcal{D}$ decides on an arbitrary message M of length $0 < |M| < n$. It defines $M_1 = \mathsf{pad}(M)$ and $M_3 = 1\|0^{n-1}$. Note that $\mathsf{pad}(M_1\|M_2) = M_1\|M_2\|M_3$ for any $M_2 \in \mathbb{Z}_2^n$;
- $\mathcal{D}$ queries M to the left oracle, to get $h_1 \leftarrow L(M)$;
- $\mathcal{D}$ queries $0\|h_1$ to the right oracle, to get $x_1\|y_1 \leftarrow R(0\|h_1)$;
- $\mathcal{D}$ queries $(x_1 \oplus M_3)\|y_1$ to the right oracle, to get $x_2\|y_2 \leftarrow R((x_1 \oplus M_3)\|y_1)$;
- $\mathcal{D}$ queries $(\mathsf{iv}^l \oplus M_1)\|\mathsf{iv}^r$ to the right oracle, to get $x_3\|y_3 \leftarrow R((\mathsf{iv}^l \oplus M_1)\|\mathsf{iv}^r)$;
- $\mathcal{D}$ queries $M_1\|x_3$ to the left oracle, to get $h_2 \leftarrow L(M_1\|x_3)$.

In the real world, where the distinguisher queries $\mathcal{H}$ and π, the answers of the oracles satisfy $h_1 = y_3$ and $h_2 = y_2$ by construction. In the simulated world however, h_2 equals $\mathcal{R}(M_1\|x_3)$. As the simulator generated y_2 without any knowledge of M_1, equality $h_2 = y_2$ holds with negligible probability only. We notice that, even though the described hash function is a parazoa design, the counterexample does not contradict Theorem 6.4.1. In particular, Theorem 6.4.1 results in a $O(1)$ indifferentiability bound (as the design has $d = n$).

6.3 Parazoa Functions

Informally, parazoa functions process a message M as follows. Firstly, the message is *padded* into several integral message blocks of m bits, using a padding function $\mathsf{pad} : \mathbb{Z}_2^\infty \to (\mathbb{Z}_2^m)^\infty$. Throughout, by k we denote the number of message blocks of a padded message. Then, these message blocks are *absorbed* by the s-bit state (compression phase), by applying sequentially a compression function $f : \mathbb{Z}_2^s \times \mathbb{Z}_2^m \to \mathbb{Z}_2^s$ on the state and the message. Next, the state is *squeezed* to obtain $l \geq 1$ output data blocks of p bits sequentially (extraction phase). The corresponding extraction function is denoted by $g : \mathbb{Z}_2^s \to \mathbb{Z}_2^s \times \mathbb{Z}_2^p$. It operates on the state, and returns an updated state and the extract. A *finalization* function $\mathsf{fin} : \mathbb{Z}_2^{pl} \to \mathbb{Z}_2^n$ combines these l data blocks of p bits into the n-bit message digest. We require that $m, p \leq s$, and that $pl \geq n$. Both the compression function and the extraction function are based on an s-bit permutation π.

These functions are further explained in Sections 6.3.1-6.3.4 (for ease of presentation, the function pad is introduced last), together with the requirements of these functions for the security proof in Section 6.4. Now, for a fixed initialization vector iv of size s, the parazoa function $\mathcal{H}$ processes a message M as follows.

$$\begin{aligned}
&\mathcal{H}(M)\\
&\quad (M_1, \ldots, M_k) \leftarrow \mathsf{pad}(M)\,,\\
&\quad h_0 \leftarrow \mathsf{iv}\,,\\
&\quad h_i \leftarrow f(h_{i-1}, M_i) \text{ for } i = 1, \ldots, k\,,\\
&\quad (h_{k+i}, P_i) \leftarrow g(h_{k+i-1}) \text{ for } i = 1, \ldots, l\,,\\
&\quad h \leftarrow \mathsf{fin}(P_1, \ldots, P_l)\,,\\
&\quad \textbf{return } h\,.
\end{aligned} \tag{6.1}$$

This function is depicted in Figure 6.2.

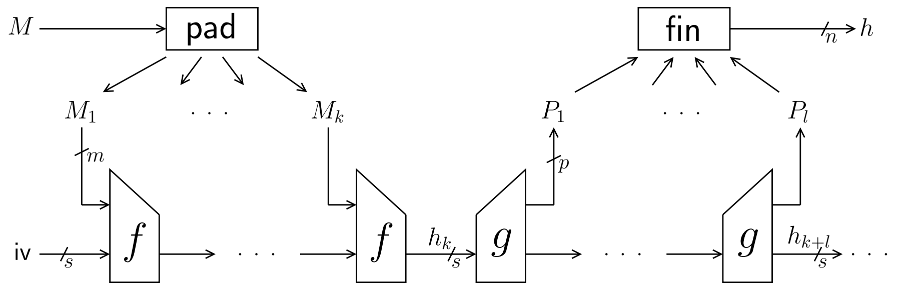


Figure 6.2. The parazoa hash function $\mathcal{H}$ of (6.1).

6.3.1 Compression Function f

On input of a state value h_{i-1} and a message input M_i, the compression function first uses an injection function $\mathsf{L}_{\text{in}} : \mathbb{Z}_2^s \times \mathbb{Z}_2^m \to \mathbb{Z}_2^s$ to inject the message into the state, and then permutes the state with π. This state is then transformed and combined with a feed-forward using a function $\mathsf{L}_{\mathrm{out}} : \mathbb{Z}_2^s \times \mathbb{Z}_2^s \times \mathbb{Z}_2^m \to \mathbb{Z}_2^s$. Formally, the compression function f is defined as $f(h_{i-1}, M_i) = h_i$, where $x \leftarrow \mathsf{L}_{\mathrm{in}}(h_{i-1}, M_i)$, $y \leftarrow \pi(x)$ and $h_i \leftarrow \mathsf{L}_{\mathrm{out}}(y, h_{i-1}, M_i)$. The compression function f is depicted in Figure 6.3.

For any $x \in \mathbb{Z}_2^s$ we define its *capacity set* $\mathcal{C}(x) = \{h \in \mathbb{Z}_2^s \mid \exists\ M \in \mathbb{Z}_2^m \text{ s.t. } \mathsf{L}_{\mathrm{in}}(h, M) = x\}$. Intuitively, $\mathcal{C}(x)$ denotes the set of state values $h \in \mathbb{Z}_2^s$ for which some message injection results in x as input to the permutation. For two values $h, h' \in \mathbb{Z}_2^s$, we define the function $\mathsf{same}\mathcal{C}(h, h')$, that outputs **true** if and only if h and h' are both a member of a capacity set $\mathcal{C}(x)$ for some x.

Requirement from L_{in}. We require L_{in} to satisfy the following properties: (i) for any $x \in \mathbb{Z}_2^s$ and $h \in \mathcal{C}(x)$, there exists *exactly one* $M \in \mathbb{Z}_2^m$ such that $\mathsf{L}_{\mathrm{in}}(h, M) = x$, and (ii) if $\mathcal{C}(x) \cap \mathcal{C}(x') \neq \emptyset$, then $\mathcal{C}(x) = \mathcal{C}(x')$. Intuitively, the first requirement guarantees that for a state value $h \in \mathbb{Z}_2^s$, a different M results in a different $x = \mathsf{L}_{\mathrm{in}}(h, M)$. The second requirement intuitively guarantees that two elements $x, x' \in \mathbb{Z}_2^s$ either have the same or disjoint capacity sets. As becomes clear in the proof, this requirement can be relaxed at a security loss of factor 2^m. We notice that these requirements are easily satisfied, and standard injection functions L_{in} satisfy both. In particular, commonly used injection functions, e.g., functions that consist of XORing the message with and/or inserting it in a part of the state, clearly satisfy both properties. Note that the second requirement is satisfied for any linear transformation.

Requirement from $\mathsf{L}_{\mathrm{out}}$. We require that for any $(h, M) \in \mathbb{Z}_2^s \times \mathbb{Z}_2^m$, the function $\mathsf{L}_{\mathrm{out}}(\cdot, h, M)$ is a bijection on the state. Its inverse function is denoted by $\mathsf{L}_{\mathrm{out}}^{-1}[h, M]$.

6.3.2 Extraction Function g

On input of a state value h_{k+i-1}, the extraction function g employs an extracting transformation $\mathsf{L}_{\mathrm{ex}} : \mathbb{Z}_2^s \to \mathbb{Z}_2^p$ that outputs a data block, and then permutes the state with π. Formally, the extraction function g is defined as $g(h_{k+i-1}) = (h_{k+i}, P_i)$, where $P_i \leftarrow \mathsf{L}_{\mathrm{ex}}(h_{k+i-1})$ and $h_{k+i} \leftarrow \pi(h_{k+i-1})$. The function g is depicted in Figure 6.3. Similar to f, one can consider an additional transformation

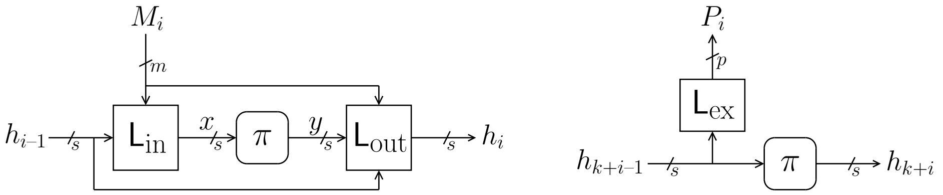


Figure 6.3. The compression function f (left) and the extraction function g (right).

after the call to the permutation, which may have h_{k+i-1} as extra input. This generalization would, however, make the proof considerably more complex (see Section 6.5).

Requirement from $\mathsf{L}_{\mathbf{ex}}$. We require L_{ex} to be balanced. Intuitively, this requirement means that each extract $P \in \mathbb{Z}_2^p$ is equally likely to occur. Accordingly, the function $\mathsf{L}_{\text{ex}}^{-1}$ is defined as described in Section 6.1, i.e., $\mathsf{L}_{\text{ex}}^{-1}(P) = \{h \in \mathbb{Z}_2^s \mid \mathsf{L}_{\text{ex}}(h) = P\}$.

6.3.3 Finalization Function fin

The function fin combines the l bit strings, obtained from squeezing the state, into the message digest. In most of the existing sponge based designs, the finalization function simply consists of concatenating a required number of blocks, $l = \lceil n/p \rceil$, and chopping it to the required length of n bits. Parazoa functions allow for a generalized finalization function.

Requirement from fin. We require fin to be balanced. Intuitively, this requirements means that each digest h is equally likely to occur. Accordingly, the function fin^{-1} is defined as described in Section 6.1, i.e., $\mathsf{fin}^{-1}(h) = \{(P_1, \ldots, P_l) \mid \mathsf{fin}(P_1, \ldots, P_l) = h\}$.

6.3.4 Padding Function pad

The padding function pad is an injective mapping that transforms messages of arbitrary length into messages of length an integral multiple of the block size m. Associated to pad is the function depad, that processes a message M' as follows. If $M' = \mathsf{pad}(M)$ for some message M, it outputs this M, otherwise it outputs $\perp$. Note that the output is unique as the padding function is injective.

Requirement from pad. We require pad to satisfy the following property: we either have $l = 1$, or the last block of a padded message, M_k, satisfies for any $x \in \mathbb{Z}_2^s$ and $(h', M') \in \mathbb{Z}_2^s \times \mathbb{Z}_2^m$:

$$\mathsf{L}_{\text{in}}(x, M_k) \neq x \text{ and } \mathsf{L}_{\text{in}}(\mathsf{L}_{\text{out}}(x, h', M'), M_k) \neq x\,. \tag{6.2}$$

As explained in Section 6.4.3 in more detail, this requirement comes from the fact that permutation queries to the simulator corresponding to the extraction phase of (6.1), may correspond to compression function executions f as well. We notice that for the original sponge design, condition (6.2) translates to requiring that the last block of a padded message is not a zero-block (which is exactly the requirement as posited by the authors of the sponge design in [42]). Because parazoa functions generalize these functions significantly, this requirement has become more complex accordingly.

6.4 Indifferentiability of Parazoa Functions

In this section, we prove the parazoa function of Section 6.3 indifferentiable from a random oracle, under the assumption that the underlying permutation π behaves like an ideal primitive. Intuitively, we demonstrate that there exists a simulator such that no distinguisher can differentiate the real world $(\mathcal{H}^\pi, \pi)$ from the simulated world $(\mathcal{R}, \mathcal{S}^\mathcal{R})$, except with negligible probability.

For the purpose of the proof, we introduce a technical variable d which we refer to as the "capacity loss." Consider the set of all couples (h, x) such that $\mathsf{L}_\text{in}(h, M) = x$ for some M (M is uniquely determinable from h, x). We define $d \geq 0$ to be the minimal value such that

Criterion 1. For fixed x and fixed $P \in \mathbb{Z}_2^p$, there are at most 2^d couples (h, x) such that $h \in \mathsf{L}_\text{ex}^{-1}(P)$;

Criterion 2. For fixed h and fixed $P \in \mathbb{Z}_2^p$, there are at most 2^d couples (h, x) such that $x \in \mathsf{L}_\text{ex}^{-1}(P)$.

As a consequence of the first criterion, we obtain $|\mathcal{C}(x)| \leq 2^{p+d}$ for any x, as any $h \in \mathbb{Z}_2^s$ satisfies $\mathsf{L}_\text{ex}(h) = P$ for exactly one $P \in \mathbb{Z}_2^p$. We note that the second criterion is not needed in case $l = 1$ (see Section 6.4.3). The reason why we opt for the name "capacity loss," as well as an intuition behind this parameter, is given in Section 6.4.1.

Theorem 6.4.1. *Let π be a random s-bit permutation, and let $\mathcal{R}$ be a random oracle. Let $\mathcal{H}$ be a parazoa function parameterized by l, m, n, p, s, t. Let $\mathcal{D}$ be a distinguisher that makes at most q_L left queries of maximal length $(K-1)m$ bits, where $K \geq 1$, and q_R right queries. Then:*

$$\mathbf{Adv}^{\text{iff}}_{\mathcal{H}^\pi, \mathcal{S}}(\mathcal{D}) \leq 42\frac{q(q+1)}{2^{s-p-d}} + 4\frac{q^2}{2^s},$$

where $q = (K+l)q_L + q_R$ and $\mathcal{S}$ makes at most $q_\mathcal{S} \leq q_R$ queries to $\mathcal{R}$.

We note that the bound is optimal for the generic parazoa design: for the original sponge design, as a particular instantiation of the parazoa functions, the best generic attack requires about $2^{(s-p-d)/2}$ queries [41] and meets the derived indifferentiability bound. In the remainder of this section, an intuition behind the capacity loss d is given in Section 6.4.1, basic preliminary definitions for the description of our simulator are given in Section 6.4.2, and the simulator used in the proof is introduced and explained in more detail in Section 6.4.3 and Figure 6.5. Then, Theorem 6.4.1 is formally proven in Section 6.4.4.

6.4.1 On the Capacity Loss d

We will provide an intuition for the technical parameter d. It is used in the computation of the indifferentiability security bound: in some cases, the simulator has to generate a value h such that it is *not* a member of $\mathcal{C}(x)$, for one or more values of x, and moreover such that $\mathsf{L}_{\text{ex}}(h) = P$ for some fixed P. Then, by the first criterion, the simulator can choose out of at least 2^{s-p-d} values for h. Intuitively, the value $s - p - d$ represents the number of bits of information of a state value that (1) cannot be affected by the distinguisher by ways of message injection, and (2) that cannot be obtained by the distinguisher by ways of extraction. We demonstrate this by ways of two examples.

- Example 1. the sponge hash function [41] (cf. Section 2.5.2). The functions L_{in} and L_{ex} of the sponge hash function design are given in Figure 6.4. We start with the first criterion that defines parameter d. Fix any $x \in \mathbb{Z}_2^s$ and fix any $P \in \mathbb{Z}_2^p$, where $p = r$. We are considering the number of choices for h that satisfy $\mathsf{L}_{\text{in}}(h, M) = x$ for some M, and that satisfy $\mathsf{L}_{\text{ex}}(h) = P$. As becomes clear from Figure 6.4, the first requirement uniquely fixes the last c bits of h, while the second statement fixes the first r bits of c. Altogether, the value h is uniquely determined by the fixed values x and P, and criterion 1 of the definition of d is satisfied for $d = 0$. Criterion 2 is equivalent, and we find that for sponge functions $d = 0$, hence the value attains its minimum;
- Example 2. the sponge-like hash function of Section 6.2. The functions L_{in} and L_{ex} of this hash function design are given in Figure 6.4. Fix any $x \in \mathbb{Z}_2^s$ and fix any $P \in \mathbb{Z}_2^p$, where $p = c$. Again, we are considering the number of choices for h that satisfy $\mathsf{L}_{\text{in}}(h, M) = x$ for some M, and that satisfy $\mathsf{L}_{\text{ex}}(h) = P$. As can be observed from Figure 6.4, if the rightmost c bits of x are equal to P, there are exactly 2^r choices for h that satisfy these requirement, namely $\{(M \| 0^c) \oplus x \mid M \in \mathbb{Z}_2^r\}$. Again, criterion 2 is equivalent; we thus obtain $d = r = s - p$. In this example the distinguisher has "full control over the state": the first r bits can be freely adjusted by message injection, and the last c bits can be obtained (in most cases) by message extraction.

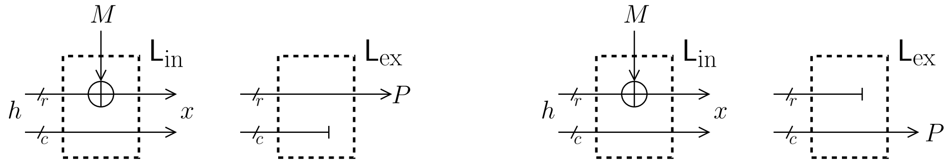


Figure 6.4. The functions L_{in} and L_{ex} of the sponge hash function (left) and the sponge-like hash function of Section 6.2 (right), where for sake of explanation we consider the state size to be $r + c$.

We note that the sponge-like hash function of Section 6.2 (example 2) still fits in the parazoa framework, and the indifferentiability result of Theorem 6.4.1 applies. Yet, as $d = s - p$ we obtain a trivial bound. The same occurs if we consider an example parazoa function where the message input size m equals the state size

s: it turns out that $d = s - p$, and the indifferentiability result of Theorem 6.4.1 results in a trivial bound.

The value d varies between 0 and $s-p$ by construction, and ideally attains its minimum. In this case, the number of bits of information that cannot be controlled by the adversary is $s-p$. This value is exactly the capacity of the sponge hash function. For increasing d the value $s-p-d$ decreases, and therefore we call d the "capacity loss."

6.4.2 Defining the Simulator

The simulator maintains an initially empty database $\mathcal{L}_\pi$ that represents the simulated permutation. It consists of tuples $(x, y) \in \mathbb{Z}_2^s \times \mathbb{Z}_2^s$, where y denotes the sampled image of x under π. The simulator has two interfaces, denoted by $\mathcal{S}, \mathcal{S}^{-1}$, and access to $\mathcal{R}$. It maintains a graph (V, A), which initially consists of the node iv and includes no arcs. The arcs in A are labeled by messages $M \in \mathbb{Z}_2^m$ and define input-output pairs of the compression function f: an arc $v \xrightarrow{M} w$ means that $f(v, M) = w$. Intuitively, if there is a path $IV \xrightarrow{M_1} h_1 \xrightarrow{M_2} \cdots \xrightarrow{M_k} h_k$ in (V, A), then $f(\ldots f(f(IV, M_1), M_2)\ldots, M_k) = h_k$. Abusing notation, we denote such path by $\mathsf{iv} \xrightarrow{\overline{M}} h_k$ for $\overline{M} = (M_1, \ldots, M_k)$. By definition of L_{in}, one query pair (x, y) adds at most 2^{p+d} arcs to the graph, namely the arcs leaving from the vertices in $\mathcal{C}(x)$. By V_{out} we denote the set of nodes in V with an outgoing arc in (V, A). Notice that $V_{\mathrm{out}} = \bigcup_{x \in \mathsf{dom}(\mathcal{L}_\pi)} \mathcal{C}(x)$. By $t(V)$ we denote the tree in (V, A) rooted in iv. Additionally, by $\bar{t}(V)$ we denote the subset of nodes of $t(V)$ that are labeled by a correctly padded message.

In addition, the simulator maintains a database Δ. This database will consist of future query inputs $x \in \mathbb{Z}_2^s$ of which the simulator knows that they correspond to the extraction phase of the parazoa execution of (6.1). Associated to each $x \in \Delta$ is a tuple $(i, P_{i+1} \cdots P_l)$ with $i \in \{1, \ldots, l-1\}$. Essentially, i denotes the number of executions of g that are already simulated for this specific path, and $P_{i+1}, \ldots, P_l$ denote the output values of the subsequent executions of g, determined by the simulator before. The idea behind Δ is further explained in Section 6.4.3.

6.4.3 Intuition

As is common in indifferentiability proofs, the simulator needs to be constructed in such a way that the answers from the oracles $(\mathcal{H}^\pi, \pi)$ and $(\mathcal{R}, \mathcal{S}^\mathcal{R})$ are close to identically distributed. In other words, the oracle answers made by the simulator need to be in consistency with the random oracle, in such a sense that any relation among the query answers in real world, holds in the simulated world as well. In particular, the simulator needs to pay attention to the following scenario: suppose a path $\mathsf{iv} \xrightarrow{\overline{M}} h_k$ is in the graph, for $\overline{M} \in \mathsf{rng}(\mathsf{pad})$ (i.e., $h_k \in \bar{t}(V)$). Suppose moreover that $h_k, \ldots, h_{k+l-1} \in \mathsf{dom}(\mathcal{L}_\pi)$, where $P_i = \mathsf{L}_{\mathrm{ex}}(h_{k+i-1})$ and $h_{k+i} = \mathcal{L}_\pi(h_{k+i-1})$ (for $i = 1, \ldots, l$). Then, these values $(P_1, \ldots, P_l)$ should satisfy

```
Forward Query $\mathcal{S}(x)$
000 if $x \in \mathsf{dom}(\mathcal{L}_\pi)$ :
001     return $y = \mathcal{L}_\pi(x)$
002 if $x \in \Delta$ assoc. with some $(i; P_{i+1} \cdots P_l)$ :
003     $h_{\mathrm{nxt}} \xleftarrow{\$} \mathsf{L}_{\mathrm{ex}}^{-1}(P_{i+1}) \backslash (\mathsf{dom}(\mathcal{L}_\pi) \cup \Delta)$
004     if $\mathcal{C}(h_{\mathrm{nxt}}) \cap t(V) \neq \emptyset$ :
005         GOTO 003
006     $y \leftarrow h_{\mathrm{nxt}}$
007     if $y \in \mathsf{rng}(\mathcal{L}_\pi)$ : GOTO 003
008     if $\mathcal{C}(x) \cap t(V) \neq \emptyset$ :
009         { find unique $h \in t(V)$ and $M$
              s.t. $\mathsf{L}_{\mathrm{in}}(h, M) = x$
010         if $\mathsf{L}_{\mathrm{out}}(y, h, M) \in V_{\mathrm{out}} \cup \left(\bigcup_{x \in \Delta} \mathcal{C}(x)\right)$
011         or if $\exists\, h' \in t(V)$ : $\mathsf{same}\mathcal{C}(h', \mathsf{L}_{\mathrm{out}}(y, h, M))$ :
012             GOTO 003
013     end if
014     $\Delta \leftarrow \Delta \backslash \{(x; i, P_{i+1} \cdots P_l)\}$
015     if $i + 1 < l$ : $\Delta \leftarrow \Delta \cup \{(h_{\mathrm{nxt}}; i + 1, P_{i+2} \cdots P_l)\}$
016 else if $\mathcal{C}(x) \cap t(V) = \emptyset$ :
017     $y \xleftarrow{\$} \mathbb{Z}_2^s \backslash \mathsf{rng}(\mathcal{L}_\pi)$
018 else if $\mathcal{C}(x) \cap t(V) \neq \emptyset$ :
019     { find unique $h \in t(V), \overline{M}$ and $M$
          s.t. $\mathsf{L}_{\mathrm{in}}(h, M) = x$ and iv $\xrightarrow{\overline{M}} h$
020     if $\overline{M} \| M \notin \mathsf{rng}(\mathsf{pad})$ :
021         $y \xleftarrow{\$} \mathbb{Z}_2^s \backslash \mathsf{rng}(\mathcal{L}_\pi)$
022         if $\mathsf{L}_{\mathrm{out}}(y, h, M) \in V_{\mathrm{out}} \cup \mathcal{C}(x) \cup \left(\bigcup_{x \in \Delta} \mathcal{C}(x)\right)$
023         or if $\exists\, h' \in t(V)$ : $\mathsf{same}\mathcal{C}(h', \mathsf{L}_{\mathrm{out}}(y, h, M))$ :
024             GOTO 021
025     else if $\overline{M} \| M \in \mathsf{rng}(\mathsf{pad})$ :
026         $h \xleftarrow{\$} \mathcal{R}(\mathsf{depad}(\overline{M} \| M))$
027         $(P_1, \ldots, P_l) \xleftarrow{\$} \mathsf{fin}^{-1}(h)$
028         $h_{\mathrm{nxt}} \xleftarrow{\$} \mathsf{L}_{\mathrm{ex}}^{-1}(P_1) \backslash (\mathsf{dom}(\mathcal{L}_\pi) \cup \{x\} \cup \Delta)$
029         if $\mathcal{C}(h_{\mathrm{nxt}}) \cap t(V) \neq \emptyset$ :
030             GOTO 028
031         $y \leftarrow \mathsf{L}_{\mathrm{out}}^{-1}[h, M](h_{\mathrm{nxt}})$
032         if $y \in \mathsf{rng}(\mathcal{L}_\pi)$ : GOTO 028
033         if $\mathsf{L}_{\mathrm{out}}(y, h, M) \in V_{\mathrm{out}} \cup \mathcal{C}(x) \cup \left(\bigcup_{x \in \Delta} \mathcal{C}(x)\right)$
034         or if $\exists\, h' \in t(V)$ : $\mathsf{same}\mathcal{C}(h', \mathsf{L}_{\mathrm{out}}(y, h, M))$ :
035             GOTO 028
036         if $1 < l$ : $\Delta \leftarrow \Delta \cup \{(h_{\mathrm{nxt}}; 1, P_2 \cdots P_l)\}$
037     end if
038 end if
039 return $\mathcal{L}_\pi(x) \leftarrow y$
```

```
Inverse Query $\mathcal{S}^{-1}(y)$
100 if $y \in \mathsf{rng}(\mathcal{L}_\pi)$ :
101     return $x = \mathcal{L}_\pi^{-1}(y)$
102 $x \xleftarrow{\$} \mathbb{Z}_2^s \backslash \mathsf{dom}(\mathcal{L}_\pi)$
103 if $x \in \Delta$ or $\mathcal{C}(x) \cap t(V) \neq \emptyset$ :
104     GOTO 102
105 return $\mathcal{L}_\pi^{-1}(y) \leftarrow x$
```

Figure 6.5. The simulator $\mathcal{S}$ for π used in the proof of Theorem 6.4.1. The simulator aborts if a certain **GOTO**-statement is executed $C > 0$ times consecutively, for some constant C.

$\mathsf{fin}(P_1, \ldots, P_l) = \mathcal{R}(\mathsf{depad}(\overline{M}))$ in order for the simulator to maintain consistency. However, in general the simulator can only guarantee this equation to hold if the P_i's are decided *after* $\overline{M}$ is known, but *before* h_k is known (notice that h_k determines $P_1 = \mathsf{L}_{\mathrm{ex}}(h_k)$ deterministically). The simulator of Figure 6.5 handles this problem in a smart way: in the query where the last arc (h_{k-1} to h_k) is added, the simulator decides on $(P_1, \dots, P_l)$ on forehand, and based on these, it fixes the next state value h_k such that $h_k \in \mathsf{L}_{\text{ex}}^{-1}(P_1)$. This value equals the input to the next execution of π in the chaining. It stores $(h_k; 1, P_2, \dots, P_l)$ in its database Δ. As soon as the simulator is then queried h_k, the simulator will apply the same trick: the answer $h_{k+1} \leftarrow \mathcal{L}_\pi(h_k)$ will be generated such that $h_{k+1} \in \mathsf{L}_{\text{ex}}^{-1}(P_2)$. This value h_{k+1} equals the input to the next execution of π in the chaining. Subsequently it replaces the corresponding entry in Δ with $(h_{k+1}; 2; P_3, \dots, P_l)$.

Before describing the simulator in more detail, we note that it includes several **GOTO**-statements. These statements guarantee the randomly generated values to satisfy certain properties. Associated to the simulator is a constant $C > 0$: the simulator aborts if a certain **GOTO**-statement is executed C times consecutively. Due to the inclusion of this parameter C, the simulator operates in polynomial time, rather than in expected polynomial time. In the security proof (Section 6.4.4) this constant is fixed to a certain value.

The simulator will answer its queries such that the tree $t(V)$ grows as little as possible: indeed, any path in the tree may emerge in the need of an extra element in Δ, and eventually in an evaluatable query. However, as mentioned before, one query pair (x, y) defines at most 2^{p+d} arcs in the graph, leaving from the nodes in the set $\mathcal{C}(x)$. The simulator will answer its queries so as to satisfy the following properties concerning the growth of the graph:

(i) Out of all newly added arcs, at most one will be added to the tree. More generally, the simulator assures the following property at any time in the execution:

$$|\mathcal{C}(x) \cap t(V)| \le 1 \text{ for any } x \in \mathbb{Z}_2^s\,; \tag{6.3}$$

(ii) When a query adds a new arc to the tree, its end node has no outgoing arc. Together with (i), this implies that per query at most one arc is added to the tree;

(iii) The tree does not contain any colliding paths.

Notice that a query pair (x, y) adds a new arc to the tree *if and only if* $\mathcal{C}(x) \cap t(V) \neq \emptyset$. Indeed, $\mathcal{C}(x)$ corresponds to all nodes with an outgoing arc defined by the query pair (x, y). In this case, the simulator needs to assure that properties (i)-(iii) are satisfied. Secondly if $x \in \Delta$, the simulator needs to handle as described above. Ideally, the value x satisfies $x \notin \Delta$ and $\mathcal{C}(x) \cap t(V) = \emptyset$, which explains the algorithm for $\mathcal{S}^{-1}$. In forward queries to $\mathcal{S}$, however, x is chosen by the distinguisher, and it may be possible that $\mathcal{C}(x) \cap t(V) \neq \emptyset$ or $x \in \Delta$. We will now explain the algorithm for a forward query x to $\mathcal{S}$, based on the above observations.

In case $\mathcal{C}(x) \cap t(V) = \emptyset$ (lines 016-017), no arc will be added to the tree, and (i)-(iii) are trivially satisfied. In case $\mathcal{C}(x) \cap t(V) \neq \emptyset$, by (6.3) and the definition of L_{in}, there exists one unique couple $h \in t(V)$ and $M \in \mathbb{Z}_2^m$ such that $\mathsf{L}_{\text{in}}(h, M) = x$. By construction, the current query adds the arc $h \xrightarrow{M} v$ to the tree, where $v = \mathsf{L}_{\text{out}}(y, v, M)$. In lines 022 and 033, the simulator assures that this is the only arc added to the tree, i.e., that y is chosen such that $v \notin V_{\text{out}}$ (there is no outgoing arc from v) and $v \notin \mathcal{C}(x)$ (no outgoing arc from v will accidentally be added in the current round). Additionally, requirements (i) and (iii) are covered by requiring that v does not share a capacity set with a node already in $t(V)$: lines 023 and 034.

Hence, a query adds at most one arc to the tree, and in particular, in case of evaluatable queries, the last arc $h_{k-1} \xrightarrow{M} h_k$ is really added last. We still need to consider this specific case that $\bar{t}(V)$ is increased. Additionally, we still need to explain the case of $x \in \Delta$.

$\bar{\mathbf{t}}(\mathbf{V})$ gets increased but $\mathbf{x} \notin \boldsymbol{\Delta}$. This specific case corresponds to the **else**-clause of line 025, the simulator will proceed as previously described: the next state value h_{nxt} (which equals the first permutation input of the extraction phase) is generated, and the original answer y is generated accordingly in line 031. Notice that we need to assure that no collision in $\mathcal{L}_\pi$ occurs (line 032). In line 036, the node h_{nxt} is added to Δ, provided $1 < l$. Note that, by (iii), the path to h_k is unique, and $(P_1, \ldots, P_l)$ is generated in a non-ambiguous way;

$\bar{\mathbf{t}}(\mathbf{V})$ does not get increased and $\mathbf{x} \in \boldsymbol{\Delta}$. This case corresponds to the **if**-clause of line 002: again, the simulator will generate the next state value h_{nxt} (which equals the next permutation input of the extraction phase), and generate the original answer y accordingly (line 006). It will update Δ in lines 014-015. Note that in this if-clause it may still be possible that $\mathcal{C}(x) \cap t(V) \neq \emptyset$. Then, the simulator assures properties (i)-(iii) as mentioned before (by lines 010 and 011);

$\bar{\mathbf{t}}(\mathbf{V})$ gets increased and $\mathbf{x} \in \boldsymbol{\Delta}$. It may be the case that a query x to $\mathcal{S}$ is an element of Δ, and moreover adds an arc to the tree (this is checked in the **if**-clause of line 008). However, as we will now explain, the message block that labels this arc can never be the last block of a padded message, and hence, $\bar{t}(V)$ will not be increased.

As $x \in \Delta$, this specific value is fixed by the simulator on forehand (in the previous query of the extraction phase defined as h_{nxt}). In particular, it is generated in the query where $(x', y') \in \mathcal{L}_\pi$ is generated such that either $\mathsf{L}_{\text{out}}(y', h', M') = x$ for some h', M' (line 031, or such that $y' = x$ (line 006). However, x had been generated such that $\mathcal{C}(x)$, all start-nodes of the arcs defined by x, had an empty intersection with $t(V)$ (lines 004 and 029).[2] Also, in all future queries, it is assured that new queries cannot make the link to this specific x (due to "$\bigcup_{x \in \Delta} \mathcal{C}(x)$" in lines 010, 022, and 033). As a consequence, if the query (x, y) adds an arc to the tree, this only happens for an arc leaving from the end node of an arc defined by (x', y') (as this is the only query round in which $\mathcal{C}(x)$ is not avoided as end node of a newly

[2]Notice that, if $l = 1$, one does not need to assure this property. In particular, lines 004 and 029 become unnecessary. In the proof, these are the lines that make us require the second property concerning the capacity loss d (cf. Section 6.4).

added arc to the tree). In other words, after the forward query x to $\mathcal{S}$, we have the path $\mathsf{iv} \xrightarrow{\overline{M}} h_0 \xrightarrow{M_1} h_1 \xrightarrow{M_2} h_2$ in the tree for $\overline{M} \in (\mathbb{Z}_2^m)^\infty$ and $M_1, M_2 \in \mathbb{Z}_2^m$, where the arc (h_0, h_1) is defined by (x', y') and the arc (h_1, h_2) by (x, y). However, the value M_2 satisfying this path, particularly satisfies $x = \mathsf{L}_{\text{in}}(h_1, M_2)$, where $h_1 = \mathsf{L}_{\text{out}}(y', h', M')$. Given the specific property of x (in the beginning of this paragraph), it satisfies either

$$x = \mathsf{L}_{\text{in}}(x, M_2) \text{ or } x = \mathsf{L}_{\text{in}}(\mathsf{L}_{\text{out}}(x, h', M'), M_2)\,,$$

for some $(h', M') \in \mathbb{Z}_2^s \times \mathbb{Z}_2^m$. By the requirement in Section 6.3.4, we either have $l = 1$ (which means that $\Delta = \emptyset$ at all time), or that M_2 can never be the last block of a padded message. Summarizing, a query $x \in \Delta$ *never* increases $\bar{t}(V)$.

The full proof of Theorem 6.4.1 is given in Section 6.4.4.

6.4.4 Proof of Theorem 6.4.1

Let $\mathcal{S}$ be the simulator of Figure 6.5, and let $\mathcal{D}$ be any distinguisher that makes at most q_L left queries of maximal length $(K-1)m$ bits, where $K \geq 1$, and q_R right queries. Recall from Definition 2.3.5 that the goal is to bound:

$$\mathbf{Adv}^{\text{iff}}_{\mathcal{H}^\pi,\mathcal{S}}(\mathcal{D}) = \left|\mathbf{Pr}\left(\mathcal{D}^{\mathcal{H}^\pi,\pi} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{\mathcal{R},\mathcal{S}^{\mathcal{R}}} = 1\right)\right|\,. \qquad (6.4)$$

Theorem 6.4.1 will be proven via a game-playing argument. In this aspect, our result is fundamentally different from the result obtained by Bertoni et al. for the original sponge design [42]. Each game consists of a left and a right oracle. In the proof, G_1 will equal the simulated world, and G_9 the real world. We will go from game 1 to game 9 stepwise, and obtain a bound on (6.4) using a hybrid argument. We define $C = 1$, hence the simulator of Figure 6.5 aborts if a certain **GOTO**-statement is executed.

Game 1: $G_1 = (L_1, R_1^{L_1})$ (Figure 6.6). The left oracle L_1 of game 1 is a lazily-sampled random oracle, and the right oracle R_1 consists of the two interfaces defined by the simulator of Figure 6.5, with an additional difference that a failure condition **bad** is added to the **GOTO**-statements in lines 005, 007, 012, 024, 030, 032, 035, and 104. The distinguisher does not see the difference until the adversary in game 1 sets **bad** (recall that we put $C = 1$). We obtain $\left|\mathbf{Pr}\left(\mathcal{D}^{\mathcal{R},\mathcal{S}^{\mathcal{R}}} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{G_1} = 1\right)\right| \leq \mathbf{Pr}\left(\mathcal{D}^{G_1} \text{ sets } \mathbf{bad}\right)$.

Game 2: $G_2 = (L_2^{L_1}, R_1^{L_1})$ (Figure 6.6). The left oracle of game 1 is replaced by a so-called relay oracle L_2 that passes the queries made by the distinguisher to L_1, and returns its responses. The right oracle remains unchanged. The distinguisher has identical views in G_1 and G_2, and we obtain $\mathbf{Pr}\left(\mathcal{D}^{G_1} = 1\right) = \mathbf{Pr}\left(\mathcal{D}^{G_2} = 1\right)$.

Game 3: $\boldsymbol{G_3 = (L_3^{R_1^{L_1}}, R_1^{L_1})}$ (Figure 6.6). The left oracle of game 2 is now replaced by an implementation of the parazoa function, which moreover uses the right oracle as a subroutine, rather than L_1 directly. The right oracle itself remains unchanged. In Lemma 6.4.2 it is proven that, as long as the **bad** flag is not set in any of the two games, both are identical. Formally, we obtain $\left|\mathbf{Pr}\left(\mathcal{D}^{G_2} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{G_3} = 1\right)\right| \leq \mathbf{Pr}\left(\mathcal{D}^{G_2} \text{ sets } \mathbf{bad}\right) + \mathbf{Pr}\left(\mathcal{D}^{G_3} \text{ sets } \mathbf{bad}\right)$.

Note that in game 3, as well as in all subsequent games, the right oracle will be queried at most $q := (K + l)q_L + q_R$ times. Indeed, in all of the following games, the left oracle queries the right oracle at most K times in the compressing phase, and l times in the extraction phase. All subsequent right oracles are constructed in such a way that each query to this oracle adds at most one element to $\mathcal{L}_\pi$.

Game 4: $\boldsymbol{G_4 = (L_3^{R_2}, R_2)}$. The right oracle R_2 of game 4 differs from R_1 of game 3 in the sense that $h \xleftarrow{\$} L_1(\mathsf{depad}(\overline{M}\|M))$; $(P_1, \ldots, P_l) \xleftarrow{\$} \mathsf{fin}^{-1}(h)$ (lines 026 and 027) is replaced by $(P_1, \ldots, P_l) \xleftarrow{\$} \mathbb{Z}_2^{pl}$. Observe that the games are perfectly indistinguishable: in game 3, R_1 is the only algorithm querying L_1, and as the padding is injective, it never queries L_1 twice on the same value. Therefore, it can just as well generate the random values h himself. Then, as the function fin is balanced, we have, for any $\alpha \in \mathbb{Z}_2^{pl}$: $\mathbf{Pr}\left((P_1, \ldots, P_l) = \alpha : h \xleftarrow{\$} \mathbb{Z}_2^n, (P_1, \ldots, P_l) \xleftarrow{\$} \mathsf{fin}^{-1}(h)\right) = 1/2^{pl}$. In other words, the values $(P_1, \ldots, P_l)$ follow the uniform random distribution on pl bits, and therefore the right oracle can just generate them directly. Formally, we have $\mathbf{Pr}\left(\mathcal{D}^{G_3} = 1\right) = \mathbf{Pr}\left(\mathcal{D}^{G_4} = 1\right)$.

Game 5: $\boldsymbol{G_5 = (L_3^{R_3}, R_3)}$ (Figure 6.7). In game 4, all values $(P_1, \ldots, P_l)$ are randomly generated as soon as the first one is needed. The remaining $l - 1$ values are then associated to a node $x \in \Delta$ (line 036), and as soon as x is queried, the next value, P_{i+1}, is taken off of the list and processed. In game 5, these values P_i are not anymore generated in advance, but generated when needed. As a consequence, we implicitly adjust the definition of Δ, in the sense that each element is labeled by an index $i \in \{1, \ldots, l-1\}$ only. As in both cases the values $(P_1, \ldots, P_l)$ are generated uniformly at random, a distinguisher cannot see the difference. Consequently, we obtain $\mathbf{Pr}\left(\mathcal{D}^{G_4} = 1\right) = \mathbf{Pr}\left(\mathcal{D}^{G_5} = 1\right)$.

Game 6: $\boldsymbol{G_6 = (L_3^{R_4}, R_4)}$ (Figure 6.7). In game 5, the values P_i ($i = 1, \ldots, l$) are taken uniformly at random, and the state values h_{nxt} are taken according to the property that $P_i = \mathsf{L}_{\text{ex}}(h_{\text{nxt}})$ for all $i = 1, \ldots, l$ (lines 004 and 028 in Figure 6.7). In game 6, this is the other way around: h_{nxt} (recall that this value equals the input to the next permutation execution in the extraction phase) is taken randomly (permutation-wise), and the value P_i is taken such that $P_i = \mathsf{L}_{\text{ex}}(h_{\text{nxt}})$ still holds. Hence, the only changes are in lines 003-004 and 027-028 of game 5. In Lemma 6.4.3 it is proven that $\left|\mathbf{Pr}\left(\mathcal{D}^{G_5} = 1 \;\middle|\; \mathcal{D}^{G_5} \text{ sets } \neg\mathbf{bad}\right) - \mathbf{Pr}\left(\mathcal{D}^{G_6} = 1 \;\middle|\; \mathcal{D}^{G_6} \text{ sets } \neg\mathbf{bad}\right)\right| \leq \frac{2q^2}{2^s}$.

Game 7: $\boldsymbol{G_7 = (L_3^{R_5}, R_5)}$ (Figure 6.8). In game 6, concretely in the blocks 003-008, where the oracle is queried on an x belonging to the extraction phase, and 027-032 where the query answer to x will initiate the extraction phase, the oracle decides on its answer y based on the next state value h_{nxt}. In game 7, the answer y is taken uniformly at random, and the next state h_{nxt} is generated accordingly. The values $P_i = \mathsf{L}_{\text{ex}}(x_i)$ are not used in R_5, and their generation is omitted. In Lemma 6.4.4 it is proven that $\left|\mathbf{Pr}\left(\mathcal{D}^{G_6} = 1 \mid \mathcal{D}^{G_6} \text{ sets } \neg\mathbf{bad}\right) - \mathbf{Pr}\left(\mathcal{D}^{G_7} = 1 \mid \mathcal{D}^{G_7} \text{ sets } \neg\mathbf{bad}\right)\right| \leq \frac{2q^2}{2^s}$.

Game 8: $\boldsymbol{G_8 = (L_3^{R_6}, R_6)}$ (Figure 6.8). The right oracle R_6 in game 8 differs from R_5 in game 7 in the sense that the **GOTO**-statements that are accompanied with a **bad**-statement are removed. As a consequence, game 7 and 8 proceed identically as long as the **bad** flag is not set in game 7. Formally, we obtain $\left|\mathbf{Pr}\left(\mathcal{D}^{G_7} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{G_8} = 1\right)\right| \leq \mathbf{Pr}\left(\mathcal{D}^{G_7} \text{ sets } \mathbf{bad}\right)$.

Game 9: $\boldsymbol{G_9 = (L_3^{R_7}, R_7)}$. The right oracle R_7 mimics a lazily-sampled random permutation π, and the left oracle is the parazoa specification querying this right oracle. Hence, $G_9 = (\mathcal{H}^\pi, \pi)$, and thus $\mathbf{Pr}\left(\mathcal{D}^{G_9} = 1\right) = \mathbf{Pr}\left(\mathcal{D}^{\mathcal{H}^\pi,\pi} = 1\right)$. It turns out that R_6 of game 8 also mimics a lazily-sampled permutation, due to the removal of the **GOTO**-statements. In particular, any forward query to R_6 is answered with a $y \in \mathbb{Z}_2^s \backslash \mathsf{rng}(\mathcal{L}_\pi)$. As a consequence, we obtain $\mathbf{Pr}\left(\mathcal{D}^{G_8} = 1\right) = \mathbf{Pr}\left(\mathcal{D}^{G_9} = 1\right)$.

We conclude that (6.4) reduces to:

$$\mathbf{Adv}^{\text{iff}}_{\mathcal{H}^\pi,\mathcal{S}}(\mathcal{D}) \leq \sum_{\substack{i\in\{1,2,\\3,5,6,7\}}} \mathbf{Pr}\left(\mathcal{D}^{G_i} \text{ sets } \mathbf{bad}\right) + \frac{4q^2}{2^s}\,. \tag{6.5}$$

In the remainder of the proof, we will evaluate the probability that the distinguisher sets bad in game 3. Thereafter, we will elaborate on the other probabilities.

Consider the j^{th} query ($j = 1, \ldots, q$) to R_1. Notice that, by Lemma 6.4.5, we have $|\Delta| \leq j-1$ and $|t(V)| \leq j$. By the union bound, the probability that **bad** is set in this round, $\mathbf{Pr}_j$, equals the probability that **bad** is set in either of the lines 005, 007, 012, 024, 030, 032, 035, or 104. Denote by $\mathbf{Pr}_j^{\mathsf{xxx}}$ the probability that **bad** is set in the j^{th} query in line xxx. Observe that $\mathbf{Pr}_j^{012} \leq \mathbf{Pr}_j^{024} \leq \mathbf{Pr}_j^{035}$, and the simulator will either execute *one of these lines.* Similar observation holds for $\mathbf{Pr}_j^{005} \leq \mathbf{Pr}_j^{030}$ and $\mathbf{Pr}_j^{007} \leq \mathbf{Pr}_j^{032}$. Therefore, we obtain $\mathbf{Pr}\left(D^{G_3} \text{ sets } \mathbf{bad} \text{ inquery } j\right) = \mathbf{Pr}_j \leq \mathbf{Pr}_j^{030\vee032\vee035} + \mathbf{Pr}_j^{104}$. We first consider $\mathbf{Pr}_j^{030\vee032\vee035}$. Boolean **bad** is set in either of these lines if h_{nxt}, taken uniformly at random from a set of size at least $2^{s-p} - 2q$ (notice that $(P_1, \ldots, P_l)$ are fixed), violates lines 029, 032, 033, or 034.

029. This line is violated if h_{nxt} corresponds to any capacity set already represented in $t(V)$. Consider a node $h \in t(V)$. By the definition of the capacity loss d in Section 6.4 (as $\mathsf{L}_{\text{ex}}(h_{\text{nxt}}) = P$ is fixed), there are at most 2^d values h_{nxt} such that $h \in \mathcal{C}(h_{\text{nxt}})$. In total, there are at most $j2^d$ possible values for h_{nxt} that make this line violated;

032. Line 032 is violated if $\mathsf{L}_{\text{out}}^{-1}[h,M](h_{\text{nxt}})$ hits a set of size at most $j-1$. Recall that L_{out} forms a bijection on the state (for fixed h,M);

033. Recall that $V_{\text{out}} = \bigcup_{x_0 \in \mathsf{dom}(\mathcal{L}_\pi)} \mathcal{C}(x_0)$. Line 033 is violated if h_{nxt} hits any of the sets $\mathcal{C}(x_0)$, for $x_0 \in \mathsf{dom}(\mathcal{L}_\pi) \cup \{x\} \cup \Delta$. However, by the definition of d, for each x_0, there are at most 2^d values h_{nxt} that would satisfy $h_{\text{nxt}} \in \mathcal{C}(x_0)$. In total, there are at most $(2j-1)2^d$ possible values for h_{nxt} that make this line violated;

034. This line is violated if h_{nxt} shares the same capacity set with any of the elements in $t(V)$. Consider a node $h \in t(V)$: this element exactly defines *one* capacity set,[3] say $\mathcal{C}(x_0)$. By the definition of d (as $\mathsf{L}_{\text{ex}}(h_{\text{nxt}}) = P$ is fixed), there are at most 2^d values h_{nxt} that would satisfy $h_{\text{nxt}} \in \mathcal{C}(x_0)$. In total, there are at most $j2^d$ possible values for h_{nxt} that make this line violated.

Summarizing, we obtain $\mathbf{Pr}_j^{030 \vee 032 \vee 035} \leq \frac{(4j-1)2^d+j-1}{2^{s-p}-2q}$. Now for $\mathbf{Pr}_j^{104}$, notice that **bad** is set in line 104 if x, taken uniformly at random from a set of size at least $2^s - q$, hits a set of size at most $j2^{p+d} + j - 1$ (by the definition of d, each h is a member of at most 2^{p+d} capacity sets). Concluding, $\mathbf{Pr}_j^{104} \leq \frac{j2^{p+d}+j-1}{2^s-q}$. By the union bound, and under the assumption that $2q < 2^{s-p-1}$, we thus obtain

$$\mathbf{Pr}\left(\mathcal{D}^{G_3} \text{ sets } \mathbf{bad}\right) \leq \frac{7q(q+1)}{2^{s-p-d}}\,.$$

For games G_1, G_2, the same analysis holds. For games G_5, G_6, G_7, the same bound can be obtained similarly, and we only highlight the major differences: (i) h_{nxt} is now generated randomly from a set of size at least $2^s - 2q$, and (ii) we cannot use the fact that $\mathsf{L}_{\text{ex}}(h_{\text{nxt}}) = P$ is fixed anymore, but still the same analysis holds with 2^d replaced by 2^{p+d}. In general, however, the same bound is obtained for these games. Now, combined with (6.5), these bounds give the claimed result.

Lemma 6.4.2. *As long as* **bad** *is not set in games 2 and 3, both are identical. Formally,* $\mathbf{Pr}\left(\mathcal{D}^{G_2} = 1 \mid \mathcal{D}^{G_2} \text{ sets } \neg\mathbf{bad}\right) = \mathbf{Pr}\left(\mathcal{D}^{G_3} = 1 \mid \mathcal{D}^{G_3} \text{ sets } \neg\mathbf{bad}\right)$.

Proof. We need to prove that, until **bad** is set in either one of the two games, the query outcomes in games 2 and 3 are identically distributed. As both games employ the same right oracle, $\mathcal{D}$ can differentiate game 2 and 3 only if it discovers any inconsistencies in the answers by the left oracles (L_2 for game 2 and L_3 for game 3), given any list of queries made by $\mathcal{D}$ to the right oracle.

Consider an execution of game 2 or game 3. Recall that $\mathcal{L}_\pi$ consists of a list of query pairs to the right oracle, and that (V, A) is the graph defined by these. Denote by $\mathcal{V} = (\mathcal{V}_L, \mathcal{V}_R)$ any view of a distinguisher on an execution of the oracle (G_2 or G_3). Here, $\mathcal{V}_L$ is a list of *different* query pairs (M, h), and $\mathcal{V}_R$ a list of query

[3]Here we require the property of L_{in} that all capacity sets are the same if they share one element. It is clear that this requirement can be relaxed at a cost of 2^m, as mentioned in Section 6.3.1.

pairs for the right world. In game 2, we have $\mathcal{V}_R = \mathcal{L}_\pi$, and in game 3, we have $\mathcal{V}_R \subseteq \mathcal{L}_\pi$ (by construction). Denote by $(\overline{V}, \overline{A})$ the subgraph of (V, A) generated by the query pairs in $\mathcal{V}_R$. We need to prove that, given any view $\mathcal{V}$, outcomes of new queries to the left oracle are identically distributed in both games. Formally, we need to prove that for any $M \in \mathbb{Z}_2^\infty$ and any $h \in \mathbb{Z}_2^n$, we have

$$\begin{aligned} &\mathbf{Pr}\left(L_2(M) = h \text{ in } G_2 \mid \mathcal{V};\ M \notin \mathsf{dom}(\mathcal{V}_L);\ \mathcal{D}^{G_2} \text{ sets } \neg\mathbf{bad}\right) \\ =\ &\mathbf{Pr}\left(L_3(M) = h \text{ in } G_3 \mid \mathcal{V};\ M \notin \mathsf{dom}(\mathcal{V}_L);\ \mathcal{D}^{G_3} \text{ sets } \neg\mathbf{bad}\right). \end{aligned} \tag{6.6}$$

Define $(M_1, \ldots, M_k) = \mathsf{pad}(M)$ to be the padded message of M. We will call the queried message M "determined" by $\mathcal{V}_R$ if there exists a path $\mathsf{iv} \xrightarrow{M_1} \cdots \xrightarrow{M_k} h_k$ in $(\overline{V}, \overline{A})$. We will prove that if M is not determined by $\mathcal{V}_R$, both probabilities in (6.6) equal $1/2^n$. On the other hand, if M is determined by $\mathcal{V}_R$, both properties are still equal (although they may naturally have a higher value).

M is not determined by $\mathcal{V}_\mathbf{R}$. As the tree in $(\overline{V}, \overline{A})$ contains no path labeled by $M_1 \cdots M_k$, and moreover $M \notin \mathsf{dom}(\mathcal{V}_L)$, in both games the oracle L_1 had never been queried on M. Now, in game 2, L_2 passes the query through to L_1, which will generate its answer uniformly at random from $\mathbb{Z}_2^n$ (line 201). In game 3, the **for**-loop of line 402 will force the right oracle to grow the path $\mathsf{iv} \xrightarrow{M_1} \cdots \xrightarrow{M_k} h_k$ to some node h_k. By Lemma 6.4.5 and as M is not determined by $\mathcal{V}_R$, at least the last arc $h_{k-1} \xrightarrow{M_k} h_k$ will be newly added to the tree. By construction, the oracle R_3 will generate the query answer via the **else**-clause of line 025. It will query a *new* value to L_1, which samples the value $h \xleftarrow{\$} \mathbb{Z}_2^n$. The oracle R_3 will represent this value h in an obfuscated way in a list of l values $(P_1, \ldots, P_l)$ and answer all future queries from L_3 such that this oracle will exactly extract the same values $(P_1, \ldots, P_l)$. Consequently, the value h outputted by L_3 in line 411, exactly equals the one randomly sampled;

M is determined by $\mathcal{V}_\mathbf{R}$. By construction, the path $\mathsf{iv} \xrightarrow{M_1} \cdots h_{k-1} \xrightarrow{M_k} h_k$ is in the tree, for some h_{k-1}, h_k. But by Lemma 6.4.5, the tree is only increased with one arc at a time, and as a consequence, the arc (h_{k-1}, h_k) labeled by M_k must have been added to the tree *after* $M_1 \cdots M_{k-1}$ are known. Additionally, the tree is never increased in inverse queries (Lemma 6.4.5), and a determinatable path is never created in queries for $x \in \Delta$ (Lemma 6.4.6). In other words, this specific arc had been added in a forward query via the **else**-clause of line 025. In particular, the oracle R_1 already generated $(P_1, \ldots, P_l)$ such that $\mathsf{fin}(P_1, \ldots, P_l) = L_1(M)$ (by Lemma 6.4.5 there are no collisions in the tree, and thus the oracle indeed queried L_1 on M). It additionally defined the node h_k as the next state in the extraction phase corresponding to the above-described path. Additionally, it saved $(h_k; 1, P_2, \ldots, P_l)$ in Δ. By construction, this value h_k satisfies $\mathsf{L}_{\mathrm{ex}}(h_k) = P_1$. In particular, $\mathcal{V}_R$ and Δ jointly deterministically define $(P_1, \ldots, P_l)$, and therewith $L_1(M)$.

However, the distinguisher does not know Δ. Let $i^* \in \{0, \dots, l-1\}$ be the maximal index such that $h_k, \dots, h_{k+i^*-1} \in \mathsf{dom}(\mathcal{V}_R)$. Recall that we have $P_i = \mathsf{L}_{\text{ex}}(h_{k+i-1})$ and $h_{k+i} = \mathcal{L}_\pi(h_{k+i-1})$, for $i = 1, \dots, l$. Then, using this equation, the distinguisher can deterministically determine $P_1, \dots, P_{i^*}$ from $\mathcal{V}_R$. As $\mathcal{L}_\pi(h_{k+i^*})$ is unknown to the distinguisher, and all other values in $(\mathcal{V}_L, \mathcal{V}_R)$ are unrelated,[4] the distinguisher is oblivious to the values $P_{i^*+1}, \dots, P_l$ such that $\mathsf{fin}(P_1, \dots, P_l) = L_1(M)$. This analysis holds for both games: in game 2, the oracle L_2 will output this pre-determined h. In game 3, by construction of the **if**-clause of line 002, the oracle L_3 will output the same pre-determined h. □

Lemma 6.4.3. *As long as* **bad** *is not set in any of the games 5 and 6 (Figure 6.7), both games are statistically indistinguishable. Formally, we have:*

$$\left|\mathbf{Pr}\left(\mathcal{D}^{G_5} = 1 \;\middle|\; \mathcal{D}^{G_5} \text{ sets } \neg\mathbf{bad}\right) - \mathbf{Pr}\left(\mathcal{D}^{G_6} = 1 \;\middle|\; \mathcal{D}^{G_6} \text{ sets } \neg\mathbf{bad}\right)\right| \leq \frac{2q^2}{2^s}\,.$$

Proof. For the purpose of the proof, we construct two new games 5a and 6a. Game 5a differs from game 5 in the sense that line 028 is replaced by:

```
028a $h_{\text{nxt}} \xleftarrow{\$} \mathsf{L}_{\text{ex}}^{-1}(P_1)$
028b if $h_{\text{nxt}} \in \mathsf{dom}(\mathcal{L}_\pi) \cup \{x\} \cup \Delta$ :
028c     $\mathbf{bad}' \leftarrow$ true; GOTO 028a
```

and similar for line 004 of game 5. $\mathcal{D}$ does not see the difference, and hence $\mathbf{Pr}\left(\mathcal{D}^{G_5} = 1\right) = \mathbf{Pr}\left(\mathcal{D}^{G_{5a}} = 1\right)$. Game 6a differs from game 6 in the sense that line 027 is replaced by:

```
027a $h_{\text{nxt}} \xleftarrow{\$} \mathbb{Z}_2^s$
027b if $h_{\text{nxt}} \in \mathsf{dom}(\mathcal{L}_\pi) \cup \{x\} \cup \Delta$ :
027c     $\mathbf{bad}' \leftarrow$ true; GOTO 027a
```

and similar for line 003 of game 6. As before, $\mathbf{Pr}\left(\mathcal{D}^{G_6} = 1\right) = \mathbf{Pr}\left(\mathcal{D}^{G_{6a}} = 1\right)$, and the problem reduces to bounding $\left|\mathbf{Pr}\left(\mathcal{D}^{G_{5a}} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{G_{6a}} = 1\right)\right|$. By the properties of L_{ex}, both games distribute the (P_i, h_{nxt})'s identically as long as **bad** and **bad′** are not set. As a consequence, we obtain:

$$\begin{aligned}&\left|\mathbf{Pr}\left(\mathcal{D}^{G_{5a}} = 1 \;\middle|\; \mathcal{D}^{G_{5a}} \text{ sets } \neg\mathbf{bad}\right) - \mathbf{Pr}\left(\mathcal{D}^{G_{6a}} = 1 \;\middle|\; \mathcal{D}^{G_{6a}} \text{ sets } \neg\mathbf{bad}\right)\right| \\ &\leq \mathbf{Pr}\left(\mathcal{D}^{G_{5a}} \text{ sets } \mathbf{bad}'\right) + \mathbf{Pr}\left(\mathcal{D}^{G_{6a}} \text{ sets } \mathbf{bad}'\right).\end{aligned}$$

Consider game 5a, and assume $\mathcal{D}^{G_{5a}}$ did not set **bad**. Suppose the right oracle has been queried $j-1$ times, and consider the j^{th} query x to the right oracle.

[4] It may be the case that $h_{k+i} \in \mathsf{dom}(\mathcal{V}_R)$ for some $i > i^*$, but this only happens with negligible probability. Moreover, from the view of the distinguisher, this arc is equally likely to be a part of the tail as any other arc.

As the simulator will either execute line 004 or 028 (not both), and will set $\mathbf{bad}'$ with a higher probability in 028c, it suffices to consider this line only. Notice that we have $|\mathsf{dom}(\mathcal{L}_\pi) \cup \{x\} \cup \Delta| \leq 2j-1$ (by Lemma 6.4.5). The probability that $\mathbf{bad}'$ is set in 028c equals the probability that a value h_{nxt}, randomly sampled from a set of size 2^s (recall that line 028 is preceded by $P_1 \xleftarrow{\$} \mathbb{Z}_2^p$ and that L_{ex} is balanced), hits a value in a set of size at most $2j-1$. As a consequence, $\mathbf{bad}'$ is set in the j^{th} query with probability at most $\frac{2j-1}{2^s}$. Summed over j, we obtain $\mathbf{Pr}\left(\mathcal{D}^{G_{5a}} \text{ sets } \mathbf{bad}'\right) \leq \frac{q^2}{2^s}$. Similarly, we obtain $\mathbf{Pr}\left(\mathcal{D}^{G_{6a}} \text{ sets } \mathbf{bad}'\right) \leq \frac{q^2}{2^s}$. □

Lemma 6.4.4. *As long as* **bad** *is not set in any of the games 6 and 7 (Figures 6.7 and 6.8), both games are statistically indistinguishable. Formally, we have:*

$$\left|\mathbf{Pr}\left(\mathcal{D}^{G_6} = 1 \mid \mathcal{D}^{G_6} \text{ sets } \neg\mathbf{bad}\right) - \mathbf{Pr}\left(\mathcal{D}^{G_7} = 1 \mid \mathcal{D}^{G_7} \text{ sets } \neg\mathbf{bad}\right)\right| \leq \frac{2q^2}{2^s}\,.$$

Proof. The proof is similar to the one of Lemma 6.4.3, and we only highlight the details. We adjust game 6 as in the proof of Lemma 6.4.3. In game 7, we expand lines 003 and 026 similarly. As long as **bad** and $\mathbf{bad}'$ are not set in both games, the values (y, h_{nxt}) are generated identically, and no distinguisher can see the difference. As a consequence, it remains to bound the probability that $\mathbf{bad}'$ is set in any of the two games. Straightforward computations now result in the required bound. □

Lemma 6.4.5. *For games G_i ($i = 1, \ldots, 8$), the following holds. Let (V, A) be the graph generated during the execution of the games, and let $(t(V), t(A))$ be the subgraph spanned by the nodes in $t(V)$. Let $(t(V)_j, t(A)_j)$ be the subgraph of $(t(V), t(A))$ after the j^{th} query. Let Δ_j denote the set Δ right after the j^{th} query. Under the condition that in the execution of the game,* **bad** *is not set, the following properties are satisfied for any $j \geq 0$:*

(i) We have $|\mathcal{C}(x) \cap t(V)_j| \leq 1$ for any $x \in \mathbb{Z}_2^s$;
(ii) We have $|t(V)_j| \leq j+1$ and $|t(A)_j| \leq j$. In particular, $t(V)$ is a tree in (V, A);
(iii) We have $|\Delta_j| \leq j$.

only for game 7. First of all, (iii) is satisfied as any query to the R_5 adds at most one element to Δ. The proof of the other properties is done by mathematical induction. Before the first query to R_5 is made, we have $|t(V)_0| = 1$ and $|t(A)_0| = 0$ and the claims are naturally satisfied. Now, assume the claims hold after $j-1$ queries, and consider the j^{th} query. We distinguish between forward and inverse queries.
Inverse query. On input of y, the simulator outputs some value x. This query adds at most 2^{p+d} arcs to the graph, leaving from the vertices in $\mathcal{C}(x)$. As the simulator did not set **bad** in line 104, we have $|\mathcal{C}(x) \cap t(V)_{j-1}| = 0$, and as a consequence, the size of the tree is not increased. The claims now follow by the induction hypothesis.

Forward query. On input of x, the simulator outputs some value y, and this is the only query pair added to $\mathcal{L}_\pi$. This query adds at most 2^{p+d} arcs to the graph, leaving from the vertices in $\mathcal{C}(x)$. If $|\mathcal{C}(x) \cap t(V)_{j-1}| = 0$, the same happens as for inverse queries, and all properties are satisfied by induction. Suppose $\mathcal{C}(x) \cap t(V)_{j-1} \neq \emptyset$. By the induction hypothesis for (i), there is exactly *one* value h in this intersection, which by the properties of L_{in} uniquely defines M such that an arc $h \xrightarrow{M} w$ to $w = \mathsf{L}_{\text{out}}(y, h, M)$ will be added. Thus, this (x, y)-pair defines exactly *one* new arc in the tree starting from a node in $t(V)_{j-1}$. In terms of Figure 6.8, either the **if**-clause of line 008, or the **else**-clause of 018 will be executed. In both cases, as **bad** is not set, y is generated in the same manner, and for simplicity we will analyze the run of the **if**-clause of line 008 only. As **bad** is not set via line 010, the end node w is *no* element of $V_{\text{out}} \cup \mathcal{C}(x)$. In other words, w has no outgoing arc in the updated graph. As a consequence, $t(A)_j = t(A)_{j-1} \cup \{(h, w)\}$. Also as **bad** is not set via line 011, the end node w is not yet in the tree $t(V)_{j-1}$, and will henceforth be newly added. These two observations prove property (ii). Remains to prove that the first property is satisfied. To the contrary, suppose $|\mathcal{C} \cap t(V)_j| = 2$ for some capacity set $\mathcal{C}$. This implies that $h', w \in \mathcal{C}$ for some $h' \in t(V)_{j-1}$, which contradicts with the fact that **bad** is not set via line 011. □

Lemma 6.4.6. *Consider games 2 and 3 (Figure 6.6), and assume that* **bad** *is not set. A query $R_1(x)$, with $x \in \Delta$, never increases $\overline{t}(V)$.*

Proof. First of all, if $l = 1$, lines 015 and 036 never increase Δ, and the claim is naturally satisfied as $\Delta = \emptyset$ throughout. Otherwise, by virtue of the condition in Section 6.3.4, the last block of a padded message always satisfies (6.2). We will show that, if a query $R_1(x)$ for $x \in \Delta$ adds an arc to the tree, the message block M that labels this path satisfies $\mathsf{L}_{\text{in}}(x, M) = x$ or $\mathsf{L}_{\text{in}}(\mathsf{L}_{\text{out}}(x, h', M'), M) = x$ for some (h', M'), therewith proving the claim.

Inverse queries never increase the tree (Lemma 6.4.5), and we will ignore them for simplicity. For readability, we will add a subscripts to the values x and the tree. Denote by $t(V)_j$ the tree in (V, A) after the j^{th} query, for $j = 1, \ldots, q$. Suppose x_j is the j^{th} query to the right oracle, and suppose $x_j \in \Delta$ (notice that x_j occurs exactly once in Δ, due to lines 003 and 028). The node x_j had been added to Δ in either line 015 or 036, in one of the previous queries, say the i^{th} query, for $i < j$. By lines 004 and 029, x_j had been generated such that $\mathcal{C}(x_j) \cap t(V)_{i-1} = \emptyset$. Due to the statement "$\bigcup_{x \in \Delta} \mathcal{C}(x)$" in lines 010, 022 and 033, it is assured that any new node added to the tree in the $(i+1)^{\text{th}}$ up to the $(j-1)^{\text{th}}$ query, is not in $\mathcal{C}(x_j)$. Formally, we have $\mathcal{C}(x_j) \cap \big(t(V)_{j-1} \backslash t(V)_i\big) = \emptyset$. As moreover $\mathcal{C}(x_j) \cap t(V)_{i-1} = \emptyset$, this means that the query x_j to R_1 increases $t(V)$ *only if* $\mathsf{L}_{\text{out}}(y_i, h', M') \in \mathcal{C}(x_j)$ for some h', M' (coming from the i^{th} execution). By the properties of L_{in}, these values uniquely define the message block M that labels the new arc created by the query x_j, namely: M satisfies $\mathsf{L}_{\text{in}}(\mathsf{L}_{\text{out}}(y_i, h', M'), M) = x_j$. However, by construction (lines 006 or 031), we either have $y_i = x_j$ or $y_i = \mathsf{L}_{\text{out}}^{-1}[h', M'](x_j)$.

Consequently, the message block violates the (6.2). Therefore, M cannot be the last block of any padded message, and in particular $\bar{t}(V)$ is not increased. $\square$

6.5 Conclusions

We present some comments on the parazoa hash function design.

Ambiguity. The design as described in Section 6.3 allows for ambiguous interpretations. In particular, it is straightforward to construct schemes that can be described as a parazoa function in different ways:

- Let $\mathcal{P}$ be any s-bit permutation. For any parazoa design with $l = 1$, the same design is described if $\mathsf{L}_{\text{in}}, \mathsf{L}_{\text{out}}$ and L_{ex} are replaced by $\mathsf{L}'_{\text{in}} = \mathsf{L}_{\text{in}} \circ \mathcal{P}^{-1}$, $\mathsf{L}'_{\text{out}} = \mathcal{P} \circ \mathsf{L}_{\text{out}}$ and $\mathsf{L}'_{\text{ex}} = \mathsf{L}_{\text{ex}} \circ \mathcal{P}^{-1}$, and with the initial chaining value redefined as $\mathsf{iv}' = \mathcal{P}(\mathsf{iv})$. Although this modification does not harm the security of the described parazoa design, the obtained indifferentiability bound may differ. In particular, this modification may affect the value d;
- Consider a parazoa design where $\mathsf{L}_{\text{ex}}(h) = \mathsf{left}_p(h)$, and $\mathsf{fin}(P_1, \dots, P_l) = P_1 \| \cdots \| P_l$. Then, the same design is described if these functions are replaced by $\mathsf{L}'_{\text{ex}}(h) = h$ and $\mathsf{fin}'(P_1, \dots, P_l) = \mathsf{left}_p(P_1) \| \cdots \| \mathsf{left}_p(P_l)$. However, our bound becomes meaningless for the second description, whereas this need not be the case for the first description. This paradoxical ambiguity is because the parazoa design allows for *any* type of finalization function, and therefore, we cannot base security of the parazoa function on specific properties of the finalization. For instance, our scheme does not make any distinction between fin' and $\mathsf{fin}''(P_1, \dots, P_l) = \mathsf{left}_{pl}(P_1 \| \cdots \| P_l)$.

In general, different descriptions of a parazoa design may result in different bounds, and the best bound naturally applies.

Generalization of g. It is possible to consider the parazoa hash function design with a more complicated function g, namely to define it as $g(h_{k+i-1}) = (h_{k+i}, P_i)$, where $(x, P_i) \leftarrow \mathsf{M}_{\text{in}}(h_{k+i-1})$, $y \leftarrow \pi(x)$, and $h_{k+i} \leftarrow \mathsf{M}_{\text{out}}(y, h_{k+i-1})$ (cf. Figure 6.9). It is straightforward to generalize the simulator and the proof to this case. The simulator of Figure 6.5 is modified mainly in the lines 003 and 028 (one randomly generates x_{nxt} as input to the next permutation and generates h_{nxt} randomly from $\mathsf{M}_{\text{in}}^{-1}(x_{\text{nxt}}, P_{i+1})$ rather than $\mathsf{L}_{\text{ex}}^{-1}(P_{i+1})$) and in line 006 (one deterministically finds the previous state value h_{prev} and computes y such that $h_{\text{nxt}} = \mathsf{M}_{\text{out}}(y, h_{\text{prev}})$). The proof results in the same bound. For this proof, we would require both M_{in} and M_{out} to be bijections on the state, and additionally that, restricted to the extract P_i, the function M_{in} is balanced.

Generalization to (multiple) different ideal primitives. The description of the parazoa design is based on one permutation π, which is utilized by both f

Forward Query $R_1(x)$

```
000 if $x \in \mathsf{dom}(\mathcal{L}_\pi)$ :
001     return $y = \mathcal{L}_\pi(x)$
002 if $x \in \Delta$ assoc. with some $(i; P_{i+1} \cdots P_l)$ :
003     $h_{\mathrm{nxt}} \xleftarrow{\$} \mathsf{L}_{\mathrm{ex}}^{-1}(P_{i+1}) \backslash (\mathsf{dom}(\mathcal{L}_\pi) \cup \Delta)$
004     if $\mathcal{C}(h_{\mathrm{nxt}}) \cap t(V) \neq \emptyset$ :
005         bad $\leftarrow$ true; GOTO 003
006     $y \leftarrow h_{\mathrm{nxt}}$
007     if $y \in \mathsf{rng}(\mathcal{L}_\pi)$ : bad $\leftarrow$ true; GOTO 003
008     if $\mathcal{C}(x) \cap t(V) \neq \emptyset$ :
009         $\left\{ \begin{array}{l} \text{find unique } h \in t(V) \text{ and } M \\ \text{s.t. } \mathsf{L}_{\mathrm{in}}(h, M) = x \end{array} \right.$
010         if $\mathsf{L}_{\mathrm{out}}(y, h, M) \in V_{\mathrm{out}} \cup \left(\bigcup_{x \in \Delta} \mathcal{C}(x)\right)$
011         or if $\exists\, h' \in t(V) : \mathsf{same}\mathcal{C}(h', \mathsf{L}_{\mathrm{out}}(y, h, M))$ :
012             bad $\leftarrow$ true; GOTO 003
013     end if
014     $\Delta \leftarrow \Delta \backslash \{(x; i, P_{i+1} \cdots P_l)\}$
015     if $i + 1 < l$ : $\Delta \leftarrow \Delta \cup \{(h_{\mathrm{nxt}}; i+1, P_{i+2} \cdots P_l)\}$
016 else if $\mathcal{C}(x) \cap t(V) = \emptyset$ :
017     $y \xleftarrow{\$} \mathbb{Z}_2^s \backslash \mathsf{rng}(\mathcal{L}_\pi)$
018 else if $\mathcal{C}(x) \cap t(V) \neq \emptyset$ :
019     $\left\{ \begin{array}{l} \text{find unique } h \in t(V), \overline{M} \text{ and } M \\ \text{s.t. } \mathsf{L}_{\mathrm{in}}(h, M) = x \text{ and } \mathsf{iv} \xrightarrow{\overline{M}} h \end{array} \right.$
020     if $\overline{M} \| M \notin \mathsf{rng}(\mathsf{pad})$ :
021         $y \xleftarrow{\$} \mathbb{Z}_2^s \backslash \mathsf{rng}(\mathcal{L}_\pi)$
022         if $\mathsf{L}_{\mathrm{out}}(y, h, M) \in V_{\mathrm{out}} \cup \mathcal{C}(x) \cup \left(\bigcup_{x \in \Delta} \mathcal{C}(x)\right)$
023         or if $\exists\, h' \in t(V) : \mathsf{same}\mathcal{C}(h', \mathsf{L}_{\mathrm{out}}(y, h, M))$ :
024             bad $\leftarrow$ true; GOTO 021
025     else if $\overline{M} \| M \in \mathsf{rng}(\mathsf{pad})$ :
026         $h \xleftarrow{\$} L_1(\mathsf{depad}(\overline{M} \| M))$
027         $(P_1, \ldots, P_l) \xleftarrow{\$} \mathsf{fin}^{-1}(h)$
028         $h_{\mathrm{nxt}} \xleftarrow{\$} \mathsf{L}_{\mathrm{ex}}^{-1}(P_1) \backslash (\mathsf{dom}(\mathcal{L}_\pi) \cup \{x\} \cup \Delta)$
029         if $\mathcal{C}(h_{\mathrm{nxt}}) \cap t(V) \neq \emptyset$ :
030             bad $\leftarrow$ true; GOTO 028
031         $y \leftarrow \mathsf{L}_{\mathrm{out}}^{-1}[h, M](h_{\mathrm{nxt}})$
032         if $y \in \mathsf{rng}(\mathcal{L}_\pi)$ : bad $\leftarrow$ true; GOTO 028
033         if $\mathsf{L}_{\mathrm{out}}(y, h, M) \in V_{\mathrm{out}} \cup \mathcal{C}(x) \cup \left(\bigcup_{x \in \Delta} \mathcal{C}(x)\right)$
034         or if $\exists\, h' \in t(V) : \mathsf{same}\mathcal{C}(h', \mathsf{L}_{\mathrm{out}}(y, h, M))$ :
035             bad $\leftarrow$ true; GOTO 028
036         if $1 < l$ : $\Delta \leftarrow \Delta \cup \{(h_{\mathrm{nxt}}; 1, P_2 \cdots P_l)\}$
037     end if
038 end if
039 return $\mathcal{L}_\pi(x) \leftarrow y$
```

Inverse Query $R_1^{-1}(y)$

```
100 if $y \in \mathsf{rng}(\mathcal{L}_\pi)$ :
101     return $x = \mathcal{L}_\pi^{-1}(y)$
102 $x \xleftarrow{\$} \mathbb{Z}_2^s \backslash \mathsf{dom}(\mathcal{L}_\pi)$
103 if $x \in \Delta$ or $\mathcal{C}(x) \cap t(V) \neq \emptyset$ :
104     bad $\leftarrow$ true; GOTO 102
105 return $\mathcal{L}_\pi^{-1}(y) \leftarrow x$
```

Query $L_1(M)$

```
200 if $M \in \mathsf{dom}(\mathcal{H})$ return $h = \mathcal{H}(M)$
201 $h \xleftarrow{\$} \mathbb{Z}_2^n$
202 return $\mathcal{H}(M) \leftarrow h$
```

Query $L_2(M)$

```
300 return $h \xleftarrow{\$} L_1(M)$
```

Query $L_3(M)$

```
400 $(M_1, \ldots, M_k) \leftarrow \mathsf{pad}(M)$
401 $h_0 \leftarrow \mathsf{iv}$
402 for $i = 1, \ldots, k$ :
403     $x \leftarrow \mathsf{L}_{\mathrm{in}}(h_{i-1}, M_i)$
404     $y \leftarrow R_1(x)$
405     $h_i \leftarrow \mathsf{L}_{\mathrm{out}}(y, h_{i-1}, M_i)$
406 end for
407 for $i = 1, \ldots, l$ :
408     $P_i \leftarrow \mathsf{L}_{\mathrm{ex}}(h_{k+i-1})$
409     $h_{k+i} \leftarrow R_1(h_{k+i-1})$
410 end for
411 $h \leftarrow \mathsf{fin}(P_1, \ldots, P_l)$
412 return $h$
```

Figure 6.6. Games 1, 2, and 3. In game 1, the distinguisher has access to $L_1, R_1^{L_1}$ (oracles L_2, L_3 are ignored). In game 2, the distinguisher has access to $L_2^{L_1}, R_1^{L_1}$ (oracle L_3 is ignored). In game 3, the distinguisher has access to $L_3^{R_1^{L_1}}, R_1^{L_1}$ (oracle L_2 is ignored). Oracle L_1 maintains an initially empty table $\mathcal{H}$.

Forward Query $R_\alpha(x)$

```
000 if $x \in \mathsf{dom}(\mathcal{L}_\pi)$ :
001     return $y = \mathcal{L}_\pi(x)$
002 if $x \in \Delta$ assoc. with some $i$ :
    for $\alpha = 3$ :
003     $P_{i+1} \xleftarrow{\$} \mathbb{Z}_2^p$
004     $h_{\text{nxt}} \xleftarrow{\$} \mathsf{L}_{\text{ex}}^{-1}(P_{i+1}) \backslash (\mathsf{dom}(\mathcal{L}_\pi) \cup \Delta)$
    for $\alpha = 4$ :
003     $h_{\text{nxt}} \xleftarrow{\$} \mathbb{Z}_2^s \backslash (\mathsf{dom}(\mathcal{L}_\pi) \cup \Delta)$
004     $P_{i+1} \leftarrow \mathsf{L}_{\text{ex}}(h_{\text{nxt}})$
005     if $\mathcal{C}(h_{\text{nxt}}) \cap t(V) \neq \emptyset$ :
006         bad $\leftarrow$ true; GOTO 004
007     $y \leftarrow h_{\text{nxt}}$
008     if $y \in \mathsf{rng}(\mathcal{L}_\pi)$ : bad $\leftarrow$ true; GOTO 004
009     if $\mathcal{C}(x) \cap t(V) \neq \emptyset$ :
010         $\left\{ \begin{array}{l} \text{find unique } h \in t(V) \text{ and } M \\ \text{s.t. } \mathsf{L}_{\text{in}}(h, M) = x \end{array} \right.$
011         if $\mathsf{L}_{\text{out}}(y, h, M) \in V_{\text{out}} \cup \left(\bigcup_{x \in \Delta} \mathcal{C}(x)\right)$
012         or if $\exists\, h' \in t(V) : \mathsf{same}\mathcal{C}(h', \mathsf{L}_{\text{out}}(y, h, M))$ :
013             bad $\leftarrow$ true; GOTO 004
014     end if
015     $\Delta \leftarrow \Delta \backslash \{(x; i)\}$
016     if $i + 1 < l$ : $\Delta \leftarrow \Delta \cup \{(h_{\text{nxt}}; i + 1)\}$
017 else if $\mathcal{C}(x) \cap t(V) = \emptyset$ :
018     $y \xleftarrow{\$} \mathbb{Z}_2^s \backslash \mathsf{rng}(\mathcal{L}_\pi)$
019 else if $\mathcal{C}(x) \cap t(V) \neq \emptyset$ :
020     $\left\{ \begin{array}{l} \text{find unique } h \in t(V), \bar{M} \text{ and } M \\ \text{s.t. } \mathsf{L}_{\text{in}}(h, M) = x \text{ and iv} \xrightarrow{\bar{M}} h \end{array} \right.$
021     if $\bar{M} \| M \notin \mathsf{rng}(\mathsf{pad})$ :
022         $y \xleftarrow{\$} \mathbb{Z}_2^s \backslash \mathsf{rng}(\mathcal{L}_\pi)$
023         if $\mathsf{L}_{\text{out}}(y, h, M) \in V_{\text{out}} \cup \mathcal{C}(x) \cup \left(\bigcup_{x \in \Delta} \mathcal{C}(x)\right)$
024         or if $\exists\, h' \in t(V) : \mathsf{same}\mathcal{C}(h', \mathsf{L}_{\text{out}}(y, h, M))$ :
025             bad $\leftarrow$ true; GOTO 022
026     else if $\bar{M} \| M \in \mathsf{rng}(\mathsf{pad})$ :
    for $\alpha = 3$ :
027         $P_1 \xleftarrow{\$} \mathbb{Z}_2^p$
028         $h_{\text{nxt}} \xleftarrow{\$} \mathsf{L}_{\text{ex}}^{-1}(P_1) \backslash (\mathsf{dom}(\mathcal{L}_\pi) \cup \{x\} \cup \Delta)$
    for $\alpha = 4$ :
027         $h_{\text{nxt}} \xleftarrow{\$} \mathbb{Z}_2^s \backslash (\mathsf{dom}(\mathcal{L}_\pi) \cup \{x\} \cup \Delta)$
028         $P_1 \leftarrow \mathsf{L}_{\text{ex}}(h_{\text{nxt}})$
029         if $\mathcal{C}(h_{\text{nxt}}) \cap t(V) \neq \emptyset$ :
030             bad $\leftarrow$ true; GOTO 028
031         $y \leftarrow \mathsf{L}_{\text{out}}^{-1}[h, M](h_{\text{nxt}})$
032         if $y \in \mathsf{rng}(\mathcal{L}_\pi)$ : bad $\leftarrow$ true; GOTO 028
033         if $\mathsf{L}_{\text{out}}(y, h, M) \in V_{\text{out}} \cup \mathcal{C}(x) \cup \left(\bigcup_{x \in \Delta} \mathcal{C}(x)\right)$
034         or if $\exists\, h' \in t(V) : \mathsf{same}\mathcal{C}(h', \mathsf{L}_{\text{out}}(y, h, M))$ :
035             bad $\leftarrow$ true; GOTO 028
036         if $1 < l$ : $\Delta \leftarrow \Delta \cup \{(h_{\text{nxt}}; 1)\}$
037     end if
038 end if
039 return $\mathcal{L}_\pi(x) \leftarrow y$
```

Inverse Query $R_\alpha^{-1}(y)$

```
100 if $y \in \mathsf{rng}(\mathcal{L}_\pi)$ :
101     return $x = \mathcal{L}_\pi^{-1}(y)$
102 $x \xleftarrow{\$} \mathbb{Z}_2^s \backslash \mathsf{dom}(\mathcal{L}_\pi)$
103 if $x \in \Delta$ or $\mathcal{C}(x) \cap t(V) \neq \emptyset$ :
104     bad $\leftarrow$ true; GOTO 102
105 return $\mathcal{L}_\pi^{-1}(y) \leftarrow x$
```

Query $L_3(M)$

```
400 $(M_1, \ldots, M_k) \leftarrow \mathsf{pad}(M)$
401 $h_0 \leftarrow \mathsf{iv}$
402 for $i = 1, \ldots, k$ :
403     $x \leftarrow \mathsf{L}_{\text{in}}(h_{i-1}, M_i)$
404     $y \leftarrow R_\alpha(x)$
405     $h_i \leftarrow \mathsf{L}_{\text{out}}(y, h_{i-1}, M_i)$
406 end for
407 for $i = 1, \ldots, l$ :
408     $P_i \leftarrow \mathsf{L}_{\text{ex}}(h_{k+i-1})$
409     $h_{k+i} \leftarrow R_\alpha(h_{k+i-1})$
410 end for
411 $h \leftarrow \mathsf{fin}(P_1, \ldots, P_l)$
412 return $h$
```

Figure 6.7. Game 5 (for $\alpha = 3$) and game 6 (for $\alpha = 4$). In both games, the distinguisher has access to $L_3^{R_\alpha}, R_\alpha$.

Forward Query $R_\alpha(x)$

```
000 if $x \in \mathsf{dom}(\mathcal{L}_\pi)$ :
001     return $y = \mathcal{L}_\pi(x)$
002 if $x \in \Delta$ assoc. with some $i$ :
003     $y \xleftarrow{\$} \mathbb{Z}_2^s \backslash \mathsf{rng}(\mathcal{L}_\pi)$
004     $h_{\text{nxt}} \leftarrow y$
005     if $h_{\text{nxt}} \in \mathsf{dom}(\mathcal{L}_\pi) \cup \Delta$
006     or if $\mathcal{C}(h_{\text{nxt}}) \cap t(V) \neq \emptyset$ :
007         bad ← true; $\boxed{\textbf{GOTO } 003}$
008     if $\mathcal{C}(x) \cap t(V) \neq \emptyset$ :
009         $\left\{ \begin{array}{l} \text{find unique } h \in t(V) \text{ and } M \\ \text{s.t. } \mathsf{L}_{\text{in}}(h, M) = x \end{array} \right.$
010         if $\mathsf{L}_{\text{out}}(y, h, M) \in V_{\text{out}} \cup \left( \bigcup_{x \in \Delta} \mathcal{C}(x) \right)$
011         or if $\exists\, h' \in t(V) : \mathsf{same}\mathcal{C}(h', \mathsf{L}_{\text{out}}(y, h, M))$ :
012             bad ← true; $\boxed{\textbf{GOTO } 003}$
013     end if
014     $\Delta \leftarrow \Delta \backslash \{(x; i)\}$
015     if $i + 1 < l$ :  $\Delta \leftarrow \Delta \cup \{(h_{\text{nxt}}; i + 1)\}$
016 else if $\mathcal{C}(x) \cap t(V) = \emptyset$ :
017     $y \xleftarrow{\$} \mathbb{Z}_2^s \backslash \mathsf{rng}(\mathcal{L}_\pi)$
018 else if $\mathcal{C}(x) \cap t(V) \neq \emptyset$ :
019     $\left\{ \begin{array}{l} \text{find unique } h \in t(V), \bar{M} \text{ and } M \\ \text{s.t. } \mathsf{L}_{\text{in}}(h, M) = x \text{ and } \mathsf{iv} \xrightarrow{\bar{M}} h \end{array} \right.$
020     if $\bar{M} \| M \notin \mathsf{rng}(\mathsf{pad})$ :
021         $y \xleftarrow{\$} \mathbb{Z}_2^s \backslash \mathsf{rng}(\mathcal{L}_\pi)$
022         if $\mathsf{L}_{\text{out}}(y, h, M) \in V_{\text{out}} \cup \mathcal{C}(x) \cup \left( \bigcup_{x \in \Delta} \mathcal{C}(x) \right)$
023         or if $\exists\, h' \in t(V) : \mathsf{same}\mathcal{C}(h', \mathsf{L}_{\text{out}}(y, h, M))$ :
024             bad ← true; $\boxed{\textbf{GOTO } 021}$
025     else if $\bar{M} \| M \in \mathsf{rng}(\mathsf{pad})$ :
026         $y \xleftarrow{\$} \mathbb{Z}_2^s \backslash \mathsf{rng}(\mathcal{L}_\pi)$
027         $h_{\text{nxt}} \leftarrow \mathsf{L}_{\text{out}}(y, h, M)$
028         if $h_{\text{nxt}} \in \mathsf{dom}(\mathcal{L}_\pi) \cup \{x\} \cup \Delta$
029         or if $\mathcal{C}(h_{\text{nxt}}) \cap t(V) \neq \emptyset$ :
030             bad ← true; $\boxed{\textbf{GOTO } 026}$
031         if $\mathsf{L}_{\text{out}}(y, h, M) \in V_{\text{out}} \cup \mathcal{C}(x) \cup \left( \bigcup_{x \in \Delta} \mathcal{C}(x) \right)$
032         or if $\exists\, h' \in t(V) : \mathsf{same}\mathcal{C}(h', \mathsf{L}_{\text{out}}(y, h, M))$ :
033             bad ← true; $\boxed{\textbf{GOTO } 026}$
034         if $1 < l$ :  $\Delta \leftarrow \Delta \cup \{(h_{\text{nxt}}; 1)\}$
035     end if
036 end if
037 return $\mathcal{L}_\pi(x) \leftarrow y$
```

Inverse Query $R_\alpha^{-1}(y)$

```
100 if $y \in \mathsf{rng}(\mathcal{L}_\pi)$ :
101     return $x = \mathcal{L}_\pi^{-1}(y)$
102 $x \xleftarrow{\$} \mathbb{Z}_2^s \backslash \mathsf{dom}(\mathcal{L}_\pi)$
103 if $x \in \Delta$ or $\mathcal{C}(x) \cap t(V) \neq \emptyset$ :
104     bad ← true; $\boxed{\textbf{GOTO } 102}$
105 return $\mathcal{L}_\pi^{-1}(y) \leftarrow x$
```

Query $L_3(M)$

```
400 $(M_1, \ldots, M_k) \leftarrow \mathsf{pad}(M)$
401 $h_0 \leftarrow \mathsf{iv}$
402 for $i = 1, \ldots, k$ :
403     $x \leftarrow \mathsf{L}_{\text{in}}(h_{i-1}, M_i)$
404     $y \leftarrow R_\alpha(x)$
405     $h_i \leftarrow \mathsf{L}_{\text{out}}(y, h_{i-1}, M_i)$
406 end for
407 for $i = 1, \ldots, l$ :
408     $P_i \leftarrow \mathsf{L}_{\text{ex}}(h_{k+i-1})$
409     $h_{k+i} \leftarrow R_\alpha(h_{k+i-1})$
410 end for
411 $h \leftarrow \mathsf{fin}(P_1, \ldots, P_l)$
412 return $h$
```

Figure 6.8. Game 7 (for $\alpha = 5$, including the boxed statements) and game 8 (for $\alpha = 6$, with the boxed statements removed). In both games, the distinguisher has access to $L_3^{R_\alpha}, R_\alpha$.

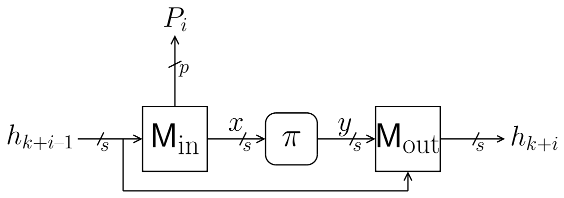


Figure 6.9. A generalized variant of the extraction function g.

and g. We notice that in the design and the proof, π can easily be replaced by a random function. Also, it is straightforward to consider two different permutations π_1, π_2 (e.g., f is build on π_1 and g on π_2). We notice that, in some cases, the usage of different permutations may improve the indifferentiability bound. For instance, if $l = 1$, and the last execution of f is defined to use a different permutation, the variable d (cf. Section 6.4) becomes superfluous.

Remarks on the applications. In the introduction of this chapter, we sketched several applications of parazoa functions, and we will briefly elaborate on this. The original sponge follows the parazoa design with $m = p$; if we relax this requirement to allow for $p \neq m$, the indifferentiability bound results in $O((Kq)^2/2^{s-\max\{p,m\}})$ ($d = \max\{m-p, 0\}$), where s is the state size of the sponge. The indifferentiability bounds for the lightweight hash functions Quark [24], PHOTON [55], and SPONGENT [106] follow directly. The Grindahl hash function (simplified, with zero final blank rounds) satisfies the parazoa function design with $d = m$, which is caused by the fact that L_{in} inserts the message in the leftmost part, while L_{ex} outputs the rightmost part of the state. With respect to NIST's SHA-3 hash function competition, we can consider the following second round candidates. CubeHash consists of a permutation P executed 16 times iteratively, and the result holds if P^{16} is assumed to be a random permutation. In its final transformation, one bit in the chaining is swapped, which results in $d = 1$. Hamsi [135] employs two different permutations, one of which is exclusively used in the last compression function. We note that, even though the permutation and compression function input sizes of Hamsi differ, the design can be described as a parazoa function: instead of chopping half of the state at the end of a compression function evaluation f, and concatenating the remaining state with the expanded message in the subsequent call to f, one can just as well leave the state unchopped, and overwrite the redundant part of the state with the expanded message in the next evaluation of f. A simplified version of Hamsi, where each compression function employs *one* single permutation, would be insecure as the attack of Section 6.2 would apply. However, it is fair to believe that for the original Hamsi a better bound can be obtained [134]. The Fugue design can be considered to have a final compression function execution based on a permutation π' differing from the permutation π used in the iteration (for instance, for Fugue-256 we have

$\pi = (\mathbf{SMIX} \circ \mathbf{CMIX} \circ \mathbf{ROR3})^2$ and $\pi' = \mathbf{G}$ [107]). Still, the parazoa design can handle Fugue, and the results carry over. The Luffa compression function consists of a linear function operating on the state, and multiple smaller permutations executed in parallel. The results hold under the assumption that this parallel execution behaves like *one* random permutation. (For Luffa-384 and Luffa-512, which satisfy the parazoa design with $l = 2$, (6.2) says that the last block of a padded message should satisfy $\mathsf{L}_{\text{in}}(x, M) \neq x$ for all $x \in \mathbb{Z}_2^s$. However, the last block of a padded message of Luffa-384 or Luffa-512 is a zero-block, and for both versions of Luffa we have $\mathsf{L}_{\text{in}}(0^s, 0^m) = 0^s$. It is straightforward to see that indifferentiability can be re-obtained by artificially assuring that $t(V) \cap \mathcal{C}(0^s) = \emptyset$ throughout (for instance, this is satisfied if the simulator defines a query pair $(0^s, y)$ for randomly chosen $y \in \mathbb{Z}_2^s$ in its initialization phase).) For both Fugue and Luffa, our result has been confirmed by Bhattacharyya and Mandal [43].

7 NIST's SHA-3 Hash Function Competition

In 2004, a series of differential attacks by Wang et al. [220, 221] have exposed security vulnerabilities in the design of the most widely adopted and deployed SHA-1 hash function. As a result, in 2007 the US National Institute for Standards and Technology (NIST) recommended the replacement of SHA-1 by the SHA-2 hash function family and announced a call for the design of a new SHA-3 hashing algorithm. The call prescribed that SHA-3 must allow for message digests of length $224, 256, 384$, and 512 bits, it should be efficient, and it should provide an adequate level of security. NIST received 64 submissions, 51 of which advanced to the first round of the competition. In the second round, 14 candidate hash functions were considered, and five candidates of them advanced to the third and final round: BLAKE [25], Grøstl [104], JH [224], Keccak [40], and Skein [93]. On October 2nd, 2012, NIST announced that Keccak has been selected as the winner of the SHA-3 competition. Keccak is about to be standardized and may (or may not) become the most prominent hash function in the next decennia. Below we recall the security requirements by NIST in their call for the SHA-3 hash function.

NIST Security Requirements

The SHA-3 hash function is required to satisfy the following security requirements [180]: `(i)` at least one variant of the hash function must securely support HMAC [30] and randomized hashing [109]. Next, for all n-bit digest values, the hash function must provide `(ii)` preimage resistance of approximately n bits, `(iii)` second preimage resistance of approximately $n - L$ bits, where the first preimage is of length at most 2^L blocks, `(iv)` collision resistance of approximately $n/2$ bits, and `(v)` it must be resistant to the length extension attack. Finally, `(vi)` for any $m \leq n$, the hash function specified by taking a fixed subset of m bits of the function's output is required to satisfy properties `(ii)-(v)` with n replaced by m.

Notation

Throughout this chapter, we use a unified notation for all candidates. The value n denotes the output size of the hash function, l the size of the chaining value, and m the number of message bits compressed in one iteration of the compression function. A padded message is always parsed as a sequence of $k \geq 1$ message blocks of length m bits: $(M_1, \ldots, M_k)$.

Contributions of This Chapter

Several classifications of the SHA-3 candidates appeared in literature [92, 97, 133, 216], but the focus in those works is on performance. In this chapter, we provide a detailed survey of the five SHA-3 finalists, in which we compare their security reductions. We consider preimage, second preimage, and collision resistance (security requirements `(ii)`-`(iv)`) for the $n = 256$ and $n = 512$ variants. We also include their indifferentiability security results. The security analysis is realized in the ideal model, where one or more of the underlying integral building blocks (e.g., the underlying block cipher or permutation(s)) are assumed to be ideal, i.e., random primitives.

Observe that for the five SHA-3 finalists, the algorithms that output 224 or 384 bits are the same as the 256- or 512-bit algorithms with an additional chopping at the end. As for requirement `(i)`, all designers claim that their proposal can safely be used in HMAC mode [30] or for randomized hashing [109], and for requirement `(v)`, we note that all designs are secure against the length extension attack.

In an attempt to complete "open gaps" in the classification, in this chapter we prove several results of the five finalists:

BLAKE. We include a full security analysis of BLAKE in the ideal cipher model in Section 7.1 and give proofs of optimal collision, preimage, and second preimage resistance;

Grøstl. We present an indifferentiability analysis of Grøstl in the ideal permutation model in Section 7.2, as well as an optimal second preimage security proof;

JH. We present a (second) preimage security proof of JH, which is optimal for $n = 256$;

Skein. We present an optimal second preimage security proof of Skein.

We refer to Table 7.1 for the summary of all provable security results of the SHA-3 finalists. It shows that all candidates, with the exception of the (second) preimage security of JH-512, achieve optimal collision, preimage, and second preimage security for both their 256 and 512 variants. The optimal results refer to the general iterative structure of all the algorithms. The analysis in all cases is performed in the ideal setting. More importantly, we claim that the provided comparison is sufficiently fair due to the fact that the ideality assumption is hypothesized on basic underlying primitives, such as block ciphers and permutations, as opposed to higher level compression function building blocks.

On the other hand, while optimality results hold for the five hash function finalists, the security of their compression functions again in the ideal model differs. The security here varies from trivially insecure compression functions for JH and Keccak to optimally secure ones for BLAKE, Grøstl, and Skein. We want to note that the latter remark does not reflect any security criteria indicated in the security requirements of NIST. In addition to the classical notions of collision, preimage, and second preimage security, we also investigate the notion of indifferentiability. We include the indifferentiability notion not only because it is relevant by itself, but it is also an important tool to derive further security results: it encompasses structural attacks, such as the length extension attack in single round interactive protocols [157, 192]. JH and Skein offer 256-bit indifferentiability security for both their variants, and BLAKE and Grøstl offer 128-bit and 256-bit security for their respective 256 and 512 variants. Keccak provides higher indifferentiability guarantees: 256-bit and 512-bit, respectively, and that is achieved by increasing the iterated state size to 1600 bits as compared to sizes from 256 bits to 1024 bits for the other hash function candidates.

Outline

In Sections 7.1-7.5, we consider the five SHA-3 finalists from a provable security point of view. We give a high level algorithmic description of each hash function, and discuss the existing and new security results. All results are summarized in detail in Table 7.2. The chapter is concluded in Section 7.6, where we discuss the provable security results.

Bibliographic Notes

The first classification of SHA-3 candidates based on the by NIST specified security criteria has been published by Andreeva, Mennink, and Preneel at the Information Security Conference 2010 [12]. This classification consisted of a summary of the state of the art provable security results (regarding requirements `(ii)-(iv)`) of all 14 second round candidates. In the International Journal of Information Security [3], Andreeva, Bogdanov, Mennink, Preneel, and Rechberger extended this result with a classification of security against differential attacks [47]. Later, during the third and final round of the competition, Andreeva, Mennink, Preneel, and Škrobot reconsidered this classification for the five finalists, and filled the "open gaps" (results in Sections 7.2, 7.3, and 7.5 of this chapter). This work was published at AFRICACRYPT 2012 [16]. The classification as given in this chapter is mostly taken from [16].

Section 7.1 furthermore contains results published by Andreeva, Luykx, and Mennink at Selected Areas in Cryptography 2012 [6], and Section 7.2 contains results published by Andreeva, Mennink, and Preneel at Security and Cryptography for Networks [9].

Parts of these results have also been presented at NIST's 2nd SHA-3 Candidate Conference 2010 [10,13], NIST's 3rd SHA-3 Candidate Conference 2012 [7,17], and Bulgarian Cryptography Days 2012 [18].

Table 7.1. Security guarantees of the SHA-3 finalists. Here, l and m denote the chaining value and the message input sizes, respectively. The next four columns of both tables correspond to the preimage, second preimage, collision, and indifferentiability security in bits. Regarding second preimage resistance, the first preimage is of length 2^L blocks. The results in **bold** are derived in this chapter. For a more detailed summary we refer to Table 7.2.

	l	m	epre	esec	col	iff	assumption
BLAKE-256	256	512	**256**	**256**	**128**	**128**	E ideal
Grøstl-256	512	512	256	**256–L**	128	**128**	π, ρ ideal
JH-256	1024	512	**256**	**256**	128	256	π ideal
Keccak-256	1600	1088	256	256	128	256	π ideal
Skein-256	512	512	256	**256**	128	256	E ideal
NIST's requirements			256	256–L	128	—	—

	l	m	epre	esec	col	iff	assumption
BLAKE-512	512	1024	**512**	**512**	**256**	**256**	E ideal
Grøstl-512	1024	1024	512	**512–L**	256	**256**	π, ρ ideal
JH-512	1024	512	**256**	**256**	256	256	π ideal
Keccak-512	1600	576	512	512	256	512	π ideal
Skein-512	512	512	512	**512**	256	256	E ideal
NIST's requirements			512	512–L	256	—	—

7.1 BLAKE

The BLAKE hash function [25] is a HAIFA construction (Section 2.5.1). The message blocks are accompanied with a HAIFA-counter, and the function employs a suffix- and prefix-free padding rule. The underlying compression function f is based on a block cipher $E : \mathbb{Z}_2^{2l} \times \mathbb{Z}_2^{m} \to \mathbb{Z}_2^{2l}$. It moreover employs an injective linear function L, and a linear function L' that XORs the first and second halves

BLAKE:
$(n,l,m,s,t) \in \{(256,256,512,128,64), (512,512,1024,256,128)\}$
$E : \mathbb{Z}_2^{2l} \times \mathbb{Z}_2^m \to \mathbb{Z}_2^{2l}$ block cipher
$L : \mathbb{Z}_2^{l+s+t} \to \mathbb{Z}_2^{2l}$, $L' : \mathbb{Z}_2^{2l} \to \mathbb{Z}_2^l$ linear functions
$f(h,S,M,T) = L'(E_M(L(h,S,T))) \oplus h \oplus [S]_2$

```
BLAKE($M$)
    $(M_1,\ldots,M_k) \leftarrow \mathsf{pad}_b(M),\ h_0 \leftarrow \mathsf{iv}$
    $S \in \mathbb{Z}_2^s,\ (T_i)_{i=1}^k$ HAIFA-counter
    $h_i \leftarrow f(h_{i-1},S,M_i,T_i)$ for $i = 1,\ldots,k$
    return $h_k$
```

Grøstl:
$(n,l,m) \in \{(256,512,512),(512,1024,1024)\}$
$\pi, \rho : \mathbb{Z}_2^l \to \mathbb{Z}_2^l$ permutations
$f(h,M) = \pi(h \oplus M) \oplus \rho(M) \oplus h$
$g(h) = \pi(h) \oplus h$

```
Grøstl($M$)
    $(M_1,\ldots,M_k) \leftarrow \mathsf{pad}_g(M),\ h_0 \leftarrow \mathsf{iv}$
    $h_i \leftarrow f(h_{i-1},M_i)$ for $i = 1,\ldots,k$
    $h \leftarrow \mathsf{left}_n(g(h_k))$
    return $h$
```

JH:
$(n,l,m) \in \{(256,1024,512),(512,1024,512)\}$
$\pi : \mathbb{Z}_2^l \to \mathbb{Z}_2^l$ permutation
$f(h,M) = \pi(h \oplus (0^{l-m}\|M)) \oplus (M\|0^{l-m})$

```
JH($M$)
    $(M_1,\ldots,M_k) \leftarrow \mathsf{pad}_j(M),\ h_0 \leftarrow \mathsf{iv}$
    $h_i \leftarrow f(h_{i-1},M_i)$ for $i = 1,\ldots,k$
    $h \leftarrow \mathsf{left}_n(h_k)$
    return $h$
```

Keccak:
$(n,l,m) \in \{(256,1600,1088),(512,1600,576)\}$
$\pi : \mathbb{Z}_2^l \to \mathbb{Z}_2^l$ permutation
$f(h,M) = \pi(h \oplus (M\|0^{l-m}))$

```
Keccak($M$)
    $(M_1,\ldots,M_k) \leftarrow \mathsf{pad}_k(M),\ h_0 \leftarrow \mathsf{iv}$
    $h_i \leftarrow f(h_{i-1},M_i)$ for $i = 1,\ldots,k$
    $h \leftarrow \mathsf{left}_n(h_k)$
    return $h$
```

Skein:
$(n,l,m) \in \{(256,512,512),(512,512,512)\}$
$E : \mathbb{Z}_2^m \times \mathbb{Z}_2^{128} \times \mathbb{Z}_2^l \to \mathbb{Z}_2^m$ tweakable block cipher
$f(h,T,M) = E_{h,T}(M) \oplus M$

```
Skein($M$)
    $(M_1,\ldots,M_k) \leftarrow \mathsf{pad}_s(M),\ h_0 \leftarrow \mathsf{iv}$
    $(T_i)_{i=1}^k$ round-specific tweaks
    $h_i \leftarrow f(h_{i-1},T_i,M_i)$ for $i = 1,\ldots,k$
    $h \leftarrow \mathsf{left}_n(h_k)$
    return $h$
```

Padding functions:
$\mathsf{pad}_b(M) = M\|10^{-|M|-t-2 \bmod m}1\|\langle|M|\rangle_t$
$\mathsf{pad}_g(M) = M\|10^{-|M|-65 \bmod l}\|\langle\lceil\frac{|M|+65}{l}\rceil\rangle_{64}$
$\mathsf{pad}_j(M) = M\|10^{383+(-|M| \bmod m)}\|\langle|M|\rangle_{128}$
$\mathsf{pad}_k(M) = M\|10^{-|M|-2 \bmod m}1$
$\mathsf{pad}_s(M) = M'\|0^{(-|M'| \bmod m)+m}$
where $M' = \begin{cases} M \text{ if } |M| \equiv 0 \bmod 8 \\ M\|10^{-|M|-1 \bmod 8} \text{ otherwise} \end{cases}$

Figure 7.1. Algorithmic descriptions of the five SHA-3 finalist hash functions. iv denotes an initialization vector, h denotes state values, M denotes message blocks, S denotes a (fixed) salt, T denotes a counter or tweak. The notations for BLAKE are explained in Section 7.1. The padding rules of BLAKE and Skein are additionally defined by a counter or tweak (see Sections 7.1 and 7.5).

of the input. By $[x]_2 = x\|x$ we denote the concatenation of two copies of x. For ease of presentation, in the technical part of this section we regularly write l, m, s, and t in terms of n, and say $E \in \mathrm{Bloc}(2n)$. The BLAKE hash function design is given in Figure 7.1. A diagram of the compression function f is displayed in Figure 7.2 and an algorithmic description follows, where $C \in \mathbb{Z}_2^n$ is a constant.

```
$f(h_{i-1},S,M_i,T_i)$
    $v_i \leftarrow (h_{i-1}\|S\|[T_i^l]_2\|[T_i^r]_2) \oplus (0^n\|C)$,
    $w_i \leftarrow E(M_i,v_i)$,
    $h_i \leftarrow w_i^l \oplus w_i^r \oplus h_{i-1} \oplus [S]_2$,
    return $h_i$.
```

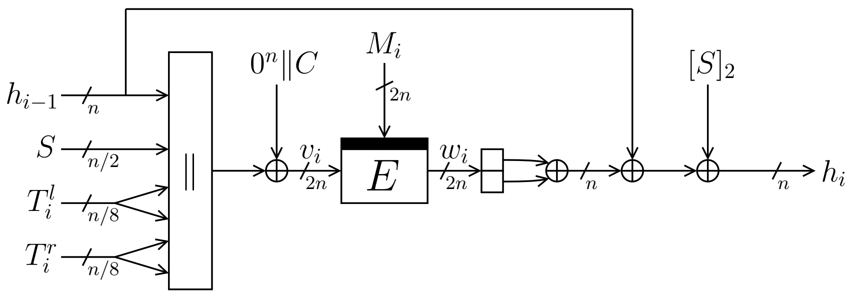


Figure 7.2. The BLAKE compression function f.

Regarding the security definitions of Section 2.3.1, if $\mathcal{C} = \mathcal{H}$ the domain points are parsed as $y = (S, M) \in \mathbb{Z}_2^s \times \mathbb{Z}_2^\infty$, and for the second preimage notion we require λ to be of length at least $n/2$ bits. In case $\mathcal{C}$ denotes the BLAKE compression function f, its domain points are of the form $y = (h, S, M, T) \in \mathbb{Z}_2^l \times \mathbb{Z}_2^s \times \mathbb{Z}_2^m \times \mathbb{Z}_2^t$. Note that in all definitions $\mathcal{A}$ can freely choose the salt, e.g., a collision with different salts is counted valid. Our results directly apply to the setting of a fixed salt. These queries are stored in a query history $\mathcal{Q}$ as elements of the form (M_j, v_j, w_j), where j is the query index, M_j is the key input to the block cipher (note that for BLAKE the message input to f is the key input to E), and v_j and w_j denote the plaintext and ciphertext, respectively. Associated to query (M_j, v_j, w_j) we define the value $x_j = w_j^l \oplus w_j^r \oplus h_j \oplus [S_j]_2$ as the output of the compression function f, where we parse $h_j \| S_j \| T_j^{(1)} \| T_j^{(2)} \| T_j^{(3)} \| T_j^{(4)} \leftarrow v_j \oplus (0^n \| C)$.

Compression Function Security

As the mode of operation of BLAKE is based on the HAIFA structure, all security properties regarding this type (cf. Section 2.4) hold [46], provided the compression function is assumed to be ideal. We show, however, that the BLAKE compression function shows non-random behavior: it is differentiable from a random compression function in about $2^{n/4}$ queries, making the above-mentioned security properties invalid. This result was independently shown by Chang et al. [69].

We consider the indifferentiability of the BLAKE compression function f from a random oracle $\mathcal{R}$ (with the same domain and range as f), when the underlying block cipher E is sampled uniformly at random, $E \xleftarrow{\$} \text{Bloc}(2n)$. In more detail, we construct a distinguisher $\mathcal{D}$, such that for any simulator $\mathcal{S}$, $\mathcal{D}$ differentiates (f^E, E) from $(\mathcal{R}, \mathcal{S}^\mathcal{R})$ in about $2^{n/4}$ queries, hence significantly faster than expected.

The differentiability attack considers fixed-points for f, by which we in this case mean values (h, S, M, T) such that $f(h, S, M, T) = h \oplus [S]_2$. The presence of fixed-points of these form have already been pointed out in [25, Section 5.2.4]. However, not every block cipher evaluation renders a valid compression function evaluation, due to the duplication of T in the block cipher input. The simulator may be able to take advantage of this, which makes the attack proof more complicated.

Theorem 7.1.1. *Let* $E \xleftarrow{\$} \mathrm{Bloc}(2n)$*, and let* $\mathcal{R} : \mathbb{Z}_2^{n+n/2+2n+n/4} \to \mathbb{Z}_2^n$ *be a random compression function. For any simulator* $\mathcal{S}$ *that makes at most* $q_\mathcal{S}$ *queries to* $\mathcal{R}$*, there exists a distinguisher* $\mathcal{D}$ *that makes at most* $2^{n/4}+1$ *queries to its oracles, such that*

$$\mathbf{Adv}^{\mathrm{iff}}_{f^E,\mathcal{S}}(\mathcal{D}) \geq 1 - e^{-1} - \frac{q_\mathcal{S}}{2^n}\,.$$

Proof. Let $\mathcal{S}$ be any simulator making at most $q_\mathcal{S}$ queries to $\mathcal{R}$. We construct a distinguisher $\mathcal{D}$ that differentiates (f^E, E) from $(\mathcal{R}, \mathcal{S}^\mathcal{R})$ with a significant probability. $\mathcal{D}$ has query access to (L, R, R^{-1}) and operates as follows.

(i) $\mathcal{D}$ picks $2^{n/4}$ distinct messages M_j, queries $v_j \leftarrow R^{-1}(M_j, 0)$, and parses $h_j \| S_j \| T_j^{(1)} \| T_j^{(2)} \| T_j^{(3)} \| T_j^{(4)} \leftarrow v_j \oplus (0^n \| C)$;

(ii) If $T_j^{(1)} \| T_j^{(3)} \neq T_j^{(2)} \| T_j^{(4)}$ for all $j \in \{1, \ldots, 2^{n/4}\}$, $\mathcal{D}$ guesses $(L, R) = (\mathcal{R}, \mathcal{S})$ and halts;

(iii) Let $j \in \{1, \ldots, 2^{n/4}\}$ be such that $T_j^{(1)} \| T_j^{(3)} = T_j^{(2)} \| T_j^{(4)}$. $\mathcal{D}$ queries

$$h \leftarrow L(h_j, S_j, M_j, T_j^{(1)} \| T_j^{(3)})\,,$$

and guesses $(L, R) = (f, E)$ if and only if $h = h_j \oplus [S_j]_2$.

The distinguisher guesses its oracles correctly *except* if one of the following events occur:

$$\begin{aligned}
&\mathsf{E}_1: \quad \forall\, j,\ T_j^{(1)} \| T_j^{(3)} \neq T_j^{(2)} \| T_j^{(4)} && \big|\ (L, R) = (f, E)\,;\\
&\mathsf{E}_2: \quad \exists\, j,\ T_j^{(1)} \| T_j^{(3)} = T_j^{(2)} \| T_j^{(4)} \text{ and } h = h_j \oplus [S_j]_2 && \big|\ (L, R) = (\mathcal{R}, \mathcal{S})\,.
\end{aligned}$$

In particular, $\mathbf{Adv}^{\mathrm{iff}}_f(\mathcal{D}) \geq 1 - \mathbf{Pr}(\mathsf{E}_1) - \mathbf{Pr}(\mathsf{E}_2)$. We start with $\mathbf{Pr}(\mathsf{E}_2)$, and we suppose $(L, R) = (\mathcal{R}, \mathcal{S})$. E_2 in fact covers the event that $\mathcal{S}$ finds a fixed-point for $\mathcal{R}$, namely inputs h_j, S_j, M_j, T_j such that $\mathcal{R}(h_j, S_j, M_j, T_j) = h_j \oplus [S_j]_2$. As $\mathcal{S}$ makes at most $q_\mathcal{S}$ queries, it can find such fixed-point with probability at most $q_\mathcal{S}/2^n$. Next, we consider $\mathbf{Pr}(\mathsf{E}_1)$, and we suppose $(L, R) = (f, E)$. As E is a random block cipher and the message blocks M_j are all different, the probabilities $\mathbf{Pr}\left(T_j^{(1)} \| T_j^{(3)} \neq T_j^{(2)} \| T_j^{(4)}\right)$ are independent for different indices j, and satisfy

$$\mathbf{Pr}\left(T_j^{(1)} \| T_j^{(3)} \neq T_j^{(2)} \| T_j^{(4)}\right) = 1 - \mathbf{Pr}\left(T_j^{(1)} \| T_j^{(3)} = T_j^{(2)} \| T_j^{(4)}\right) = 1 - 1/2^{n/4}\,.$$

Therefore, $\mathbf{Pr}(\mathsf{E}_1) = \left(1 - 1/2^{n/4}\right)^{2^{n/4}} \leq e^{-1}$. We thus obtain $\mathbf{Adv}^{\mathrm{iff}}_f(\mathcal{D}) \geq 1 - e^{-1} - q_\mathcal{S}/2^n$. □

The BLAKE compression function shows similarities with the PGV5, or Davies-Meyer, compression function, but the results of [52, 53, 210] (see Section 2.6.1) do not directly carry over. We prove that optimal collision and preimage resistance is achieved. In the ideal model, everywhere second preimage resistance of the compression function can be proven similar as the preimage resistance, up to a constant (the security analysis differs only in that we give the adversary one query for free).

Theorem 7.1.2. *Let $n \in \mathbb{N}$. The advantage of any adversary $\mathcal{A}$ in finding a collision for f after $q < 2^{2n-1}$ queries can be upper bounded by*

$$\mathbf{Adv}^{\mathrm{col}}_{f^E}(q) \leq \frac{q(q+1)}{2^n}\,.$$

Proof. Let $j = 1, \ldots, q$. We consider the probability that the j^{th} query results in a collision. We distinguish between *forward* and *inverse* queries.

Collision by a forward query. The adversary makes an encryption query of the form (M_j, v_j) to receive a ciphertext w_j such that $E(M_j, v_j) = w_j$. Parse $h_j \| S_j \| T_j^{(1)} \| T_j^{(2)} \| T_j^{(3)} \| T_j^{(4)} \leftarrow v_j \oplus (0^n \| C)$. If $T_j^{(1)} \| T_j^{(3)} \neq T_j^{(2)} \| T_j^{(4)}$ the block cipher query does not correspond to a compression function evaluation and a collision is obtained with probability 0. Hence, we assume $T_j^{(1)} \| T_j^{(3)} = T_j^{(2)} \| T_j^{(4)}$. In this case, the block cipher query corresponds to the compression function evaluation $f(h_j, S_j, M_j, T_j^{(1)} \| T_j^{(3)}) = w_j^l \oplus w_j^r \oplus h_j \oplus [S_j]_2$. The query renders a collision for f only if

$$w_j^l \oplus w_j^r \oplus h_j \oplus [S_j]_2 \in \{x_i \mid i = 1, \ldots, j-1\} \cup \{h_0\}\,, \tag{7.1}$$

where the x_i are defined as in the beginning of this section. Let x denote any of the j elements from the set at the right hand side of (7.1). The j^{th} query collides with x with probability $\mathbf{Pr}\left(w_j^l \oplus w_j^r = x \oplus h_j \oplus [S_j]_2\right)$, where h_j and S_j are fixed by the adversarial input v_j to E. As w_j is generated from a set of size at least $2^{2n} - q$, and at most 2^n values w_j satisfy the equation, this probability is upper bounded by $\frac{2^n}{2^{2n}-q}$. Considering any choice of x, then the j^{th} query results in a collision with probability at most $\frac{j2^n}{2^{2n}-q}$.

Collision by an inverse query. The adversary makes a decryption query of the form (M_j, w_j) to receive a plaintext v_j. Parse $h_j \| S_j \| T_j^{(1)} \| T_j^{(2)} \| T_j^{(3)} \| T_j^{(4)} \leftarrow v_j \oplus (0^n \| C)$. This query only renders a collision if v_j is consistent with the definition of f, i.e., it satisfies $T_j^{(1)} \| T_j^{(3)} = T_j^{(2)} \| T_j^{(4)}$. Additionally, the query constitutes a collision for f only if

$$w_j^l \oplus w_j^r \oplus h_j \oplus [S_j]_2 \in \{x_i \mid i = 1, \ldots, j-1\} \cup \{h_0\}\,. \tag{7.2}$$

Let x denote any of the j elements from the set at the right hand side of (7.2). The j^{th} query results in a collision with this x with probability

$$\begin{aligned}&\mathbf{Pr}\left(h_j \oplus [S_j]_2 = x \oplus w_j^l \oplus w_j^r \ \wedge\ T_j^{(1)} \| T_j^{(3)} = T_j^{(2)} \| T_j^{(4)}\right)\\ &= \sum_{S \in \mathbb{Z}_2^{n/2}} \sum_{T \in \mathbb{Z}_2^{n/4}} \mathbf{Pr}\left(\begin{array}{l} h_j = x \oplus w_j^l \oplus w_j^r \oplus [S]_2 \ \wedge\ S_j = S \\ \wedge\ T_j^{(1)} \| T_j^{(2)} \| T_j^{(3)} \| T_j^{(4)} = [T^l]_2 \| [T^r]_2 \end{array}\right),\end{aligned}$$

where the equality holds simply by conditioning on the values attained by S_j and the T_j's. As v_j is generated from a set of size at least $2^{2n} - q$, for any fixed S and T this probability is upper bounded by $\frac{1}{2^{2n}-q}$. Considering any choice of x, S, and T, then the j^{th} query results in a collision with probability at most $\frac{j 2^{n/2} 2^{n/4}}{2^{2n}-q} = \frac{j 2^{3n/4}}{2^{2n}-q}$.

A collision for the compression function f is generated by either a forward or inverse query, and the j^{th} query thus renders a collision with probability at most $\max\left\{\frac{j2^n}{2^{2n}-q}, \frac{j2^{3n/4}}{2^{2n}-q}\right\} = \frac{j2^n}{2^{2n}-q}$. Summing over all q queries, we obtain

$$\mathbf{Adv}_{f^E}^{\text{col}}(q) \le \sum_{j=1}^{q} \frac{j2^n}{2^{2n}-q} \le \frac{q(q+1)2^n}{2(2^{2n}-q)}\,.$$

For $q < 2^{2n-1}$, we have $\frac{1}{2^{2n}-q} \le \frac{2}{2^{2n}}$, which completes the proof. $\square$

Theorem 7.1.3. *Let $n \in \mathbb{N}$. The advantage of any adversary $\mathcal{A}$ in finding a preimage for f after $q < 2^{2n-1}$ queries can be upper bounded by*

$$\mathbf{Adv}_{f^E}^{\text{epre}}(q) \le \frac{2q}{2^n}\,.$$

Proof. Let $y \in \mathbb{Z}_2^n$ be any point to be inverted, as specified in Definition 2.3.2. The proof follows the proof of Theorem 7.1.2 with the only difference that the j^{th} query (for $j = 1, \ldots, q$) needs to hit this particular value y, rather than any value x from a set of size j (cf. (7.1-7.2)). More detailed, the j^{th} query needs to satisfy $w_j^l \oplus w_j^r \oplus h_j \oplus [S_j]_2 = y$, and results in a preimage for y with probability at most $\frac{2^n}{2^{2n}-q}$. When summing over all queries, we obtain

$$\mathbf{Adv}_{f^E}^{\text{epre}}(q) \le \sum_{j=1}^{q} \frac{2^n}{2^{2n}-q} \le \frac{q2^n}{2^{2n}-q} \le \frac{2q}{2^n}\,,$$

where the last inequality holds as $q < 2^{2n-1}$. $\square$

Hash Function Security

As the HAIFA mode of operation preserves collision and everywhere preimage resistance [19], the findings of Theorems 7.1.2 and 7.1.3 directly carry over to the BLAKE hash function $\mathcal{H}$, due to which we obtain $\mathbf{Adv}^{\mathrm{col}}_{\mathcal{H}^E} = \Theta(q^2/2^n)$ and $\mathbf{Adv}^{\mathrm{epre}}_{\mathcal{H}^E} = \Theta(q/2^n)$.

We prove that optimal second preimage resistance (up to a constant) is achieved for all versions. Our proof shows similarities with the second preimage proof for HAIFA by Bouillaguet and Fouque [59]. It is, however, realized in the ideal cipher (rather than ideal compression function) model.

Theorem 7.1.4. *Let $n \in \mathbb{N}$, and $\lambda \geq n/2$. The advantage of any adversary $\mathcal{A}$ in finding a second preimage for $\mathcal{H}$ after $q < 2^{2n-1}$ queries can be upper bounded by*

$$\mathbf{Adv}^{\mathrm{esec}[\lambda]}_{\mathcal{H}^E}(q) \leq \frac{4q}{2^n}\,.$$

Proof. Let $(S', M') \in \mathbb{Z}_2^{n/2} \times \mathbb{Z}_2^{\lambda-n/2}$ be the target preimage. We denote $\mathsf{pad}(M') = M'_1\|\cdots\|M'_{k'}$ and denote by $T'_1, \ldots, T'_{k'}$ the corresponding counter values. Note that by construction, $T'_{i'} = \langle i'2n\rangle_{n/4}$ for $i' \in \{1, \ldots, k'-2\}$, $T'_{k'} \neq \langle k'2n\rangle_{n/4}$, and $T'_{k'-1}$ may or may not be of the form $\langle (k'-1)2n\rangle_{n/4}$. The block cipher executions corresponding to this hash function evaluation are given to the adversary for free. That is, $\mathcal{A}$ is forced to make the k' corresponding queries but will not be charged for this. Denote by $h'_0, \ldots, h'_{k'}$ the state values corresponding to the evaluation of $\mathcal{H}(S', M')$.

The goal of the adversary is to find a tuple $(S, M) \neq (S', M')$ such that $\mathcal{H}(S, M) = \mathcal{H}(S', M')$ and such that the query history contains all block cipher evaluations required for the computation of $\mathcal{H}(S, M)$. We pose the following claim:

Lemma 7.1.5. *Suppose $\mathcal{A}$ finds $(S, M) \neq (S', M')$ such that $\mathcal{H}(S, M) = \mathcal{H}(S', M')$. Denote by $M_1, \ldots, M_k$, $T_1, \ldots, T_k$, and $h_0, \ldots, h_k$ the message blocks, counter values and intermediate state values corresponding to the computation of $\mathcal{H}(S, M)$. There must be $i \in \{1, \ldots, k\}$ and $i' \in \{1, \ldots, k'\}$ such that $f(h_{i-1}, S, M_i, T_i) = f(h'_{i'-1}, S', M'_{i'}, T'_{i'})$, where $(h_{i-1}, S, M_i) \neq (h'_{i'-1}, S', M'_{i'})$ and $T_i, T'_{i'}$ satisfy*

$$T_i = T'_{i'} \text{ or } \left(T_i \neq \langle i2n\rangle_{n/4} \text{ and } T'_{i'} \neq \langle i'2n\rangle_{n/4}\right)\,. \tag{7.3}$$

Proof of lemma. As $\mathcal{H}(S, M) = \mathcal{H}(S', M')$, we have $h_k = h'_{k'}$. If $|M| \neq |M'|$, then $M_k \neq M'_{k'}$, $T_k \neq \langle k2n\rangle_{n/4}$ and $T'_{k'} \neq \langle k'2n\rangle_{n/4}$, and a collision of the prescribed form is found. Thus, suppose $|M| = |M'|$. This implies $k = k'$ and $T_i = T'_i$ for $i = 1, \ldots, k$. If $S \neq S'$, a collision for h_k is directly found. If $S = S'$, we necessarily have $M \neq M'$ and by the standard collision resistance preservation proof for the Merkle-Damgård mode of operation (Theorem 2.4.1), there must by an index i such that $f(h_{i-1}, S, M_i, T_i) = f(h'_{i-1}, S', M'_i, T'_i)$ but $(h_{i-1}, M_i) \neq (h'_{i-1}, M'_i)$. This completes the proof. □

It consequently suffices to consider the probability of the adversary finding a collision with any of the k' compression function evaluations of $\mathcal{H}(S', M')$, such that the corresponding counter values satisfy (7.3). We call a collision of this form a "valid collision." Thus,

$$\mathbf{Adv}_{\mathcal{H}^E}^{\mathrm{esec}[\lambda]}(q) \leq \sum_{j=1}^{q} \mathbf{Pr}\left(j^{\mathrm{th}} \text{ query is valid collision}\right) . \tag{7.4}$$

Let $j = 1, \ldots, q$. We consider the probability that the j^{th} query results in a collision with any of the target state values. We distinguish between *forward* and *inverse* queries.

Valid collision by a forward query. The adversary makes an encryption query of the form (M_j, v_j) to receive a ciphertext w_j such that $E(M_j, v_j) = w_j$. Parse $h_j \| S_j \| T_j^{(1)} \| T_j^{(2)} \| T_j^{(3)} \| T_j^{(4)} \leftarrow v_j \oplus (0^n \| C)$. If $T_j^{(1)} \| T_j^{(3)} \neq T_j^{(2)} \| T_j^{(4)}$ the block cipher query does not correspond to a compression function evaluation and a valid collision is obtained with probability 0. Hence, we assume $T_j^{(1)} \| T_j^{(3)} = T_j^{(2)} \| T_j^{(4)}$. In this case, the block cipher query corresponds to the compression function evaluation $f(h, S, M, T_j^{(1)} \| T_j^{(3)}) = w_j^l \oplus w_j^r \oplus h_j \oplus [S_j]_2$.

If $T_j^{(1)} \| T_j^{(3)} = \langle \alpha 2n \rangle_{n/4}$ for some $\alpha \in \{1, \ldots, k' - 1\}$, this means the query corresponds to a compression function at the α^{th} position, and (7.3) may be satisfied only for $i' = \alpha$. If $T_j^{(1)} \| T_j^{(3)} \neq \alpha 2n$ for any $\alpha \in \{1, \ldots, k' - 2\}$, (7.3) can be satisfied only for $i' \in \{k' - 1, k'\}$. In any other case, there is no i' that makes (7.3) satisfied. In any case, there are at most 2 values that $w_j^l \oplus w_j^r \oplus h_j \oplus [S_j]_2$ may hit in order to render a valid collision. As in the proof of Theorem 7.1.2, the j^{th} query results in a valid collision with probability at most $\frac{2 \cdot 2^n}{2^{2n} - q}$.

Valid collision by a inverse query. The analysis follows the same lines. The j^{th} query results in a valid collision with probability at most $\frac{2 \cdot 2^{3n/4}}{2^{2n} - q}$.

A valid collision for the compression function f is generated by either a forward or inverse query, and the j^{th} query thus renders a valid collision with probability at most $\max\left\{\frac{2 \cdot 2^n}{2^{2n} - q}, \frac{2 \cdot 2^{3n/4}}{2^{2n} - q}\right\} = \frac{2 \cdot 2^n}{2^{2n} - q}$. Summing over all q queries, we obtain from (7.4):

$$\mathbf{Adv}_{\mathcal{H}^E}^{\mathrm{esec}[\lambda]}(q) \leq \sum_{j=1}^{q} \frac{2 \cdot 2^n}{2^{2n} - q} \leq \frac{2q2^n}{2^{2n} - q} \leq \frac{4q}{2^n} ,$$

where the last inequality holds as $q < 2^{2n-1}$. □

We prove that the BLAKE hash function is indifferentiable from a random oracle, under the assumption that the underlying block cipher E behaves like an ideal primitive. Intuitively, we demonstrate that there exists a simulator such that no distinguisher can differentiate the real world $(\mathcal{H}^E, E)$ from the simulated world $(\mathcal{R}, \mathcal{S}^{\mathcal{R}})$, except with negligible probability.

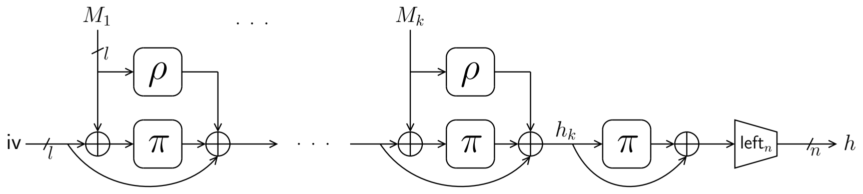


Figure 7.3. The Grøstl hash function $\mathcal{H}$.

Theorem 7.1.6. *Let $E \xleftarrow{\$} \mathrm{Bloc}(2n)$, and let $\mathcal{R}$ be a random oracle. Let $\mathcal{H}$ be the* BLAKE *hash function. Let $\mathcal{D}$ be a distinguisher that makes at most q_L left queries of maximal length $K \cdot 2n$ bits (not including the salt), where $K \geq 1$, and q_R right queries. Then:*

$$\mathbf{Adv}^{\mathrm{iff}}_{\mathcal{H}^E,\mathcal{S}}(\mathcal{D}) \leq 5\frac{q(q+1)}{2^n},$$

where $q = Kq_L + q_R$ and $\mathcal{S}$ makes $q_\mathcal{S} \leq q_R$ queries to $\mathcal{R}$.

The simulator $\mathcal{S}$ used in the proof mimics the behavior of random block cipher E such that queries to $\mathcal{S}$ and queries to $\mathcal{R}$ are consistent, which means that relations among the query outputs in the real world hold in the simulated world as well. The proof can be found in Andreeva et al. [6], and is by Luykx. It can also be found in [156].

7.2 Grøstl

The Grøstl hash function [104] is a chop Merkle-Damgård construction, with a final transformation before chopping (Section 2.4). The hash function employs a suffix-free padding rule. The underlying compression function f is based on two permutations $\pi, \rho : \mathbb{Z}_2^l \to \mathbb{Z}_2^l$. The final transformation g is defined as $g(h) = \pi(h) \oplus h$. The Grøstl hash function design is given in Figure 7.1. A graphical visualization is given in Figure 7.3.

We also consider the Grøstail function $\mathsf{F} : \mathbb{Z}_2^l \times \mathbb{Z}_2^l \to \mathbb{Z}_2^l$, a composition of the last compression function f with the final transformation (i.e., Grøstail is the "tail" of Grøstl):

$$\mathsf{F}(h,m) = \pi(f(h,m)) \oplus f(h,m)\,. \tag{7.5}$$

Note the similarities of this function with the class of functions of Chapter 5, where in this case two primitives are the same. We remark that — even if the three permutations are independent — in order to find collisions for F, it suffices to find them for $f(h,m)$: F allows for collisions in $2^{l/4}$ queries (see next section).

Compression Function Security

The compression function of Grøstl is permutation based, and the results of [199, 209] apply. Furthermore, the preimage resistance of the compression function is analyzed by Fouque et al. [101], and an upper bound for collision resistance can be obtained easily. As a consequence, we obtain tight security bounds on the compression function, $\mathbf{Adv}^{\text{epre}}_{f^{\pi,\rho}} = \Theta(q^2/2^l)$ and $\mathbf{Adv}^{\text{col}}_{f^{\pi,\rho}} = \Theta(q^4/2^l)$. In the ideal model, everywhere second preimage resistance of the compression function can be proven similar as the preimage resistance, up to a constant (the security analysis differs only in that we give the adversary one query for free).

A recent result by Dodis et al. [88] prescribes how to prove indifferentiability of hash functions by ways of preimage awareness (cf. Section 2.4.3). Loosely speaking, Dodis et al. proved that if $\mathcal{H} : \mathbb{Z}_2^\infty \to \mathbb{Z}_2^l$ is a preimage aware hash function and $\mathcal{R} : \mathbb{Z}_2^l \to \mathbb{Z}_2^l$ is a random function, then the composition $\mathcal{R} \circ \mathcal{H}$ is indifferentiable from a random oracle. One might be tempted to consider this approach for the indifferentiability analysis of Grøstl, i.e., by assuming that the output transformation is a random oracle and then proving the Grøstl hash function (without the output transformation) to be preimage aware. However, the behavior of the output transformation $\pi(x) \oplus x$ deviates significantly from a random function: similarly to the Davies-Meyer construction, fixed points $\pi(x) \oplus x = x$ are easy to compute by making the inverse query $\pi^{-1}(0) = x$. A second attempt is to go one step backwards in the iteration and view the last compression function together with the output transformation, i.e., Grøstail (7.5), as a random function. We show that this approach also fails since Grøstail is easily differentiable from a random function.

Theorem 7.2.1. *Let π, ρ be two random l-bit permutations, let F be the* Grøstail *compression function (7.5), and let $\mathcal{R} : \mathbb{Z}_2^l \times \mathbb{Z}_2^l \to \mathbb{Z}_2^l$ be a random function. For any simulator $\mathcal{S}$ that makes at most $q_\mathcal{S}$ queries to $\mathcal{R}$, there exists a distinguisher $\mathcal{D}$ that makes at most* 3 *queries to its oracle, such that $\mathbf{Adv}^{\text{iff}}_{\mathsf{F}^{\pi,\rho},\mathcal{S}}(\mathcal{D}) \geq 1 - q_\mathcal{S}/2^l$.*

Proof. Let $\mathcal{S}$ be any simulator that makes at most $q_\mathcal{S}$ queries to $\mathcal{R}$. We construct a distinguisher $\mathcal{D}$ that with overwhelming probability distinguishes Grøstail from a random function in 3 oracle queries. The distinguisher proceeds as follows. First, it makes inverse queries $x_2 = R_{\rho^{-1}}(0)$ and $x_1 = R_{\pi^{-1}}(x_2)$. Then, it makes a query to the left oracle to obtain $y = L(x_1 \oplus x_2, x_2)$. If $\mathcal{D}$ converses with $(\mathsf{F}^{\pi,\rho}, (\pi, \rho))$, then $y = \mathsf{F}^{\pi,\rho}(x_1 \oplus x_2, x_2) = \pi(x_1) \oplus x_1 = x_1 \oplus x_2$. If $\mathcal{D}$ converses with $(\mathcal{R}, \mathcal{S}^\mathcal{R})$, this equation holds only if the simulator can find x_1, x_2 such that $\mathcal{R}(x_1 \oplus x_2, x_2) = x_1 \oplus x_2$, i.e., only if the simulator can find a fixed point for $\mathcal{R}$. As the probability for the simulator to find fixed points for $\mathcal{R}$ is upper bounded by $q_\mathcal{S}/2^l$, the advantage for $\mathcal{D}$ to distinguish, $\mathbf{Adv}^{\text{iff}}_{\mathsf{F}^{\pi,\rho},\mathcal{S}}(\mathcal{D})$, is lower bounded by $1 - q_\mathcal{S}/2^l$. □

If the final truncation is included in Grøstail as well, a lower bound $1 - q_\mathcal{S}/2^n$ can be obtained similarly.

Hash Function Security

The Grøstl mode of operation preserves collision resistance and everywhere preimage resistance due to which we obtain $\mathbf{Adv}^{\mathrm{col}}_{\mathcal{H}^{\pi,\rho}} = \Theta(q^2/2^n)$ and $\mathbf{Adv}^{\mathrm{epre}}_{\mathcal{H}^{\pi,\rho}} = \Theta(q/2^n)$.

We prove that optimal second preimage resistance (up to a constant) is achieved for all versions. Our proof shows similarities with the second preimage proof for Merkle-Damgård by Bouillaguet and Fouque [59]. It is, however, realized in the ideal permutation (rather than ideal compression function) model.

Theorem 7.2.2. *Let $n \in \mathbb{N}$, and $\lambda \geq 0$. The advantage of any adversary $\mathcal{A}$ in finding a second preimage for the* Grøstl *hash function $\mathcal{H}$ after $q < 2^{l-1}$ queries can be upper bounded by*

$$\mathbf{Adv}^{\mathrm{esec}[\lambda]}_{\mathcal{H}^{\pi,\rho}}(q) \leq \frac{((\lambda+65)/m+2)q(q-1)}{2^l} + \frac{2q}{2^n}\,.$$

Proof. Let $M' \in \mathbb{Z}_2^\lambda$ be any target preimage. Denote by $h'_0, \ldots, h'_{k'}$ the state values corresponding to the evaluation of $\mathcal{H}(M')$, and let $h = \mathsf{left}_n(\pi(h'_{k'}) \oplus h'_{k'})$.

We consider any adversary $\mathcal{A}$ making q queries to its underlying permutations π and ρ. Associated to these queries, we introduce an initially empty graph G that indicates compression function calls for Grøstl that can be derived from these queries. Note that any π-query (x_π, y_π) and any ρ-query (x_ρ, y_ρ) correspond to exactly one compression function call, namely $x_\pi \oplus x_\rho \to x_\pi \oplus y_\pi \oplus x_\rho \oplus y_\rho$ where the message input is x_ρ. In order to find a second preimage, the adversary

(i) either needs to end up with a graph that contains a path (labeled differently from the first preimage) from iv to any node of $\{h'_0, \ldots, h'_{k'}\}$,

(ii) or it needs to find a π-query (x_π, y_π) with $x_\pi \neq h'_{k'}$ such that $\mathsf{left}_n(x_\pi \oplus y_\pi) = h$ and G contains a path from iv to x_π.

A proof of this claim is similar to the one in Theorem 7.1.4. To achieve the first goal, the adversary needs to find a preimage for the Grøstl compression function, for any image in $\{h'_0, \ldots, h'_{k'}\}$. To achieve the second goal, the adversary needs to find a preimage for the final transformation of the Grøstl compression function. For $i = 1, \ldots, q$, we consider the probability of the i^{th} query to render success. We distinguish between the two success cases.

Case (1). Without loss of generality the i^{th} query is a forward query x_π to π, let y_π be the oracle answer drawn uniformly at random from a set of size at least $2^l - q$. Let (x_ρ, y_ρ) be any ρ-query in the query history. The query results in a compression function call $x_\pi \oplus x_\rho \to x_\pi \oplus y_\pi \oplus x_\rho \oplus y_\rho$. This value hits any of $\{h'_0, \ldots, h'_{k'}\}$ with probability at most $\frac{k'+1}{2^l-q}$. Considering any of the at most $i-1$ possible ρ-queries, case (1) is achieved with probability at most $\frac{(k'+1)(i-1)}{2^l-q}$. The same bound is found for queries to ρ and for inverse queries.

Case (2). Case (2) can only be achieved in a query to π. Without loss of generality, the i^{th} query is a forward query x_π, let y_π be the oracle answer drawn uniformly at random from a set of size at least $2^l - q$. This value satisfies $\mathsf{left}_n(x_\pi \oplus y_\pi) = h$ with probability at most $\frac{2^{l-n}}{2^l-q}$.

By the union bound, we obtain the following bound on the second preimage resistance of Grøstl:

$$\mathbf{Adv}^{\text{esec}[\lambda]}_{\mathcal{H}^{\pi,\rho}}(q) \leq \sum_{i=1}^{q} \left(\frac{(k'+1)(i-1)}{2^l - q} + \frac{2^{l-n}}{2^l-q} \right) \leq \frac{(k'+1)q(q-1)}{2(2^l-q)} + \frac{q2^{l-n}}{2^l-q}\,.$$

As for $q < 2^{l-1}$ we have $\frac{1}{2^l-q} \leq \frac{2}{2^l}$ and $k' \leq (\lambda+65)/m + 1$, we obtain our result. $\square$

Given that for Grøstl we have $l = 2n$, for $q < 2^n$ the result of Theorem 7.2.2 directly implies a $\Theta(\lambda/m \cdot q/2^n)$ bound on the second preimage resistance.

We prove that the Grøstl hash function is indifferentiable from a random oracle, under the assumption that the underlying permutations π, ρ behave like ideal primitives. Intuitively, we demonstrate that there exists a simulator such that no distinguisher can differentiate the real world $(\mathcal{H}^{\pi,\rho}, (\pi, \rho))$ from the simulated world $(\mathcal{R}, \mathcal{S}^{\mathcal{R}})$, except with negligible probability.

Theorem 7.2.3. *Let π, ρ be two random l-bit permutations, and let $\mathcal{R}$ be a random oracle. Let $\mathcal{H}$ be the* Grøstl *hash function. Let $\mathcal{D}$ be a distinguisher that makes at most q_L left queries of maximal length $(K-1)l$ bits, where $K \geq 1$, q_π right queries to π and q_ρ right queries to ρ. Then:*

$$\mathbf{Adv}^{\text{iff}}_{\mathcal{H}^{\pi,\rho},\mathcal{S}}(\mathcal{D}) \leq 18\frac{q^3(2q+1)}{2^l} + 4\frac{q^2}{2^l}\,,$$

where $q = (K+1)q_L + \max\{q_\pi, q_\rho\}$ and $\mathcal{S}$ makes at most $q_\mathcal{S} \leq q_R$ queries to $\mathcal{R}$.

The simulator $\mathcal{S}$ used in the proof mimics the behavior of random permutations π and ρ such that queries to $\mathcal{S}$ and queries to $\mathcal{R}$ are consistent, which means that relations among the query outputs in the real world hold in the simulated world as well. To this end, the construction of the simulator is based on several designing decisions. The proof is, for ease of presentation, included in Appendix C.1.

7.3 JH

The JH hash function [224] is a sponge-like function, but can also be considered as a parazoa function (Chapter 6) or a chop Merkle-Damgård construction (Section 2.4). The hash function employs a suffix-free padding rule. The compression function f is based on a permutation $\mathbb{Z}_2^l \to \mathbb{Z}_2^l$. The JH hash function design is given in Figure 7.1.

Compression Function Security

The compression function of JH is based on one permutation, and collisions and preimages for the compression function can be found in one query to the permutation [50, 51].

Hash Function Security

The JH hash function is proven optimally collision resistant [140], and we obtain $\mathbf{Adv}^{\mathrm{col}}_{\mathcal{H}^\pi} = \Theta(q^2/2^n)$. Furthermore, it is proven indifferentiable from a random oracle up to bound $O\left(\frac{q^3}{2^{l-m}} + \frac{Kq^3}{2^{l-n}}\right)$ if the underlying permutation is assumed to be ideal by Bhattacharyya et al. [45]. This bound is improved by Moody et al. [171] to $O((Kq)^2/2^{l-m})$. Using (2.1), this indifferentiability bound additionally renders an improved upper bound $O\left(\frac{q}{2^n} + \frac{q^2}{2^{l-m}}\right)$ on the preimage and second preimage resistance.

Independently of Moody et al. [171], we considered the preimage and second preimage resistance of JH, given that the bound of Bhattacharyya did not result in optimal bounds for these security notions. We analyze these notions in Theorems 7.3.1 and 7.3.2. Although the new bounds are still not better than the trivial bound for $n = 512$ (as was the previous bound), they are optimal (up to a constant) for the 256 variant. They meet the bounds independently derived by Moody et al..

Theorem 7.3.1. *Let $n \in \mathbb{N}$. The advantage of any adversary $\mathcal{A}$ in finding a preimage for the* JH *hash function $\mathcal{H}$ after $q < 2^{l-1}$ queries can be upper bounded by*

$$\mathbf{Adv}^{\mathrm{epre}}_{\mathcal{H}^\pi}(q) \leq \frac{4q^2}{2^{l-m}} + \frac{2q}{2^n}\,.$$

Proof. Let $h \in \mathbb{Z}_2^n$ be any point to be inverted (cf. Definition 2.3.2). iv denotes the initialization vector of size l bits. We consider any adversary $\mathcal{A}$ making q queries to its underlying permutation π. Associated to these queries, we introduce an initially empty graph G that indicates compression function calls for JH that can be derived from these queries. We denote G_i as the graph after the i^{th} query ($i = 0, \ldots, q$). Each query adds 2^m edges to the graph, and G_i thus contains $i2^m$ edges. In order to find a preimage, the adversary must necessarily end up with a graph that contains a path from node iv to any node in $H := \{h \| h' \mid h' \in \mathbb{Z}_2^{l-n}\}$. We denote by winA_i the event that after the i^{th} query this property is satisfied.

We denote by G_i^{out}, resp. G_i^{in}, the set of nodes in G_i with an outgoing, resp. incoming, edge. We denote by τ_i^{iv} the subgraph of G_i consisting of all nodes and edges reachable from iv. Similarly, τ_i^H denotes the subgraph of G_i consisting of all nodes and edges from which any node in H can be reached. Next to event winA_i, we say the adversary also wins if either of the following events occurs for any $i = 1, \ldots, q$:

$$\begin{aligned} \mathsf{winB}_i &: \ \tau_i^{\mathsf{iv}} \text{ contains two nodes } v, v' \text{ with } \mathsf{left}_{l-m}(v) = \mathsf{left}_{l-m}(v')\,; \\ \mathsf{winC}_i &: \ \tau_i^{H} \backslash H \text{ contains two nodes } v, v' \text{ with } \mathsf{right}_{l-m}(v) = \mathsf{right}_{l-m}(v')\,. \end{aligned}$$

We denote by $\mathsf{win}_i = \mathsf{winA}_i \vee \mathsf{winB}_i \vee \mathsf{winC}_i$ the event that after the i^{th} query the adversary has won. We have

$$\mathbf{Adv}^{\text{epre}}_{\mathcal{H}^\pi}(q) \le \mathbf{Pr}\left(\mathsf{winA}_q\right) \le \mathbf{Pr}\left(\mathsf{win}_q\right) = \sum_{i=1}^{q} \mathbf{Pr}\left(\mathsf{win}_i \wedge \neg\mathsf{win}_{i-1}\right)\ . \tag{7.6}$$

For $i = 1, \ldots, q$, we consider the probability of the i^{th} query to render success. We distinguish between forward and inverse queries.

Forward query. Suppose the adversary makes a forward query x_i to receive a random y_i. By $\neg\mathsf{winB}_{i-1}$, there is at most one $v \in \tau_{i-1}^{\mathsf{iv}}$ such that $\mathsf{left}_{l-m}(v) = \mathsf{left}_{l-m}(x_i)$. Denote $M = \mathsf{right}_{l-m}(v) \oplus \mathsf{right}_{l-m}(x_i)$; this query will add only the edge $v \to y_i \oplus (M \| 0^{l-m}) =: w$ to the tree. We define the following events:

$$\begin{aligned} \mathsf{badA}_i &: \ \mathsf{right}_{l-m}(y_i) \in \{\mathsf{right}_{l-m}(w) \mid w \in \tau_{i-1}^{H}\}\,; \\ \mathsf{badB}_i &: \ \mathsf{left}_{l-m}(w) \in \{\mathsf{left}_{l-m}(v) \mid v \in \tau_{i-1}^{\mathsf{iv}}\}\,; \\ \mathsf{badC}_i &: \ w \in G_i^{\text{out}}\,; \\ \mathsf{badD}_i &: \ \mathsf{left}_n(w) = h\,. \end{aligned}$$

Here, badA_i covers the event that τ_{i-1}^{H} is extended. Event badB_i covers the case that the updated τ_i^{iv} contains two nodes with the same left half (note that this would directly make winB_i satisfied). The case badC_i covers the event that the newly added edge to τ_i^{iv} hits any node with outgoing edge, and badD_i covers the event that the newly added edge to the tree would hit h (in both cases a valid preimage path may have been established). Denote $\mathsf{bad}_i = \mathsf{badA}_i \vee \mathsf{badB}_i \vee \mathsf{badC}_i \vee \mathsf{badD}_i$.

By basic probability theory, we have in case of forward queries

$$\mathbf{Pr}\left(\mathsf{win}_i \wedge \neg\mathsf{win}_{i-1}\right) \le \mathbf{Pr}\left(\mathsf{win}_i \wedge \neg\mathsf{win}_{i-1} \wedge \neg\mathsf{bad}_i\right) + \mathbf{Pr}\left(\mathsf{bad}_i \wedge \neg\mathsf{win}_{i-1}\right)\ .$$

We consider the first probability. Assume $\neg\mathsf{win}_{i-1} \wedge \neg\mathsf{bad}_i$. Recall that by $\neg\mathsf{winB}_{i-1}$, $v \to w$ is the only edge added to τ_{i-1}^{iv}. Now, we have $\neg\mathsf{winA}_i$ by $\neg\mathsf{winA}_{i-1}$, and as by $\neg\mathsf{badC}_i \wedge \neg\mathsf{badD}_i$ this new edge does not connect τ_i^{iv} with H. Case $\neg\mathsf{winB}_i$ follows from $\neg\mathsf{winB}_{i-1} \wedge \neg\mathsf{badB}_i$. Finally, by $\neg\mathsf{badA}_i$, the tree τ_{i-1}^{H} is not extended, and hence $\neg\mathsf{winC}_i$ follows from $\neg\mathsf{winC}_{i-1}$. Thus, the first probability equals 0 and for forward queries we have $\mathbf{Pr}\left(\mathsf{win}_i \wedge \neg\mathsf{win}_{i-1}\right) \le \mathbf{Pr}\left(\mathsf{bad}_i \wedge \neg\mathsf{win}_{i-1}\right)$. This probability will be analyzed later.

Inverse query. Suppose the adversary makes an inverse query y_i to receive a random x_i. By $\neg\mathsf{winC}_{i-1}$, there is at most one $v \in \tau_{i-1}^{H}$ such that $\mathsf{right}_{l-m}(v) =$

$\mathsf{right}_{l-m}(y_i)$. Denote $M = \mathsf{left}_{l-m}(v) \oplus \mathsf{left}_{l-m}(y_i)$; this query will add only the edge $w := x_i \oplus (0^{l-m} \| M) \to v$ to the tree. We define the following events:

$$\begin{aligned}
\mathsf{badA}'_i &: \ \mathsf{left}_{l-m}(x_i) \in \{\mathsf{left}_{l-m}(v) \mid v \in \tau^{\mathsf{iv}}_{i-1}\}\,; \\
\mathsf{badB}'_i &: \ \mathsf{right}_{l-m}(v) \in \{\mathsf{right}_{l-m}(w) \mid w \in \tau^{H}_{i-1}\}\,; \\
\mathsf{badC}'_i &: \ v \in G^{\mathrm{in}}_i\,; \\
\mathsf{badD}'_i &: \ v = \mathsf{iv}\,.
\end{aligned}$$

Here, badA'_i covers the event that τ^{iv}_{i-1} is extended. Event badB'_i covers the case that the updated τ^H_i contains two nodes with the same right half (note that this would directly make winC_i satisfied). The case badC'_i covers the event that the newly added edge to τ^H_i hits any node with incoming edge, and badD'_i covers the event that the newly added edge to the tree would hit iv (in both cases a valid preimage path may have been established). Denote $\mathsf{bad}'_i = \mathsf{badA}'_i \vee \mathsf{badB}'_i \vee \mathsf{badC}'_i \vee \mathsf{badD}'_i$.

By basic probability theory, we have in case of inverse queries

$$\mathbf{Pr}\left(\mathsf{win}_i \wedge \neg\mathsf{win}_{i-1}\right) \leq \mathbf{Pr}\left(\mathsf{win}_i \wedge \neg\mathsf{win}_{i-1} \wedge \neg\mathsf{bad}'_i\right) + \mathbf{Pr}\left(\mathsf{bad}'_i \wedge \neg\mathsf{win}_{i-1}\right)\,.$$

We consider the first probability. Assume $\neg\mathsf{win}_{i-1} \wedge \neg\mathsf{bad}'_i$. Recall that by $\neg\mathsf{winC}_{i-1}$, $v \to w$ is the only edge added to τ^H_{i-1}. Now, we have $\neg\mathsf{winA}_i$ by $\neg\mathsf{winA}_{i-1}$, and as by $\neg\mathsf{badC}'_i \wedge \neg\mathsf{badD}'_i$ this new edge does not connect iv with τ^H_i. By $\neg\mathsf{badA}'_i$, the tree τ^{iv}_{i-1} is not extended, and hence $\neg\mathsf{winB}_i$ follows from $\neg\mathsf{winB}_{i-1}$. Finally, case $\neg\mathsf{winC}_i$ follows from $\neg\mathsf{winC}_{i-1} \wedge \neg\mathsf{badB}'_i$. Thus, the first probability equals 0 and for inverse queries we have $\mathbf{Pr}\left(\mathsf{win}_i \wedge \neg\mathsf{win}_{i-1}\right) \leq \mathbf{Pr}\left(\mathsf{bad}'_i \wedge \neg\mathsf{win}_{i-1}\right)$. This probability will be analyzed later.

As each query is either a forward or an inverse query, we obtain for $i = 1, \ldots, q$:

$$\begin{aligned}
\mathbf{Pr}\left(\mathsf{win}_i \wedge \neg\mathsf{win}_{i-1}\right) \leq \max\{&\mathbf{Pr}\left(\mathsf{bad}_i \mid \neg\mathsf{win}_{i-1}; \text{forward query}\right)\,, \\
&\mathbf{Pr}\left(\mathsf{bad}'_i \mid \neg\mathsf{win}_{i-1}; \text{inverse query}\right)\}\,.
\end{aligned} \tag{7.7}$$

As explained above, provided $\neg\mathsf{win}_{i-1}$, the i^{th} query adds at most one node to τ^{iv}_{i-1} and at most one node to τ^H_{i-1}, regardless whether it is a forward or inverse query. This particularly means that $|\tau^{\mathsf{iv}}_{i-1}| \leq i$ and $|\tau^H_{i-1}| \leq i-1$. Additionally, $|G^{\mathrm{out}}_i|, |G^{\mathrm{in}}_i| \leq i2^m$. It is now straightforward to analyze the success probabilities of $\mathsf{bad}_i, \mathsf{bad}'_i$ to occur. As the answer from π is drawn uniformly at random from a set of size at least $2^l - q$, we obtain from (7.7):

$$\mathbf{Pr}\left(\mathsf{win}_i \wedge \neg\mathsf{win}_{i-1}\right) \leq \frac{(2i-1)2^m}{2^l - q} + \frac{i2^m}{2^l - q} + \frac{2^{l-n}}{2^l - q}\,. \tag{7.8}$$

This combines with (7.6) to

$$\mathbf{Adv}^{\mathrm{epre}}_{\mathcal{H}^\pi}(q) \leq \sum_{i=1}^{q} \left(\frac{(3i-1)2^m}{2^l - q} + \frac{2^{l-n}}{2^l - q} \right) \leq \frac{2q^2 2^m}{2^l - q} + \frac{q2^{l-n}}{2^l - q}\,,$$

The result is now obtained as for $q < 2^{l-1}$ we have $\frac{1}{2^l - q} \leq \frac{2}{2^l}$. □

The proof of second preimage resistance of JH is similar. Note that the attack by Kelsey and Schneier [123] only impacts JH in the internal state, which is reflected by the second part of the bound. In accordance with NIST's security requirements, we can assume $q < 2^{n-L}$, or in particular that $\lambda/m \cdot q \lesssim 2^n$ (see the remark below Definition 2.3.3). Consequently, the second term of the second preimage bound is negligible.

Theorem 7.3.2. *Let $n \in \mathbb{N}$, and $\lambda \geq 0$. The advantage of any adversary $\mathcal{A}$ in finding a second preimage for the* JH *hash function $\mathcal{H}$ after $q < 2^{l-1}$ queries can be upper bounded by*

$$\mathbf{Adv}^{\mathrm{esec}[\lambda]}_{\mathcal{H}^\pi}(q) \leq \frac{4q^2}{2^{l-m}} + \frac{2(\lambda/m+2)q}{2^l} + \frac{2q}{2^n}.$$

Proof. The proof follows the same argument as the proof of Theorem 7.3.1; we only highlight the differences. Let $M' \in \mathbb{Z}_2^\lambda$ be any target preimage. Denote by $h'_0, \ldots, h'_{k'}$ the state values corresponding to the evaluation of $\mathcal{H}(M')$, and set $\mathsf{left}_n(h'_{k'}) = h$. Now, the adversary necessarily needs to end up with a graph that contains a path from iv to any node in

$$\{h'_0, \ldots, h'_{k'}\} \cup \{h \| h' \mid h' \in \mathbb{Z}_2^{l-n}\}.$$

This path must be labeled by a message different from M'. The analysis of Theorem 7.3.1 carries over, with the minor difference that badC_i and badC'_i are replaced by

$$\begin{aligned} \mathsf{badC}_i &: \quad w \in G_i^{\mathrm{out}} \cup \{h'_0, \ldots, h'_{k'-1}\}; \\ \mathsf{badC}'_i &: \quad v \in G_i^{\mathrm{in}} \cup \{h'_1, \ldots, h'_{k'}\}. \end{aligned}$$

Similar as before, we obtain

$$\begin{aligned} \mathbf{Adv}^{\mathrm{esec}[\lambda]}_{\mathcal{H}^\pi}(q) &\leq \sum_{i=1}^{q} \left(\frac{(3i-1)2^m}{2^l - q} + \frac{k'}{2^l - q} + \frac{2^{l-n}}{2^l - q} \right) \\ &\leq \frac{2q^2 2^m}{2^l - q} + \frac{k'q}{2^l - q} + \frac{q2^{l-n}}{2^l - q}. \end{aligned}$$

The result is now obtained from the fact that $k' \leq \lambda/m + 2$. □

7.4 Keccak

The Keccak hash function [40] is a sponge function (Section 2.5.2), but can also be considered as a parazoa function (Chapter 6) or a chop Merkle-Damgård construction (Section 2.4). The compression function f is based on a permutation

$\mathbb{Z}_2^l \to \mathbb{Z}_2^l$. The hash function output is obtained by taking the n leftmost bits of the state.[1] Notice that the parameters of Keccak satisfy $l = 2n + m$. The Keccak hash function design is given in Figure 7.1.

Compression Function Security

The compression function of Keccak is based on one permutation, and collisions and preimages for the compression function can be found in one query to the permutation [50, 51].

Hash Function Security

The Keccak hash function is proven indifferentiable from a random oracle up to bound $\Theta((Kq)^2/2^{l-m})$ if the underlying permutation is assumed to be ideal [42]. Using (2.1), this indifferentiability bound renders an optimal collision resistance bound for Keccak, $\mathbf{Adv}_{\mathcal{H}^\pi}^{\mathrm{col}} = \Theta(q^2/2^n)$, as well as optimal preimage second preimage resistance bounds $\Theta(q/2^n)$.

7.5 Skein

The Skein hash function [93] is a chop Merkle-Damgård construction (Section 2.4). The message blocks are accompanied with a round-specific tweak,[2] and the function employs a suffix- and prefix-free padding rule. The compression function f is based on a tweakable block cipher $E : \mathbb{Z}_2^m \times \mathbb{Z}_2^{128} \times \mathbb{Z}_2^l \to \mathbb{Z}_2^m$. The Skein hash function design is given in Figure 7.1.

Compression Function Security

The compression function of Skein is the PGV1, or Matyas-Meyer-Oseas, compression function, with a difference that a tweak is involved. As claimed in [31], the results of [52, 53, 210] carry over, giving optimal preimage and collision security bounds on the compression function, $\mathbf{Adv}_{f^E}^{\mathrm{epre}} = \Theta(q/2^l)$ and $\mathbf{Adv}_{f^E}^{\mathrm{col}} = \Theta(q^2/2^l)$. In the ideal model, everywhere second preimage resistance of the compression function can be proven similar as the preimage resistance, up to a constant (the security analysis differs only in that we give the adversary one query for free).

[1]We notice that sponge function designs are more general [41], but for Keccak this description suffices.

[2]More formally, the design is based on the UBI (unique block identifier) chaining mode which queries its underlying tweakable block cipher on additional tweaks, that differ in each iteration. The general description of Skein involves a specific final transformation. In the primary proposal of the hash function, however, this final transformation consists of another execution of the compression function, with an output-specific tweak and with message 0^m. As we included this final message block in the padding, the given description of Skein suffices.

Hash Function Security

The Skein mode of operation preserves collision resistance and everywhere preimage resistance due to which we obtain $\mathbf{Adv}_{\mathcal{H}^E}^{\mathrm{col}} = \Theta(q^2/2^n)$ and $\mathbf{Adv}_{\mathcal{H}^E}^{\mathrm{epre}} = \Theta(q/2^n)$. Furthermore, the Skein hash function is proven indifferentiable from a random oracle up to bound $O((Kq)^2/2^l)$ if the underlying tweakable block cipher is assumed to be ideal [31]. This proof is based on the preimage awareness approach [88]. Using (2.1), this indifferentiability bound additionally renders an improved upper bound $O\left(\frac{q}{2^n} + \frac{q^2}{2^l}\right)$ on the second preimage resistance.

The second preimage bound for Skein is optimal for the $n = 256$ variant, but meets the trivial bound for the $n = 512$ variant. Therefore, we reconsider second preimage resistance of the Skein hash function. We prove that optimal second preimage resistance (up to a constant) is achieved for all versions.

Theorem 7.5.1. *Let $n \in \mathbb{N}$, and $\lambda \geq 0$. The advantage of any adversary $\mathcal{A}$ in finding a second preimage for the* Skein *hash function $\mathcal{H}$ after $q < 2^{l-1}$ queries can be upper bounded by*

$$\mathbf{Adv}_{\mathcal{H}^E}^{\mathrm{esec}[\lambda]}(q) \leq \frac{2q}{2^l} + \frac{2q}{2^n}\,.$$

Proof. The proof follows a similar reasoning as the proof of Theorem 7.2.2, and we only highlight the differences. Let $M' \in \mathbb{Z}_2^\lambda$ be any target preimage. Denote by $h'_0, \ldots, h'_{k'}$ the state values corresponding to the evaluation of $\mathcal{H}(M')$, and let $h = \mathsf{left}_n(h'_{k'})$.

We consider any adversary $\mathcal{A}$ making q queries to its underlying block cipher E. Associated to these queries, we introduce an initially empty graph G that indicates compression function calls for Skein that can be derived from these queries. Note that any query tuple $(M, T, h) \to C$ corresponds to exactly one compression function call, namely $h \to C \oplus M$ where the message input is M and where T is a tweak value. These tweaks are round-specific (see Figure 7.1). In order to find a second preimage, the adversary needs to end up with a graph that contains a path (labeled different from the first preimage) from iv to any node of $\{h'_0, \ldots, h'_{k'}\} \cup \{h \| h' \mid h' \in \mathbb{Z}_2^{l-n}\}$, where the associated tweaks need to be compliant with the hash function evaluation corresponding to the path. A proof of this claim is similar to the one in Theorem 7.1.4. To achieve the first goal, the adversary needs to find a preimage for the Skein compression function, for any image in $H_1 := \{h'_0, \ldots, h'_{k'}\}$ or $H_2 := \{h \| h' \mid h' \in \mathbb{Z}_2^{l-n}\}$ (where the tweak is compliant). For $i = 1, \ldots, q$, we consider the probability of the i^{th} query to render success. We distinguish between the two sets H_1, H_2. Without loss of generality, let the i^{th} query be a forward query M, T, h, and let C be the oracle answer drawn uniformly at random from a set of size at least $2^l - q$. The same bounds are found for inverse queries.

Set H_1. As the tweaks need to be compliant, depending on T there is at most one value $h \in H_1$ for which a collision $h = C \oplus M$ may result in a valid second preimage. The i^{th} query thus renders a collision with H_1 probability at most $\frac{1}{2^l - q}$.

Set H_2. A collision with any element from H_2 may result in a valid preimage. $C \oplus M$ collides with any element from H_2 with probability at most $\frac{2^{l-n}}{2^l - q}$.

By the union bound, we obtain the following bound on the second preimage resistance of Skein:

$$\mathbf{Adv}_{\mathcal{H}^E}^{\text{esec}[\lambda]}(q) \le \sum_{i=1}^{q} \left(\frac{1}{2^l - q} + \frac{2^{l-n}}{2^l - q} \right) \le \frac{q}{2^l - q} + \frac{q 2^{l-n}}{2^l - q} .$$

The result is now obtained as for $q < 2^{l-1}$ we have $\frac{1}{2^l - q} \le \frac{2}{2^l}$. □

7.6 Conclusions

In this chapter we classified the provable security results of the five finalist SHA-3 candidate hash functions. More concretely, we analyzed and summarized the state of the art, and proved almost all remaining gaps in the classification. A detailed summary of all results is given in Table 7.2. All results in this classification hold for comparable ideal underlying primitives, either ideal ciphers or permutations. The few open problems left for the security analysis of the five finalist hash functions in the ideal model is achieving an optimal preimage and second preimage bound of the 512 variant of the JH hash function.

We note that our security analysis needs to be read with care and for this purpose we provide the following discussion:

- As already discussed in Chapter 1, ideal model proofs only give an indication for security. In particular, by idealizing the candidates' underlying block ciphers or permutations, we abstracted away any properties of these building blocks, and only considered security of the modes of operation. However, due to the lack of security proofs in the standard model (other than preserving collision and preimage security of the compression function in Merkle-Damgård based designs), assuming ideality of these underlying primitives gives significantly more confidence in the security of the higher level hash function structure than any ad-hoc analysis or no proof at all;
- While assuming ideality of sizable underlying building blocks like permutations and block ciphers allows for a fair security comparison of the candidates on one hand, it disregards internal differences between the idealized primitives on the other hand. Such specific design details can distort the security results for the distinct hash functions when concrete attacks exploiting the internal primitive weaknesses are applied. Moreover, further differences, such as chaining sizes and message input sizes, are also not fully reflected in the comparison.

On October 2nd, 2012, NIST announced that Keccak has been selected as the winner of the SHA-3 competition. From a provable security point of view, we support this choice. Indeed, given the classifications in Tables 7.1 and 7.2, (almost) all finalists are proven to achieve an adequate level of collision, preimage, and second preimage security. Yet, Keccak provides higher indifferentiability guarantees than the others, n-bit security where n is the size of the hash function output. It means that Keccak achieves a higher level of security than any other design when used as underlying primitive in a higher level cryptosystem $\mathcal{C}$ (see Section 2.3.3). Remarkably, Keccak is the only design whose results have already been proven in the first phase of the competition and shortly after the design has been introduced, for the other candidates the results have only been completed in 2012 [6, 16]. The designers seem to have made a deliberate choice to also focus on achieving a high level of provable security of their design, which we believe has given Keccak a significant advantage over the other candidates and shows the importance of provable security.

Table 7.2. Detailed summary of all security guarantees of the SHA-3 finalists. The second column summarizes the parameters n, l, m, which denote the hash function output size, the chaining value size and the message input size, respectively. The last row of the table gives a representation of the security requirements `(ii)-(iv)` by NIST [180].

	$n/l/m$	$\mathbf{Adv}_f^{\text{epre}}$	$\mathbf{Adv}_f^{\text{esec}[\lambda]}$	$\mathbf{Adv}_f^{\text{col}}$	$\mathbf{Adv}_{\mathcal{H}}^{\text{epre}}$	$\mathbf{Adv}_{\mathcal{H}}^{\text{esec}[\lambda]}$	$\mathbf{Adv}_{\mathcal{H}}^{\text{col}}$	$\mathbf{Adv}_{\mathcal{H}}^{\text{iff}}$
BLAKE	256/256/512, 512/512/1024	$\Theta(q/2^n)$ E ideal	$\Theta(q/2^n)$ E ideal	$\Theta(q^2/2^n)$ E ideal	$\Theta(q/2^n)$ E ideal	$\Theta(q/2^n)$ E ideal	$\Theta(q^2/2^n)$ E ideal	$\Theta((Kq)^2/2^n)$ E ideal
Grøstl	256/512/512, 512/1024/1024	$\Theta(q^2/2^l)$ π, ρ ideal	$\Theta(q^2/2^l)$ π, ρ ideal	$\Theta(q^4/2^l)$ π, ρ ideal	$\Theta(q/2^n)$ π ideal	$\Theta(\lambda/m \cdot q/2^n)$ π, ρ ideal	$\Theta(q^2/2^n)$ π, ρ ideal	$O((Kq)^4/2^l)$ π, ρ ideal
JH	256/1024/512, 512/1024/512	$\Theta(1)$ π ideal	$\Theta(1)$ π ideal	$\Theta(1)$ π ideal	$O\left(\frac{q}{2^n} + \frac{q^2}{2^{l-m}}\right)$ π ideal	$O\left(\frac{q}{2^n} + \frac{q^2}{2^{l-m}}\right)$ π ideal	$\Theta(q^2/2^n)$ π ideal	$O((Kq)^2/2^{l-m})$ π ideal
Keccak	256/1600/1088, 512/1600/576	$\Theta(1)$ π ideal	$\Theta(1)$ π ideal	$\Theta(1)$ π ideal	$\Theta(q/2^n)$ π ideal	$\Theta(q/2^n)$ π ideal	$\Theta(q^2/2^n)$ π ideal	$\Theta((Kq)^2/2^{l-m})$ π ideal
Skein	256/512/512, 512/512/512	$\Theta(q/2^l)$ E ideal	$\Theta(q/2^l)$ E ideal	$\Theta(q^2/2^l)$ E ideal	$\Theta(q/2^n)$ E ideal	$\Theta(q/2^n)$ E ideal	$\Theta(q^2/2^n)$ E ideal	$O((Kq)^2/2^l)$ E ideal
NIST's security requirements		(not specified)	(not specified)	(not specified)	$O(q/2^n)$	$O(\lambda/m \cdot q/2^n)$	$O(q^2/2^n)$	(not specified)

8 Conclusions and Open Problems

"Richtiges Auffassen einer Sache und Mißverstehen der gleichen Sache schließen einander nicht vollständig aus."

Franz Kafka, Der Proceß

Information security relies heavily on cryptographic hash functions. These functions should be efficient, but more importantly, they should be secure. Recent breakthroughs [219–221] have exposed serious security problems in mainstream hash functions MD5, SHA-1, and RIPEMD, and the launch of NIST's SHA-3 hash function competition has sparked new interest in hash function security.

Our research, presented in this dissertation, can be divided into two parts: research on the basic principles of hash function security and contributions to the SHA-3 competition.

In the former direction, we analyzed various aspects of hash function security. First of all, we considered the foundational concepts of hash function design, by formalizing and analyzing security against the Nostradamus attack of Kelsey and Kohno [122]: we showed, among others, that the attack is actually optimal (Section 2.4.2). Secondly, we analyzed the security of several block cipher and permutation based compression functions, and introduced new families of functions. Our search for minimal compression function designs with maximal security has lead to the introduction of a family of optimally collision secure double block length compression functions (Chapter 4) and a full security classification of all $2n$-to-n-bit compression functions solely built of XOR operators and three permutations (Chapter 5). Finally, we introduced the parazoa hash function family, as a generalization of sponge functions, and proved them to be indifferentiable from a random oracle (Chapter 6).

Regarding the impact of our work on NIST's SHA-3 hash function competition, we provided in Chapter 7 a detailed survey of the five SHA-3 finalists (following an earlier work [12] on all fourteen second round candidates), in which we considered collision resistance, preimage resistance, second preimage resistance, and indifferentiability of all finalists. In this direction, we discussed the state of the art, and proved almost all remaining gaps in the security classification. We not only believe that our contributions in this direction have helped NIST in making their exceptionally hard decision, but we also think they have made the society conscious of the importance of provable security, the base for the design cryptographic primitives.

8.1 Directions for Future Research

Although the SHA-3 competition has come to an end, many research questions remain open. A formal theoretical foundation of cryptographic primitives will always be of utmost importance. With CAESAR [64], a novel competition on authenticated encryption mechanisms, as well as a password hashing competition [184] approaching, challenging security problems are ahead of us for the next years. We conclude with some challenging research problems related to the findings in this dissertation.

Exploring the boundaries. The quest for design minimalism with maximal security exposes the boundaries of symmetric cryptology: the possibilities and impossibilities of design strategies. The works of Chapters 4 and 5 are a shot in the right direction, but many open questions regarding block cipher or permutation based compression functions exist. We highlight the open problem of designing an optimally secure $3n$-to-$2n$-bit compression function making *two* calls to a block cipher with n-bit message and key space: this question has previously been considered by Jetchev et al. [118], and in Section 4.2 we ruled out a wide class of designs, yet it remains open to either prove impossibility in general or to come up with a (possibly impractical) function that provably achieves optimal collision resistance.

Tightness of proofs. Apart from the search for optimally secure designs, many functions — most prominently double length functions using a cipher with n-bit message and key space — with a loose security bound exist. For instance, the security bounds on Jetchev et al.'s construction [118] and the (iterated) MJH hash function (see Section 2.6) do not meet the best known attack. In Table 3.1 we list all bounds and gaps for MDC-2 and MDC-4 in detail. Also for permutation based hash functions, open problems of similar kind exist [198]. Powerful proof techniques, such as using thresholds (see, e.g., Chapters 3-5) and wish lists (see, e.g., Chapters 3-4), appeared in the last years. These techniques allowed for achieving better security bounds. A new creative idea is likely to improve the current state of the art.

Generalizations. Indifferentiability proofs are nowadays still design-specific. In Chapter 6 we presented an indifferentiability proof that applies to a class of hash functions, and several follow-up works appeared [67, 77]. However, as shown in Chapter 6, generality comes with a toll, in a sense that the proofs are more complex and that design-specific proofs may render better bounds. Further research in this direction is needed.

Automated Proofs. Although applied to specific designs, some proofs in this dissertation are generic. For instance, the proofs of Theorem 3.2.1 (Section 3.2.1), Theorem 4.3.1 (Section 4.3.1.1), and Lemma 5.3.5(i) (Appendix B.1) all rely on the fundamental idea that the hard problem is divided into smaller sub-problems which are easier to analyze. In [198], Rogaway and Steinberger developed an automated proof technique to analyze permutation based compression functions, and it is of interest to investigate the possibilities of automated analysis for other constructions. In a similar fashion, Backes et al. [26] and Daubignard et al. [77] analyze indifferentiability of the Merkle-Damgård mode of operation in an automated way.

Indifferentiability for multi-stage games. Novel definitions rarely come with no technicalities, and should always be handled with care. Indifferentiability, one of the most prominent notions in hash function security, is shown to apply to single stage games only [192]. However, a recent work by Luykx et al. [157] shows that no practical domain extender meets the generalized reset-indifferentiability for multi-stage games. A major question is, how this affects the security of hash functions proven indifferentiable in the original model, such as SHA-3. For which practical applications is this a problem, and how can this problem be solved?

Beyond the ideal model. All proofs in this dissertation are in the ideal cipher or ideal permutation model. The next step is to break these primitives into pieces. Several works considered the indifferentiability of the Feistel cipher [71, 72, 85, 115, 202], and in a recent work by Andreeva et al. [1], we analyze the indifferentiability of the Even-Mansour block cipher construction, assuming ideality of the underlying primitives. However, this research direction is still in its infancy. Furthermore, these approaches still focus on structural security, as the underlying primitives are assumed to be ideal. The ultimate goal is, of course, to step away from the ideal model, and consider security of compressing functions based on block ciphers or permutations in the standard model. However, this approach involves many technicalities. For instance, Simon [207] showed that the standard security model for block ciphers, PRP (pseudo-random permutation) security, is insufficient to prove security of a block cipher based hash function, and additionally, schemes are known that are secure in the ideal model but insecure with any instantiation of it [29, 49, 65, 66, 178]. A fruitful approach may be to consider security in the almost ideal model (as, e.g., done by [61]), security from unpredictable ciphers (see, e.g., [89, 148]), or schemes based on hard mathematical problems.

Bibliography

[1] Elena Andreeva, Andrey Bogdanov, Yevgeniy Dodis, Bart Mennink, and John P. Steinberger. On the indifferentiability of key-alternating ciphers. Cryptology ePrint Archive, Report 2013/061, 2013.

[2] Elena Andreeva, Andrey Bogdanov, and Bart Mennink. Towards understanding the known-key security of block ciphers. In *Fast Software Encryption – FSE 2013*, Lecture Notes in Computer Science, Singapore, March 11–13, 2013. Springer, Berlin, Germany. To appear.

[3] Elena Andreeva, Andrey Bogdanov, Bart Mennink, Bart Preneel, and Christian Rechberger. On security arguments of the second round SHA-3 candidates. *International Journal of Information Security*, 11(2):103–120, 2012.

[4] Elena Andreeva, Charles Bouillaguet, Orr Dunkelman, and John Kelsey. Herding, second preimage and trojan message attacks beyond Merkle-Damgård. In Michael J. Jacobson Jr., Vincent Rijmen, and Reihaneh Safavi-Naini, editors, *SAC 2009: 16th Annual International Workshop on Selected Areas in Cryptography*, volume 5867 of *Lecture Notes in Computer Science*, pages 393–414, Calgary, Alberta, Canada, August 13–14, 2009. Springer, Berlin, Germany.

[5] Elena Andreeva, Charles Bouillaguet, Pierre-Alain Fouque, Jonathan J. Hoch, John Kelsey, Adi Shamir, and Sébastien Zimmer. Second preimage attacks on dithered hash functions. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 270–288, Istanbul, Turkey, April 13–17, 2008. Springer, Berlin, Germany.

[6] Elena Andreeva, Atul Luykx, and Bart Mennink. Provable security of BLAKE with non-ideal compression function. In Lars R. Knudsen, editor, *SAC 2012: 19th Annual International Workshop on Selected Areas in Cryptography*, volume 7707 of *Lecture Notes in Computer Science*, pages 322–339, Windsor, Ontario, Canada, August 16–17, 2012. Springer, Berlin, Germany.

[7] Elena Andreeva, Atul Luykx, and Bart Mennink. Provable security of BLAKE with non-ideal compression function. NIST's 3rd SHA-3 Candidate Conference 2012, 2012.

[8] Elena Andreeva and Bart Mennink. Provable chosen-target-forced-midfix preimage resistance. In Ali Miri and Serge Vaudenay, editors, *SAC 2011: 18th Annual International Workshop on Selected Areas in Cryptography*, volume 7118 of *Lecture Notes in Computer Science*, pages 37–54, Toronto, Ontario, Canada, August 11–12, 2011. Springer, Berlin, Germany.

[9] Elena Andreeva, Bart Mennink, and Bart Preneel. On the indifferentiability of the Grøstl hash function. In Juan A. Garay and Roberto De Prisco, editors, *SCN 2010: 7th International Conference on Security in Communication Networks*, volume 6280 of *Lecture Notes in Computer Science*, pages 88–105, Amalfi, Italy, September 13–15, 2010. Springer, Berlin, Germany.

[10] Elena Andreeva, Bart Mennink, and Bart Preneel. On the indifferentiability of the Grøstl hash function. NIST's 2nd SHA-3 Candidate Conference 2010, 2010.

[11] Elena Andreeva, Bart Mennink, and Bart Preneel. Security properties of domain extenders for cryptographic hash functions. *Journal of Information Processing Systems – JIPS*, 6(4):453–480, 2010.

[12] 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, *ISC 2010: 13th International Conference on Information Security*, volume 6531 of *Lecture Notes in Computer Science*, pages 39–53, Boca Raton, FL, USA, October 25–28, 2010. Springer, Berlin, Germany.

[13] Elena Andreeva, Bart Mennink, and Bart Preneel. Security reductions of the SHA-3 candidates. NIST's 2nd SHA-3 Candidate Conference 2010, 2010.

[14] Elena Andreeva, Bart Mennink, and Bart Preneel. The parazoa family: Generalizing the sponge hash functions. ECRYPT II Hash Workshop 2011, 2011.

[15] Elena Andreeva, Bart Mennink, and Bart Preneel. The parazoa family: Generalizing the sponge hash functions. *International Journal of Information Security*, 11(3):149–165, 2012.

[16] Elena Andreeva, Bart Mennink, Bart Preneel, and Marjan Škrobot. Security analysis and comparison of the SHA-3 finalists BLAKE, Grøstl, JH, Keccak, and Skein. In Aikaterini Mitrokotsa and Serge Vaudenay, editors, *AFRICACRYPT 2012: 5th International Conference on Cryptology in*

Africa, volume 7374 of *Lecture Notes in Computer Science*, pages 287–305, Ifrance, Morocco, July 10–12, 2012. Springer, Berlin, Germany.

[17] Elena Andreeva, Bart Mennink, Bart Preneel, and Marjan Škrobot. Security analysis and comparison of the SHA-3 Finalists BLAKE, Grøstl, JH, Keccak, and Skein. NIST's 3rd SHA-3 Candidate Conference 2012, 2012.

[18] Elena Andreeva, Bart Mennink, Bart Preneel, and Marjan Škrobot. Security analysis and comparison of the SHA-3 Finalists BLAKE, Grøstl, JH, Keccak, and Skein. Bulgarian Cryptography Days 2012, 2012.

[19] Elena Andreeva, Gregory Neven, Bart Preneel, and Thomas Shrimpton. Seven-property-preserving iterated hashing: ROX. In Kaoru Kurosawa, editor, *Advances in Cryptology – ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 130–146, Kuching, Malaysia, December 2–6, 2007. Springer, Berlin, Germany.

[20] Elena Andreeva and Martijn Stam. The symbiosis between collision and preimage resistance. In Liqun Chen, editor, *13th IMA International Conference on Cryptography and Coding*, volume 7089 of *Lecture Notes in Computer Science*, pages 152–171, Oxford, UK, December 12–15, 2011. Springer, Berlin, Germany.

[21] Yuriy Arbitman, Gil Dogon, Vadim Lyubashevsky, Daniele Micciancio, Chris Peikert, and Alon Rosen. SWIFFTX: A proposal for the SHA-3 standard. Submission to NIST's SHA-3 competition, 2008.

[22] 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*, volume 7073 of *Lecture Notes in Computer Science*, pages 233–251, Seoul, South Korea, December 4–8, 2011. Springer, Berlin, Germany.

[23] Daniel Augot, Matthieu Finiasz, and Nicolas Sendrier. A fast provably secure cryptographic hash function. Cryptology ePrint Archive, Report 2003/230, 2003.

[24] 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*, volume 6225 of *Lecture Notes in Computer Science*, pages 1–15, Santa Barbara, California, USA, August 17–20, 2010. Springer, Berlin, Germany.

[25] Jean-Philippe Aumasson, Luca Henzen, Willi Meier, and Raphael Phan. SHA-3 proposal BLAKE. Submission to NIST's SHA-3 competition, 2010.

[26] Michael Backes, Gilles Barthe, Matthias Berg, Benjamin Grégoire, César Kunz, Malte Skoruppa, and Santiago Zanella Béguelin. Verified security of Merkle-Damgård. In Stephen Chong, editor, *CSF 2012: 25th IEEE Computer Security Foundations Symposium*, pages 354–368, Cambridge, Massachusetts, USA, June 25–27, 2012. IEEE Computer Society.

[27] Nasour Bagheri, Praveen Gauravaram, Majid Naderi, and Søren S. Thomsen. On the collision and preimage resistance of certain two-call hash functions. In Swee-Huay Heng, Rebecca N. Wright, and Bok-Min Goi, editors, *CANS 2010: 9th International Conference on Cryptology and Network Security*, volume 6467 of *Lecture Notes in Computer Science*, pages 96–105, Kuala Lumpur, Malaysia, December 12–14, 2010. Springer, Berlin, Germany.

[28] Mihir Bellare. New proofs for NMAC and HMAC: Security without collision-resistance. In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 602–619, Santa Barbara, CA, USA, August 20–24, 2006. Springer, Berlin, Germany.

[29] Mihir Bellare, Alexandra Boldyreva, and Adriana Palacio. An uninstantiable random-oracle-model scheme for a hybrid-encryption problem. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 171–188, Interlaken, Switzerland, May 2–6, 2004. Springer, Berlin, Germany.

[30] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In Neal Koblitz, editor, *Advances in Cryptology – CRYPTO'96*, volume 1109 of *Lecture Notes in Computer Science*, pages 1–15, Santa Barbara, CA, USA, August 18–22, 1996. Springer, Berlin, Germany.

[31] Mihir Bellare, Tadayoshi Kohno, Stefan Lucks, Niels Ferguson, Bruce Schneier, Doug Whiting, Jon Callas, and Jesse Walker. Provable security support for the Skein hash family, 2009.

[32] Mihir Bellare and Thomas Ristenpart. Multi-property-preserving hash domain extension and the EMD transform. In Xuejia Lai and Kefei Chen, editors, *Advances in Cryptology – ASIACRYPT 2006*, volume 4284 of *Lecture Notes in Computer Science*, pages 299–314, Shanghai, China, December 3–7, 2006. Springer, Berlin, Germany.

[33] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press.

[34] 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*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416, Saragossa, Spain, May 12–16, 1996. Springer, Berlin, Germany.

[35] Mihir Bellare and Phillip Rogaway. Collision-resistant hashing: Towards making UOWHFs practical. In Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 470–484, Santa Barbara, CA, USA, August 17–21, 1997. Springer, Berlin, Germany.

[36] Kamel Bentahar, Dan Page, Markku-Juhani O. Saarinen, Joseph H. Silverman, and Nigel Smart. LASH. NIST Second Cryptographic Hash Workshop 2006, 2006.

[37] Daniel J. Bernstein. CubeHash specification. Submission to NIST's SHA-3 competition, 2009.

[38] Daniel J. Bernstein, Tanja Lange, Christiane Peters, and Peter Schwabe. Really fast syndrome-based hashing. In Abderrahmane Nitaj and David Pointcheval, editors, *AFRICACRYPT 2011: 4th International Conference on Cryptology in Africa*, volume 6737 of *Lecture Notes in Computer Science*, pages 134–152, Dakar, Senegal, July 5–7, 2011. Springer, Berlin, Germany.

[39] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Sufficient conditions for sound tree and sequential hashing modes. Cryptology ePrint Archive, Report 2009/210, 2009.

[40] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. The KECCAK sponge function family. Submission to NIST's SHA-3 competition, 2011.

[41] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Sponge functions. ECRYPT Hash Function Workshop 2007, 2007.

[42] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. On the indifferentiability of the sponge construction. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 181–197, Istanbul, Turkey, April 13–17, 2008. Springer, Berlin, Germany.

[43] Rishiraj Bhattacharyya and Avradip Mandal. On the indifferentiability of Fugue and Luffa. In Javier Lopez and Gene Tsudik, editors, *ACNS 2011: 9th International Conference on Applied Cryptography and Network Security*, volume 6715 of *Lecture Notes in Computer Science*, pages 479–497, Nerja, Spain, June 7–10, 2011. Springer, Berlin, Germany.

[44] Rishiraj Bhattacharyya, Avradip Mandal, and Mridul Nandi. Indifferentiability characterization of hash functions and optimal bounds of popular domain extensions. In Bimal K. Roy and Nicolas Sendrier, editors, *Progress in Cryptology - INDOCRYPT 2009: 10th International Conference in Cryptology in India*, volume 5922 of *Lecture Notes in Computer Science*, pages 199–218, New Delhi, India, December 13–16, 2009. Springer, Berlin, Germany.

[45] 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 – FSE 2010*, volume 6147 of *Lecture Notes in Computer Science*, pages 168–191, Seoul, Korea, February 7–10, 2010. Springer, Berlin, Germany.

[46] Eli Biham and Orr Dunkelman. A framework for iterative hash functions – HAIFA. Cryptology ePrint Archive, Report 2007/278, 2007.

[47] Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. In Alfred J. Menezes and Scott A. Vanstone, editors, *Advances in Cryptology – CRYPTO'90*, volume 537 of *Lecture Notes in Computer Science*, pages 2–21, Santa Barbara, CA, USA, August 11–15, 1991. Springer, Berlin, Germany.

[48] Alex Biryukov, Dmitry Khovratovich, and Ivica Nikolić. Distinguisher and related-key attack on the full AES-256. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 231–249, Santa Barbara, CA, USA, August 16–20, 2009. Springer, Berlin, Germany.

[49] John Black. The ideal-cipher model, revisited: An uninstantiable blockcipher-based hash function. In Matthew J. B. Robshaw, editor, *Fast Software Encryption – FSE 2006*, volume 4047 of *Lecture Notes in Computer Science*, pages 328–340, Graz, Austria, March 15–17, 2006. Springer, Berlin, Germany.

[50] John Black, Martin Cochran, and Thomas Shrimpton. On the impossibility of highly-efficient blockcipher-based hash functions. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 526–541, Aarhus, Denmark, May 22–26, 2005. Springer, Berlin, Germany.

[51] John Black, Martin Cochran, and Thomas Shrimpton. On the impossibility of highly-efficient blockcipher-based hash functions. *Journal of Cryptology*, 22(3):311–329, July 2009.

[52] John Black, Phillip Rogaway, and Thomas Shrimpton. Black-box analysis of the block-cipher-based hash-function constructions from PGV. In Moti

Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 320–335, Santa Barbara, CA, USA, August 18–22, 2002. Springer, Berlin, Germany.

[53] John Black, Phillip Rogaway, Thomas Shrimpton, and Martijn Stam. An analysis of the blockcipher-based hash functions from PGV. *Journal of Cryptology*, 23(4):519–545, October 2010.

[54] Simon R. Blackburn, Douglas R. Stinson, and Jalaj Upadhyay. On the complexity of the herding attack and some related attacks on hash functions. *Designs, Codes and Cryptography*, 64(1–2):171–193, 2012.

[55] Andrey Bogdanov, Miroslav Knezevic, Gregor Leander, Deniz Toz, Kerem Varıcı, and Ingrid Verbauwhede. Spongent: A lightweight hash function. In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems – CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*, pages 312–325, Nara, Japan, September 28 – October 1, 2011. Springer, Berlin, Germany.

[56] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, François-Xavier Standaert, John P. Steinberger, and Elmar Tischhauser. Key-alternating ciphers in a provable setting: Encryption using a small number of public permutations - (extended abstract). In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 45–62, Cambridge, UK, April 15–19, 2012. Springer, Berlin, Germany.

[57] Béla Bollobás. *Extremal Graph Theory.* Academic Press, 1978.

[58] Joppe W. Bos, Onur Özen, and Martijn Stam. Efficient hashing using the AES instruction set. In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems – CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*, pages 507–522, Nara, Japan, September 28 – October 1, 2011. Springer, Berlin, Germany.

[59] Charles Bouillaguet and Pierre-Alain Fouque. Practical hash functions constructions resistant to generic second preimage attacks beyond the birthday bound, 2010. Submitted to Information Processing Letters.

[60] Bruno O. Brachtl, Don Coppersmith, Myrna M. Hyden, Stephen M. Matyas, Carl H. Meyer, Jonathan Oseas, Shaiy Pilpel, and Michael Schilling. *Data Authentication Using Modification Detection Codes Based on a Public One Way Encryption Function.* U.S. Patent Number 4,908,861, March 13, 1990.

[61] Emmanuel Bresson, Anne Canteaut, Benoît Chevallier-Mames, Christophe Clavier, Thomas Fuhr, Aline Gouget, Thomas Icart, Jean-François Misarsky, Marìa Naya-Plasencia, Pascal Paillier, Thomas Pornin, Jean-René Reinhard,

Céline Thuillet, and Marion Videau. Indifferentiability with distinguishers: Why Shabal does not require ideal ciphers. Cryptology ePrint Archive, Report 2009/199, 2009.

[62] Daniel R. L. Brown, Adrian Antipa, Matt Campagna, and Rene Struik. ECOH: the elliptic curve only hash. Submission to NIST's SHA-3 competition, 2008.

[63] Lawrence Brown, Josef Pieprzyk, and Jennifer Seberry. LOKI - a cryptographic primitive for authentication and secrecy applications. In Jennifer Seberry and Josef Pieprzyk, editors, *Advances in Cryptology – AUSCRYPT'90*, volume 453 of *Lecture Notes in Computer Science*, pages 229–236, Sydney, Australia, January 8–11, 1990. Springer, Berlin, Germany.

[64] CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness, April 2013. `http://competitions.cr.yp.to/caesar.html`.

[65] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *30th Annual ACM Symposium on Theory of Computing*, pages 209–218, Dallas, Texas, USA, May 23–26, 1998. ACM Press.

[66] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *Journal of the ACM*, 51(4):557–594, 2004.

[67] 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.

[68] Donghoon Chang, Sangjin Lee, Mridul Nandi, and Moti Yung. Indifferentiable security analysis of popular hash functions with prefix-free padding. In Xuejia Lai and Kefei Chen, editors, *Advances in Cryptology – ASIACRYPT 2006*, volume 4284 of *Lecture Notes in Computer Science*, pages 283–298, Shanghai, China, December 3–7, 2006. Springer, Berlin, Germany.

[69] Donghoon Chang, Mridul Nandi, and Moti Yung. Indifferentiability of the hash algorithm BLAKE. Cryptology ePrint Archive, Report 2011/623, 2011.

[70] Scott Contini, Arjen Lenstra, and Ron Steinfeld. VSH, an efficient and provable collision-resistant hash function. Cryptology ePrint Archive, Report 2005/193, 2005.

[71] 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*, volume 3621 of *Lecture Notes in Computer Science*, pages 430–448, Santa Barbara, CA, USA, August 14–18, 2005. Springer, Berlin, Germany.

[72] Jean-Sébastien Coron, Jacques Patarin, and Yannick Seurin. The random oracle model and the ideal cipher model are equivalent. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 1–20, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Berlin, Germany.

[73] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25, Santa Barbara, CA, USA, August 23–27, 1998. Springer, Berlin, Germany.

[74] Joan Daemen and Vincent Rijmen. The wide trail design strategy. In Bahram Honary, editor, *8th IMA International Conference on Cryptography and Coding*, volume 2260 of *Lecture Notes in Computer Science*, pages 222–238, Cirencester, UK, December 17–19, 2001. Springer, Berlin, Germany.

[75] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES – The Advanced Encryption Standard.* Springer, 2002.

[76] Ivan Damgård. A design principle for hash functions. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 416–427, Santa Barbara, CA, USA, August 20–24, 1990. Springer, Berlin, Germany.

[77] Marion Daubignard, Pierre-Alain Fouque, and Yassine Lakhnech. Generic indifferentiability proofs of hash designs. In Stephen Chong, editor, *CSF 2012: 25th IEEE Computer Security Foundations Symposium*, pages 340–353, Cambridge, Massachusetts, USA, June 25–27, 2012. IEEE Computer Society.

[78] Christophe De Cannière, Hisayoshi Sato, and Dai Watanabe. Hash function Luffa. Submission to NIST's SHA-3 competition, 2009.

[79] Richard Dean. *Formal Aspects of Mobile Code Security.* PhD thesis, Princeton University, Princeton, 1999.

[80] Grégory Demay, Peter Gaži, Martin Hirt, and Ueli Maurer. Resource-restricted indifferentiability. In *Advances in Cryptology – EUROCRYPT 2013*, Lecture Notes in Computer Science, Athens, Greece, May 26–30, 2013. Springer, Berlin, Germany. To appear.

[81] Bert den Boer and Antoon Bosselaers. Collisions for the compressin function of MD5. In Tor Helleseth, editor, *Advances in Cryptology – EUROCRYPT'93*, volume 765 of *Lecture Notes in Computer Science*, pages 293–304, Lofthus, Norway, May 23–27, 1993. Springer, Berlin, Germany.

[82] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):655–654, 1976.

[83] Itai Dinur, Orr Dunkelman, and Adi Shamir. Collision attacks on up to 5 rounds of SHA-3 using generalized internal differentials. In *Fast Software Encryption – FSE 2013*, Lecture Notes in Computer Science, Singapore, March 11–13, 2013. Springer, Berlin, Germany. To appear.

[84] Hans Dobbertin. The status of MD5 after a recent attack. *CryptoBytes*, 2(2):1–6, 1996.

[85] Yevgeniy Dodis and Prashant Puniya. On the relation between the ideal cipher and the random oracle models. In Shai Halevi and Tal Rabin, editors, *TCC 2006: 3rd Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 184–206, New York, NY, USA, March 4–7, 2006. Springer, Berlin, Germany.

[86] Yevgeniy Dodis and Prashant Puniya. Getting the best out of existing hash functions; or what if we are stuck with SHA? In Steven M. Bellovin, Rosario Gennaro, Angelos D. Keromytis, and Moti Yung, editors, *ACNS 2008: 6th International Conference on Applied Cryptography and Network Security*, volume 5037 of *Lecture Notes in Computer Science*, pages 156–173, New York, NY, USA, June 3–6, 2008. Springer, Berlin, Germany.

[87] Yevgeniy Dodis, Leonid Reyzin, Ronald L. Rivest, and Emily Shen. Indifferentiability of permutation-based compression functions and tree-based modes of operation, with applications to MD6. In Orr Dunkelman, editor, *Fast Software Encryption – FSE 2009*, volume 5665 of *Lecture Notes in Computer Science*, pages 104–121, Leuven, Belgium, February 22–25, 2009. Springer, Berlin, Germany.

[88] Yevgeniy Dodis, Thomas Ristenpart, and Thomas Shrimpton. Salvaging Merkle-Damgård for practical applications. In Antoine Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 371–388, Cologne, Germany, April 26–30, 2009. Springer, Berlin, Germany.

[89] Yevgeniy Dodis and John P. Steinberger. Message authentication codes from unpredictable block ciphers. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 267–285, Santa Barbara, CA, USA, August 16–20, 2009. Springer, Berlin, Germany.

[90] Lei Duo and Chao Li. Improved collision and preimage resistance bounds on PGV schemes. Cryptology ePrint Archive, Report 2006/462, 2006.

[91] 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*, volume 739 of *Lecture Notes in Computer Science*, pages 210–224, Fujiyoshida, Japan, November 11–14, 1991. Springer, Berlin, Germany.

[92] Niels Ferguson, Stefan Lucks, Bruce Schneier, Doug Whiting, Mihir Bellare, Tadayoshi Kohno, Jon Callas, and Jesse Walker. Engineering comparison of SHA-3 candidates, 2010.

[93] Niels Ferguson, Stefan Lucks, Bruce Schneier, Doug Whiting, Mihir Bellare, Tadayoshi Kohno, Jon Callas, and Jesse Walker. The Skein hash function family. Submission to NIST's SHA-3 competition, 2010.

[94] 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*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, Santa Barbara, CA, USA, August 1987. Springer, Berlin, Germany.

[95] FIPS 140-2: Security Policy for IBM 'CryptoLite in C' (CLiC), 2003.

[96] Marc Fischlin. Security of NMAC and HMAC based on non-malleability. In Tal Malkin, editor, *Topics in Cryptology – CT-RSA 2008*, volume 4964 of *Lecture Notes in Computer Science*, pages 138–154, San Francisco, CA, USA, April 7–11, 2008. Springer, Berlin, Germany.

[97] Ewan Fleischmann, Christian Forler, and Michael Gorski. Classification of the SHA-3 candidates. Cryptology ePrint Archive, Report 2008/511, 2008.

[98] Ewan Fleischmann, Christian Forler, and Stefan Lucks. The collision security of MDC-4. In Aikaterini Mitrokotsa and Serge Vaudenay, editors, *AFRICACRYPT 2012: 5th International Conference on Cryptology in Africa*, volume 7374 of *Lecture Notes in Computer Science*, pages 252–269, Ifrance, Morocco, July 10–12, 2012. Springer, Berlin, Germany.

[99] Ewan Fleischmann, Christian Forler, Stefan Lucks, and Jakob Wenzel. The collision security of MDC-4. Cryptology ePrint Archive, Report 2012/096, 2012. Full version of [98].

[100] Ewan Fleischmann, Michael Gorski, and Stefan Lucks. Security of cyclic double block length hash functions. In Matthew G. Parker, editor, *12th IMA International Conference on Cryptography and Coding*, volume 5921 of *Lecture Notes in Computer Science*, pages 153–175, Cirencester, UK, December 15–17, 2009. Springer, Berlin, Germany.

[101] Pierre-Alain Fouque, Jacques Stern, and Sébastien Zimmer. Cryptanalysis of tweaked versions of SMASH and reparation. In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *SAC 2008: 15th Annual International Workshop on Selected Areas in Cryptography*, volume 5381 of *Lecture Notes in Computer Science*, pages 136–150, Sackville, New Brunswick, Canada, August 14–15, 2008. Springer, Berlin, Germany.

[102] Praveen Gauravaram and John Kelsey. Linear-XOR and additive checksums don't protect Damgård-Merkle hashes from generic attacks. In Tal Malkin, editor, *Topics in Cryptology – CT-RSA 2008*, volume 4964 of *Lecture Notes in Computer Science*, pages 36–51, San Francisco, CA, USA, April 7–11, 2008. Springer, Berlin, Germany.

[103] Praveen Gauravaram, John Kelsey, Lars R. Knudsen, and Søren Thomsen. On hash functions using checksums. *International Journal of Information Security*, 9(2):137–151, 2010.

[104] Praveen Gauravaram, Lars R. Knudsen, Krystian Matusiewicz, Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. Grøstl – a SHA-3 candidate. Submission to NIST's SHA-3 competition, 2011.

[105] Zheng Gong, Xuejia Lai, and Kefei Chen. A synthetic indifferentiability analysis of some block-cipher-based hash functions. *Designs, Codes and Cryptography*, 48(3):293–305, 2008.

[106] Jian Guo, Thomas Peyrin, and Axel Poschmann. The PHOTON family of lightweight hash functions. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 222–239, Santa Barbara, CA, USA, August 14–18, 2011. Springer, Berlin, Germany.

[107] Shai Halevi, William E. Hall, and Charanjit S. Jutla. The hash function "Fugue". Submission to NIST's SHA-3 competition, 2009.

[108] Shai Halevi, William E. Hall, Charanjit S. Jutla, and Arnab Roy. Weak ideal functionalities for designing random oracles with applications to Fugue, 2010.

[109] Shai Halevi and Hugo Krawczyk. Strengthening digital signatures via randomized hashing. In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 41–59, Santa Barbara, CA, USA, August 20–24, 2006. Springer, Berlin, Germany.

[110] Shoichi Hirose. Secure block ciphers are not sufficient for one-way hash functions in the Preneel-Govaerts-Vandewalle model. In Kaisa Nyberg and Howard M. Heys, editors, *SAC 2002: 9th Annual International Workshop on*

Selected Areas in Cryptography, volume 2595 of *Lecture Notes in Computer Science*, pages 339–352, St. John's, Newfoundland, Canada, August 15–16, 2003. Springer, Berlin, Germany.

[111] Shoichi Hirose. Provably secure double-block-length hash functions in a black-box model. In Choonsik Park and Seongtaek Chee, editors, *ICISC 2004: 7th International Conference on Information Security and Cryptology*, volume 3506 of *Lecture Notes in Computer Science*, pages 330–342, Seoul, Korea, December 2–3, 2004. Springer, Berlin, Germany.

[112] Shoichi Hirose. Some plausible constructions of double-block-length hash functions. In Matthew J. B. Robshaw, editor, *Fast Software Encryption – FSE 2006*, volume 4047 of *Lecture Notes in Computer Science*, pages 210–225, Graz, Austria, March 15–17, 2006. Springer, Berlin, Germany.

[113] 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*, volume 4833 of *Lecture Notes in Computer Science*, pages 113–129, Kuching, Malaysia, December 2–6, 2007. Springer, Berlin, Germany.

[114] Walter Hohl, Xuejia Lai, Thomas Meier, and Christian Waldvogel. Security of iterated hash functions based on block ciphers. In Douglas R. Stinson, editor, *Advances in Cryptology – CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, pages 379–390, Santa Barbara, CA, USA, August 22–26, 1994. Springer, Berlin, Germany.

[115] Thomas Holenstein, Robin Künzler, and Stefano Tessaro. The equivalence of the random oracle model and the ideal cipher model, revisited. In Lance Fortnow and Salil P. Vadhan, editors, *43rd Annual ACM Symposium on Theory of Computing*, pages 89–98, San Jose, California, USA, June 6–8, 2011. ACM Press.

[116] Deukjo Hong and Daesung Kwon. New preimage attack on MDC-4. Cryptology ePrint Archive, Report 2012/633, 2012.

[117] ISO/IEC 10118-2:2010. Information technology – Security techniques – Hash-functions – Part 2: Hash-functions using an n-bit block cipher, 1994, revised in 2010.

[118] Dimitar Jetchev, Onur Özen, and Martijn Stam. Collisions are not incidental: A compression function exploiting discrete geometry. In Ronald Cramer, editor, *TCC 2012: 9th Theory of Cryptography Conference*, volume 7194 of *Lecture Notes in Computer Science*, pages 303–320, Taormina, Sicily, Italy, March 19–21, 2012. Springer, Berlin, Germany.

[119] Antoine Joux. Multicollisions in iterated hash functions. application to cascaded constructions. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 306–316, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Berlin, Germany.

[120] Antoine Joux and Thomas Peyrin. Hash functions and the (amplified) boomerang attack. In Alfred Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 244–263, Santa Barbara, CA, USA, August 19–23, 2007. Springer, Berlin, Germany.

[121] David Kahn. *The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet.* Scribner, 1996.

[122] John Kelsey and Tadayoshi Kohno. Herding hash functions and the Nostradamus attack. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 183–200, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Berlin, Germany.

[123] John Kelsey and Bruce Schneier. Second preimages on n-bit hash functions for much less than 2^{n} work. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 474–490, Aarhus, Denmark, May 22–26, 2005. Springer, Berlin, Germany.

[124] Auguste Kerckhoffs. La cryptographie militaire. *Journal des Sciences Militaires*, IX:5–83, 1883.

[125] Dmitry Khovratovich, Ivica Nikolic, and Christian Rechberger. Rotational rebound attacks on reduced Skein. In Masayuki Abe, editor, *Advances in Cryptology – ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 1–19, Singapore, December 5–9, 2010. Springer, Berlin, Germany.

[126] Lars R. Knudsen, Xuejia Lai, and Bart Preneel. Attacks on fast double block length hash functions. *Journal of Cryptology*, 11(1):59–72, 1998.

[127] Lars R. Knudsen, Florian Mendel, Christian Rechberger, and Søren S. Thomsen. Cryptanalysis of MDC-2. In Antoine Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 106–120, Cologne, Germany, April 26–30, 2009. Springer, Berlin, Germany.

[128] Lars R. Knudsen and Bart Preneel. Hash functions based on block ciphers and quaternary codes. In Kwangjo Kim and Tsutomu Matsumoto, editors,

Advances in Cryptology – ASIACRYPT'96, volume 1163 of *Lecture Notes in Computer Science*, pages 77–90, Kyongju, Korea, November 3–7, 1996. Springer, Berlin, Germany.

[129] Lars R. Knudsen and Bart Preneel. Fast and secure hashing based on codes. In Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 485–498, Santa Barbara, CA, USA, August 17–21, 1997. Springer, Berlin, Germany.

[130] Lars R. Knudsen and Bart Preneel. Construction of secure and fast hash functions using nonbinary error-correcting codes. *IEEE Transactions on Information Theory*, 48(9):2524–2539, 2002.

[131] Lars R. Knudsen, Christian Rechberger, and Søren S. Thomsen. The Grindahl hash functions. In Alex Biryukov, editor, *Fast Software Encryption – FSE 2007*, volume 4593 of *Lecture Notes in Computer Science*, pages 39–57, Luxembourg, Luxembourg, March 26–28, 2007. Springer, Berlin, Germany.

[132] Lars R. Knudsen and Vincent Rijmen. Known-key distinguishers for some block ciphers. In Kaoru Kurosawa, editor, *Advances in Cryptology – ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 315–324, Kuching, Malaysia, December 2–6, 2007. Springer, Berlin, Germany.

[133] Kazuyuki Kobayashi, Jun Ikegami, Shin'ichiro Matsuo, Kazuo Sakiyama, and Kazuo Ohta. Evaluation of hardware performance for the SHA-3 candidates using SASEBO-GII. Cryptology ePrint Archive, Report 2010/010, 2010.

[134] Özgül Küçük. *Design and Analysis of Cryptographic Hash Functions.* PhD thesis, Katholieke Universiteit Leuven, Leuven, 2012.

[135] Özgül Küçük. The hash function Hamsi. Submission to NIST's SHA-3 competition, 2009.

[136] Hidenori Kuwakado and Masakatu Morii. Indifferentiability of single-block-length and rate-1 compression functions. *IEICE Transactions*, 90-A(10):2301–2308, 2007.

[137] Xuejia Lai and James L. Massey. Hash function based on block ciphers. In Rainer A. Rueppel, editor, *Advances in Cryptology – EUROCRYPT'92*, volume 658 of *Lecture Notes in Computer Science*, pages 55–70, Balatonfüred, Hungary, May 24–28, 1992. Springer, Berlin, Germany.

[138] Xuejia Lai, James L. Massey, and Sean Murphy. Markov ciphers and differential cryptoanalysis. In Donald W. Davies, editor, *Advances in Cryptology – EUROCRYPT'91*, volume 547 of *Lecture Notes in Computer*

Science, pages 17–38, Brighton, UK, April 8–11, 1991. Springer, Berlin, Germany.

[139] Jooyoung Lee. Provable security of the Knudsen-Preneel compression functions. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, volume 7658 of *Lecture Notes in Computer Science*, pages 504–525, Beijing, China, December 2–6, 2012. Springer, Berlin, Germany.

[140] Jooyoung Lee and Deukjo Hong. Collision resistance of the JH hash function. *IEEE Transactions on Information Theory*, 58(3):1992–1995, 2012.

[141] Jooyoung Lee and Daesung Kwon. Security of single-permutation-based compression functions. Cryptology ePrint Archive, Report 2009/145, 2009.

[142] Jooyoung Lee and Daesung Kwon. The security of Abreast-DM in the ideal cipher model. *IEICE Transactions*, 94-A(1):104–109, 2011.

[143] Jooyoung Lee and Je Hong Park. Preimage resistance of lpmkr with r=m-1. *Information Processing Letters*, 110(14–15):602–608, 2010.

[144] Jooyoung Lee and Martijn Stam. MJH: A faster alternative to MDC-2. In Aggelos Kiayias, editor, *Topics in Cryptology – CT-RSA 2011*, volume 6558 of *Lecture Notes in Computer Science*, pages 213–236, San Francisco, CA, USA, February 14–18, 2011. Springer, Berlin, Germany.

[145] Jooyoung Lee, Martijn Stam, and John P. Steinberger. The collision security of Tandem-DM in the ideal cipher model. Cryptology ePrint Archive, Report 2010/409, 2010. Full version of [146].

[146] 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*, volume 6841 of *Lecture Notes in Computer Science*, pages 561–577, Santa Barbara, CA, USA, August 14–18, 2011. Springer, Berlin, Germany.

[147] Jooyoung Lee, Martijn Stam, and John P. Steinberger. The preimage security of double-block-length compression functions. Cryptology ePrint Archive, Report 2011/210, 2011.

[148] Jooyoung Lee and John P. Steinberger. Multi-property-preserving domain extension using polynomial-based modes of operation. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 573–596, French Riviera, May 30 – June 3, 2010. Springer, Berlin, Germany.

[149] Wonil Lee, Donghoon Chang, Sangjin Lee, Soo Hak Sung, and Mridul Nandi. New parallel domain extenders for UOWHF. In Chi-Sung Laih, editor, *Advances in Cryptology – ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 208–227, Taipei, Taiwan, November 30 – December 4, 2003. Springer, Berlin, Germany.

[150] Wonil Lee, Donghoon Chang, Sangjin Lee, Soo Hak Sung, and Mridul Nandi. Construction of UOWHF: Two new parallel methods. *IEICE Transactions*, 88-A(1):49–58, 2005.

[151] Anja Lehmann and Stefano Tessaro. A modular design for hash functions: Towards making the mix-compress-mix approach practical. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 364–381, Tokyo, Japan, December 6–10, 2009. Springer, Berlin, Germany.

[152] Moses Liskov. Constructing an ideal hash function from weak ideal compression functions. In Eli Biham and Amr M. Youssef, editors, *SAC 2006: 13th Annual International Workshop on Selected Areas in Cryptography*, volume 4356 of *Lecture Notes in Computer Science*, pages 358–375, Montreal, Canada, August 17–18, 2006. Springer, Berlin, Germany.

[153] Stefan Lucks. A failure-friendly design principle for hash functions. In Bimal K. Roy, editor, *Advances in Cryptology – ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 474–494, Chennai, India, December 4–8, 2005. Springer, Berlin, Germany.

[154] Stefan Lucks. A collision-resistant rate-1 double-block-length hash function. Symmetric Cryptography, Dagstuhl Seminar Proceedings 07021, 2007.

[155] Yiyuan Luo, Zheng Gong, Ming Duan, Bo Zhu, and Xuejia Lai. Revisiting the indifferentiability of PGV hash functions. Cryptology ePrint Archive, Report 2009/265, 2009.

[156] Atul Luykx. The scope of indifferentiability and an application to BLAKE. Master's thesis, Katholieke Universiteit Leuven, Leuven, 2012.

[157] Atul Luykx, Elena Andreeva, Bart Mennink, and Bart Preneel. Impossibility results for indifferentiability with resets. Cryptology ePrint Archive, Report 2012/644, 2012.

[158] Stephen Matyas, Carl Meyer, and Jonathan Oseas. Generating strong one-way functions with cryptographic algorithm. *IBM Technical Disclosure Bulletin*, 27(10A):5658–5659, 1985.

[159] Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle

methodology. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39, Cambridge, MA, USA, February 19–21, 2004. Springer, Berlin, Germany.

[160] Florian Mendel, Christian Rechberger, Martin Schläffer, and Søren S. Thomsen. The rebound attack: Cryptanalysis of reduced Whirlpool and Grøstl. In Orr Dunkelman, editor, *Fast Software Encryption – FSE 2009*, volume 5665 of *Lecture Notes in Computer Science*, pages 260–276, Leuven, Belgium, February 22–25, 2009. Springer, Berlin, Germany.

[161] Alfred Menezes, Paul van Oorschot, and Scott Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[162] Bart Mennink. Increasing the flexibility of the herding attack. *Information Processing Letters*, 112(3):98–105, 2012.

[163] 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*, volume 7658 of *Lecture Notes in Computer Science*, pages 526–543, Beijing, China, December 2–6, 2012. Springer, Berlin, Germany.

[164] Bart Mennink. On the collision and preimage security of MDC-4 in the ideal cipher model. *Designs, Codes and Cryptography*, 2013. To appear.

[165] 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*, volume 7417 of *Lecture Notes in Computer Science*, pages 330–347, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Berlin, Germany.

[166] Ralph C. Merkle. Protocols for public key cryptosystems. In *1980 IEEE Symposium on Security and Privacy*, pages 122–134, Oakland, California, USA, April 14–16, 1980. IEEE Computer Society Press.

[167] Ralph C. Merkle. One way hash functions and DES. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 428–446, Santa Barbara, CA, USA, August 20–24, 1990. Springer, Berlin, Germany.

[168] Ralph Charles Merkle. *Secrecy, Authentication and Public Key Systems.* PhD thesis, UMI Research Press, 1979.

[169] Carl Meyer and Michael Schilling. Secure program load with manipulation detection code. In *Proceedings of Securicom*, pages 111–130, 1988.

[170] Shoji Miyaguchi, Kazuo Ohta, and Masahiko Iwata. Confirmation that some hash functions are not collision free. In Ivan Damgård, editor, *Advances in Cryptology – EUROCRYPT'90*, volume 473 of *Lecture Notes in Computer Science*, pages 326–343, Aarhus, Denmark, May 21–24, 1990. Springer, Berlin, Germany.

[171] Dustin Moody, Souradyuti Paul, and Daniel Smith-Tone. Improved indifferentiability security bound for the JH mode. NIST's 3rd SHA-3 Candidate Conference 2012, 2012.

[172] Yusuke Naito. On the indifferentiable hash functions in the multi-stage security games. Cryptology ePrint Archive, Report 2012/014, 2012.

[173] Yusuke Naito, Kazuki Yoneyama, Lei Wang, and Kazuo Ohta. How to confirm cryptosystems security: The original Merkle-Damgård is still alive! In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 382–398, Tokyo, Japan, December 6–10, 2009. Springer, Berlin, Germany.

[174] Mridul Nandi. Towards optimal double-length hash functions. In Subhamoy Maitra, C. E. Veni Madhavan, and Ramarathnam Venkatesan, editors, *Progress in Cryptology - INDOCRYPT 2005: 6th International Conference in Cryptology in India*, volume 3797 of *Lecture Notes in Computer Science*, pages 77–89, Bangalore, India, December 10–12, 2005. Springer, Berlin, Germany.

[175] Mridul Nandi. Characterizing padding rules of MD hash functions preserving collision security. In Colin Boyd and Juan Manuel González Nieto, editors, *ACISP 2009: 14th Australasian Conference on Information Security and Privacy*, volume 5594 of *Lecture Notes in Computer Science*, pages 171–184, Brisbane, Australia, July 1–3, 2009. Springer, Berlin, Germany.

[176] Mridul Nandi, Wonil Lee, Kouichi Sakurai, and Sangjin Lee. Security analysis of a 2/3-rate double length compression function in the black-box model. In Henri Gilbert and Helena Handschuh, editors, *Fast Software Encryption – FSE 2005*, volume 3557 of *Lecture Notes in Computer Science*, pages 243–254, Paris, France, February 21–23, 2005. Springer, Berlin, Germany.

[177] Gregory Neven, Nigel P. Smart, and Bogdan Warinschi. Hash function requirements for Schnorr signatures. *Journal of Mathematical Cryptology*, 3(1):69–87, 2009.

[178] Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in*

Computer Science, pages 111–126, Santa Barbara, CA, USA, August 18–22, 2002. Springer, Berlin, Germany.

[179] National Institute for Standards and Technology. Announcing Request for Candidate Algorithm Nominations for the Advance Encryption Standard (AES), September 1997. http://csrc.nist.gov/archive/aes/pre-round1/aes_9709.htm.

[180] National Institute for Standards and Technology. Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA3) Family, November 2007. http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf.

[181] Onur Özen, Thomas Shrimpton, and Martijn Stam. Attacking the Knudsen-Preneel compression functions. In Seokhie Hong and Tetsu Iwata, editors, *Fast Software Encryption – FSE 2010*, volume 6147 of *Lecture Notes in Computer Science*, pages 94–115, Seoul, Korea, February 7–10, 2010. Springer, Berlin, Germany.

[182] Onur Özen and Martijn Stam. Another glance at double-length hashing. In Matthew G. Parker, editor, *12th IMA International Conference on Cryptography and Coding*, volume 5921 of *Lecture Notes in Computer Science*, pages 176–201, Cirencester, UK, December 15–17, 2009. Springer, Berlin, Germany.

[183] Onur Özen and Martijn Stam. Collision attacks against the Knudsen-Preneel compression functions. In Masayuki Abe, editor, *Advances in Cryptology – ASIACRYPT 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 76–93, Singapore, December 5–9, 2010. Springer, Berlin, Germany.

[184] Password Hashing Competition, April 2013. https://password-hashing.net.

[185] Thomas Peyrin, Henri Gilbert, Frédéric Muller, and Matt Robshaw. Combining compression functions and block cipher-based hash functions. In Xuejia Lai and Kefei Chen, editors, *Advances in Cryptology – ASIACRYPT 2006*, volume 4284 of *Lecture Notes in Computer Science*, pages 315–331, Shanghai, China, December 3–7, 2006. Springer, Berlin, Germany.

[186] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli M. Maurer, editor, *Advances in Cryptology – EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 387–398, Saragossa, Spain, May 12–16, 1996. Springer, Berlin, Germany.

[187] Bart Preneel, Antoon Bosselaers, René Govaerts, and Joos Vandewalle. Collision free hash functions based on blockcipher algorithms. In *1989 IEEE*

International Carnahan Conference on Security Technology, pages 203–210, 1989.

[188] 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*, volume 773 of *Lecture Notes in Computer Science*, pages 368–378, Santa Barbara, CA, USA, August 22–26, 1994. Springer, Berlin, Germany.

[189] Michael Rabin. Digitalized signatures. In *Foundations of Secure Computation '78*, pages 155–166, New York, 1978. Academic Press.

[190] Mohammad Reza Reyhanitabar and Willy Susilo. Mix-compress-mix revisited: Dispensing with non-invertible random injection oracles. Cryptology ePrint Archive, Report 2012/479, 2012.

[191] Mohammad Reza Reyhanitabar, Willy Susilo, and Yi Mu. Enhanced security notions for dedicated-key hash functions: Definitions and relationships. In Seokhie Hong and Tetsu Iwata, editors, *Fast Software Encryption – FSE 2010*, volume 6147 of *Lecture Notes in Computer Science*, pages 192–211, Seoul, Korea, February 7–10, 2010. Springer, Berlin, Germany.

[192] 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*, volume 6632 of *Lecture Notes in Computer Science*, pages 487–506, Tallinn, Estonia, May 15–19, 2011. Springer, Berlin, Germany.

[193] Thomas Ristenpart and Thomas Shrimpton. How to build a hash function from any collision-resistant function. In Kaoru Kurosawa, editor, *Advances in Cryptology – ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 147–163, Kuching, Malaysia, December 2–6, 2007. Springer, Berlin, Germany.

[194] Ronald Rivest. The MD5 message-digest algorithm. Request for Comments (RFC) 1321, April 1992. `http://tools.ietf.org/html/rfc1321`.

[195] Ronald L. Rivest. The MD4 message digest algorithm. In Alfred J. Menezes and Scott A. Vanstone, editors, *Advances in Cryptology – CRYPTO'90*, volume 537 of *Lecture Notes in Computer Science*, pages 303–311, Santa Barbara, CA, USA, August 11–15, 1991. Springer, Berlin, Germany.

[196] Phillip Rogaway. Formalizing human ignorance. In Phong Q. Nguyen, editor, *Progress in Cryptology - VIETCRYPT 06: 1st International Conference on Cryptology in Vietnam*, volume 4341 of *Lecture Notes in Computer Science*, pages 211–228, Hanoi, Vietnam, September 25–28, 2006. Springer, Berlin, Germany.

[197] 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 – FSE 2004*, volume 3017 of *Lecture Notes in Computer Science*, pages 371–388, New Delhi, India, February 5–7, 2004. Springer, Berlin, Germany.

[198] Phillip Rogaway and John P. Steinberger. Constructing cryptographic hash functions from fixed-key blockciphers. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 433–450, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Berlin, Germany.

[199] Phillip Rogaway and John P. Steinberger. Security/efficiency tradeoffs for permutation-based hashing. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 220–236, Istanbul, Turkey, April 13–17, 2008. Springer, Berlin, Germany.

[200] Palash Sarkar. Construction of universal one-way hash functions: Tree hashing revisited. *Discrete Applied Mathematics*, 155(16):2174–2180, 2007.

[201] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[202] Yannick Seurin. *Primitives et protocoles cryptographiques à sécurité prouvée.* PhD thesis, Université de Versailles Saint-Quentin-en-Yvelines, France, 2009.

[203] Yannick Seurin and Thomas Peyrin. Security analysis of constructions combining FIL random oracles. In Alex Biryukov, editor, *Fast Software Encryption – FSE 2007*, volume 4593 of *Lecture Notes in Computer Science*, pages 119–136, Luxembourg, Luxembourg, March 26–28, 2007. Springer, Berlin, Germany.

[204] Claude Shannon. A mathematical theory of communication. *Bell Systems Technical Journal*, 27:379–423, 623–656, 1948.

[205] Claude Shannon. Communication theory of secrecy systems. *Bell Systems Technical Journal*, 28(4):656–715, 1949.

[206] 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, *ICALP 2008: 35th International Colloquium on Automata, Languages and Programming, Part II*, volume 5126 of *Lecture Notes in Computer Science*, pages 643–654, Reykjavik, Iceland, July 7–11, 2008. Springer, Berlin, Germany.

[207] Daniel R. Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In Kaisa Nyberg, editor, *Advances in Cryptology – EUROCRYPT'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 334–345, Espoo, Finland, May 31 – June 4, 1998. Springer, Berlin, Germany.

[208] John L. Smith. *The design of Lucifer: a cryptographic device for data communications.* IBM Research Report RC 3326, 1971.

[209] Martijn Stam. Beyond uniformity: Better security/efficiency tradeoffs for compression functions. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 397–412, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Berlin, Germany.

[210] Martijn Stam. Blockcipher-based hashing revisited. In Orr Dunkelman, editor, *Fast Software Encryption – FSE 2009*, volume 5665 of *Lecture Notes in Computer Science*, pages 67–83, Leuven, Belgium, February 22–25, 2009. Springer, Berlin, Germany.

[211] John P. Steinberger. The collision intractability of MDC-2 in the ideal-cipher model. In Moni Naor, editor, *Advances in Cryptology – EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 34–51, Barcelona, Spain, May 20–24, 2007. Springer, Berlin, Germany.

[212] John P. Steinberger. Stam's collision resistance conjecture. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 597–615, French Riviera, May 30 – June 3, 2010. Springer, Berlin, Germany.

[213] 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*, volume 7417 of *Lecture Notes in Computer Science*, pages 384–405, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Berlin, Germany.

[214] Douglas R. Stinson. Some observations on the theory of cryptographic hash functions. *Designs, Codes and Cryptography*, 38(2):259–277, 2006.

[215] Kazuhiro Suzuki, Dongvu Tonien, Kaoru Kurosawa, and Koji Toyota. Birthday paradox for multi-collisions. In Min Surp Rhee and Byoungcheon Lee, editors, *ICISC 2006: 9th International Conference on Information Security and Cryptology*, volume 4296 of *Lecture Notes in Computer Science*, pages 29–40, Busan, Korea, November 30 – December 1, 2006. Springer, Berlin, Germany.

[216] Stefan Tillich, Martin Feldhofer, Mario Kirschbaum, Thomas Plos, Jörn-Marc Schmidt, and Alexander Szekely. High-speed hardware implementations of BLAKE, Blue Midnight Wish, CubeHash, ECHO, Fugue, Grøstl, Hamsi, JH, Keccak, Luffa, Shabal, SHAvite-3, SIMD, and Skein. Cryptology ePrint Archive, Report 2009/510, 2009.

[217] Serge Vaudenay. Decorrelation: A theory for block cipher security. *Journal of Cryptology*, 16(4):249–286, September 2003.

[218] David Wagner. A generalized birthday problem. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 288–303, Santa Barbara, CA, USA, August 18–22, 2002. Springer, Berlin, Germany.

[219] Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu. Cryptanalysis of the hash functions MD4 and RIPEMD. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 1–18, Aarhus, Denmark, May 22–26, 2005. Springer, Berlin, Germany.

[220] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding collisions in the full SHA-1. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 17–36, Santa Barbara, CA, USA, August 14–18, 2005. Springer, Berlin, Germany.

[221] Xiaoyun Wang and Hongbo Yu. How to break MD5 and other hash functions. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 19–35, Aarhus, Denmark, May 22–26, 2005. Springer, Berlin, Germany.

[222] Dai Watanabe. A note on the security proof of Knudsen-Preneel construction of a hash function, 2006.

[223] Wikipedia. Parazoa, April 2013. `http://en.wikipedia.org/wiki/Parazoa`.

[224] Hongjun Wu. The hash function JH. Submission to NIST's SHA-3 competition, 2011.

[225] Kazuki Yoneyama, Satoshi Miyagawa, and Kazuo Ohta. Leaky random oracle (extended abstract). In Joonsang Baek, Feng Bao, Kefei Chen, and Xuejia Lai, editors, *ProvSec 2008: 2nd International Conference on Provable Security*, volume 5324 of *Lecture Notes in Computer Science*, pages 226–240, Shanghai, China, October 31 – November 1, 2008. Springer, Berlin, Germany.

[226] Gideon Yuval. How to swindle Rabin. *Cryptologia*, 3(3):187–191, 1979.

A Appendix to Chapter 2

A.1 Chosen-Target-Forced-Midfix Preimage Security of Iterated Hashing

In this section, we prove chosen-target-forced-midfix security of the Merkle-Damgård domain extender, Theorem 2.4.3. We consider adversaries making $q = q_1 + q_2$ queries to their random oracle f. Associated to the queries made in the attack we consider a directed graph (V, A). A compression function execution $f(h_{i-1}, M_i) = h_i$ corresponds to an arc $h_{i-1} \xrightarrow{M_i} h_i$ in the graph. The graph is initialized as $(\{\mathsf{iv}\}, \emptyset)$. We denote by $(V(j), A(j))$ (for $j = -q_1, \ldots, q_2$) the subgraph of (V, A) after the j^{th} query. Hence, after the first phase of the attack we are left with graph $(V(0), A(0))$.

A path $h_i \xrightarrow{M_{i+1}} h_{i+1} \cdots \xrightarrow{M_l} h_l$ in a graph is called "$\mathsf{v\text{-}tail}$" (for valid tail) if $M_{i+1}\| \cdots \|M_l$ forms the suffix of a correctly padded message. Formally, it is $\mathsf{v\text{-}tail}$ if there exists $M_i \in (\mathbb{Z}_2^m)^\infty$ such that

$$M_i\|M_{i+1}\| \cdots \|M_l\|M_{l+1} \in \mathsf{rng}(\mathsf{pad})\,.$$

Intuitively, it means that $h_i \to h_l$ is formed by a valid sequence of message blocks and can possibly occur at the end of a hash function execution. Notice that the $\mathsf{v\text{-}tail}$ property of a path carries over to its sub-suffixes. For two state values $h_i, h_l \in \mathbb{Z}_2^n$, we define

$$\mathsf{dist}_{h_i \to h_l}(j) = \big\{0 \leq k < \infty \;\big|\; (V(j), A(j)) \text{ contains } \mathsf{v\text{-}tail}\ h_i \longrightarrow h_l \text{ of length } k\big\}\,.$$

For a $P \in \mathbb{Z}_2^p$, a path $h_i \xrightarrow{M_{i+1}} h_{i+1} \cdots \xrightarrow{M_k} h_k$ is called "P-comprising" if $M_{i+1}\| \cdots \|M_k$ contains P as a substring.

Proof of Theorem 2.4.3. Let $\mathcal{A}$ be any CTFM adversary making $q_1 + q_2$ queries to its random function f. In other words, at a high level the attack works as follows. (1) The adversary makes q_1 queries to f, (2) it commits to digest z, (3) it receives a random challenge $P \xleftarrow{\$} \mathbb{Z}_2^p$, (4) it makes q_2 queries to f, and (5) it responds with R_1, R_2 such that $\mathcal{H}(R_1\|P\|R_2) = z$ and $\big|R_1\|P\|R_2\big| \leq Lm$. Denote by $\mathbf{Pr}(\mathsf{suc}_{\mathcal{A}}(j))$ for $j = -q_1, \ldots, q_2$ the success probability of $\mathcal{A}$ after the j^{th} query. Obviously, $\mathbf{Adv}^{\text{ctfm}}_{\mathcal{H}^f}(\mathcal{A}) = \mathbf{Pr}(\mathsf{suc}_{\mathcal{A}}(q_2))$. In the first phase of the attack, the adversary arbitrarily makes q_1 queries to f, and then outputs a commitment z. Consequently, it receives challenge $P \xleftarrow{\$} \mathbb{Z}_2^p$. Consider the following event E_0:

$$\mathsf{E}_0 : \quad (V(0), A(0)) \text{ contains a } P\text{-comprising path.}$$

E_0 captures the event of $\mathcal{A}$ guessing P in the first phase of the attack. We will split the success probability of the adversary, using the fact that it either did or did not guess P in the first phase. In other words, we can write $\mathbf{Pr}(\mathsf{suc}_{\mathcal{A}}(q_2)) \leq \mathbf{Pr}(\mathsf{suc}_{\mathcal{A}}(q_2) \mid \neg\mathsf{E}_0) + \mathbf{Pr}(\mathsf{E}_0)$. However, for the purpose of the analysis we introduce two more events $\mathsf{E}_1, \mathsf{E}_2$. Let $\tau > 0$ be any integral threshold.

$$\begin{aligned}
\mathsf{E}_1 : \quad & \max_{0 \leq k \leq L} |\{h \in V(q_2) \mid k = \min \mathsf{dist}_{h \to z}(q_2)\}| =: \alpha_1 > \tau\,; \\
\mathsf{E}_2 : \quad & \max_{h \in V(q_2)} |\{h' \in V(q_2) \mid (h', h) \in A(q_2)\}| =: \alpha_2 > 2\,.
\end{aligned}$$

E_1 sorts all nodes in $V(q_2)$ in classes with elements at (minimal) distance $0, 1, 2, \ldots, L$ from z, and considers the class with the maximal number of nodes. E_2 considers the event that $\mathcal{Q}$ contains a multi-collision of more than two compression function executions. By basic probability theory, we have

$$\begin{aligned}
\mathbf{Pr}(\mathsf{suc}_{\mathcal{A}}(q_2)) &\leq \mathbf{Pr}(\mathsf{suc}_{\mathcal{A}}(q_2) \mid \neg\mathsf{E}_0 \wedge \neg\mathsf{E}_1) + \mathbf{Pr}(\mathsf{E}_0 \vee \mathsf{E}_1) \\
&\leq \mathbf{Pr}(\mathsf{suc}_{\mathcal{A}}(q_2) \mid \neg\mathsf{E}_0 \wedge \neg\mathsf{E}_1) + \mathbf{Pr}(\mathsf{E}_0 \vee \mathsf{E}_1 \mid \neg\mathsf{E}_2) + \mathbf{Pr}(\mathsf{E}_2) \\
&\leq \mathbf{Pr}(\mathsf{suc}_{\mathcal{A}}(q_2) \mid \neg\mathsf{E}_0 \wedge \neg\mathsf{E}_1) + \mathbf{Pr}(\mathsf{E}_0 \mid \neg\mathsf{E}_2) \\
&\qquad + \mathbf{Pr}(\mathsf{E}_1 \mid \neg\mathsf{E}_2) + \mathbf{Pr}(\mathsf{E}_2)\,, \qquad \text{(A.1)}
\end{aligned}$$

and we consider the probabilities on the right hand side separately.

$\mathbf{Pr}(\mathsf{suc}_{\mathcal{A}}(q_2) \mid \neg\mathsf{E}_0 \wedge \neg\mathsf{E}_1)$. By $\neg\mathsf{E}_0$, P is not contained in $(V(0), A(0))$ yet, but it may be contained partially and hence the adversary will at least need to make one compression function execution. It may be the case that the adversary makes calls to the compression function for multiple strings P, and it may be the case that after it queried for P, it knows multiple paths of different length including P, but this does not violate the analysis. In general, the suffix R_2 of the attack covers at most $L-1$ message blocks. At any time in the attack, there are at most

$$|\{h \in V(q_2) \mid \mathsf{dist}_{h \to z}(q_2) \cap \{0, \ldots, L-1\}|$$

possible nodes for which a hit results in a collision. By $\neg\mathsf{E}_1$, this set is upper bounded by $(L-1)\tau$. As the adversary makes at most $q_2 \leq q$ compression function calls that may result in success, the total probability is upper bounded by $\frac{(L-1)\tau q}{2^n}$.

$\mathbf{Pr}(\mathsf{E}_0 \mid \neg\mathsf{E}_2)$. Notice that $\neg\mathsf{E}_2$ implies that all nodes in $(V(q_2), A(q_2))$, as well as all nodes in $(V(0), A(0))$, have at most two incoming arcs. We consider the probability that there exists a P-comprising path. The existence of such path implies the existence of an arc that supplies the last bit of P. Consider any arc $h_{j-1} \xrightarrow{M_j} h_j$, and let $M_j^{(i)}$ for $i = 1, \ldots, m$ denote the mth bit. Now, we can analyze the probability that $P \xleftarrow{\$} \mathbb{Z}_2^p$ is included as a substring of a path in $(V(0), A(0))$, with $M_j^{(i)}$ corresponding to the last bit of P. Then, $\mathbf{Pr}\left(\mathsf{E}_0 \mid \neg\mathsf{E}_2\right)$ is upper bounded by this probability summed over all i and the number of arcs. We consider the probability for different values of i:

- $i \geq p$. P is integrally captured in M_j as $M_j^{(i-p+1)} \| \ldots \| M_j^{(i)} = P$. This happens with probability $1/2^p$ for predetermined M_j and random P;
- $i < p$. The first i bits of M_j correspond to the last i bits of P, and that the first $p-i$ bits of P are a suffix of any path ending in h_{j-1}. Let $\beta = \lceil (p-i)/m \rceil$. As by $\neg\mathsf{E}_2$ there are at most 2^β paths of length β blocks to h_{j-1}, we can upper bound the probability by $\frac{1}{2^i} \cdot \frac{2^\beta}{2^{p-i}} = \frac{2^\beta}{2^p}$.

Now, we can sum over all possible values of i and the number of queries q_1. We obtain

$$\mathbf{Pr}\left(\mathsf{E}_0 \mid \neg\mathsf{E}_2\right) \leq \begin{cases} \frac{(m+p-1)q_1}{2^p}, \text{ if } p \leq m\,, \\ \frac{m2^{\lceil p/m \rceil} q_1}{2^p}, \text{ if } p > m\,. \end{cases}$$

In both cases, we derive upper bound $\frac{m2^{\lceil p/m \rceil} q}{2^p}$, given $q_1 \leq q$.

$\mathbf{Pr}\left(\mathsf{E}_1 \mid \neg\mathsf{E}_2\right)$. Let k^* be minimal such that the maximum is achieved, and let $h_1, \ldots, h_{\alpha_1}$ be all nodes with distance k^* from z. Consider the subgraph $(\overline{V}, \overline{A})$ of $(V(q_2), A(q_2))$ consisting of all[1] paths $h_i \to z$ of length k^* edges (for $i = 1, \ldots, \alpha_1$). By ways of an elaborate case distinction (see Lemma A.1.1), one can show that for each node h in $(\overline{V}, \overline{A})$, all paths to z are of the same length. This in particular implies that the h_i's ($i = 1, \ldots, \alpha_1$) have no ingoing edge, and that z has no outgoing edge. Therefore, we can classify the nodes in the subgraph into sets: $\alpha_1^{k^*} = \alpha_1$ at distance k^* from z, $\alpha_1^{k^*-1}$ at distance $k^* - 1$, etc., $\alpha_1^0 = 1$ at distance 0. Notice that $\alpha_1^0, \ldots, \alpha_1^{k^*-1} < \alpha_1^{k^*}$ by definition, but it can be the case that $\alpha_1^{i-1} > \alpha_1^i$ (for $1 < i < k^*$) for technical reasons. By $\neg\mathsf{E}_2$, $\mathcal{Q}$ does not contain any 3-way collisions, but only 2-way collisions. The number of 2-way collisions between the nodes at distances i and $i-1$ equals $\max\{\alpha_1^i - \alpha_1^{i-1}, 0\}$. Consequently, the described subgraph, and hence $(V(q_2), A(q_2))$ itself, contains at least

$$\sum_{i=1}^{k^*} \max\{\alpha_1^i - \alpha_1^{i-1}, 0\} \geq \alpha_1^{k^*} - \alpha_1^0 = \alpha_1 - 1 \geq \tau$$

[1]In case of multiple paths of the same length starting from a node h_i, one arbitrarily chooses a path.

2-way collisions. Thus, the probability is upper bounded by $\binom{q}{\tau}\left(\frac{q}{2^n}\right)^{\tau} \leq \left(\frac{q^2 e}{\tau 2^n}\right)^{\tau}$, where the inequality holds due to Stirling's approximation.

Pr $(\mathsf{E_2})$. The occurrence of E_2 implies the presence of a 3-way collision in $\mathcal{Q}$, which exists with probability at most $q^3/2^{2n}$ only [215].

From (A.1) and above upper bounds on the three probabilities, we obtain:

$$\mathbf{Adv}^{\mathrm{ctfm}}_{\mathcal{H}^f}(\mathcal{A}) = \mathbf{Pr}\left(\mathsf{suc}_{\mathcal{A}}(q_2)\right) \leq \frac{(L-1)\tau q}{2^n} + \frac{m 2^{\lceil p/m \rceil} q}{2^p} + \left(\frac{q^2 e}{\tau 2^n}\right)^{\tau} + \frac{q^3}{2^{2n}}\,.$$

As this holds for any adversary making q queries, this completes the proof.

Lemma A.1.1. *For each node h in $(\overline{V}, \overline{A})$, all paths to z are of the same length.*

Proof. We show that the graph $(\overline{V}, \overline{A})$ defined in Theorem 2.4.3 does not contain a node with two paths of different lengths to z. Recall that $(\overline{V}, \overline{A})$ is constructed in the following manner. k^* is the minimal value for which there are the maximum number of nodes, α_1 with distance k^* to z. For each of the α_1 nodes $h_1, \ldots, h_{\alpha_1}$, we take any path of length k^* to z, and add it to $(\overline{V}, \overline{A})$. By definition, for each $i = 1, \ldots, \alpha_1$, there does not exist a path $h_i \to z$ of length shorter than k^* arcs. We show that for any node $h \in \overline{V}$, all paths to z are of the same length. The proof is done by contradiction: we will show that the existence of an h contradicting aforementioned property implies the existence of a path $h_i \to z$ (for some i) of length strictly shorter than k^* arcs. Notice that this result in particular implies that the h_i's ($i = 1, \ldots, \alpha_1$) have no ingoing edge, and that z has no outgoing edge.

Suppose there exists $h \in \overline{V}$ such that for some $M_1, M_2 \in (\mathbb{Z}_2^m)^\infty$ with $|M_1| < |M_2|$ the paths $h \xrightarrow{M_1} z$ and $h \xrightarrow{M_2} z$ are included in $(\overline{V}, \overline{A})$. If the path $h \xrightarrow{M_2} z$ is a subpath of any $h_i \to z$ for some i, one can replace this subpath by $h \xrightarrow{M_1} z$ to obtain a path $h_i \to z$ of length strictly shorter than k^* arcs, rendering contradiction. Thus, we assume that $h \xrightarrow{M_2} z$ is not integrally included as a subpath of any $h_i \to z$. We split up the path $h \xrightarrow{M_2} z$ intro three parts. Let $i \in \{1, \ldots, \alpha_1\}$ be such that the first edge of $h \xrightarrow{M_2} z$ is included in the path $h_i \to z$. Let $M_2^{(1)}$ be the maximal prefix of M_2 such that $h \xrightarrow{M_2^{(1)}} h^{(1)}$ (for some $h^{(1)}$) is a subpath of $h_i \to z$. Secondly, identify the edge leaving[2] $h^{(1)}$ in the path $h \xrightarrow{M_2} z$, and let i' be such that this edge is included in the path $h_{i'} \to z$. Let $M_2^{(2)}$ be of maximal length such that $M_2^{(1)} \| M_2^{(2)}$ is a prefix of M_2 and $h^{(1)} \xrightarrow{M_2^{(2)}} h^{(2)}$ (for some $h^{(2)}$) is a subpath of $h_{i'} \to z$. Thus, we split $h \xrightarrow{M_2} z$ into

$$h \xrightarrow{M_2^{(1)}} h^{(1)} \xrightarrow{M_2^{(2)}} h^{(2)} \xrightarrow{M_2^{(3)}} z\,, \tag{A.2}$$

[2]This edge exists, as $h \xrightarrow{M_2} z$ is not an integral subpath of any path $h_i \to z$.

where $|M_2^{(1)}|, |M_2^{(2)}| > 0$, $|M_2^{(3)}| \geq 0$, and

$$h_i \xrightarrow{M_3} h \xrightarrow{M_2^{(1)}} h^{(1)} \xrightarrow{M_4} z \text{ and } h_{i'} \xrightarrow{M_5} h^{(1)} \xrightarrow{M_2^{(2)}} h^{(2)} \xrightarrow{M_6} z\,, \tag{A.3}$$

for some $M_3, M_4, M_5, M_6 \in (\mathbb{Z}_2^m)^\infty$. Here, $M_2^{(1)}$ and $M_2^{(2)}$ are of maximal possible length, i.e., the first arcs of $h^{(1)} \xrightarrow{M_4} z$ and $h^{(1)} \xrightarrow{M_2^{(2)}} h^{(2)}$ are different and the first arcs of $h^{(2)} \xrightarrow{M_6} z$ and $h^{(2)} \xrightarrow{M_2^{(3)}} z$ are different.

If $h^{(1)} = h$, the path $h_i \xrightarrow{M_3} h \xrightarrow{M_4} z$ is in $(V(q_2), A(q_2))$ and of length shorter than k^* blocks, rendering contradiction. Similarly, if $h^{(2)} = h^{(1)}$, a shorter path $h_{i'} \to z$ can be found. Hence, we consider the case $h \neq h^{(1)} \neq h^{(2)}$, and make the following case distinction:

$|\mathbf{M_4}| \neq |\mathbf{M_2^{(2)} M_6}|$. One can combine the two paths described in (A.3) to obtain either a path $h_i \to z$ or $h_{i'} \to z$ of length strictly shorter than k^* arcs;

$|\mathbf{M_4}| = |\mathbf{M_2^{(2)} M_6}|$. We make the following case distinction:

- **$|\mathbf{M_6}| \geq |\mathbf{M_2^{(3)}}|$.** This means that $|M_4| \geq |M_2^{(2)} M_2^{(3)}|$ and hence $|M_2^{(1)} M_4| \geq |M_2| > |M_1|$. The path $h_i \xrightarrow{M_3} h \xrightarrow{M_1} z$ is thus strictly shorter than k^* arcs;
- **$|\mathbf{M_6}| < |\mathbf{M_2^{(3)}}|$.** One can do the same analysis with paths $h^{(2)} \xrightarrow{M_6} z$ and $h^{(2)} \xrightarrow{M_2^{(3)}} z$. But by construction $|M_2^{(3)}| < |M_2| - 2m$ so one will eventually end up with the same problem with $|M_2^{(3)}| = 0$, in which case one will not arrive in case 2b.

Concluding, there does not exist any node in $(\overline{V}, \overline{A})$ which has two paths of different lengths to z. □

B Appendix to Chapter 5

B.1 Proof of Lemma 5.3.5(i)

For $\mathsf{F}_{\mathsf{A}_k}$ ($k = 1, \ldots, 4$), where the matrices A_k are given in (5.5), the goal is to prove that $\lim_{n\to\infty} \mathbf{Adv}^{\mathrm{col}}_{\mathsf{F}_{\mathsf{A}_k}}(2^{n/2(1-\varepsilon)}) = 0$ for any $\varepsilon > 0$, demonstrating the asymptotic collision security of $\mathsf{F}_{\mathsf{A}_k}$. In the remainder of this section, π_1, π_2, π_3 are assumed to be three permutations taken uniformly at random from $\mathrm{Perm}(n)$.

The approach followed in this proof is as follows. Finding a collision for a function F_{A}, with A of the form (5.6), corresponds to obtaining query pairs $(x_1, y_1), (x_1', y_1')$ for π_1, $(x_2, y_2), (x_2', y_2')$ for π_2, and $(x_3, y_3), (x_3', y_3')$ for π_3 in the query history, such that:

$$(x_1, x_2) \neq (x_1', x_2')\,, \tag{B.1a}$$

$$a_{31}x_1 \oplus a_{32}x_2 \oplus a_{33}y_1 \oplus a_{34}y_2 = x_3\,, \tag{B.1b}$$

$$a_{31}x_1' \oplus a_{32}x_2' \oplus a_{33}y_1' \oplus a_{34}y_2' = x_3'\,, \tag{B.1c}$$

$$a_{41}x_1 \oplus a_{42}x_2 \oplus a_{43}y_1 \oplus a_{44}y_2 \oplus y_3 = a_{41}x_1' \oplus a_{42}x_2' \oplus a_{43}y_1' \oplus a_{44}y_2' \oplus y_3' \tag{B.1d}$$

(recall that the adversary is required to make the correct queries in order to form the collision). We will analyze the maximum probability of any adversary, making at most q queries to its oracles, in breaking (B.1), which equals $\mathbf{Adv}^{\mathrm{col}}_{\mathsf{F}_{\mathsf{A}}}(q)$ by definition. Denote by $\mathcal{Q}_i$ for $i = 1, \ldots, q$ the first i queries of the query history $\mathcal{Q}_q$. To bound $\mathbf{Adv}^{\mathrm{col}}_{\mathsf{F}_{\mathsf{A}}}(q)$, we distinguish among the possibilities that $x_j = x_j'$ (for $j = 1, 2, 3$). Formally, we obtain

$$\mathbf{Adv}^{\mathrm{col}}_{\mathsf{F}_{\mathsf{A}}}(q) \leq \sum_{\substack{\alpha_1,\alpha_2,\\ \alpha_3\in\{0,1\}}} \mathbf{Pr}\left(\text{solution for (B.1)} \;\wedge\; \bigwedge_{j=1}^{3} x_j = x_j' \equiv \alpha_j\right). \tag{B.2}$$

Returning to the four compression functions $\mathsf{F}_{\mathsf{A}_k}$ ($k \in \{1, 2, 3, 4\}$), this leaves 32 cases to be evaluated, but for some choices of $\alpha_1\alpha_2\alpha_3$ the probability on the right hand side of (B.2) equals 0. Starting with $\mathsf{F}_{\mathsf{A}_1}$, the cases $\alpha_1\alpha_2\alpha_3 \in \{110, 111\}$ violate (B.1a), the case 010 would give contradiction for (B.1d), and 101 would give a contradiction in lines (B.1b-B.1c). On the other side, the case $\alpha_1\alpha_2\alpha_3 = 000$

Table B.1. Case distinction for the analysis of (B.2) for $\mathsf{F}_{\mathsf{A}_k}$ ($k \in \{1,2,3,4\}$). In case of ✗, the choice $\alpha_1\alpha_2\alpha_3$ renders violation of one or more equations of (B.1), otherwise the case corresponds to event $\mathsf{E}_l(\mathcal{Q}_q)$ ($l \in \{1,\dots,13\}$) given in Figure B.1.

$\alpha_1\alpha_2\alpha_3$	000	001	010	011	100	101	110	111
$\mathsf{F}_{\mathsf{A}_1}$	$\mathsf{E}_1(\mathcal{Q}_q)$	$\mathsf{E}_5(\mathcal{Q}_q)$	✗ (B.1d)	$\mathsf{E}_9(\mathcal{Q}_q)$	$\mathsf{E}_{10}(\mathcal{Q}_q)$	✗ (B.1b-B.1c)	✗ (B.1a)	✗ (B.1a)
$\mathsf{F}_{\mathsf{A}_2}$	$\mathsf{E}_2(\mathcal{Q}_q)$	$\mathsf{E}_5(\mathcal{Q}_q)$	$\mathsf{E}_7(\mathcal{Q}_q)$	$\mathsf{E}_9(\mathcal{Q}_q)$	$\mathsf{E}_{11}(\mathcal{Q}_q)$	✗ (B.1b-B.1c)	✗ (B.1a)	✗ (B.1a)
$\mathsf{F}_{\mathsf{A}_3}$	$\mathsf{E}_3(\mathcal{Q}_q)$	$\mathsf{E}_6(\mathcal{Q}_q)$	$\mathsf{E}_8(\mathcal{Q}_q)$	$\mathsf{E}_9(\mathcal{Q}_q)$	✗ (B.1d)	$\mathsf{E}_{13}(\mathcal{Q}_q)$	✗ (B.1a)	✗ (B.1a)
$\mathsf{F}_{\mathsf{A}_4}$	$\mathsf{E}_4(\mathcal{Q}_q)$	✗ (B.1b-B.1d)	$\mathsf{E}_8(\mathcal{Q}_q)$	$\mathsf{E}_9(\mathcal{Q}_q)$	$\mathsf{E}_{12}(\mathcal{Q}_q)$	✗ (B.1d)	✗ (B.1a)	✗ (B.1a)

corresponds to $\mathsf{E}_1(\mathcal{Q}_q)$ of Figure B.1, and similarly the cases 001, 011, and 100 correspond to events $\mathsf{E}_5(\mathcal{Q}_q), \mathsf{E}_9(\mathcal{Q}_q), \mathsf{E}_{10}(\mathcal{Q}_q)$ of Figure B.1, respectively. A similar analysis can be applied to $\mathsf{F}_{\mathsf{A}_2}, \mathsf{F}_{\mathsf{A}_3}, \mathsf{F}_{\mathsf{A}_4}$ to obtain the results of Table B.1. In general, the following holds for $\mathsf{F}_{\mathsf{A}_k}$, where $k \in \{1,2,3,4\}$:

$$\text{collision for } \mathsf{F}_{\mathsf{A}_1} \implies \mathsf{E}_1(\mathcal{Q}_q) \vee \mathsf{E}_5(\mathcal{Q}_q) \vee \mathsf{E}_9(\mathcal{Q}_q) \vee \mathsf{E}_{10}(\mathcal{Q}_q)\,, \tag{B.3a}$$
$$\text{collision for } \mathsf{F}_{\mathsf{A}_2} \implies \mathsf{E}_2(\mathcal{Q}_q) \vee \mathsf{E}_5(\mathcal{Q}_q) \vee \mathsf{E}_7(\mathcal{Q}_q) \vee \mathsf{E}_9(\mathcal{Q}_q) \vee \mathsf{E}_{11}(\mathcal{Q}_q)\,, \tag{B.3b}$$
$$\text{collision for } \mathsf{F}_{\mathsf{A}_3} \implies \mathsf{E}_3(\mathcal{Q}_q) \vee \mathsf{E}_6(\mathcal{Q}_q) \vee \mathsf{E}_8(\mathcal{Q}_q) \vee \mathsf{E}_9(\mathcal{Q}_q) \vee \mathsf{E}_{13}(\mathcal{Q}_q)\,, \tag{B.3c}$$
$$\text{collision for } \mathsf{F}_{\mathsf{A}_4} \implies \mathsf{E}_4(\mathcal{Q}_q) \vee \mathsf{E}_8(\mathcal{Q}_q) \vee \mathsf{E}_9(\mathcal{Q}_q) \vee \mathsf{E}_{12}(\mathcal{Q}_q)\,. \tag{B.3d}$$

Here, the events $\mathsf{E}_l(\mathcal{Q}_q)$ ($l \in \{1,\dots,13\}$) are given in Figure B.1. Thus, it remains to analyze the probabilities of the events $\mathsf{E}_l(\mathcal{Q}_q)$ to occur, but we will analyze these under the condition that the previous query did not result in success, and some additional condition $\mathsf{C}(\mathcal{Q}_q) = \mathsf{C}_{1\vee\dots\vee 4}(\mathcal{Q}_q)$, where the claims $\mathsf{C}_1(\mathcal{Q}_q), \dots, \mathsf{C}_4(\mathcal{Q}_q)$ are given in Figure B.2:

$$\mathbf{Pr}\left(\mathsf{E}_l(\mathcal{Q}_q)\right) \le \mathbf{Pr}\left(\mathsf{E}_l(\mathcal{Q}_q) \mid \neg\mathsf{E}_l(\mathcal{Q}_{q-1}) \wedge \neg\mathsf{C}(\mathcal{Q}_q)\right) + \mathbf{Pr}\left(\mathsf{E}_l(\mathcal{Q}_{q-1}) \vee \mathsf{C}(\mathcal{Q}_q)\right)\,.$$

Similarly, the second probability of this bound can be split up further:

$$\begin{aligned}\mathbf{Pr}\left(\mathsf{E}_l(\mathcal{Q}_{q-1}) \vee \mathsf{C}(\mathcal{Q}_q)\right) \le \mathbf{Pr}\left(\mathsf{E}_l(\mathcal{Q}_{q-1}) \mid \neg\mathsf{E}_l(\mathcal{Q}_{q-2}) \wedge \neg\mathsf{C}(\mathcal{Q}_{q-1})\right) +&\\ \mathbf{Pr}\left(\mathsf{C}(\mathcal{Q}_q) \wedge \neg\mathsf{C}(\mathcal{Q}_{q-1})\right) + \mathbf{Pr}\left(\mathsf{E}_l(\mathcal{Q}_{q-2}) \vee \mathsf{C}(\mathcal{Q}_{q-1})\right)\,.&\end{aligned}$$

Applying this trick q times eventually gives the following probability bound on $\mathsf{E}_l(\mathcal{Q}_q)$:

$$\begin{aligned}\mathbf{Pr}\left(\mathsf{E}_l(\mathcal{Q}_q)\right) \le &\sum_{i=1}^{q} \mathbf{Pr}\left(\mathsf{E}_l(\mathcal{Q}_i) \mid \neg\mathsf{E}_l(\mathcal{Q}_{i-1}) \wedge \neg\mathsf{C}(\mathcal{Q}_i)\right) +\\ &\sum_{i=1}^{q} \mathbf{Pr}\left(\mathsf{C}(\mathcal{Q}_i) \wedge \neg\mathsf{C}(\mathcal{Q}_{i-1})\right)\,.\end{aligned} \tag{B.4}$$

The remainder of the section is now divided as follows. In Appendix B.1.1, we will bound the conditioned events $\mathsf{E}_l(\mathcal{Q}_q)$ to occur for $l \in \{1, \ldots, 13\}$ (first sum of (B.4)). Then, in Appendix B.1.2, a bound on the occurrence of $\mathsf{C}(\mathcal{Q}_q)$ is computed (second sum of (B.4)). The results are assembled in Appendix B.1.3, to prove Lemma 5.3.5(i).

$$\begin{aligned}
\mathsf{E}_{\mathbf{1}}(\mathcal{Q}) :\ & (x_1, y_1), (x_1', y_1'), (x_2, y_2), (x_2', y_2'), \\
& (x_3, y_3), (x_3', y_3') \in \mathcal{Q} \textbf{ s.t.} \\
& x_1 \neq x_1', x_2 \neq x_2', x_3 \neq x_3', \\
& x_1 \oplus y_1 \oplus x_2 = x_3, \\
& x_1' \oplus y_1' \oplus x_2' = x_3', \\
& x_2 \oplus y_2 \oplus y_3 = x_2' \oplus y_2' \oplus y_3'.
\end{aligned}$$

$$\begin{aligned}
\mathsf{E}_{\mathbf{2}}(\mathcal{Q}) :\ & (x_1, y_1), (x_1', y_1'), (x_2, y_2), (x_2', y_2'), \\
& (x_3, y_3), (x_3', y_3') \in \mathcal{Q} \textbf{ s.t.} \\
& x_1 \neq x_1', x_2 \neq x_2', x_3 \neq x_3', \\
& x_1 \oplus y_1 \oplus x_2 = x_3, \\
& x_1' \oplus y_1' \oplus x_2' = x_3', \\
& x_1 \oplus y_1 \oplus y_2 \oplus y_3 = x_1' \oplus y_1' \oplus y_2' \oplus y_3'.
\end{aligned}$$

$$\begin{aligned}
\mathsf{E}_{\mathbf{3}}(\mathcal{Q}) :\ & (x_1, y_1), (x_1', y_1'), (x_2, y_2), (x_2', y_2'), \\
& (x_3, y_3), (x_3', y_3') \in \mathcal{Q} \textbf{ s.t.} \\
& x_1 \neq x_1', x_2 \neq x_2', x_3 \neq x_3', \\
& x_1 \oplus y_1 \oplus x_2 \oplus y_2 = x_3, \\
& x_1' \oplus y_1' \oplus x_2' \oplus y_2' = x_3', \\
& x_1 \oplus y_1 \oplus y_3 = x_1' \oplus y_1' \oplus y_3'.
\end{aligned}$$

$$\begin{aligned}
\mathsf{E}_{\mathbf{4}}(\mathcal{Q}) :\ & (x_1, y_1), (x_1', y_1'), (x_2, y_2), (x_2', y_2'), \\
& (x_3, y_3), (x_3', y_3') \in \mathcal{Q} \textbf{ s.t.} \\
& x_1 \neq x_1', x_2 \neq x_2', x_3 \neq x_3', \\
& x_1 \oplus y_1 \oplus x_2 \oplus y_2 = x_3, \\
& x_1' \oplus y_1' \oplus x_2' \oplus y_2' = x_3', \\
& x_1 \oplus y_1 \oplus x_2 \oplus y_3 = x_1' \oplus y_1' \oplus x_2' \oplus y_3'.
\end{aligned}$$

$$\begin{aligned}
\mathsf{E}_{\mathbf{5}}(\mathcal{Q}) :\ & (x_1, y_1), (x_1', y_1'), (x_2, y_2), (x_2', y_2') \in \mathcal{Q} \textbf{ s.t.} \\
& x_1 \neq x_1', x_2 \neq x_2', \\
& x_1 \oplus y_1 \oplus x_2 = x_1' \oplus y_1' \oplus x_2', \\
& x_2 \oplus y_2 = x_2' \oplus y_2'.
\end{aligned}$$

$$\begin{aligned}
\mathsf{E}_{\mathbf{6}}(\mathcal{Q}) :\ & (x_1, y_1), (x_1', y_1'), (x_2, y_2), (x_2', y_2') \in \mathcal{Q} \textbf{ s.t.} \\
& x_1 \neq x_1', x_2 \neq x_2', \\
& x_1 \oplus y_1 = x_1' \oplus y_1', \\
& x_2 \oplus y_2 = x_2' \oplus y_2'.
\end{aligned}$$

$$\begin{aligned}
\mathsf{E}_{\mathbf{7}}(\mathcal{Q}) :\ & (x_1, y_1), (x_1', y_1'), (x_3, y_3), (x_3', y_3') \in \mathcal{Q} \textbf{ s.t.} \\
& x_1 \neq x_1', x_3 \neq x_3', \\
& x_1 \oplus y_1 \oplus x_3 = x_1' \oplus y_1' \oplus x_3', \\
& x_3 \oplus y_3 = x_3' \oplus y_3'.
\end{aligned}$$

$$\begin{aligned}
\mathsf{E}_{\mathbf{8}}(\mathcal{Q}) :\ & (x_1, y_1), (x_1', y_1'), (x_2, y_2), \\
& (x_3, y_3), (x_3', y_3') \in \mathcal{Q} \textbf{ s.t.} \\
& x_1 \neq x_1', x_3 \neq x_3', \\
& x_1 \oplus y_1 \oplus x_3 = x_1' \oplus y_1' \oplus x_3' = x_2 \oplus y_2, \\
& x_3 \oplus y_3 = x_3' \oplus y_3'.
\end{aligned}$$

$$\begin{aligned}
\mathsf{E}_{\mathbf{9}}(\mathcal{Q}) :\ & (x_1, y_1), (x_1', y_1') \in \mathcal{Q} \textbf{ s.t.} \\
& x_1 \neq x_1', \\
& x_1 \oplus y_1 = x_1' \oplus y_1'.
\end{aligned}$$

$$\begin{aligned}
\mathsf{E}_{\mathbf{10}}(\mathcal{Q}) :\ & (x_1, y_1), (x_2, y_2), (x_2', y_2'), \\
& (x_3, y_3), (x_3', y_3') \in \mathcal{Q} \textbf{ s.t.} \\
& x_2 \neq x_2', x_3 \neq x_3', \\
& x_2 \oplus x_3 = x_2' \oplus x_3' = x_1 \oplus y_1, \\
& x_2 \oplus y_2 \oplus y_3 = x_2' \oplus y_2' \oplus y_3'.
\end{aligned}$$

$$\begin{aligned}
\mathsf{E}_{\mathbf{11}}(\mathcal{Q}) :\ & (x_1, y_1), (x_2, y_2), (x_2', y_2'), \\
& (x_3, y_3), (x_3', y_3') \in \mathcal{Q} \textbf{ s.t.} \\
& x_2 \neq x_2', x_3 \neq x_3', \\
& x_2 \oplus x_3 = x_2' \oplus x_3' = x_1 \oplus y_1, \\
& y_2 \oplus y_3 = y_2' \oplus y_3'.
\end{aligned}$$

$$\begin{aligned}
\mathsf{E}_{\mathbf{12}}(\mathcal{Q}) :\ & (x_1, y_1), (x_2, y_2), (x_2', y_2'), \\
& (x_3, y_3), (x_3', y_3') \in \mathcal{Q} \textbf{ s.t.} \\
& x_2 \neq x_2', x_3 \neq x_3', \\
& x_2 \oplus y_2 \oplus x_3 = x_2' \oplus y_2' \oplus x_3' = x_1 \oplus y_1, \\
& x_2 \oplus y_3 = x_2' \oplus y_3'.
\end{aligned}$$

$$\begin{aligned}
\mathsf{E}_{\mathbf{13}}(\mathcal{Q}) :\ & (x_2, y_2), (x_2', y_2') \in \mathcal{Q} \textbf{ s.t.} \\
& x_2 \neq x_2', \\
& x_2 \oplus y_2 = x_2' \oplus y_2'.
\end{aligned}$$

Figure B.1. The events $\mathsf{E}_l(\mathcal{Q})$ ($l \in \{1, \ldots, 13\}$).

$\mathsf{C}_1(\mathcal{Q})$ **: for some c there exist more than τ_1 solutions in $\mathcal{Q}$ to one of the 3 equations:**

$$x_1 \oplus y_1 = c, \qquad x_2 \oplus y_2 = c, \qquad x_3 \oplus y_3 = c.$$

$\mathsf{C}_2(\mathcal{Q})$ **: for some c there exist more than τ_2 solutions in $\mathcal{Q}$ to one of the 3 equations:**

$$x_1 \oplus y_1 \oplus x_2 \oplus y_2 = c, \qquad x_1 \oplus y_1 \oplus x_3 \oplus y_3 = c, \qquad x_2 \oplus y_2 \oplus x_3 \oplus y_3 = c.$$

$\mathsf{C}_3(\mathcal{Q})$ **: there exist more than $\tau_2 q$ solutions in $\mathcal{Q}$ to one of the 3 equations:**

$$x_1 \oplus y_1 \oplus x_2 \oplus y_2 = x_3, \qquad x_1 \oplus y_1 \oplus x_3 \oplus y_3 = x_2, \qquad x_2 \oplus y_2 \oplus x_3 \oplus y_3 = x_1.$$

$\mathsf{C}_4(\mathcal{Q})$ **: there exist more than $q-1$ solutions in $\mathcal{Q}$ to one of the 3 equations:**

$x_2 \neq x_2',\ x_3 \neq x_3'$, **and**	$x_2 \neq x_2',\ x_3 \neq x_3'$, **and**	$x_2 \neq x_2',\ x_3 \neq x_3'$, **and**
$x_2 \oplus x_3 = x_2' \oplus x_3'$, **and**	$x_2 \oplus x_3 = x_2' \oplus x_3'$, **and**	$x_2 \oplus y_3 = x_2' \oplus y_3'$, **and**
$y_2 \oplus y_3 = y_2' \oplus y_3'$	$x_2 \oplus y_2 \oplus y_3 = x_2' \oplus y_2' \oplus y_3'$	$x_2 \oplus y_2 \oplus x_3 = x_2' \oplus y_2' \oplus x_3'$.

Figure B.2. Claims $\mathsf{C}_1(\mathcal{Q}), \ldots, \mathsf{C}_4(\mathcal{Q})$. We denote $\mathsf{C}(\mathcal{Q}) = \mathsf{C}_{1\vee\ldots\vee 4}(\mathcal{Q})$. The parameters $\tau_1 \geq 1$, $\tau_2 > 1$ are any fixed constants.

B.1.1 Bounding Occurrence of $\mathsf{E}_l(\mathcal{Q}_q)$, $l = 1, \ldots, 13$

In this section, we bound the conditioned events $\mathsf{E}_l(\mathcal{Q}_q)$ (for $l = 1, \ldots, 13$) to occur, more specifically the first sum of (B.4). The cases $l = 1, 2, 3, 4$ are found in Lemmas B.1.1-B.1.4, respectively. The cases $l = 10, 11, 12$ are found in Lemma B.1.5, and the remaining cases in Lemma B.1.6.

Let d_1 be the constant defined in Conjecture 5.3.2. On input of parameters (q, n), we define $\varepsilon_c(q, n) = \mathbf{Pr}\left(\beta > d_1 q \log q\right)$ of Conjecture 5.3.2. This quantity tends to 0 for $n \to \infty$ and for $q < 2^{n/2}$. In the remainder of this chapter, we define $q_1 = d_1 q \log q$. We will also consider Conjecture 5.3.2 on inputs (q_1, n) and $(\tau_2 q, n)$, where τ_2 is a parameter used in Figure B.2.

Lemma B.1.1. $\displaystyle\sum_{i=1}^{q} \mathbf{Pr}\left(\mathsf{E}_1(\mathcal{Q}_i) \mid \neg\mathsf{E}_1(\mathcal{Q}_{i-1}) \wedge \neg\mathsf{C}(\mathcal{Q}_i)\right) \leq \frac{\tau_2 q_1^2 \log q_1}{2^n - q} + 2\varepsilon_c(q, n) + \varepsilon_c(q_1, n)$.

Proof. We write $\mathsf{E}_1\mathsf{a}, \mathsf{E}_1\mathsf{b}$, and $\mathsf{E}_1\mathsf{c}$ for the three equations of $\mathsf{E}_1(\mathcal{Q})$ (Figure B.1). We first bound the success probability of the i^{th} query ($i = 1, \ldots, q$), and then we sum over all values of i.

Assume first the adversary makes a query $x_3^{(i)} \to y_3^{(i)} = \pi_3(x_3^{(i)})$ for $i = 1, \ldots, q$ (the same treatment holds for queries $x_3' \to y_3'$, $x_2 \to y_2$, or $x_2' \to y_2'$). By Conjecture 5.3.2, there exist at most q_1 tuples $(x_1', y_1'), (x_2', y_2'), (x_3', y_3')$ such that $\mathsf{E}_1\mathsf{b}$ is satisfied, except w.p. at most $\varepsilon_c(q, n)$. Denote L_i to be the number of tuples $(x_1, y_1), (x_2, y_2)$ that make equation $\mathsf{E}_1\mathsf{a}$ satisfied for $x_3^{(i)}$. For any of the tuples satisfying $\mathsf{E}_1\mathsf{b}$, and any of the tuples satisfying $\mathsf{E}_1\mathsf{a}$, equation $\mathsf{E}_1\mathsf{c}$ is satisfied with probability at most $\frac{1}{2^n - q}$. Considering all queries $x_3 \to y_3$, the adversary succeeds in breaking $\mathsf{E}_1(\mathcal{Q}_q)$ with probability at most $\sum_{i=1}^{q} \frac{L_i q_1}{2^n - q} + \varepsilon_c(q, n)$. Notice that by Conjecture 5.3.2, $\sum_{i=1}^{q} L_i \leq q_1$, except w.p. at most $\varepsilon_c(q,n)$. We thus obtain upper bound $\frac{q_1^2}{2^n-q} + 2\varepsilon_c(q,n)$.

Assume the adversary makes a query $y_2^{(i)} \to x_2^{(i)} = \pi_2^{-1}(y_2^{(i)})$ for $i = 1,\ldots,q$ (the same treatment holds for query $y_2' \to x_2'$). By Conjecture 5.3.2, there exist at most q_1 tuples $(x_1', y_1'), (x_2', y_2')$, (x_3', y_3') such that $\mathsf{E}_1\mathsf{b}$ is satisfied, except w.p. at most $\varepsilon_c(q,n)$. This consequently leads to at most q_1 possible values of $y_2^{(i)} \oplus x_2' \oplus y_2' \oplus y_3'$. For any of these values, by $\neg\mathsf{C}_2(\mathcal{Q}_i)$ there exist at most τ_2 combinations of queries $(x_1,y_1),(x_3,y_3)$ such that $y_2^{(i)} \oplus x_2' \oplus y_2' \oplus y_3' = x_1 \oplus y_1 \oplus x_3 \oplus y_3$. For any of these $\tau_2 q_1$ choices, the adversary breaks $\mathsf{E}_1(\mathcal{Q}_i)$ if $x_2^{(i)}$ hits $y_2^{(i)} \oplus y_3 \oplus x_2' \oplus y_2' \oplus y_3' = x_1 \oplus y_1 \oplus x_3$, hence with probability at most $\frac{1}{2^n-q}$. Considering all queries $y_2 \to x_2$, the adversary succeeds with probability at most $\frac{\tau_2 q q_1}{2^n-q} + \varepsilon_c(q,n)$.

Assume the adversary makes a query $y_3^{(i)} \to x_3^{(i)} = \pi_3^{-1}(y_3^{(i)})$ for $i = 1,\ldots,q$ (the same treatment holds for query $y_3' \to x_3'$). By Conjecture 5.3.2, there exist at most q_1 tuples $(x_1', y_1'), (x_2', y_2')$, (x_3', y_3') such that $\mathsf{E}_1\mathsf{b}$ is satisfied, except w.p. at most $\varepsilon_c(q,n)$. This consequently leads to at most q_1 possible values of $x_2' \oplus y_2' \oplus y_3'$. Denote L_i to be the number of choices $(x_2,y_2), x_2' \oplus y_2' \oplus y_3'$ that make equation $\mathsf{E}_1\mathsf{c}$ satisfied for $y_3^{(i)}$. For any of these L_i tuples, equation $\mathsf{E}_1\mathsf{a}$ is satisfied with probability at most $\frac{q}{2^n-q}$. Considering all queries $y_3 \to x_3$, the adversary succeeds in breaking $\mathsf{E}_1(\mathcal{Q}_q)$ with probability at most $\sum_{i=1}^{q} \frac{L_i q}{2^n-q} + \varepsilon_c(q,n)$. Notice that by Conjecture 5.3.2, $\sum_{i=1}^{q} L_i \leq d_1 q_1 \log q_1$, except w.p. at most $\varepsilon_c(q_1,n)$.[1] We thus obtain upper bound $\frac{d_1 q q_1 \log q_1}{2^n-q} + \varepsilon_c(q,n) + \varepsilon_c(q_1,n)$.

Assume the adversary makes a query $x_1^{(i)} \to y_1^{(i)} = \pi_1(x_1^{(i)})$ for $i = 1,\ldots,q$ (the same treatment holds for queries $y_1 \to x_1$, $x_1' \to y_1'$, or $y_1' \to x_1'$). By Conjecture 5.3.2, there exist at most q_1 tuples $(x_1', y_1'), (x_2', y_2'), (x_3', y_3')$ such that $\mathsf{E}_1\mathsf{b}$ is satisfied, except w.p. at most $\varepsilon_c(q,n)$. This consequently leads to at most q_1 possible values of $x_2' \oplus y_2' \oplus y_3'$. Again by Conjecture 5.3.2, there exist at most $d_1 q_1 \log q_1$ combinations of queries $(x_1', y_1'), (x_2', y_2'), (x_3', y_3'), (x_2, y_2), (x_3, y_3)$ such that $\mathsf{E}_1\mathsf{c}$ is satisfied, except w.p. at most $\varepsilon_c(q_1,n)$. For any of these combinations, the adversary breaks $\mathsf{E}_1(\mathcal{Q}_q)$ if $y_1^{(i)}$ hits $x_1^{(i)} \oplus x_2 \oplus x_3$, hence with probability at most $\frac{1}{2^n-q}$. Considering all queries $x_1 \to y_1$, the adversary succeeds with probability at most $\frac{d_1 q q_1 \log q_1}{2^n-q} + \varepsilon_c(q,n) + \varepsilon_c(q_1,n)$.

In any case, the success probability is at most $\frac{\tau_2 q_1^2 \log q_1}{2^n-q} + 2\varepsilon_c(q,n) + \varepsilon_c(q_1,n)$. $\square$

[1]Here, Z (the values $x_2 \oplus y_2$) is a set of q random elements, and the adversary is challenged to find two sets, one of size q (the values x_3) and one of size at most q_1 (the values $x_2' \oplus y_2' \oplus y_3'$), to maximize the number of matches. The success probability for this is upper bounded by the success probability in breaking this problem if all sets are of size q_1. Then, we can apply Conjecture 5.3.2 on (q_1, n).

Lemma B.1.2. $\sum_{i=1}^{q} \mathbf{Pr}\left(\mathsf{E}_2(\mathcal{Q}_i) \mid \neg\mathsf{E}_2(\mathcal{Q}_{i-1}) \wedge \neg\mathsf{C}(\mathcal{Q}_i)\right) \le \dfrac{\tau_2 q_1^2 \log q_1}{2^n - q} + 2\varepsilon_c(q,n) + \varepsilon_c(q_1,n)$.

Proof. We write $\mathsf{E}_2\mathsf{a}, \mathsf{E}_2\mathsf{b}$, and $\mathsf{E}_2\mathsf{c}$ for the three equations of $\mathsf{E}_2(\mathcal{Q})$ (Figure B.1). The approach is similar as before.

Assume first the adversary makes a query $x_3^{(i)} \to y_3^{(i)} = \pi_3(x_3^{(i)})$ for $i = 1, \dots, q$ (the same treatment holds for queries $x_3' \to y_3'$, $x_2 \to y_2$, or $x_2' \to y_2'$). By Conjecture 5.3.2, there exist at most q_1 tuples $(x_1', y_1'), (x_2', y_2'), (x_3', y_3')$ such that $\mathsf{E}_2\mathsf{b}$ is satisfied, except w.p. at most $\varepsilon_c(q,n)$. Denote L_i to be the number of tuples $(x_1, y_1), (x_2, y_2)$ that make equation $\mathsf{E}_2\mathsf{a}$ satisfied for $x_3^{(i)}$. For any of the tuples satisfying $\mathsf{E}_2\mathsf{b}$, and any of the tuples satisfying $\mathsf{E}_2\mathsf{a}$, equation $\mathsf{E}_2\mathsf{c}$ is satisfied with probability at most $\frac{1}{2^n-q}$. Considering all queries $x_3 \to y_3$, the adversary succeeds in breaking $\mathsf{E}_2(\mathcal{Q}_q)$ with probability at most $\sum_{i=1}^{q} \frac{L_i q_1}{2^n-q} + \varepsilon_c(q,n)$. Notice that by Conjecture 5.3.2, $\sum_{i=1}^{q} L_i \le q_1$, except w.p. at most $\varepsilon_c(q,n)$. We thus obtain upper bound $\frac{q_1^2}{2^n-q} + 2\varepsilon_c(q,n)$.

Assume the adversary makes a query $y_3^{(i)} \to x_3^{(i)} = \pi_3^{-1}(y_3^{(i)})$ for $i = 1, \dots, q$ (the same treatment holds for query $y_3' \to x_3'$, $y_2 \to x_2$, or $y_2' \to x_2'$). By Conjecture 5.3.2, there exist at most q_1 tuples $(x_1', y_1'), (x_2', y_2'), (x_3', y_3')$ such that $\mathsf{E}_2\mathsf{b}$ is satisfied, except w.p. at most $\varepsilon_c(q,n)$. This consequently leads to at most q_1 possible values of $y_3^{(i)} \oplus x_1' \oplus y_1' \oplus y_2' \oplus y_3'$. Again by Conjecture 5.3.2, there exist at most $d_1 q_1 \log q_1$ combinations of queries $(x_1', y_1'), (x_2', y_2'), (x_3', y_3'), (x_1, y_1)$, (x_2, y_2) such that $\mathsf{E}_2\mathsf{c}$ is satisfied for $y_3^{(i)}$, except w.p. at most $\varepsilon_c(q_1,n)$. For any of these combinations, the adversary breaks $\mathsf{E}_2(\mathcal{Q}_q)$ if $x_3^{(i)}$ hits $x_1 \oplus y_1 \oplus x_2$, hence with probability at most $\frac{1}{2^n-q}$. Considering all queries $y_3 \to x_3$, the adversary succeeds with probability at most $\frac{d_1 q q_1 \log q_1}{2^n-q} + \varepsilon_c(q,n) + \varepsilon_c(q_1,n)$.

Assume the adversary makes a query $x_1^{(i)} \to y_1^{(i)} = \pi_1(x_1^{(i)})$ for $i = 1, \dots, q$ (the same treatment holds for queries $y_1 \to x_1$, $x_1' \to y_1'$, or $y_1' \to x_1'$). By Conjecture 5.3.2, there exist at most q_1 tuples $(x_1', y_1'), (x_2', y_2'), (x_3', y_3')$ such that $\mathsf{E}_2\mathsf{b}$ is satisfied, except w.p. at most $\varepsilon_c(q,n)$. This consequently leads to at most q_1 possible values of $x_1' \oplus y_1' \oplus y_2' \oplus y_3'$. For any of these values, by $\neg\mathsf{C}_2(\mathcal{Q}_i)$ there exist at most τ_2 combinations of queries $(x_2, y_2), (x_3, y_3)$ such that $x_2 \oplus y_2 \oplus x_3 \oplus y_3 = x_1' \oplus y_1' \oplus y_2' \oplus y_3'$. For any of these $\tau_2 q_1$ choices, the adversary breaks $\mathsf{E}_2(\mathcal{Q}_i)$ if $y_1^{(i)}$ hits $x_1^{(i)} \oplus x_2 \oplus x_3$, hence with probability at most $\frac{1}{2^n-q}$. Considering all queries $x_1 \to y_1$, the adversary succeeds with probability at most $\frac{\tau_2 q q_1}{2^n-q} + \varepsilon_c(q,n)$.

In any case, the success probability is at most $\frac{\tau_2 q_1^2 \log q_1}{2^n-q} + 2\varepsilon_c(q,n) + \varepsilon_c(q_1,n)$. □

Lemma B.1.3. $\sum_{i=1}^{q} \mathbf{Pr}\left(\mathsf{E}_3(\mathcal{Q}_i) \mid \neg\mathsf{E}_3(\mathcal{Q}_{i-1}) \wedge \neg\mathsf{C}(\mathcal{Q}_i)\right) \le \dfrac{\tau_2^2 q^2}{2^n - q}$.

Proof. The case of $\mathsf{E}_3(\mathcal{Q}_q)$ is fairly similar to the case of $\mathsf{E}_2(\mathcal{Q}_q)$ (see Lemma B.1.2), with the difference that the usage of Conjecture 5.3.2 is replaced with $\neg\mathsf{C}_3(\mathcal{Q}_i)$. We write $\mathsf{E}_3\mathsf{a}, \mathsf{E}_3\mathsf{b}$, and $\mathsf{E}_3\mathsf{c}$ for the three equations of $\mathsf{E}_3(\mathcal{Q})$ (Figure B.1). The cases of queries $x_1, x_1', y_1, y_1', x_3$, and x_3' are the same as before (notice that $\mathsf{E}_3\mathsf{a}$ and $\mathsf{E}_3\mathsf{b}$ can be substituted in $\mathsf{E}_3\mathsf{c}$), resulting in a success probability upper bounded by $\frac{\tau_2^2 q^2}{2^n-q}$.

Assume the adversary makes a query $x_2^{(i)} \to y_2^{(i)} = \pi_2(x_2^{(i)})$ for $i = 1, \ldots, q$ (the same treatment holds for queries $y_2 \to x_2$, $x_2' \to y_2'$, or $y_2' \to x_2'$). As $\neg\mathsf{C}_3(\mathcal{Q}_i)$, there exist at most $\tau_2 q$ tuples $(x_1', y_1'), (x_2', y_2'), (x_3', y_3')$ such that $\mathsf{E}_3\mathsf{b}$ is satisfied, and this consequently leads to at most $\tau_2 q$ possible values of $x_2' \oplus y_2' \oplus x_3' \oplus y_3'$, which behave random. By a slight variant of $\neg\mathsf{C}_3(\mathcal{Q}_i)$ (as the previously mentioned values behave random), there exist at most $\tau_2^2 q$ combinations of queries $(x_1', y_1'), (x_2', y_2'), (x_3', y_3'), (x_1, y_1), (x_3, y_3)$ such that $x_1 \oplus y_1 \oplus y_3 = x_2' \oplus y_2' \oplus x_3' \oplus y_3'$. For any of these combinations, the adversary breaks $\mathsf{E}_3(\mathcal{Q}_q)$ if $y_2^{(i)}$ hits $x_1 \oplus y_1 \oplus x_2^{(i)} \oplus x_3$, hence with probability at most $\frac{1}{2^n-q}$. Considering all queries $x_2 \to y_2$, the adversary succeeds with probability at most $\frac{\tau_2^2 q^2}{2^n-q}$.

Assume the adversary makes a query $y_3^{(i)} \to x_3^{(i)} = \pi_3^{-1}(y_3^{(i)})$ for $i = 1, \ldots, q$ (the same treatment holds for query $y_3' \to x_3'$). As $\neg\mathsf{C}_3(\mathcal{Q}_i)$, there exist at most $\tau_2 q$ tuples $(x_1', y_1'), (x_2', y_2'), (x_3', y_3')$ such that $\mathsf{E}_3\mathsf{b}$ is satisfied, and this consequently leads to at most $\tau_2 q$ possible values of $x_2' \oplus y_2' \oplus x_3' \oplus y_3'$. Denote L_i to be the number of choices $(x_1, y_1), x_1' \oplus y_1' \oplus y_3'$ that make equation $\mathsf{E}_3\mathsf{c}$ satisfied for $y_3^{(i)}$. For any of these tuples, equation $\mathsf{E}_3\mathsf{a}$ is satisfied with probability at most $\frac{q}{2^n-q}$. Considering all queries $y_3 \to x_3$, the adversary succeeds in breaking $\mathsf{E}_3(\mathcal{Q}_q)$ with probability at most $\sum_{i=1}^{q} \frac{L_i q}{2^n-q}$. Notice that by a slight variant of $\neg\mathsf{C}_3(\mathcal{Q}_i)$, $\sum_{i=1}^{q} L_i \le \tau_2^2 q$. We thus obtain upper bound $\frac{\tau_2^2 q^2}{2^n-q}$.

In any case, the success probability is at most $\frac{\tau_2^2 q^2}{2^n-q}$. □

The slight variant of $\mathsf{C}_3(\mathcal{Q}_i)$ employed in the proof of Lemma B.1.3 embraces the case the adversary has i different tuples (x_1, y_1) and i different tuples (x_3, y_3), but (at most) $\tau_2 i$ different random values $z_2 = x_2' \oplus y_2' \oplus x_3' \oplus y_3'$, and aims at finding combinations such that $x_1 \oplus y_1 \oplus z_2 = x_3$ (or similar variants). Rather than introducing a separate claim for this, it suffices to condition for $\mathsf{E}_3(\mathcal{Q})$ on claim $\mathsf{C}(\mathcal{Q})$ where $\tau_2 q$ queries are allowed. This observation is used in Appendix B.1.3, where the results are assembled.

Lemma B.1.4. $\displaystyle\sum_{i=1}^{q} \mathbf{Pr}\left(\mathsf{E}_4(\mathcal{Q}_i) \mid \neg\mathsf{E}_4(\mathcal{Q}_{i-1}) \wedge \neg\mathsf{C}(\mathcal{Q}_i)\right) \le \frac{d_1 \tau_2^2 q^2 \log(\tau_2 q)}{2^n - q} + \varepsilon_c(\tau_2 q, n)$.

Proof. The case of $\mathsf{E}_4(\mathcal{Q}_q)$ is fairly similar to the case of $\mathsf{E}_1(\mathcal{Q}_q)$ (see Lemma B.1.1), with the difference that most of the usages of Conjecture 5.3.2 are replaced with $\neg\mathsf{C}_3(\mathcal{Q}_i)$. We write $\mathsf{E}_4\mathsf{a}, \mathsf{E}_4\mathsf{b}$, and $\mathsf{E}_4\mathsf{c}$ for the three equations of $\mathsf{E}_4(\mathcal{Q})$ (Figure B.1).

Note that equation $\mathsf{E}_4\mathsf{c}$ reduces to $y_2 \oplus x_3 \oplus y_3 = y_2' \oplus x_3' \oplus y_3'$ by substituting equations $\mathsf{E}_4\mathsf{a}, \mathsf{E}_4\mathsf{b}$. The cases of queries $x_1, x_1', y_1, y_1', x_3$, and x_3' are the same as before. For queries x_2, x_2', y_3, and y_3', one follows the same reasoning as for queries $y_2 \to x_2$ in the analysis of $\mathsf{E}_1(\mathcal{Q}_q)$, and for queries y_2, y_2' one follows the same reasoning as for $y_3 \to x_3$ in the analysis of $\mathsf{E}_1(\mathcal{Q}_q)$. Concretely, the success probability is at most $\frac{d_1\tau_2^2 q^2 \log(\tau_2 q)}{2^n - q} + \varepsilon_c(\tau_2 q, n)$. $\square$

Lemma B.1.5.

$$\sum_{i=1}^{q} \mathbf{Pr}\left(\mathsf{E}_l(\mathcal{Q}_i) \mid \neg\mathsf{E}_l(\mathcal{Q}_{i-1}) \wedge \neg\mathsf{C}(\mathcal{Q}_i)\right) \leq \begin{cases} \frac{\tau_1 q q_1 + q^2}{2^n - q} + \varepsilon_c(q,n) \; \textit{for } l = 10\,, \\ \frac{q q_1 + q^2}{2^n - q} + \varepsilon_c(q,n) \; \textit{for } l = 11\,, \\ \frac{\tau_1\tau_2 q^2 + q^2}{2^n - q} \; \textit{for } l = 12\,. \end{cases}$$

Proof. We start with $\mathsf{E}_{11}(\mathcal{Q}_q)$, and write $\mathsf{E}_{11}\mathsf{a}, \mathsf{E}_{11}\mathsf{b}$ for the two equations of $\mathsf{E}_{11}(\mathcal{Q})$ (Figure B.1). The approach is similar as before.

Assume first the adversary makes a query $x_3^{(i)} \to y_3^{(i)} = \pi_3(x_3^{(i)})$ for $i = 1, \ldots, q$ (the same treatment holds for queries $x_3' \to y_3'$, $x_2 \to y_2$, or $x_2' \to y_2'$). By Conjecture 5.3.2, there exist at most q_1 tuples $(x_2', y_2'), (x_3', y_3'), (x_1, y_1)$ such that the second equality of $\mathsf{E}_{11}\mathsf{a}$ is satisfied, except w.p. at most $\varepsilon_c(q,n)$. For any such choice, as $x_3^{(i)}$ is fixed there exists at most one (x_2, y_2) such that equation $\mathsf{E}_{11}\mathsf{a}$ is satisfied. For any of the combinations, $\mathsf{E}_{11}\mathsf{b}$ is satisfied with probability at most $\frac{1}{2^n - q}$. Considering all queries $x_3 \to y_3$, the adversary succeeds in breaking $\mathsf{E}_{11}(\mathcal{Q}_q)$ with probability at most $\frac{q q_1}{2^n - q} + \varepsilon_c(q,n)$.

Assume the adversary makes a query $y_3^{(i)} \to x_3^{(i)} = \pi_3^{-1}(y_3^{(i)})$ for $i = 1, \ldots, q$ (the same treatment holds for queries $y_3' \to x_3'$, $y_2 \to x_2$, or $y_2' \to x_2'$). By Conjecture 5.3.2, there exist at most q_1 tuples $(x_2', y_2'), (x_3', y_3'), (x_1, y_1)$ such that the second equality of $\mathsf{E}_{11}\mathsf{a}$ is satisfied, except w.p. at most $\varepsilon_c(q,n)$. For any such choice, as $y_3^{(i)}$ is fixed there exists at most one (x_2, y_2) such that equation $\mathsf{E}_{11}\mathsf{b}$ is satisfied. For any of the combinations, $\mathsf{E}_{11}\mathsf{a}$ is satisfied with probability at most $\frac{1}{2^n - q}$. Considering all queries $y_3 \to x_3$, the adversary succeeds in breaking $\mathsf{E}_{11}(\mathcal{Q}_q)$ with probability at most $\frac{q q_1}{2^n - q} + \varepsilon_c(q,n)$.

Assume the adversary makes a query $x_1^{(i)} \to y_1^{(i)} = \pi_1(x_1^{(i)})$ for $i = 1, \ldots, q$ (the same treatment holds for queries $y_1 \to x_1$). By $\neg\mathsf{C}_4(\mathcal{Q}_i)$, there exist at most $q - 1$ tuples (x_2, y_2), $(x_3, y_3), (x_2', y_2'), (x_3', y_3')$ such that $x_2 \oplus x_3 = x_2' \oplus x_3'$ and $y_2 \oplus y_3 = y_2' \oplus y_3'$. For any such tuple, the adversary succeeds with probability at most $\frac{1}{2^n - q}$. Considering all queries $x_1 \to y_1$, the adversary succeeds in breaking $\mathsf{E}_{11}(\mathcal{Q}_q)$ with probability at most $\frac{q^2}{2^n - q}$.

In any case, the success probability is at most $\frac{q q_1 + q^2}{2^n - q} + \varepsilon_c(q,n)$. The cases of $\mathsf{E}_{10}(\mathcal{Q}_q)$, $\mathsf{E}_{12}(\mathcal{Q}_q)$ are fairly similar, with the major difference that one needs to take into account that by $\neg\mathsf{C}_1(\mathcal{Q}_i)$ the query history contains at most τ_1 collisions $x_2 \oplus y_2 = c$, for any c, and similar for x_3. Additionally, for $\mathsf{E}_{12}(\mathcal{Q}_q)$ the usage of

Conjecture 5.3.2 is replaced with $\neg\mathsf{C}_3(\mathcal{Q}_i)$. Concretely, the success probabilities are upper bounded by $\frac{\tau_1 q q_1 + q^2}{2^n - q} + \varepsilon_c(q,n)$ for $\mathsf{E}_{10}(\mathcal{Q}_q)$ and $\frac{\tau_1\tau_2 q^2 + q^2}{2^n - q}$ for $\mathsf{E}_{12}(\mathcal{Q}_q)$. $\square$

Lemma B.1.6. $\sum_{i=1}^{q} \mathbf{Pr}\left(\mathsf{E}_l(\mathcal{Q}_i) \mid \neg\mathsf{E}_l(\mathcal{Q}_{i-1}) \wedge \neg\mathsf{C}(\mathcal{Q}_i)\right) = 0$ *for* $l = 5,\ldots,9,13$, *provided* $\tau_1 = 1$.

Proof. Starting with $\mathsf{E}_5(\mathcal{Q}_q)$, for the i^{th} query $x_2^{(i)} \leftrightarrow y_2^{(i)}$ for $i = 1,\ldots,q$. As $\neg\mathsf{C}_1(\mathcal{Q}_i)$ for $\tau_1 = 1$ there does not exist any other query (x_2', y_2') such that $x_2^{(i)} \oplus y_2^{(i)} = x_2' \oplus y_2'$. The same reasoning applies to the other events. $\square$

B.1.2 Bounding Occurrence of $\mathsf{C}(\mathcal{Q}_q)$

In this section, we bound the event $\mathsf{C}(\mathcal{Q}_q)$ to occur, more specifically the second sum of (B.4). Notice that this sum by probability theory equals $\mathbf{Pr}\left(\mathsf{C}(\mathcal{Q}_q)\right)$. However, we can split up the probability as follows.

$$\mathbf{Pr}\left(\mathsf{C}_{1\vee\ldots\vee4}\right) \le \mathbf{Pr}\left(\mathsf{C}_1\right) + \mathbf{Pr}\left(\mathsf{C}_2 \mid \neg\mathsf{C}_1\right) + \mathbf{Pr}\left(\mathsf{C}_3 \mid \neg\mathsf{C}_{1\vee2}\right) + \mathbf{Pr}\left(\mathsf{C}_4\right). \qquad \text{(B.5)}$$

The probability bounds on $\mathsf{C}_1(\mathcal{Q}_q),\ldots,\mathsf{C}_4(\mathcal{Q}_q)$ (the four quantities of (B.5)) are obtained in Lemmas B.1.7-B.1.10. The proofs rely on the following bound, which holds due to Stirling's approximation:

$$\binom{a}{b} \le \frac{a^b}{b!} \le \left(\frac{ae}{b}\right)^b.$$

Lemma B.1.7. $\mathbf{Pr}\left(\mathsf{C}_1(\mathcal{Q}_q)\right) \le 3\cdot 2^n \left(\frac{qe}{(\tau_1+1)(2^n-q)}\right)^{\tau_1+1}$.

Proof. We start with the first equation of $\mathsf{C}_1(\mathcal{Q}_q)$. Fix any c. For any (x_1, y_1), the equation is satisfied with probability at most $\frac{1}{2^n-q}$. More than τ_1 such tuples give a collision with probability at most

$$\binom{q}{\tau_1+1}\left(\frac{1}{2^n-q}\right)^{\tau_1+1} \le \left(\frac{qe}{(\tau_1+1)(2^n-q)}\right)^{\tau_1+1}.$$

Now, the result follows by quantifying over the number of choices for c and the number of equations of $\mathsf{C}_1(\mathcal{Q}_q)$. $\square$

Lemma B.1.8. $\mathbf{Pr}\left(\mathsf{C}_2(\mathcal{Q}_q) \mid \neg\mathsf{C}_1(\mathcal{Q}_q)\right) \le 3\cdot 2^n \left(\frac{\tau_1 2q^2 e}{(\tau_2+1)(2^n-q)}\right)^{(\tau_2+1)/\tau_1}$.

Proof. We start with the first equation of $\mathsf{C}_2(\mathcal{Q}_q)$. Fix any c. Assume the adversary makes a query $x_1^{(i)} \to y_1^{(i)}$ for $i = 1,\ldots,q$ (the same treatment holds for queries $y_1 \to x_1$, $x_2 \to y_2$, or $y_2 \to x_2$). For any tuple (x_2, y_2), the equation is satisfied with probability at most $\frac{1}{2^n-q}$. Thus, the query results in a solution with probability at most $\frac{q}{2^n-q}$. The adversary makes $2q$ queries, and as $\neg\mathsf{C}_1(\mathcal{Q}_q)$, each "hit" adds at most τ_1 solutions. Therefore, the adversary needs at least $(\tau_2+1)/\tau_1$ out of at most $2q$ hits. Consequently, $\mathcal{Q}_q$ contains more than τ_2 solutions to the first equation of $\mathsf{C}_2(\mathcal{Q}_q)$ with probability at most

$$\binom{2q}{(\tau_2+1)/\tau_1}\left(\frac{q}{2^n-q}\right)^{(\tau_2+1)/\tau_1} \le \left(\frac{\tau_1 2q^2 e}{(\tau_2+1)(2^n-q)}\right)^{(\tau_2+1)/\tau_1}.$$

Now, the result follows by quantifying over the number of choices for c and the number of equations of $\mathsf{C}_2(\mathcal{Q}_q)$. □

Lemma B.1.9. $\mathbf{Pr}\left(\mathsf{C}_3(\mathcal{Q}_q) \mid \neg\mathsf{C}_{1\vee 2}(\mathcal{Q}_q)\right) = 0$.

Proof. We start with the first equation of $\mathsf{C}_3(\mathcal{Q}_q)$, a similar reasoning applies to the other equations. As $\neg\mathsf{C}_2(\mathcal{Q}_q)$, for any (x_3, y_3) there are at most τ_2 solutions to $x_1 \oplus y_1 \oplus x_2 \oplus y_2 = x_3$. As $\mathcal{Q}_q$ contains q tuples (x_3, y_3), it contains at most $\tau_2 q$ solutions to the first equation of $\mathsf{C}_3(\mathcal{Q}_q)$. □

Lemma B.1.10. $\mathbf{Pr}\left(\mathsf{C}_4(\mathcal{Q}_q)\right) \le 3\left(\frac{q^2 e}{2^n-q}\right)^q$.

Proof. We start with the first equation of $\mathsf{C}_4(\mathcal{Q}_q)$. By construction, there are at most q^3 tuples of queries that satisfy $x_2 \oplus x_3 = x_2' \oplus x_3'$, and the adversary can achieve this number if it makes forward queries only, and we will assume henceforth. For any of these tuples, the second equation is satisfied with probability at most $\frac{1}{2^n-q}$. More than $q-1$ such tuples give a collision with probability at most

$$\binom{q^3}{q}\left(\frac{1}{2^n-q}\right)^q \le \left(\frac{q^3 e}{q(2^n-q)}\right)^q.$$

Now, the result follows by quantifying over the number of equations of $\mathsf{C}_4(\mathcal{Q}_q)$. □

B.1.3 Assembling the Results

Denote by $\mathsf{bndE}_l(q)$ the bound obtained for conditional events $\mathsf{E}_l(\mathcal{Q}_q)$ ($l = 1,\ldots,13$, Lemmas B.1.1-B.1.6), and by $\mathsf{bndC}_l(q)$ the bound obtained for claim $\mathsf{C}_l(\mathcal{Q}_q)$ ($l = 1,\ldots,4$, Lemmas B.1.7-B.1.10). Recall that $q_1 = d_1 q \log q$. Using slight variants of (B.3-B.5), one gets

$$\mathbf{Adv}^{\mathrm{col}}_{\mathsf{F}_{\mathsf{A}_1}}(q) \leq \sum_{\substack{l\in\{1,\\5,9,10\}}} \mathsf{bndE}_l(q) + \sum_{l=1}^{4} \mathsf{bndC}_l(q)\,,$$

$$\mathbf{Adv}^{\mathrm{col}}_{\mathsf{F}_{\mathsf{A}_2}}(q) \leq \sum_{\substack{l\in\{2,5,\\7,9,11\}}} \mathsf{bndE}_l(q) + \sum_{l=1}^{4} \mathsf{bndC}_l(q)\,,$$

$$\mathbf{Adv}^{\mathrm{col}}_{\mathsf{F}_{\mathsf{A}_3}}(q) \leq \sum_{\substack{l\in\{3,6,\\8,9,13\}}} \mathsf{bndE}_l(q) + \sum_{l=1}^{3} \mathsf{bndC}_l(q) + \sum_{l=1}^{3} \mathsf{bndC}_l(\tau_2 q)\,,$$

$$\mathbf{Adv}^{\mathrm{col}}_{\mathsf{F}_{\mathsf{A}_4}}(q) \leq \sum_{\substack{l\in\{4,\\8,9,12\}}} \mathsf{bndE}_l(q) + \sum_{l=1}^{4} \mathsf{bndC}_l(q)\,.$$

Note that the bound for $\mathbf{Adv}^{\mathrm{col}}_{\mathsf{F}_{\mathsf{A}_3}}(q)$ includes a third sum, the cause of which is explained after Lemma B.1.3. Let $\varepsilon > 0$. In order to prove $\lim_{n\to\infty} \mathbf{Adv}^{\mathrm{col}}_{\mathsf{F}_{\mathsf{A}_k}}(2^{n/2(1-\varepsilon)}) = 0$ (for $k = 1,2,3,4$), it suffices to prove that the separate bounds tend to zero for $n \to \infty$. The results of Appendices B.1.1 and B.1.2 hold provided $\tau_1 = 1$, but still hold for any choice of τ_2. Set $\tau_2 = n-1$. Note that the ε_c-parts in the bounds $\mathsf{bndE}_l(q)$ all approach 0 for $q = 2^{n/2(1-\varepsilon)}$: in particular, for large enough n we have $d_1 q \log q < 2^{n/2}$ and $\tau_2 q < 2^{n/2}$, and Conjecture 5.3.2 applies. Therefore, we omit the ε_c-parts for simplicity. The evaluations are now fairly the same and mostly rely on the fact that for large enough n the bounds behave like $\frac{\alpha n^\beta}{2^{n\varepsilon}}$ for some constants α, β. This function clearly tends to 0 for $n \to \infty$. We discuss $\mathsf{bndE}_1(q)$ and $\mathsf{bndC}_2(q)$ in detail. As we consider the asymptotic behavior, without loss of generality we assume n is large enough to obtain $\frac{1}{2^n-\tau_2 q} \leq \frac{2}{2^n}$ and $\frac{1}{2^n-q} \leq \frac{2}{2^n}$ for $q < 2^{n/2}$. By elementary mathematics,

$$\mathsf{bndE}_1(q) = \frac{(n-1)(d_1 q \log q)^2 \log(d_1 q \log q)}{2^n - q} \leq \frac{d_1^2 n^4 q^2}{2^n}\,.$$

Consequently, $\mathsf{bndE}_1(2^{n/2(1-\varepsilon)}) \leq \frac{d_1^2 n^4}{2^{n\varepsilon}}$, approaching 0 for $n \to \infty$. Similarly, for bndC_2:

$$\mathsf{bndC}'_2(q) := \mathsf{bndC}_2(\tau_2 q) = 3\cdot 2^n \left(\frac{2(\tau_2 q)^2 e}{n(2^n - \tau_2 q)}\right)^n \leq 3\cdot 2^n \left(\frac{4enq^2}{2^n}\right)^n,$$

which implies $\mathsf{bndC}'_2(2^{n/2(1-\varepsilon)}) \leq 3\left(\frac{8en}{2^{n\varepsilon}}\right)^n$, approaching 0 for $n \to \infty$.

B.2 Proof of Lemma 5.3.5(ii)

For $\mathsf{F}_{\mathsf{A}_k}$ ($k = 1, 2, 3, 4$), where the matrices A_k are given in (5.5), the goal is to prove that $\mathbf{Adv}^{\text{epre}}_{\mathsf{F}_{\mathsf{A}_k}}(q) = \Theta(q^2/2^n)$ for $k = 1, 3, 4$, and to prove $\lim_{n\to\infty} \mathbf{Adv}^{\text{epre}}_{\mathsf{F}_{\mathsf{A}_2}}(2^{2n/3(1-\varepsilon)}) = 0$ for any $\varepsilon > 0$, demonstrating the asymptotic preimages security of $\mathsf{F}_{\mathsf{A}_2}$. In the remainder of this section, π_1, π_2, π_3 are assumed to be three permutations taken uniformly at random from $\mathrm{Perm}(n)$, and z denotes any challenge. We will analyze the maximum probability of any adversary $\mathcal{A}$, making at most q queries to its oracles, in finding preimage for $\mathsf{F}_{\mathsf{A}_k}$, denoted as $\mathbf{Adv}^{\text{epre}}_{\mathsf{F}_{\mathsf{A}_k}}(q)$ by definition. Denote by $\mathcal{Q}_i$ for $i = 1, \dots, q$ the first i queries of the query history $\mathcal{Q}_q$. By construction, we have $\mathbf{Adv}^{\text{epre}}_{\mathsf{F}_{\mathsf{A}_1}}(q) = \mathbf{Pr}\left(\mathsf{E}_{14}(\mathcal{Q}_q)\right)$, $\mathbf{Adv}^{\text{epre}}_{\mathsf{F}_{\mathsf{A}_2}}(q) = \mathbf{Pr}\left(\mathsf{E}_{15}(\mathcal{Q}_q)\right)$, $\mathbf{Adv}^{\text{epre}}_{\mathsf{F}_{\mathsf{A}_3}}(q) = \mathbf{Pr}\left(\mathsf{E}_{16}(\mathcal{Q}_q)\right)$, and $\mathbf{Adv}^{\text{epre}}_{\mathsf{F}_{\mathsf{A}_4}}(q) = \mathbf{Pr}\left(\mathsf{E}_{17}(\mathcal{Q}_q)\right)$, where the events $\mathsf{E}_l(\mathcal{Q}_q)$ ($l \in \{14, \dots, 17\}$) are given in Figure B.3.

$\mathsf{E}_{14}(\mathcal{Q}) : (x_1, y_1), (x_2, y_2), (x_3, y_3) \in \mathcal{Q}$ **s.t.**
$x_1 \oplus y_1 \oplus x_2 = x_3,$
$x_2 \oplus y_2 \oplus y_3 = z.$

$\mathsf{E}_{15}(\mathcal{Q}) : (x_1, y_1), (x_2, y_2), (x_3, y_3) \in \mathcal{Q}$ **s.t.**
$x_1 \oplus y_1 \oplus x_2 = x_3,$
$x_1 \oplus y_1 \oplus y_2 \oplus y_3 = z.$

$\mathsf{E}_{16}(\mathcal{Q}) : (x_1, y_1), (x_2, y_2), (x_3, y_3) \in \mathcal{Q}$ **s.t.**
$x_1 \oplus y_1 \oplus x_2 \oplus y_2 = x_3,$
$x_1 \oplus y_1 \oplus y_3 = z.$

$\mathsf{E}_{17}(\mathcal{Q}) : (x_1, y_1), (x_2, y_2), (x_3, y_3) \in \mathcal{Q}$ **s.t.**
$x_1 \oplus y_1 \oplus x_2 \oplus y_2 = x_3,$
$y_2 \oplus x_3 \oplus y_3 = z.$

Figure B.3. Events $\mathsf{E}_l(\mathcal{Q})$ ($l \in \{14, \dots, 17\}$).

In Appendix B.2.1, we will upper bound $\mathbf{Pr}\left(\mathsf{E}_{15}(\mathcal{Q}_q)\right)$. In Appendix B.2.2, we provide a tight bound for $\mathbf{Pr}\left(\mathsf{E}_l(\mathcal{Q}_q)\right)$ for $l = 14, 16, 17$.

B.2.1 Bounding Occurrence of $\mathsf{E}_{15}(\mathcal{Q}_q)$

In order to bound $\mathbf{Pr}\left(\mathsf{E}_{15}(\mathcal{Q}_q)\right)$, we define a new claim $\mathsf{C}_{2'}(\mathcal{Q})$. This claim equals $\mathsf{C}_2(\mathcal{Q})$ of Figure B.2 restricted to the third equation and for fixed c instead of for any c. In a similar fashion as in (B.4)-(B.5), we obtain:

$$\begin{aligned}\mathbf{Pr}\left(\mathsf{E}_{15}(\mathcal{Q}_q)\right) \leq \sum_{i=1}^{q} &\mathbf{Pr}\left(\mathsf{E}_{15}(\mathcal{Q}_i) \mid \neg\mathsf{E}_{15}(\mathcal{Q}_{i-1}) \wedge \neg\mathsf{C}_{1\vee 2'}(\mathcal{Q}_i)\right) + \\ &\mathbf{Pr}\left(\mathsf{C}_1(\mathcal{Q}_q)\right) + \mathbf{Pr}\left(\mathsf{C}_{2'}(\mathcal{Q}_q) \mid \neg\mathsf{C}_1(\mathcal{Q}_q)\right) .\end{aligned} \tag{B.6}$$

In Lemma B.2.1 we bound the conditioned occurrence of $\mathsf{E}_{15}(\mathcal{Q}_q)$ (the first part (B.6)), and in Lemma B.2.2 we compute probability on $\mathsf{C}_{2'}(\mathcal{Q}_q)$ (the third part of (B.6)). A bound on $\mathsf{C}_1(\mathcal{Q}_q)$ is given in Lemma B.1.7. The results are then assembled to prove $\lim_{n\to\infty} \mathbf{Adv}^{\text{epre}}_{\mathsf{F}_{\mathsf{A}_2}}(2^{2n/3(1-\varepsilon)}) = 0$ for any $\varepsilon > 0$.

Let d_2 be the constant defined in Conjecture 5.3.2. On input of parameters (q, n), we define $\varepsilon'_c(q, n) = \mathbf{Pr}\left(\beta > d_2 q^{3/2}\right)$ of Conjecture 5.3.2. This quantity tends to 0 for $n \to \infty$ and for $q < 2^{2n/3}$.

Lemma B.2.1. $\sum_{i=1}^{q} \mathbf{Pr}\left(\mathsf{E}_{15}(\mathcal{Q}_i) \mid \neg\mathsf{E}_{15}(\mathcal{Q}_{i-1}) \wedge \neg\mathsf{C}_{1\vee 2'}(\mathcal{Q}_i)\right) \leq \frac{d_2 q^{3/2} + \tau_2 q}{2^n - q} + \varepsilon_c'(q,n)$.

Proof. We write $\mathsf{E}_{15}\mathsf{a}$ and $\mathsf{E}_{15}\mathsf{b}$ for the two equations of $\mathsf{E}_{15}(\mathcal{Q})$ (Figure B.3). The approach is similar as before.

Assume first the adversary makes a query $x_3^{(i)} \to y_3^{(i)} = \pi_3(x_3^{(i)})$ for $i = 1, \ldots, q$ (the same treatment holds for queries $y_3 \to x_3$, $x_2 \to y_2$, and $y_2 \to x_2$). Denote L_i to be the number of tuples $(x_1, y_1), (x_2, y_2)$ that make equation $\mathsf{E}_{15}\mathsf{a}$ satisfied for $x_3^{(i)}$. For any of these tuples, equation $\mathsf{E}_{15}\mathsf{b}$ is satisfied with probability at most $\frac{1}{2^n - q}$. Considering all queries $x_3 \to y_3$, the adversary succeeds in breaking $\mathsf{E}_{15}(\mathcal{Q}_q)$ with probability at most $\sum_{i=1}^{q} \frac{L_i}{2^n - q}$. Notice that by Conjecture 5.3.2, $\sum_{i=1}^{q} L_i \leq d_2 q^{3/2}$, except w.p. at most $\varepsilon_c'(q,n)$. We thus obtain upper bound $\frac{d_2 q^{3/2}}{2^n - q} + \varepsilon_c'(q,n)$.

Assume the adversary makes a query $x_1^{(i)} \to y_1^{(i)} = \pi_1(x_1^{(i)})$ for $i = 1, \ldots, q$ (the same treatment holds for queries $y_1 \to x_1$). As $\neg\mathsf{C}_{2'}(\mathcal{Q}_i)$, there exist at most τ_2 tuples (x_2, y_2), (x_3, y_3) such that $x_2 \oplus y_2 \oplus x_3 \oplus y_3 = z$ is satisfied. For any of these tuples, equation $\mathsf{E}_{15}\mathsf{a}$ is satisfied with probability at most $\frac{1}{2^n - q}$. Considering all queries $x_1 \to y_1$, the adversary succeeds in breaking $\mathsf{E}_{15}(\mathcal{Q}_q)$ with probability at most $\frac{\tau_2 q}{2^n - q}$.

In any case, the success probability is at most $\frac{d_2 q^{3/2} + \tau_2 q}{2^n - q} + \varepsilon_c'(q,n)$. □

Lemma B.2.2. $\mathbf{Pr}\left(\mathsf{C}_{2'}(\mathcal{Q}_q) \mid \neg\mathsf{C}_1(\mathcal{Q}_q)\right) \leq \left(\frac{\tau_1 2 q^2 e}{(\tau_2+1)(2^n - q)}\right)^{(\tau_2+1)/\tau_1}$.

Proof. The proof is identical to the proof of the bound for $\mathsf{C}_2(\mathcal{Q}_q)$ (Lemma B.1.8). Note that $\mathsf{C}_{2'}(\mathcal{Q}_q)$ consists of one equation, and one choice for c only. □

Let $\varepsilon > 0$. As in Appendix B.1.3, to prove $\lim_{n\to\infty} \mathbf{Adv}^{\text{epre}}_{\mathsf{F}_{\mathsf{A}_2}}(2^{2n/3(1-\varepsilon)}) = 0$, it suffices to prove that the separate bounds tend to zero for $n \to \infty$. Put $\tau_1 = 2$ and $\tau_2 = 2^{n/3} - 1$. The evaluations are now fairly the same, and we only discuss the bound $\mathsf{bndE}_{15}(q)$ on $\mathsf{E}_{15}(\mathcal{Q}_q)$. Again, by Conjecture 5.3.2 the ε_c'-part goes to 0 for $q = 2^{2n/3(1-\varepsilon)}$, and we omit it for simplicity. Notice that $\frac{1}{2^n - q} \leq \frac{2}{2^n}$ for $q \leq 2^{n-1}$. We obtain:

$$\mathsf{bndE}_{15}(q) = \frac{d_2 q^{3/2} + (2^{n/3} - 1)q}{2^n - q} \leq 2\frac{d_2 q^{3/2} + 2^{n/3} q}{2^n}.$$

Consequently, $\mathsf{bndE}_{15}(2^{2n/3(1-\varepsilon)}) \leq \frac{2d_2}{2^{n\varepsilon}} + \frac{2}{2^{2n/3\varepsilon}}$, which approaches 0 for $n \to \infty$.

B.2.2 Bounding Occurrence of $\mathsf{E}_l(\mathcal{Q}_q)$, $l = 14, 16, 17$

In this section, we prove $\mathbf{Pr}\left(\mathsf{E}_l(\mathcal{Q}_q)\right) = \Theta(q^2/2^n)$ $(l = 14, 16, 17)$ by providing a lower bound in Lemma B.2.3, and an upper bound in Lemma B.2.4.

Lemma B.2.3. $\mathbf{Pr}\left(\mathsf{E}_l(\mathcal{Q}_q)\right) = \Omega(q^2/2^n)$ *for* $l = 14, 16, 17$.

Proof. We consider $\mathsf{E}_{14}(\mathcal{Q}_q)$, the analysis for the other compression functions is analogous.[2]

We construct an adversary $\mathcal{A}$ whose goal is to find tuples $(x_1, y_1) \in \pi_1, (x_2, y_2) \in \pi_2$, and $(x_3, y_3) \in \pi_3$ such that $\mathsf{E}_{14}(\mathcal{Q}_q)$ is satisfied. The adversary proceeds as follows. It sets up two lists of q random elements $X_1 := \{x_1^{(1)}, \ldots, x_1^{(q)}\}$ and $X_2 := \{x_2^{(1)}, \ldots, x_2^{(q)}\}$, and computes the corresponding values $y_1^{(k)} = \pi_1(x_1^{(k)})$ and $y_2^{(k)} = \pi_2(x_2^{(k)})$ (for $k = 1, \ldots, q$). Additionally, for each $k = 1, \ldots, q$, the adversary sets $y_3^{(k)} = x_2^{(k)} \oplus y_2^{(k)} \oplus z$ and computes the corresponding value $x_3^{(k)} = \pi_3^{-1}(y_3^{(k)})$. Fix any $k \in \{1, \ldots, q\}$, then $x_2^{(k)} \oplus y_2^{(k)} \oplus y_3^{(k)} = z$ by construction. The adversary obtains a solution for $\mathsf{E}_{14}(\mathcal{Q}_q)$ if X_1 contains an element x_1 such that $x_1 \oplus y_1 = x_2 \oplus x_3$. By basic probability theory, such x_1 exists with probability $\Omega(q/2^n)$. As the adversary needs to succeed for only one of the q choices of k, it finds a solution for $\mathsf{E}_{14}(\mathcal{Q}_q)$ with probability $\Omega(q^2/2^n)$. □

Lemma B.2.4. $\mathbf{Pr}\left(\mathsf{E}_l(\mathcal{Q}_q)\right) = O(q^2/2^n)$ *for* $l = 14, 16, 17$.

Proof. We consider $\mathsf{E}_{14}(\mathcal{Q}_q)$, the analysis for the other compression functions is analogous. We write $\mathsf{E}_{14}\mathsf{a}, \mathsf{E}_{14}\mathsf{b}$ for the two equations of $\mathsf{E}_{14}(\mathcal{Q}_q)$. The approach is similar to before. By basic probability theory,

$$\mathbf{Pr}\left(\mathsf{E}_{14}(\mathcal{Q}_q)\right) \le \sum_{i=1}^{q} \mathbf{Pr}\left(\mathsf{E}_{14}(\mathcal{Q}_i) \mid \neg\mathsf{E}_{14}(\mathcal{Q}_{i-1}) \wedge \neg\mathsf{C}_1(\mathcal{Q}_i)\right) + \mathbf{Pr}\left(\mathsf{C}_1(\mathcal{Q}_q)\right) . \quad \text{(B.7)}$$

We start with the first probability and consider $\tau_1 = 1$.

Assume first the adversary makes a query $x_3^{(i)} \to y_3^{(i)} = \pi_3(x_3^{(i)})$ for $i = 1, \ldots, q$ (the same treatment holds for queries $x_2 \to y_2$). For each of the $\le q$ tuples (x_1, y_1), by $\neg\mathsf{C}_1(\mathcal{Q}_i)$ the values $x_1 \oplus y_1$ are distinct, and there exists at most one (x_2, y_2) such that equation $\mathsf{E}_{14}\mathsf{a}$ is satisfied. For any of the combinations, $\mathsf{E}_{14}\mathsf{b}$ is satisfied with probability at most $\frac{1}{2^n - q}$. Considering all queries $x_3 \to y_3$, the adversary succeeds in breaking $\mathsf{E}_{14}(\mathcal{Q}_q)$ with probability at most $\frac{q^2}{2^n - q}$.

Assume the adversary makes a query $y_3^{(i)} \to x_3^{(i)} = \pi_3^{-1}(x_3^{(i)})$ for $i = 1, \ldots, q$. By $\neg\mathsf{C}_1(\mathcal{Q}_i)$, there exists at most one tuple (x_2, y_2) such that equation $\mathsf{E}_{14}\mathsf{b}$ is satisfied. For this tuple, $\mathsf{E}_{14}\mathsf{a}$ is satisfied with probability at most $\frac{q}{2^n - q}$. Considering all queries $x_3 \to y_3$, the adversary succeeds in breaking $\mathsf{E}_{14}(\mathcal{Q}_q)$ with probability at most $\frac{q^2}{2^n - q}$.

Assume the adversary makes a query $y_2^{(i)} \to x_2^{(i)} = \pi_2^{-1}(x_2^{(i)})$ for $i = 1, \ldots, q$. For each of the $\le q$ tuples (x_1, y_1), by $\neg\mathsf{C}_1(\mathcal{Q}_i)$ the values $x_1 \oplus y_1$ are distinct, and there exists at most one (x_3, y_3) such that $x_1 \oplus y_1 \oplus y_2 \oplus x_3 \oplus y_3 = z$. For any of

[2]The attack defining the lower bound for $\mathsf{E}_{16}(\mathcal{Q}_q)$ corresponds to an attack by Joux described in [206].

the combinations, $\mathsf{E}_{14}\mathsf{a}$ is satisfied with probability at most $\frac{1}{2^n-q}$. Considering all queries $x_2 \to y_2$, the adversary succeeds in breaking $\mathsf{E}_{14}(\mathcal{Q}_q)$ with probability at most $\frac{q^2}{2^n-q}$.

Assume the adversary makes a query $x_1^{(i)} \to y_1^{(i)} = \pi_1(x_1^{(i)})$ for $i = 1, \dots, q$ (the same treatment holds for queries $y_1 \to x_1$). For each of the $\leq q$ tuples (x_3, y_3), by $\neg\mathsf{C}_1(\mathcal{Q}_i)$ there exists at most one (x_2, y_2) such that equation $\mathsf{E}_{14}\mathsf{b}$ is satisfied. For any of the combinations, $\mathsf{E}_{14}\mathsf{a}$ is satisfied with probability at most $\frac{1}{2^n-q}$. Considering all queries $x_1 \to y_1$, the adversary succeeds in breaking $\mathsf{E}_{14}(\mathcal{Q}_q)$ with probability at most $\frac{q^2}{2^n-q}$.

In any case, we obtain $\sum_{i=1}^{q} \mathbf{Pr}\left(\mathsf{E}_{14}(\mathcal{Q}_i) \mid \neg\mathsf{E}_{14}(\mathcal{Q}_{i-1}) \wedge \neg\mathsf{C}_1(\mathcal{Q}_i)\right) \leq \frac{q^2}{2^n-q}$. The claim now immediately follows from (B.7), Lemma B.1.7, and the fact that $q \leq 2^{n-1}$. □

B.3 Generalization of Theorem 5.4.1

We generalize our findings on the single-permutation setting to cover *any* function, where affine transformations on the inputs to the permutations are taken into account. This generalization is straightforward, but technical and more elaborate. For a matrix $\mathsf{B} = (b_1, b_2, b_3, b_4)^\top$ with elements in $\mathbb{Z}_2^n$, we define the compression function F_{AB} as follows.

$$\begin{aligned}
&\mathsf{F}_{\mathsf{AB}}(x_1, x_2) \\
&\quad y_1 \leftarrow \pi_1(a_{11}x_1 \oplus a_{12}x_2 \oplus b_1)\,, \\
&\quad y_2 \leftarrow \pi_2(a_{21}x_1 \oplus a_{22}x_2 \oplus a_{23}y_1 \oplus b_2)\,, \\
&\quad y_3 \leftarrow \pi_3(a_{31}x_1 \oplus a_{32}x_2 \oplus a_{33}y_1 \oplus a_{34}y_2 \oplus b_3)\,, \\
&\quad z \leftarrow a_{41}x_1 \oplus a_{42}x_2 \oplus a_{43}y_1 \oplus a_{44}y_2 \oplus a_{45}y_3 \oplus b_4\,, \\
&\quad \textbf{return } z\,.
\end{aligned} \tag{B.8}$$

where A is as in Section 5.1.1. We note that for the multi-permutation setting, this generalization is of no added value, as the permutations are independently distributed anyway. Adding constants is, however, a customary approach to obtain *different* permutations from a single one (e.g., $\pi_i(x) = \pi(b_i \oplus x)$ for $i = 1, 2, 3$), but as we will show, the findings of Theorem 5.4.1 also apply to this extended setting.

We reformulate Propositions 5.2.2-5.2.4 to the case of F_{AB} (recall that Proposition 5.2.5 did not apply to the single-permutation setting in the first place). Propositions 5.2.2 and 5.2.3 apply to any F_{AB} and $\mathsf{F}_{\mathsf{A'B'}}$ with $\mathsf{B} = \mathsf{B}'$ and Proposition 5.2.4 holds for any B and B' with $(b'_i, b'_{i+1}) = (b_{i+1}, b_i)$. Given this, the proof of Theorem 5.4.1 almost carries over. Lemmas 5.3.3 and 5.3.4 apply with straightforward generalization. It remains to evaluate the matrices A of the form (5.6) *for any* $\mathsf{B} \in (\mathbb{Z}_2^n)^{4\times 1}$. The case $\|\mathsf{a}_{3,*}\| \leq 2$ is the same as in the proof of Theorem 5.4.1.

$\|\mathsf{a}_{3,*}\| = \mathbf{3}$. By M-reductions, it suffices to consider $a_{31}a_{32}a_{33}a_{34} \in \{1110, 0111\}$.

- $a_{31}a_{32}a_{33}a_{34} = 1110$. The same analysis as in Section 5.3.1 applies, leaving the matrices A_1 and A_2 of (5.5). In the extended single-permutation setting, the two corresponding compression functions satisfy $\mathsf{F}_{\mathsf{A}_1\mathsf{B}}(x_1 \oplus b_1, \pi(x_1) \oplus b_1 \oplus b_3) = \pi(\pi(x_1) \oplus b_1 \oplus b_2 \oplus b_3) \oplus b_1 \oplus b_3 \oplus b_4$ and $\mathsf{F}_{\mathsf{A}_2\mathsf{B}}(x_1, x_2) = \mathsf{F}_{\mathsf{A}_2\mathsf{B}}(x_1, x_1 \oplus x_2 \oplus \pi(x_1 \oplus b_1) \oplus b_2 \oplus b_3)$ for any x_1, x_2. Collisions can thus be trivially found;
- $a_{31}a_{32}a_{33}a_{34} = 0111$. By Lemma 5.3.4(ii), we have $a_{41} = 1$. Lemma 5.3.4(iii) now states that we require $a_{42} = a_{44}$, which gives the following four options for $a_{42}a_{43}a_{44}$: 000, 010, 101, and 111. The first one is vulnerable to the attack of Lemma 5.3.4(iv), the second, third, and fourth matrix satisfy $\mathsf{F}_{\mathsf{AB}}(x_1, \pi^{-1}(\pi(x_1 \oplus b_1) \oplus b_2 \oplus b_3) \oplus b_2) = x_1 \oplus b_2 \oplus b_3 \oplus b_4$, $\mathsf{F}_{\mathsf{AB}}(x_1 \oplus b_1, x_1 \oplus b_2) = \mathsf{F}_{\mathsf{AB}}(x_1 \oplus b_1 \oplus b_2 \oplus b_3, x_1 \oplus b_3)$, and $\mathsf{F}_{\mathsf{AB}}(x_1 \oplus b_1, x_1 \oplus b_2) = \pi(x_1 \oplus b_2 \oplus b_3) \oplus b_1 \oplus b_2 \oplus b_4$, respectively, for any x_1. Collisions can thus be trivially found;

$\|\mathsf{a}_{3,*}\| = \mathbf{4}$. Except for $a_{41}a_{42}a_{43}a_{44} \in \{1010, 1001, 0110, 0101\}$, all induced compression functions satisfy $\mathsf{F}_{\mathsf{AB}}(x_1 \oplus b_1, x_1 \oplus b_2) \oplus \pi(b_1 \oplus b_2 \oplus b_3) \oplus a_{41}b_1 \oplus a_{42}b_2 \oplus b_4 \in \{0, x_1, \pi(x_1)\}$ for any x_1, for which collisions can be trivially found. The cases $1001, 0110$ are vulnerable to Lemma 5.3.4(iii). The remaining two cases, which are equivalent by M-reduction, allow for trivial collisions as well: the compression function induced by $(a_{41}a_{42}a_{43}a_{44}) = (1010)$ satisfies $\mathsf{F}_{\mathsf{AB}}(x_1, \pi^{-1}(x_1 \oplus \pi(x_1 \oplus b_1) \oplus b_2 \oplus b_3) \oplus b_2) = b_2 \oplus b_3 \oplus b_4$ for any x_1.

Hence, any of the analyzed compression functions either allows for trivial collision or is vulnerable to Lemma 5.3.4, therewith allowing for collisions in at most $2^{2n/5}$ queries.

Concluding, for any compression function F_{AB} of the generalized form (B.8), collisions can be found in at most $2^{2n/5}$ queries, hence considerably faster than in $2^{n/2}$ queries.

B.4 Heuristic Argument for Conjecture 5.3.2

In this section, we provide a heuristic argument for the first part of Conjecture 5.3.2, a similar argument applies to part two. Throughout the argument it becomes clear why the conjecture is similar to but more complex than Zarankiewicz problem [57, Ch. 6.2], as claimed in Section 5.3.

In more detail, we will show that the conjecture should hold for $d_1 = 10$ for large enough n. Let Z be a given set of $q < 2^{n/2}$ random elements. Denote by $\mathbf{Pr}(\mathsf{succ}(Z))$ the probability that there exist two sets X_1, X_2 such that the number of solutions $(x_1, x_2, z) \in X_1 \times X_2 \times Z$ with $x_1 \oplus x_2 = z$ is larger than $10q \log q$. In this heuristic argument we will provide a bound on $\mathbf{Pr}(\mathsf{succ}(Z))$. In fact, we will first consider $q = 2^\alpha$ and show that the number of solutions is with high probability upper bounded by $2q\log q + q$. If q is no power of two, the number of solutions is then clearly upper bounded by the amount of solutions for $q' = 2^{\lceil \log q \rceil}$, hence upper bounded by $8q\log q + 2q \leq 10q\log q$ (provided $q \geq 2$).

Before proceeding, we pose the following claim on Z (recall that Z is a multiset):

$$\mathsf{D}_1(Z) : \textbf{there exist } z_1, z_2 \in Z \textbf{ such that } z_1 = z_2.$$

Clearly, $\mathbf{Pr}(\mathsf{D}_1(Z)) \leq q^2/2^n$, and by probability theory we have

$$\begin{aligned} \mathbf{Pr}(\mathsf{succ}(Z)) &\leq \mathbf{Pr}(\mathsf{succ}(Z) \mid \neg\mathsf{D}_1(Z)) + \mathbf{Pr}(\mathsf{D}_1(Z)) \\ &\leq \mathbf{Pr}(\mathsf{succ}(Z) \mid \neg\mathsf{D}_1(Z)) + \frac{q^2}{2^n}\,. \end{aligned} \tag{B.9}$$

Therefore, it suffices to analyze $\mathsf{succ}(Z)$ given that Z contains no collisions. The goal for an adversary now is to come up with two sets $X_1 = \{x_1^{(1)}, \ldots, x_1^{(q)}\}$ and $X_2 = \{x_2^{(1)}, \ldots, x_2^{(q)}\}$ such that $\left|\{(x_1, x_2, z) \in X_1 \times X_2 \times Z \mid x_1 \oplus x_2 = z\}\right|$ is maximized.

Consider a $q \times q$ matrix X with rows corresponding to $x_1^{(i)}$ and columns to $x_2^{(j)}$, and the coefficient x_{ij} of X equals $z \in Z$ if $x_1^{(i)} \oplus x_2^{(j)} = z$, and is empty if $x_1^{(i)} \oplus x_2^{(j)} \notin Z$. Now, the goal of the adversary is to maximize the number of filled coefficients of X, by smartly choosing $x_1^{(i)}, x_2^{(j)}$. Denote by $s(\mathsf{X})$ the maximum number of elements in the matrix X. Some restrictions apply to the choices for $x_1^{(i)}, x_2^{(j)}$.

(i) An element $z \in Z$ does not occur twice in one row or column (it would imply a collision in X_1 or X_2);

(ii) Let $i, i', j, j' \in \{1, \ldots, q\}$ and $z_1, z_2 \in Z$. If $x_{ij} = x_{i'j'} = z_1$ and $x_{ij'} = z_2$, then $x_{i'j} = z_2$ (obtained by XORing the first three equations);

(iii) Let $i, i', j, j' \in \{1, \ldots, q\}$ and $z_1, \ldots, z_4 \in Z$ satisfying $z_1 \oplus z_2 \oplus z_3 \oplus z_4 = 0$. If $x_{ij} = z_1$, $x_{ij'} = z_2$ and $x_{i'j} = z_3$, then $x_{i'j'} = z_4$ (obtained by XORing the first three equations).

The problem now reduces to smartly positioning in X as many elements from Z as possible.

Let K be maximal such that there exists a set of values $z_1, \ldots, z_{2K} \in Z$ satisfying $z_1 \oplus z_2 = \ldots = z_{2K-1} \oplus z_{2K}$. We call this set of values a K-way collision, and two consecutive elements z_{2i-1}, z_{2i} are called a twin. Using properties (ii,iii), the adversary can obtain a 4×4 submatrix of X filled with four values z_1, z_2, z_3, z_4 each occurring four times,[3] but as we will argue this is essentially the best the adversary can get. Notice that this also demonstrates that the best approach followed by the adversary is to exploit the K-way collision.

In Figure B.4, we introduce four claims $\mathsf{D}_2(Z), \ldots, \mathsf{D}_5(Z)$ to further analyze event $\mathsf{succ}(Z)$, and we bound the occurrence of these events provided $\neg\mathsf{D}_1(Z)$.

[3]The adversary chooses $x_1^{(1)}$, sets $x_2^{(j)} = z_j \oplus x_1^{(1)}$ for $j = 1, 2, 3, 4$, and $x_1^{(i)} = z_i \oplus x_2^{(1)}$ for $i = 2, 3, 4$. By properties (ii,iii), the remaining coefficients are filled.

$\mathsf{D}_2(Z)$: for some c there exist more than τ_2 solutions in Z to:

$z_1 \neq z_2$, **and**

$z_1 \oplus z_2 = c$.

$\mathsf{D}_3(Z)$: for some c there exist more than one solution in Z to (with different z_7 for each solution):

$z_1, \ldots, z_7$ **distinct, and**

$z_{2i-1} \oplus z_{2i} = c$ **(for** $i = 1, 2, 3$**), and**

$z_1 \oplus z_3 \oplus z_5 \oplus z_7 = 0$.

$\mathsf{D}_4(Z)$: for some c there exist a solution in Z to:

$z_1, \ldots, z_{12}$ **distinct, and**

$z_{2i-1} \oplus z_{2i} = c$ **(for** $i = 1, \ldots, 5$**), and**

$z_1 \oplus z_3 \oplus z_5 \oplus z_{11} = 0$, **and**

$z_7 \oplus z_9 \oplus z_{11} \oplus z_{12} = 0$.

$\mathsf{D}_5(Z)$: let $d \geq 4$, for some c there exists a solution in $\mathcal{Q}$ to:

$z_1, \ldots, z_{2d}$ **distinct, and**

$z_{2i-1} \oplus z_{2i} = c$ **(for** $i = 1, \ldots, d$**), and**

$z_1 \oplus z_3 \oplus \cdots \oplus z_{2d-1} = 0$.

Figure B.4. Claims $\mathsf{D}_2(Z), \ldots, \mathsf{D}_5(Z)$. The parameter $\tau_2 > 1$ is any fixed constant.

$\mathbf{Pr}\left(\mathsf{D}_2(\mathbf{Z}) \mid \neg\mathsf{D}_1(\mathbf{Z})\right) \leq \mathbf{2^n} \left(\frac{\mathbf{2q^2e}}{(\tau_2+\mathbf{1})\mathbf{2^n}}\right)^{(\tau_2+\mathbf{1})}$**.** The proof is similar to the proof of Lemma B.1.8, with the difference that now q random values are generated (being Z) that piece for piece may result in a solution;

$\mathbf{Pr}\left(\mathsf{D}_3(\mathbf{Z}) \mid \neg\mathsf{D}_{1\vee 2}(\mathbf{Z})\right) \leq \mathbf{2^n} \left(\frac{\tau_2^3\mathbf{q}}{\mathbf{2^n}}\right)^{\mathbf{2}}$**.** Fix any c. By $\neg\mathsf{D}_2(Z)$, there are at most $2^3\binom{\tau_2}{3}$ possible choices for z_1, z_3, z_5 (the choices for z_2, z_4, z_6 follow directly), any of which satisfies the second equation with probability at most $q/2^n$. As we require more than one *different* value z_7 to be hit, and Z contains q *different* possibilities for $z_3^{(7)}$ by $\neg\mathsf{D}_1(Z)$, Z contains more than one solution to $\mathsf{D}_3(Z)$ with probability at most

$$\binom{2^3\binom{\tau_2}{3}}{2}\left(\frac{q}{2^n}\right)^2 \leq \left(\frac{\tau_2^3 q}{2^n}\right)^2.$$

Now, the result follows by quantifying over the number of choices for c;

$\mathbf{Pr}\left(\mathsf{D}_4(\mathbf{Z}) \mid \neg\mathsf{D}_{1\vee 2}(\mathbf{Z})\right) \leq \mathbf{2^n} \left(\frac{\tau_2^3\mathbf{q}}{\mathbf{2^n}}\right)^{\mathbf{2}}$**.** Fix any c. By $\neg\mathsf{D}_2(Z)$, there are at most $2^5\binom{\tau_2}{5}$ possible choices for $z_1, z_3, \ldots, z_9$ (the choices for $z_2, z_4, \ldots, z_{10}$ follow directly). For any choice, the second and third equation are both satisfied with probability at most $q/2^n$. Therefore, Z contains a solution to $\mathsf{D}_4(Z)$ with probability at most

$$2^5\binom{\tau_2}{5}\left(\frac{q}{2^n}\right)^2 \leq \left(\frac{\tau_2^3 q}{2^n}\right)^2.$$

Now, the result follows by quantifying over the number of choices for c;

$\mathbf{Pr}\left(\mathsf{D}_5(\mathbf{Z}) \mid \neg\mathsf{D}_1(\mathbf{Z})\right) \leq \mathbf{2^n} \sum_{\mathbf{d=4}}^{\infty} \frac{\mathbf{q^{2d}}}{(\mathbf{2^n})^{\mathbf{d+1}}}$**.** Fix any c, d. Without loss of generality we can consider the values of Z to be generated piece for piece. Consider the generation of z_i for $i = 1, \ldots, q$. For any other value z, the first equation is satisfied with probability at most $1/2^n$. Thus, the query results in a solution with probability at most $q/2^n$. In total q values are generated, and as $\neg\mathsf{D}_1(Z)$, each "hit" adds at most one solution. Therefore, we need d out of at most q hits. Given that d pairs are found, the second equation of $\mathsf{D}_5(Z)$ is then satisfied with probability at most $1/2^n$, as the values $z_3^{(i)}$ are different by $\neg\mathsf{D}_1(Z)$. Consequently, Z contains a solution to $\mathsf{D}_5(Z)$ with probability at most

$$\binom{q}{d}\left(\frac{q}{2^n}\right)^d \cdot \frac{1}{2^n} \leq \frac{q^{2d}}{(2^n)^{d+1}}\,.$$

Now, the result follows by quantifying over the number of choices for c and d (in fact, we are using $\mathsf{D}_5(Z)$ for $d = 4, 6$ only but we generalize its usage for simplicity).

We obtain for (B.9):

$$\begin{aligned}\mathbf{Pr}\left(\mathsf{succ}(Z) \mid \neg\mathsf{D}_1\right) \leq \mathbf{Pr}\left(\mathsf{succ}(Z) \mid \neg\mathsf{D}_{1\vee\ldots\vee 5}\right) + \mathbf{Pr}\left(\mathsf{D}_2 \mid \neg\mathsf{D}_1\right) +& \\ \mathbf{Pr}\left(\mathsf{D}_3 \mid \neg\mathsf{D}_{1\vee 2}\right) + \mathbf{Pr}\left(\mathsf{D}_4 \mid \neg\mathsf{D}_{1\vee 2}\right) + \mathbf{Pr}\left(\mathsf{D}_5 \mid \neg\mathsf{D}_1\right)\,.&\end{aligned} \tag{B.10}$$

Similar to before (e.g., Appendix B.1.3), one can show that for $\tau_2 = n - 1$ the second part of (B.10) approaches 0 for $q = 2^{n/2(1-\varepsilon)}$ and $n \to \infty$. In what remains, we heuristically argue that $\mathbf{Pr}\left(\mathsf{succ}(Z) \mid \neg\mathsf{D}_{1\vee\ldots\vee 5}(Z)\right)$ is expected to equal 0 for $d_1 = 10$.

Naturally, the maximal number of solutions is achieved when the adversary includes in the matrix X as many boxes $\left(\begin{smallmatrix} z_a & z_b \\ z_b & z_a \end{smallmatrix}\right)$ as possible, where z_a, z_b denotes any twin of the maximal K-way collision. However, it may be the case that three values z_1, z_2, z_3 coming from *different* twins form a collision with a value $z \in Z$ no member of the K-way collision, therewith resulting in more solutions due to property (iii) described above. However, by $\neg\mathsf{D}_3(Z)$, there is only one such possible value z, and XORed with any other values from different twins it does not collide with another element from Z (by $\neg\mathsf{D}_4(Z)$). Thus, the value z appears in X at most q times, and no other value can occur. We just scrap this value out of the matrix, continue with the matrix built of the K-way collision only, and add q to the finally obtained number of elements in the matrix. Concluding, the best approach is to consider the matrix X consisting of 2×2 submatrices that satisfy the following property: each block is either entirely filled by a twin of the K-way collision, or empty. Additionally, any two rows of 2×2 submatrices share at most two "positions" (by $\neg\mathsf{D}_3(Z)$). Now, we can constrict rows $2i - 1$ and $2i$ (for $i = 1, \ldots, q/2$) and columns $2j - 1$ and $2j$ (for $j = 1, \ldots, q/2$) to obtain a matrix $\mathsf{X}^{[q/2]}$. Here we replace every 2×2 submatrix by the first element of the twin. Matrix $\mathsf{X}^{[q/2]}$ still satisfies properties (i,ii), and moreover it satisfies the next properties:

(iv) The matrix does not contain a full 2×3 (or 3×2) submatrix (by $\neg\mathsf{D}_3(Z)$);

(v) For any $d \geq 4$, the matrix does not contain d different values that XOR up to 0 (by $\neg\mathsf{D}_5(Z)$).

Now, the number of coefficients filled in X satisfies $s(\mathsf{X}) \leq 4s(\mathsf{X}^{[q/2]}) + q$, and hence we bound the maximum number of elements in any matrix $\mathsf{X}^{[q/2]}$ satisfying (i,ii,iv,v).

Before proceeding, we note the relation of our conjecture with Zarankiewicz conjecture.[4] Let $G_n = (V_1 \cup V_2, E)$ be a bipartite graph on color classes V_1, V_2 of equal size n. Zarankiewicz problem regards the case of determining the maximal graph G_n that does not contain any complete bipartite subgraph on $3+3$ vertices. With respect to our problem, we can consider $\mathsf{X}^{[q/2]}$ to represent an incidence matrix of a bipartite graph, where the rows correspond to one side of the bipartition, and the columns to the other side. An element $x_{ij} \in \mathsf{X}^{[q/2]}$ is non-empty if and only if $x_1^{(i)} \oplus x_2^{(j)} = z \in Z$, and the corresponding edge is labeled by z. By virtue of (i,ii,iv,v) several restrictions apply to this graph: for instance, it does not contain a complete bipartite subgraph on $2+3$ vertices (property (iv)), and it does not contain a complete bipartite subgraph on $2+2$ vertices where the four edges have different labels (property (v)). Various other restrictions to this matrix can be extracted from the randomness of Z, but for our heuristic argument the restrictions put forward suffice.

We proceed with the heuristic argument. Consider a matrix $\mathsf{X}^{[q/2]}$ that achieves the maximum number of solutions. $\mathsf{X}^{[q/2]}$ is likely to have two elements $z_1, z_2 \in Z$ occurring $q/2$ times, which can be seen as follows. Consider a first element z_1. Suppose z_1 occurs $q/2 - 1$ times, w.l.o.g. (by graph isomorphism) at the first $q/2 - 1$ diagonal elements of $\mathsf{X}^{[q/2]}$. This means that $x_{q/2,q/2} \neq z_1$. Without loss of generality, $\left|\{i \mid x_{i,q/2} \in Z\}\right| \leq \left|\{j \mid x_{q/2,j} \in Z\}\right|$. We construct the matrix $\overline{\mathsf{X}}^{[q/2]}$, with the first $q/2 - 1$ columns identical to the ones of $\mathsf{X}^{[q/2]}$, but with $\overline{x}_{q/2,q/2} = z_1$. By property (ii) and the fact that all diagonal elements of $\overline{\mathsf{X}}^{[q/2]}$ equal z_1, we obtain $\overline{x}_{i,q/2} = \overline{x}_{q/2,i} = x_{q/2,i}$ for $i = 1, \ldots, q/2 - 1$. It is easy to check that $\overline{\mathsf{X}}^{[q/2]}$ satisfies (i,ii,iv,v) if $\mathsf{X}^{[q/2]}$ does. In particular, it does not violate property (i) (similar argument applies to other properties): it would violate (i) if $\overline{x}_{i,q/2} = \overline{x}_{ij} = x_{ij}$ for some $i, j \in \{1, \ldots, q/2 - 1\}$, which for the original matrix $\mathsf{X}^{[q/2]}$ would imply that $x_{q/2,i} = x_{ij}$, and thus (by property (ii) for $\mathsf{X}^{[q/2]}$) $x_{q/2,j} = x_{ii} = z_1$, impossible by construction. Thus, we obtained a matrix $\overline{\mathsf{X}}^{[q/2]}$ with z_1 occurring $q/2$ times and with at least as many solutions as $\mathsf{X}^{[q/2]}$. A second element is likely to occur $q/2$ times for similar reasons.

[4]Generalizations of Zarankiewicz problem exist, but this problem is the most well-known among them.

$\mathsf{X}^{[q/2]}$ can thus be considered to be of the following form:

$$\mathsf{X}^{[q/2]} = \left(\begin{array}{c|c|c|c} \begin{matrix} z_1 & z_2 \\ z_2 & z_1 \end{matrix} & \mathsf{X}_{12} & \cdots & \mathsf{X}_{1,\frac{q}{4}} \\ \hline \mathsf{X}_{21} & \begin{matrix} z_1 & z_2 \\ z_2 & z_1 \end{matrix} & \cdots & \mathsf{X}_{2,\frac{q}{4}} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline \mathsf{X}_{\frac{q}{4},1} & \mathsf{X}_{\frac{q}{4},2} & \cdots & \begin{matrix} z_1 & z_2 \\ z_2 & z_1 \end{matrix} \end{array}\right),$$

for some 2×2 submatrices X_{ij}. Now, by property (ii) we have $\mathsf{X}_{ij} = \mathsf{X}_{ji}$ for all i, j. Additionally, as no two rows share three columns (and vice versa), each block is either empty or an (anti-)diagonal matrix where the two (anti-)diagonal elements are equal by property (ii). Consequently, we can constrict rows $2i-1$ and $2i$ (for $i = 1, \ldots, q/4$) and columns $2j-1$ and $2j$ (for $j = 1, \ldots, q/4$) to obtain a matrix $\mathsf{X}^{[q/4]}$. Here each 2×2 block on the diagonal of $\mathsf{X}^{[q/2]}$ is constricted to z_1, and each other block to its only element. Now, the number of coefficients in $\mathsf{X}^{[q']}$ (with $q' = q/2$) satisfies $s(\mathsf{X}^{[q']}) \leq 2 \cdot (q'/2) + 2s(\mathsf{X}^{[q'/2]})$: the second part counts each element in $\mathsf{X}^{[q'/2]}$ as two original elements, and we add two remaining from the original diagonal blocks. The matrix $\mathsf{X}^{[q/4]}$ satisfies (i,ii,iv,v) if $\mathsf{X}^{[q/2]}$ does. For (i,ii,v) this is clear, and we briefly consider property (iv). Notice that a diagonal element of $\mathsf{X}^{[q/4]}$ corresponds to a full 2×2 submatrix of $\mathsf{X}^{[q/2]}$ but a non-empty non-diagonal element corresponds to a diagonal or anti-diagonal 2×2 submatrix with one and the same element. Suppose $\mathsf{X}^{[q/4]}$ contains a full 2×3 submatrix $\mathsf{X}^{[2\times 3]}$. If $\mathsf{X}^{[2\times 3]}$ involves two diagonal elements, it implies $z_3 \oplus z_4 = 0$ for some $z_3, z_4 \in Z$ (impossible by $\neg\mathsf{D}_1(Z)$). If $\mathsf{X}^{[2\times 3]}$ involves one diagonal element, it implies $z_1 \oplus z_2 \oplus z_3 \oplus z_4 = 0$ for some $z_3, z_4 \in Z$ (impossible by (v)). If $\mathsf{X}^{[2\times 3]}$ involves zero diagonal elements, it implies $z_1 \oplus z_2 \oplus z_3 \oplus z_4 \oplus z_5 \oplus z_6 = 0$ for some $z_3, z_4, z_5, z_6 \in Z$ (impossible by (v)). The last property comes from selecting 6 coefficients of $\mathsf{X}^{[q/2]}$ such that in any row/column either two or zero coefficients are selected. By the form of $\mathsf{X}^{[2\times 3]}$ and the properties (i,ii), this turns out to be possible for the matrix $\mathsf{X}^{[q/2]}$.

We reduced the problem to a smaller dimension $q/4$, but with the same properties. This analysis can be applied recursively, until we are left with a 2×2 matrix $\mathsf{X}^{[2]}$. By induction to the size of q, we can now show that for matrices satisfying properties (i,ii,iv,v) the number of elements is upper bounded by $q \log(2q)$. For $q = 2$, we have $s(\mathsf{X}^{[2]}) = 4 = q\log(2q)$, so the claim holds. Suppose $s(\mathsf{X}^{[k/2]}) \leq k/2 \log k$. Then,

$$s(\mathsf{X}^{[k]}) \leq 2 \cdot (k/2) + 2s(\mathsf{X}^{[k/2]}) \leq k(1 + \log k) = k \log(2k)\,.$$

Thus, $s(\mathsf{X}^{[q/2]}) \leq q/2 \log q$. For the original matrix X, we now obtain $s(\mathsf{X}) \leq 4s(X^{[q/2]}) + q \leq 2q \log q + q$, which completes the argument for q a power of two. As explained in the beginning of this appendix, this result implies $s(\mathsf{X}) \leq 10q \log q$ for any q.

C Appendix to Chapter 7

C.1 Indifferentiability of Grøstl

In this section, we prove the indifferentiability of Grøstl, Theorem 7.2.3.

C.1.1 Defining the Simulator

The simulator maintains two, initially empty, lists $\mathcal{L}_\pi, \mathcal{L}_\rho$ that represent the simulated permutation. They consist of tuples $(x, y) \in \mathbb{Z}_2^l \times \mathbb{Z}_2^l$, where y denotes the sampled image of x under π or ρ. The simulator has four interfaces, denoted by $\mathcal{S}_\pi, \mathcal{S}_\pi^{-1}, \mathcal{S}_\rho, \mathcal{S}_\rho^{-1}$, and access to $\mathcal{R}$. It maintains a graph (V, A), which initially consists of the node iv and includes no arcs. The arcs in A are labeled by messages $M \in \mathbb{Z}_2^l$ and define input-output pairs of the compression function f: any $(x_1, y_1) \in \mathcal{L}_\pi$ and $(x_2, y_2) \in \mathcal{L}_\rho$ define an arc $x_1 \oplus x_2 \xrightarrow{x_2} x_1 \oplus x_2 \oplus y_1 \oplus y_2$ in (V, A). Intuitively, if there is a path $\mathsf{iv} \xrightarrow{M_1} h_1 \xrightarrow{M_2} \cdots \xrightarrow{M_k} h_k$ in (V, A), then $f(\dots f(f(\mathsf{iv}, M_1), M_2) \dots, M_k) = h_k$. Abusing notation, we denote such path by $\mathsf{iv} \xrightarrow{\overline{M}} h_k$ for $\overline{M} = (M_1, \dots, M_k)$. We say that (V, A) contains "colliding paths" if there exists an $h \in V$ such that $\mathsf{iv} \xrightarrow{\overline{M}} h$ and $\mathsf{iv} \xrightarrow{\overline{M}'} h$ are two paths in (V, A), for different $\overline{M}, \overline{M}' \in \left(\mathbb{Z}_2^l\right)^\infty$. By $V_{\text{out}}, V_{\text{in}}$ we denote the set of vertices in V with an outgoing or ingoing arc, respectively. Observe that if $\mathcal{L}_\pi, \mathcal{L}_\rho$ are of size q_π, q_ρ, respectively, the sets $V_{\text{out}}, V_{\text{in}}$ are of size at most $q_\pi q_\rho$. By $t(V)$ we denote the tree in (V, A) rooted in iv. Additionally, by $\bar{t}(V)$ we denote the subset of nodes of $t(V)$ that are labeled by a correctly padded message.

C.1.2 Intuition

In this section we take a closer look at the simulator of Figure C.1 by starting with an example. Consider the case that a node x is a member of both $\bar{t}(V)$ and $\mathsf{dom}(\mathcal{L}_\pi)$. This means that (1) there exists an M such that $\mathsf{iv} \xrightarrow{M} x$ and $\mathsf{depad}(M) \neq \perp$, and (2) there exists a $y \in \mathsf{rng}(\mathcal{L}_\pi)$, such that $y = \mathcal{S}_\pi(x)$. In the real world (where the left oracle is the Grøstl hash function), these values satisfy $\mathcal{H}(\mathsf{depad}(M)) = \mathsf{left}_n(x \oplus y)$ by construction. If the simulator does not answer its

Forward Query $\mathcal{S}_\pi(x_1)$

```
000 if $x_1 \in \mathsf{dom}(\mathcal{L}_\pi)$ :
001     return $y_1 = \mathcal{L}_\pi(x_1)$
002 if $x_1 \in \bar{t}(V)$ for iv $\xrightarrow{M} x_1$ :
003     $h \xleftarrow{\$} \mathcal{R}(\mathsf{depad}(M))$
004     $w \xleftarrow{\$} \mathbb{Z}_2^{l-n}$
005     $y_1 \leftarrow x_1 \oplus (h \| w)$
006     if $y_1 \in \mathsf{rng}(\mathcal{L}_\pi)$ :
007         GOTO 004
008 else : $y_1 \xleftarrow{\$} \mathbb{Z}_2^l \backslash \mathsf{rng}(\mathcal{L}_\pi)$
009 end if
010 $V_{\text{new}} \leftarrow \{x_1 \oplus x_2', x_1 \oplus x_2' \oplus y_1 \oplus y_2' \mid (x_2', y_2') \in \mathcal{L}_\rho\}$ multiset
011 forall $(x_2, y_2) \in \mathcal{L}_\rho$ with iv $\xrightarrow{M} x_1 \oplus x_2$ :
012     $h_{\text{nxt}} \leftarrow x_1 \oplus x_2 \oplus y_1 \oplus y_2$
013     if $h_{\text{nxt}} \in V \cup (V_{\text{new}} \backslash \{h_{\text{nxt}}\})$ or
014         $\big(M \| x_2 \in \mathsf{rng}(\mathsf{pad})$ and $h_{\text{nxt}} \in \mathsf{dom}(\mathcal{L}_\pi) \cup \{x_1\}\big)$ :
015         GOTO 002
016 end forall
017 return $\mathcal{L}_\pi(x_1) \leftarrow y_1$
```

Forward Query $\mathcal{S}_\rho(x_2)$

```
100 if $x_2 \in \mathsf{dom}(\mathcal{L}_\rho)$ :
101     return $y_2 = \mathcal{L}_\rho(x_2)$
102 $y_2 \xleftarrow{\$} \mathbb{Z}_2^l \backslash \mathsf{rng}(\mathcal{L}_\rho)$
103 $V_{\text{new}} \leftarrow \{x_1' \oplus x_2, x_1' \oplus x_2 \oplus y_1' \oplus y_2 \mid (x_1', y_1') \in \mathcal{L}_\pi\}$ multiset
104 forall $(x_1, y_1) \in \mathcal{L}_\pi$ with iv $\xrightarrow{M} x_1 \oplus x_2$ :
105     $h_{\text{nxt}} \leftarrow x_1 \oplus x_2 \oplus y_1 \oplus y_2$
106     if $h_{\text{nxt}} \in V \cup (V_{\text{new}} \backslash \{h_{\text{nxt}}\})$ or
107         $\big(M \| x_2 \in \mathsf{rng}(\mathsf{pad})$ and $h_{\text{nxt}} \in \mathsf{dom}(\mathcal{L}_\pi)\big)$ :
108         GOTO 102
109 end forall
110 return $\mathcal{L}_\rho(x_2) \leftarrow y_2$
```

Inverse Query $\mathcal{S}_\pi^{-1}(y_1)$

```
200 if $y_1 \in \mathsf{rng}(\mathcal{L}_\pi)$ :
201     return $x_1 = \mathcal{L}_\pi^{-1}(y_1)$
202 $x_1 \xleftarrow{\$} \mathbb{Z}_2^l \backslash \mathsf{dom}(\mathcal{L}_\pi)$
203 if $x_1 \in \bar{t}(V)$ :
204     GOTO 202
205 forall $x_2 \in \mathsf{dom}(\mathcal{L}_\rho)$ : if $x_1 \oplus x_2 \in t(V)$ :
206     GOTO 202
207 return $\mathcal{L}_\pi^{-1}(y_1) \leftarrow x_1$
```

Inverse Query $\mathcal{S}_\rho^{-1}(y_2)$

```
300 if $y_2 \in \mathsf{rng}(\mathcal{L}_\rho)$ :
301     return $x_2 = \mathcal{L}_\rho^{-1}(y_2)$
302 $x_2 \xleftarrow{\$} \mathbb{Z}_2^l \backslash \mathsf{dom}(\mathcal{L}_\rho)$
303 forall $x_1 \in \mathsf{dom}(\mathcal{L}_\pi)$ : if $x_1 \oplus x_2 \in t(V)$ :
304     GOTO 302
305 return $\mathcal{L}_\rho^{-1}(y_2) \leftarrow x_2$
```

Figure C.1. The simulator $\mathcal{S}$ for π and ρ used in the proof of Theorem 7.2.3.

queries wisely, this equality would hold with negligible probability in the simulated world. More generally, the simulator can guarantee that this equation holds only if x is added to $\mathsf{dom}(\mathcal{L}_\pi)$ *after* it was added to $\bar{t}(V)$ (reflected in requirement R3 below). Maintaining consistency, however, becomes harder when $|\bar{t}(V)|$ and $|\mathsf{dom}(\mathcal{L}_\pi)|$ increase. The idea behind the simulator is to answer its queries such that it can control the growth of $t(V)$, and in particular the growth of $\bar{t}(V)$ as a subset of $t(V)$, while still maintaining consistency in its answers. Intuitively, the simulator responds to its queries, such that the following requirements are satisfied:

R1. There are no colliding paths in $(t(V), A)$. Observe that two different paths to the same node may lead to distinguishability for $\mathcal{D}$ as the simulator can be consistent with only *one* of the paths. This requirement is satisfied if $t(V)$

is never increased with a node that has two incoming arcs in the updated[1] graph;

R2. $\mathcal{S}$ increases $t(V)$ only if it is forced to do. In particular, $t(V)$ is never increased with a node that has an outgoing arc in the updated graph. Observe that each path in $(t(V), A)$ leads to a potential node in $\bar{t}(V)$;

R3. $\mathcal{S}$ never increases $\bar{t}(V)$ with a node in the updated $\mathsf{dom}(\mathcal{L}_\pi)$;

R4. $\mathcal{S}$ increases $\mathsf{dom}(\mathcal{L}_\pi)$ with a node in $\bar{t}(V)$ only if it is forced to. Observe that in case of inverse queries to $\mathcal{S}_\pi^{-1}$, the simulator can avoid outputting elements in $\bar{t}(V)$. In forward queries to $\mathcal{S}_\pi$, the simulator may be forced to increase $\bar{t}(V) \cap \mathsf{dom}(\mathcal{L}_\pi)$. In this case, it consults its oracle $\mathcal{R}$ to generate the answer.

The first two conditions are regarding the growth of $t(V)$, and the second two concern the growth of $\bar{t}(V) \cap \mathsf{dom}(\mathcal{L}_\pi)$. We show how these conditions occur in the description of the simulator in Figure C.1. We first consider requirements R1 and R2, then we look at R3 and R4.

Restricting the growth of $t(V)$

Inverse queries. Consider an inverse query y_1 to $\mathcal{S}_\pi^{-1}$. It is easy to see that both R1 and R2 are satisfied if the simulator outputs its answer x_1, such that none of the newly added vertices $\{x_1 \oplus x_2 \mid x_2 \in \mathsf{dom}(\mathcal{L}_\rho)\}$ to V_{out} is already rooted. A similar observation holds for queries to $\mathcal{S}_\rho^{-1}$. These requirements translate to lines 205 and 303 in the description of the simulator in Figure C.1.

Forward queries. In forward queries to $\mathcal{S}_\pi, \mathcal{S}_\rho$, the simulator may be forced to increase $t(V)$. Consider a query x_1 to $\mathcal{S}_\pi$, and consider any $x_2 \in \mathsf{dom}(\mathcal{L}_\rho)$ such that $x_1 \oplus x_2 \in t(V)$. Then, the arc $x_1 \oplus x_2 \xrightarrow{x_2} x_1 \oplus x_2 \oplus y_1 \oplus y_2$ will be added to (V, A) by construction. Denote by V' the multiset of updated nodes after the query. Then, we require that $x_1 \oplus x_2 \oplus y_1 \oplus y_2$ does not occur twice in V'_{in} (in order to establish R1), and moreover that it does not occur in V'_{out} (in order to establish R2). If we define $V_{\text{new}} = \{x_1 \oplus x_2',\ x_1 \oplus x_2' \oplus y_1 \oplus y_2' \mid (x_2', y_2') \in \mathcal{L}_\rho\}$ to be the multiset of newly added nodes to V in the query to $\mathcal{S}_\pi$, both requirements are satisfied if $x_1 \oplus x_2 \oplus y_1 \oplus y_2 \notin V \cup (V_{\text{new}} \backslash \{x_1 \oplus x_2 \oplus y_1 \oplus y_2\})$ holds for all $(x_2, y_2) \in \mathcal{L}_\rho$ such that $x_1 \oplus x_2 \in t(V)$. A similar condition can be derived for queries to $\mathcal{S}_\rho$. These requirements translate to lines 013 and 106 in the description of the simulator in Figure C.1.

[1] This requirement should hold for the "updated" graph, which can be seen as follows. Suppose the distinguisher makes a forward query x_1 to $\mathcal{S}_\pi$ such that $x_1 \oplus x_2, x_1 \oplus x_2' \in t(V)$ for different $x_2, x_2' \in \mathsf{dom}(\mathcal{L}_\rho)$, and both $x_1 \oplus x_2 \oplus y_1 \oplus y_2$ and $x_1 \oplus x_2' \oplus y_1 \oplus y_2'$ are not in V yet. By construction, these nodes have zero incoming arcs in the non-updated (V, A), but it may accidentally be the case that these nodes are equal, in which case they have two incoming arcs in the updated graph.

Restricting the growth of $\bar{t}(V) \cap \mathsf{dom}(\mathcal{L}_\pi)$

Inverse queries. As explained, $\mathcal{S}$ never increases $\bar{t}(V) \subseteq t(V)$ in inverse queries. Hence, requirement R3 is naturally satisfied. Furthermore, R4 is guaranteed if queries to $\mathcal{S}_\pi^{-1}$ are never answered with a node in $\bar{t}(V)$. This requirement translates to line 203 from Figure C.1.

Forward queries. First consider requirement R3. Let the distinguisher make a query to $\mathcal{S}_\pi$ or $\mathcal{S}_\rho$, such that $\bar{t}(V)$ gets increased. By construction and the fact that requirement R2 is satisfied, this means that an arc $x_1 \oplus x_2 \xrightarrow{x_2} x_1 \oplus x_2 \oplus y_1 \oplus y_2$ is added to (V, A), such that $\mathsf{iv} \xrightarrow{M} x_1 \oplus x_2$ for some $M \in \left(\mathbb{Z}_2^l\right)^\infty$, and $M \| x_2 \in \mathsf{rng}(\mathsf{pad})$. The simulator needs to be designed such that the newly added value to $\bar{t}(V)$, $x_1 \oplus x_2 \oplus y_1 \oplus y_2$, is not a member of (the updated) $\mathsf{dom}(\mathcal{L}_\pi)$. This requirement translates to lines 014 and 107 in Figure C.1. Requirement R4 is clearly not applicable to queries to $\mathcal{S}_\rho$. Consider a query x_1 to $\mathcal{S}_\pi$, where $x_1 \in \bar{t}(V)$. Then, the simulator is forced to increase $\bar{t}(V) \cap \mathsf{dom}(\mathcal{L}_\pi)$. As $x_1 \in \bar{t}(V)$, there exists an M such that $\mathsf{iv} \xrightarrow{M} x_1$ and $\mathsf{depad}(M) \neq \bot$. The output of the simulator needs to be consistent with its random oracle, such that $\mathcal{R}(\mathsf{depad}(M)) = \mathsf{left}_n(\mathcal{S}_\pi(x_1) \oplus x_1)$. This requirement translates to lines 002-007 in the description of the simulator in Figure C.1.

C.1.3 Proof of Theorem 7.2.3

Let $\mathcal{S}$ be the simulator of Figure C.1, and let $\mathcal{D}$ be any distinguisher that makes at most q_L left queries of maximal length $(K-1)l$ bits, where $K \geq 1$, q_π right queries to π and q_ρ right queries to ρ. Recall from Definition 2.3.5 that the goal is to bound:

$$\mathbf{Adv}^{\mathrm{iff}}_{\mathcal{H}^{\pi,\rho},\mathcal{S}}(\mathcal{D}) = \left| \mathbf{Pr}\left(\mathcal{D}^{\mathcal{H}^{\pi,\rho},(\pi,\rho)} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{\mathcal{R},\mathcal{S}^{\mathcal{R}}} = 1\right) \right| . \tag{C.1}$$

Theorem 7.2.3 will be proven via a game-playing argument. Each game consists of a left and a right oracle. In the proof, G_1 will equal the simulated world, and G_6 the real world. We will go from game 1 to game 6 stepwise, and obtain a bound on (C.1) using a hybrid argument.

Game 1: $G_1 = (L_1, R_1^{L_1})$ (Figure C.2). The left oracle L_1 of game 1 is a lazily-sampled random oracle, and the four interfaces of the right oracle are the simulator of Figure C.1, except for the inclusion of a failure condition **bad** in lines 007, 015, 108, 204, 206, and 304. In other words, we have $G_1 = (\mathcal{R}, \mathcal{S}^{\mathcal{R}})$, and in particular, $\mathbf{Pr}\left(\mathcal{D}^{\mathcal{R},\mathcal{S}^{\mathcal{R}}} = 1\right) = \mathbf{Pr}\left(\mathcal{D}^{G_1} = 1\right)$.

Game 2: $G_2 = (L_2^{L_1}, R_1^{L_1})$ (Figure C.2). The left oracle of game 1 is replaced by a so-called relay oracle L_2 that passes the queries made by the distinguisher to L_1, and returns its responses. The right oracle remains unchanged. The distinguisher has identical views in G_1 and G_2, and we obtain $\mathbf{Pr}\left(\mathcal{D}^{G_1} = 1\right) = \mathbf{Pr}\left(\mathcal{D}^{G_2} = 1\right)$.

Game 3: $G_3 = (L_3^{R_1^{L_1}}, R_1^{L_1})$ (Figure C.2). The left oracle of game 2 is now replaced by an implementation of the Grøstl hash function, which moreover uses the right oracle as a subroutine, rather than L_1 directly. The right oracle itself remains unchanged. In Lemma C.1.1 it is proven that, as long as the **bad** flag is not set in any of the two games, both are identical. Formally, we obtain $\left|\mathbf{Pr}\left(\mathcal{D}^{G_2} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{G_3} = 1\right)\right| \leq \mathbf{Pr}\left(\mathcal{D}^{G_2} \text{ sets } \mathbf{bad}\right) + \mathbf{Pr}\left(\mathcal{D}^{G_3} \text{ sets } \mathbf{bad}\right)$.

Note that in game 3, as well as in all subsequent games, the right oracles π and ρ will both be queried at most $q := (K+1)q_L + \max\{q_\pi, q_\rho\}$ times. Indeed, in all of the following games, the left oracle queries π at most $K+1$ times and ρ at most K times. All subsequent right oracles are constructed in such a way that each query to this oracle adds at most one element to $\mathcal{L}_\pi$ or $\mathcal{L}_\rho$.

Game 4: $G_4 = (L_3^{R_2}, R_2)$ (Figure C.3). Game 4 differs from game 3 in the fact that the right oracle does not query subroutine L_1 anymore, but rather, it generates the outcomes itself. Concretely, in line 003, h is now randomly sampled from $\mathbb{Z}_2^n$. The distinguisher cannot notice the difference: as the padding rule is injective, in game 3 the right oracle $R_{1,\pi}$ will never query its left oracle twice on the same value, and hence it will always receive $h \overset{\$}{\leftarrow} \mathbb{Z}_2^n$. Formally, we obtain $\mathbf{Pr}\left(\mathcal{D}^{G_3} = 1\right) = \mathbf{Pr}\left(\mathcal{D}^{G_4} = 1\right)$.

Game 5: $G_5 = (L_3^{R_3}, R_3)$ (Figure C.3). Game 5 only differs from game 4 in the fact that the **GOTO**-statements are removed. In other words, game 5 and game 4 proceed identically until **bad** occurs. As a consequence, $\left|\mathbf{Pr}\left(\mathcal{D}^{G_4} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{G_5} = 1\right)\right| \leq \mathbf{Pr}\left(\mathcal{D}^{G_4} \text{ sets } \mathbf{bad}\right)$.

Game 6: $G_6 = (L_3^{R_4}, R_4)$ (Figure C.4). The left oracle of game 6 is the Grøstl algorithm, and the four interfaces of the right oracle perfectly mimic two lazily-sampled random permutations π and ρ. In other words, we have $G_6 = (\mathcal{H}^{\pi,\rho}, (\pi, \rho))$, and thus $\mathbf{Pr}\left(\mathcal{D}^{G_6} = 1\right) = \mathbf{Pr}\left(\mathcal{D}^{\mathcal{H}^{\pi,\rho},(\pi,\rho)} = 1\right)$. The only difference between games 6 and 5 is in the forward queries to $R_{3,\pi}$ and $R_{4,p}$: in game 5, some queries to $R_{3,\pi}$ are answered with uniform random samples from $\mathbb{Z}_2^l$. Therefore, distinguishing game 6 from game 5 is at least as hard as distinguishing a random permutation from a random function. We thus obtain $\left|\mathbf{Pr}\left(\mathcal{D}^{G_5} = 1\right) - \mathbf{Pr}\left(\mathcal{D}^{G_6} = 1\right)\right| \leq \frac{q^2}{2^l}$.

We conclude that (C.1) reduces to:

$$\mathbf{Adv}^{\text{iff}}_{\mathcal{H}^{\pi,\rho},\mathcal{S}}(\mathcal{D}) \leq \sum_{i=2}^{4} \mathbf{Pr}\left(\mathcal{D}^{G_i} \text{ sets } \mathbf{bad}\right) + \frac{q^2}{2^l}. \tag{C.2}$$

As we have $\mathbf{Pr}\left(\mathcal{D}^{G_2} \text{ sets } \mathbf{bad}\right) \leq \mathbf{Pr}\left(\mathcal{D}^{G_3} \text{ sets } \mathbf{bad}\right) = \mathbf{Pr}\left(\mathcal{D}^{G_4} \text{ sets } \mathbf{bad}\right)$, we focus on game G_4, and introduce an additional game 7 to simplify the analysis.

Game 7: $G_7 = (L_3^{R_5}, R_5)$ (Figure C.5). In game 7, the failure conditions for **bad** of game 4 are rewritten into sets $\Delta_0, \ldots, \Delta_4$. Suppose $\mathcal{D}^{G_4}$ sets **bad** in line 015. This means that for some $(x_2, y_2) \in \mathcal{L}_\rho$ such that $x_1 \oplus x_2 \in t(V)$ either one of the following two cases occurred:

$$y_1 = \begin{cases} x_1 \oplus x_2 \oplus y_2 \oplus v, \text{ for some } v \in V \cup (V_{\text{new}} \backslash \{x_1 \oplus x_2 \oplus y_1 \oplus y_2\}) \ , \\ x_1 \oplus x_2 \oplus y_2 \oplus x_1', \text{ for some } x_1' \in \mathsf{dom}(\mathcal{L}_\pi) \cup \{x_1\} \, . \end{cases}$$

By definition of Δ_1, this means that $y_1 \in \Delta_1$. Hence, $\mathcal{D}^{G_7}$ sets **bad** in line `011`. A similar observation holds for the other **bad** conditions of game 4. As a consequence, $\mathbf{Pr}\left(\mathcal{D}^{G_4} \text{ sets } \mathbf{bad}\right) \leq \mathbf{Pr}\left(\mathcal{D}^{G_7} \text{ sets } \mathbf{bad}\right)$, and it suffices to consider game 7.

Consider the j^{th} query ($j = 1, \ldots, q$) to R_5. By the union bound, the probability that **bad** is set in this round, $\mathbf{Pr}_j$, equals the probability that **bad** is set in either of the lines `007`, `011`, `104`, `204`, or `304`. Denote by $\mathbf{Pr}_j^{\mathtt{xxx}}$ the probability that **bad** is set in the j^{th} query in line `xxx`.

$\mathbf{Pr_j^{007}}$. **bad** is set if y_1, taken uniformly at random from a set of size 2^l, hits Δ_0. As Δ_0 is of size at most q, $\mathbf{Pr}_j^{007}$ is bounded by $\frac{q}{2^l}$;

$\mathbf{Pr_j^{011}}$. **bad** is set if y_1, taken uniformly at random from a set of size at least $2^l - q$ (we can assume **bad** is not set in line `007`), hits Δ_1. As Δ_1 is of size at most $q(2q^2+q)$, $\mathbf{Pr}_j^{011}$ is bounded by $\frac{q^2(2q+1)}{2^l-q}$;

$\mathbf{Pr_j^{104}}$. Similarly to $\mathbf{Pr}_j^{011}$, $\mathbf{Pr}_j^{104}$ is bounded by $\frac{q^2(2q+1)}{2^l-q}$;

$\mathbf{Pr_j^{204}}$. **bad** is set if x_1, taken uniformly at random from a set of size at least $2^l - q$, hits Δ_3. As Δ_3 is of size at most q^2+q^3, $\mathbf{Pr}_j^{204}$ is bounded by $\frac{q^2(q+1)}{2^l-q}$;

$\mathbf{Pr_j^{304}}$. **bad** is set if x_2, taken uniformly at random from a set of size at least $2^l - q$, hits Δ_4. As Δ_4 is of size at most q^3, $\mathbf{Pr}_j^{304}$ is bounded by $\frac{q^3}{2^l-q}$.

Summarizing, we obtain $\mathbf{Pr}_j \leq \frac{3q^2(2q+1)}{2^l-q} + \frac{q}{2^l}$. By the union bound, and under the assumption that $q < 2^l$, we thus obtain

$$\mathbf{Pr}\left(\mathcal{D}^{G_7} \text{ sets } \mathbf{bad}\right) \leq \frac{6q^3(2q+1)}{2^l} + \frac{q^2}{2^l} \, .$$

Now, combined with (C.2), this gives the claimed result.

Lemma C.1.1. *As long as* **bad** *is not set in games 2 and 3, both are identical. Formally,* $\mathbf{Pr}\left(\mathcal{D}^{G_2} = 1 \;\middle|\; \mathcal{D}^{G_2} \text{ sets } \neg\mathbf{bad}\right) = \mathbf{Pr}\left(\mathcal{D}^{G_3} = 1 \;\middle|\; \mathcal{D}^{G_3} \text{ sets } \neg\mathbf{bad}\right)$.

Proof. We need to prove that, until **bad** is set in either one of the two games, the query outcomes in games 2 and 3 are identically distributed. As both games employ the same right oracle, $\mathcal{D}$ can differentiate game 2 and 3 only if it discovers any inconsistencies in the answers by the left oracles (L_2 for game 2 and L_3 for game 3), given any list of queries made by $\mathcal{D}$ to the right oracle.

Consider an execution of game 2 or game 3. As both games have the same right oracle, we simply write R for R_i ($i = 2, 3$). Recall that $\mathcal{L}_\pi, \mathcal{L}_\rho$ consist of lists of query pairs to the right oracle, and that (V, A) is the graph defined by these. Denote by $\mathcal{V} = (\mathcal{V}_L, \mathcal{V}_\pi, \mathcal{V}_\rho)$ any view of a distinguisher on an execution of the oracle (G_2 or G_3). Here, $\mathcal{V}_L$ is a list of *different* query pairs (M, h), $\mathcal{V}_\pi$ a list of query pairs for the right oracle π, and similarly for $\mathcal{V}_\rho$. In game 2, we have $(\mathcal{V}_\pi, \mathcal{V}_\rho) = (\mathcal{L}_\pi, \mathcal{L}_\rho)$, and in game 3, we have $(\mathcal{V}_\pi, \mathcal{V}_\rho) \subseteq (\mathcal{L}_\pi, \mathcal{L}_\rho)$ (by construction). Denote by $(\overline{V}, \overline{A})$ the subgraph of (V, A) generated by the query pairs in $\mathcal{V}_\pi, \mathcal{V}_\rho$. We need to prove that, given any view $\mathcal{V}$, outcomes of new queries to the left oracle are identically distributed in both games. Formally, we need to prove that for any $M \in \mathbb{Z}_2^\infty$ and any $h \in \mathbb{Z}_2^n$, we have

$$\begin{aligned} &\mathbf{Pr}\left(L_2(M) = h \text{ in } G_2 \mid \mathcal{V};\ M \notin \mathsf{dom}(\mathcal{V}_L);\ \mathcal{D}^{G_2} \text{ sets } \neg\mathbf{bad}\right) \\ &= \mathbf{Pr}\left(L_3(M) = h \text{ in } G_3 \mid \mathcal{V};\ M \notin \mathsf{dom}(\mathcal{V}_L);\ \mathcal{D}^{G_3} \text{ sets } \neg\mathbf{bad}\right). \end{aligned} \tag{C.3}$$

Define $(M_1, \ldots, M_k) = \mathsf{pad}(M)$ to be the padded message of M. We will call the queried message M "determined" by $\mathcal{V}_\pi, \mathcal{V}_\rho$ if there exists a path $\mathsf{iv} \xrightarrow{M_1} \cdots \xrightarrow{M_k} h_k$ in $(\overline{V}, \overline{A})$ with $h_k \in \mathsf{dom}(\mathcal{V}_\pi)$. Similarly, we define determinability by $\mathcal{L}_\pi, \mathcal{L}_\rho$. We will prove that if M is not determined by $\mathcal{V}_\pi, \mathcal{V}_\rho$, both probabilities in (C.3) equal $1/2^n$. On the other hand, if M is determined by $\mathcal{V}_\pi, \mathcal{V}_\rho$, both properties are still equal (although they may naturally have a higher value).

M is determined by $\mathcal{V}_\pi, \mathcal{V}_\rho$. In both games, this means that the path $\mathsf{iv} \xrightarrow{M_1} \cdots h_{k-1} \xrightarrow{M_k} h_k$ is in the tree $\overline{t}(\overline{V})$, with $h_k \in \mathsf{dom}(\mathcal{V}_\pi)$. By Claim C.1.2, there are no colliding paths and in particular the described path is unique. Furthermore, due to Claim C.1.3, h_k had been added to $\mathsf{dom}(\mathcal{V}_\pi)$ in a forward query, *after* it was added to $\overline{t}(\overline{V})$. Therefore, by line 003, we have $R_\pi(h_k) = h_k \oplus (h \| w)$, where $h = L_1(M)$. As a consequence, $L_1(M)$, and thus $L_2(M)$ and $L_3(M)$, is fully determined by $\mathcal{V}_\pi, \mathcal{V}_\rho$, which means that the outcomes in game 2 and 3 are identically distributed;

M is determined by $\mathcal{L}_\pi, \mathcal{L}_\rho$ but not by $\mathcal{V}_\pi, \mathcal{V}_\rho$. This event is excluded for game 2, given $(\mathcal{V}_\pi, \mathcal{V}_\rho) = (\mathcal{L}_\pi, \mathcal{L}_\rho)$. In game 3, $\mathcal{L}_\pi, \mathcal{L}_\rho$ also include queries made to the right oracle via the left oracle L_3. We will show, however, that the probabilities of (C.3) equal $1/2^n$ then. Similarly to the previous case, there exists a path $\mathsf{iv} \xrightarrow{M_1} \cdots h_{k-1} \xrightarrow{M_k} h_k$ in the tree $\overline{t}(V)$, with $h_k \in \mathsf{dom}(\mathcal{L}_\pi)$. It satisfies $R_\pi(h_k) = h_k \oplus (h \| w)$, where $h = L_1(M)$. But M is *not* determined by $\mathcal{V}_\pi, \mathcal{V}_\rho$, which means that h_k had been queried to R_π independently of $\mathcal{V}_\pi, \mathcal{V}_\rho$. Furthermore, $L_3(M)$ is also independent of $\mathcal{V}_L$.[2] Concluding, (C.3) holds with probability $1/2^n$ in this case;

M is not determined by $\mathcal{L}_\pi, \mathcal{L}_\rho$. As a consequence, there either exists *no* path $\mathsf{iv} \xrightarrow{M_1} \cdots h_{k-1} \xrightarrow{M_k} h_k$ in the tree $\overline{t}(V)$, or there exists such path, but with

[2]Observe that in game 3, $\mathcal{V}_L$ consists of pairs $(\overline{M}, \overline{h})$ such that $\overline{h} = \mathsf{left}_n(R_\pi(\overline{h}_k) \oplus \overline{h}_k)$ for some $\overline{h}_k \in \overline{t}(V) \cap \mathsf{dom}(\mathcal{L}_\pi)$, where, by Claim C.1.3, $R_\pi(\overline{h}_k)$ had been generated via lines 003-007. As there are no colliding paths in (V, A) by Claim C.1.2, h_k differs from all such $\overline{h}_k$'s, and in particular $\mathcal{V}_L$ reveals nothing about $L_3(M)$.

$h_k \notin \mathsf{dom}(\mathcal{L}_\pi)$. For game 2, $M \notin \mathsf{dom}(\mathcal{V}_L)$ implies that M had not been queried to L_1 before (L_1 is queried in lines 003 and 500 only). Therefore, in this case $L_2(M)$ outputs a value h randomly sampled from $\mathbb{Z}_2^n$. For game 3, let $j \le k$ be the maximal index such that $\mathsf{iv} \xrightarrow{M_1} \cdots \xrightarrow{M_j} h_j$ is a path in (V, A). We consider the following cases:

(i) $j = k$. As M is not determined, we have $h_k \notin \mathsf{dom}(\mathcal{L}_\pi)$. In line 607, $R_\pi(h_k)$ will then be computed via lines 002-007: $R_\pi(h_k) = h_k \oplus (h\|w)$ for $h \xleftarrow{\$} \mathbb{Z}_2^n$. The outcome $L_3(M)$ thus equals $L_3(M) = \mathsf{left}_n(R_\pi(h_k) \oplus h_k) = h$. As a consequence, the outcomes of L_2 and L_3 are identically distributed in this case;

(ii) $j < k$. (V, A) contains no arc $h_j \to h_{j+1}$ labeled by M_{j+1}. By virtue of Claim C.1.2, in the $(j+1)^{\text{th}}$ iteration of lines 602-606, a new node h_{j+1} will be added to $t(V)$ such that h_{j+1} was not rooted yet and there is no outgoing arc from h_{j+1} in the updated graph. The same holds for all subsequent iterations, and in particular h_k will be newly added to $\bar{t}(V)$ in the k^{th} iteration. Due to Claim C.1.3, this newly added note is not an element of $\mathsf{dom}(\mathcal{L}_\pi)$ after this last round. Now, the same analysis as in case (i) applies. □

Lemma C.1.2. *Suppose $\mathcal{D}^{G_i}$ sets $\neg$**bad** (for $i = 2, 3$). Consider a node $h \in t(V)$, and a right oracle query in which an arc (h, v) will be added to (V, A). Denote by (V', A') the updated graph (after the query). Then, v has no incoming or outgoing arc in $(V', A' \backslash \{(v, w)\})$. As a consequence, after the execution of G_i, the final graph contains no colliding paths.*

Proof. As both games have the same right oracle, we simply write R for R_i ($i = 2, 3$). In a right query to R_π^{-1} or R_ρ^{-1}, none of the newly added arcs have a rooted node as starting point, by $\neg$**bad** in lines 206 and 304. Consider a query x_1 to R_π, and let (V, A) be the graph before the query. An outgoing arc from $h \in t(V)$ will only be added if $h = x_1 \oplus x_2$ for some $x_2 \in \mathsf{dom}(\mathcal{L}_\rho)$. By construction, the end node of the arc is $x_1 \oplus x_2 \oplus y_1 \oplus y_2 =: v$. By $\neg$**bad** in line 108, we have (i) $v \notin V$, (ii) none of the newly added arcs will leave from v, and (iii) apart from (h, v), none of the newly added arcs will arrive at v. As a consequence, v is an isolated node in $(V', A' \backslash \{(h, v)\})$. A similar argument holds for queries to R_ρ, by $\neg$**bad** in line 108.

We prove that the final graph contains no colliding paths by mathematical induction. Before the first query is made, $A = \emptyset$ and hence no colliding paths occur. Assume (V, A) contains no colliding paths and consider a right oracle query. We can sequentially apply the above reasoning and discard all newly added arcs (h, v) for $h \in t(V)$, in order to observe that colliding paths in (V', A') imply colliding paths in (V, A). By the induction hypothesis, these do not occur. □

Lemma C.1.3. *Suppose $\mathcal{D}^{G_i}$ sets $\neg$**bad** (for $i = 2, 3$). Consider a right oracle query in which a node w will be added to $\bar{t}(V)$. Then, w is no element of (the* *updated)* $\mathsf{dom}(\mathcal{L}_\pi)$. *Furthermore,* $\bar{t}(V) \cap \mathsf{dom}(\mathcal{L}_\pi)$ *will only be increased in forward queries to* $R_{i,\pi}$.

Proof. As both games have the same right oracle, we simply write R for R_i ($i = 2, 3$). As a direct consequence of Claim C.1.2, $\bar{t}(V)$ will be increased only if an arc $x_1 \oplus x_2 \xrightarrow{x_2} x_1 \oplus x_2 \oplus y_1 \oplus y_2$ is added such that $\mathsf{iv} \xrightarrow{M} x_1 \oplus x_2$ is a path in (V, A), and $M \| x_2 \in \mathsf{rng}(\mathsf{pad})$. By $\neg$**bad** in lines `015` and `108`, this newly added node is not an element of (the updated) $\mathsf{dom}(\mathcal{L}_\pi)$. Furthermore, an inverse query to R_π will never be answered with a node already in $\bar{t}(V)$, by $\neg$**bad** in line `204`, and therefore $\bar{t}(V) \cap \mathsf{dom}(\mathcal{L}_\pi)$ will only be increased in forward queries to R_π. $\square$

Forward Query $R_{1,\pi}(x_1)$

```
000 if $x_1 \in \mathsf{dom}(\mathcal{L}_\pi)$ :
001     return $y_1 = \mathcal{L}_\pi(x_1)$
002 if $x_1 \in \bar{t}(V)$ for $\mathsf{iv} \xrightarrow{M} x_1$ :
003     $h \xleftarrow{\$} L_1(\mathsf{depad}(M))$
004     $w \xleftarrow{\$} \mathbb{Z}_2^{l-n}$
005     $y_1 \leftarrow x_1 \oplus (h \| w)$
006     if $y_1 \in \mathsf{rng}(\mathcal{L}_\pi)$ :
007         bad ← true; GOTO 004
008 else : $y_1 \xleftarrow{\$} \mathbb{Z}_2^l \backslash \mathsf{rng}(\mathcal{L}_\pi)$
009 end if
010 $V_{\text{new}} \leftarrow \{x_1 \oplus x_2', x_1 \oplus x_2' \oplus y_1 \oplus y_2' \mid (x_2', y_2') \in \mathcal{L}_\rho\}$ multiset
011 forall $(x_2, y_2) \in \mathcal{L}_\rho$ with $\mathsf{iv} \xrightarrow{M} x_1 \oplus x_2$ :
012     $h_{\text{nxt}} \leftarrow x_1 \oplus x_2 \oplus y_1 \oplus y_2$
013     if $h_{\text{nxt}} \in V \cup (V_{\text{new}} \backslash \{h_{\text{nxt}}\})$ or
014         $\big(M \| x_2 \in \mathsf{rng}(\mathsf{pad})$ and $h_{\text{nxt}} \in \mathsf{dom}(\mathcal{L}_\pi) \cup \{x_1\}\big)$ :
015         bad ← true; GOTO 002
016 end forall
017 return $\mathcal{L}_\pi(x_1) \leftarrow y_1$
```

Forward Query $R_{1,\rho}(x_2)$

```
100 if $x_2 \in \mathsf{dom}(\mathcal{L}_\rho)$ :
101     return $y_2 = \mathcal{L}_\rho(x_2)$
102 $y_2 \xleftarrow{\$} \mathbb{Z}_2^l \backslash \mathsf{rng}(\mathcal{L}_\rho)$
103 $V_{\text{new}} \leftarrow \{x_1' \oplus x_2, x_1' \oplus x_2 \oplus y_1' \oplus y_2 \mid (x_1', y_1') \in \mathcal{L}_\pi\}$ multiset
104 forall $(x_1, y_1) \in \mathcal{L}_\pi$ with $\mathsf{iv} \xrightarrow{M} x_1 \oplus x_2$ :
105     $h_{\text{nxt}} \leftarrow x_1 \oplus x_2 \oplus y_1 \oplus y_2$
106     if $h_{\text{nxt}} \in V \cup (V_{\text{new}} \backslash \{h_{\text{nxt}}\})$ or
107         $\big(M \| x_2 \in \mathsf{rng}(\mathsf{pad})$ and $h_{\text{nxt}} \in \mathsf{dom}(\mathcal{L}_\pi)\big)$ :
108         bad ← true; GOTO 102
109 end forall
110 return $\mathcal{L}_\rho(x_2) \leftarrow y_2$
```

Inverse Query $R_{1,\pi}^{-1}(y_1)$

```
200 if $y_1 \in \mathsf{rng}(\mathcal{L}_\pi)$ :
201     return $x_1 = \mathcal{L}_\pi^{-1}(y_1)$
202 $x_1 \xleftarrow{\$} \mathbb{Z}_2^l \backslash \mathsf{dom}(\mathcal{L}_\pi)$
203 if $x_1 \in \bar{t}(V)$ :
204     bad ← true; GOTO 202
205 forall $x_2 \in \mathsf{dom}(\mathcal{L}_\rho)$ : if $x_1 \oplus x_2 \in t(V)$ :
206     bad ← true; GOTO 202
207 return $\mathcal{L}_\pi^{-1}(y_1) \leftarrow x_1$
```

Inverse Query $R_{1,\rho}^{-1}(y_2)$

```
300 if $y_2 \in \mathsf{rng}(\mathcal{L}_\rho)$ :
301     return $x_2 = \mathcal{L}_\rho^{-1}(y_2)$
302 $x_2 \xleftarrow{\$} \mathbb{Z}_2^l \backslash \mathsf{dom}(\mathcal{L}_\rho)$
303 forall $x_1 \in \mathsf{dom}(\mathcal{L}_\pi)$ : if $x_1 \oplus x_2 \in t(V)$ :
304     bad ← true; GOTO 302
305 return $\mathcal{L}_\rho^{-1}(y_2) \leftarrow x_2$
```

Query $L_1(M)$

```
400 if $M \in \mathsf{dom}(\mathcal{H})$ :
401     return $h = \mathcal{H}(M)$
402 $h \xleftarrow{\$} \mathbb{Z}_2^n$
403 return $\mathcal{H}(M) \leftarrow h$
```

Query $L_2(M)$

```
500 return $h \xleftarrow{\$} L_1(M)$
```

Query $L_3(M)$

```
600 $(M_1, \dots, M_k) \leftarrow \mathsf{pad}(M)$
601 $h_0 \leftarrow \mathsf{iv}$
602 for $i = 1, \dots, k$ :
603     $x \leftarrow R_{1,\rho}(M_i)$
604     $y \leftarrow R_{1,\pi}(h_{i-1} \oplus M_i)$
605     $h_i \leftarrow x \oplus y \oplus h_{i-1}$
606 end for
607 $z \leftarrow R_{1,\pi}(h_k)$
608 $h \leftarrow \mathsf{left}_{l-n}(z \oplus h_k)$
609 return $h$
```

Figure C.2. Games 1, 2, and 3. In game 1, the distinguisher has access to $L_1, R_1^{L_1}$ (oracles L_2, L_3 are ignored). In game 2, the distinguisher has access to $L_2^{L_1}, R_1^{L_1}$ (oracle L_3 is ignored). In game 3, the distinguisher has access to $L_3^{R_1^{L_1}}, R_1^{L_1}$ (oracle L_2 is ignored). Oracle L_1 maintains an initially empty table $\mathcal{H}$.

Forward Query $R_{\alpha,\pi}(x_1)$

```
000 if $x_1 \in \mathsf{dom}(\mathcal{L}_\pi)$ :
001     return $y_1 = \mathcal{L}_\pi(x_1)$
002 if $x_1 \in \bar{t}(V)$ :
003     $h \xleftarrow{\$} \mathbb{Z}_2^n$
004     $w \xleftarrow{\$} \mathbb{Z}_2^{l-n}$
005     $y_1 \leftarrow x_1 \oplus (h \| w)$
006     if $y_1 \in \mathsf{rng}(\mathcal{L}_\pi)$ :
007         bad $\leftarrow$ true; $\boxed{\textbf{GOTO } 004}$
008 else : $y_1 \xleftarrow{\$} \mathbb{Z}_2^l \backslash \mathsf{rng}(\mathcal{L}_\pi)$
009 end if
010 $V_{\text{new}} \leftarrow \{x_1 \oplus x_2', x_1 \oplus x_2' \oplus y_1 \oplus y_2' \mid (x_2', y_2') \in \mathcal{L}_\rho\}$ multiset
011 forall $(x_2, y_2) \in \mathcal{L}_\rho$ with $\mathsf{iv} \xrightarrow{M} x_1 \oplus x_2$ :
012     $h_{\text{nxt}} \leftarrow x_1 \oplus x_2 \oplus y_1 \oplus y_2$
013     if $h_{\text{nxt}} \in V \cup (V_{\text{new}} \backslash \{h_{\text{nxt}}\})$ or
014         $\Big(M \| x_2 \in \mathsf{rng}(\mathsf{pad})$ and $h_{\text{nxt}} \in \mathsf{dom}(\mathcal{L}_\pi) \cup \{x_1\}\Big)$ :
015         bad $\leftarrow$ true; $\boxed{\textbf{GOTO } 002}$
016 end forall
017 return $\mathcal{L}_\pi(x_1) \leftarrow y_1$
```

Inverse Query $R_{\alpha,\pi}^{-1}(y_1)$

```
200 if $y_1 \in \mathsf{rng}(\mathcal{L}_\pi)$ :
201     return $x_1 = \mathcal{L}_\pi^{-1}(y_1)$
202 $x_1 \xleftarrow{\$} \mathbb{Z}_2^l \backslash \mathsf{dom}(\mathcal{L}_\pi)$
203 if $x_1 \in \bar{t}(V)$ :
204     bad $\leftarrow$ true; $\boxed{\textbf{GOTO } 202}$
205 forall $x_2 \in \mathsf{dom}(\mathcal{L}_\rho)$ : if $x_1 \oplus x_2 \in t(V)$ :
206     bad $\leftarrow$ true; $\boxed{\textbf{GOTO } 202}$
207 return $\mathcal{L}_\pi^{-1}(y_1) \leftarrow x_1$
```

Inverse Query $R_{\alpha,\rho}^{-1}(y_2)$

```
300 if $y_2 \in \mathsf{rng}(\mathcal{L}_\rho)$ :
301     return $x_2 = \mathcal{L}_\rho^{-1}(y_2)$
302 $x_2 \xleftarrow{\$} \mathbb{Z}_2^l \backslash \mathsf{dom}(\mathcal{L}_\rho)$
303 forall $x_1 \in \mathsf{dom}(\mathcal{L}_\pi)$ : if $x_1 \oplus x_2 \in t(V)$ :
304     bad $\leftarrow$ true; $\boxed{\textbf{GOTO } 302}$
305 return $\mathcal{L}_\rho^{-1}(y_2) \leftarrow x_2$
```

Forward Query $R_{\alpha,\rho}(x_2)$

```
100 if $x_2 \in \mathsf{dom}(\mathcal{L}_\rho)$ :
101     return $y_2 = \mathcal{L}_\rho(x_2)$
102 $y_2 \xleftarrow{\$} \mathbb{Z}_2^l \backslash \mathsf{rng}(\mathcal{L}_\rho)$
103 $V_{\text{new}} \leftarrow \{x_1' \oplus x_2, x_1' \oplus x_2 \oplus y_1' \oplus y_2 \mid (x_1', y_1') \in \mathcal{L}_\pi\}$ multiset
104 forall $(x_1, y_1) \in \mathcal{L}_\pi$ with $\mathsf{iv} \xrightarrow{M} x_1 \oplus x_2$ :
105     $h_{\text{nxt}} \leftarrow x_1 \oplus x_2 \oplus y_1 \oplus y_2$
106     if $h_{\text{nxt}} \in V \cup (V_{\text{new}} \backslash \{h_{\text{nxt}}\})$ or
107         $\Big(M \| x_2 \in \mathsf{rng}(\mathsf{pad})$ and $h_{\text{nxt}} \in \mathsf{dom}(\mathcal{L}_\pi)\Big)$ :
108         bad $\leftarrow$ true; $\boxed{\textbf{GOTO } 102}$
109 end forall
110 return $\mathcal{L}_\rho(x_2) \leftarrow y_2$
```

Query $L_3(M)$

```
600 $(M_1, \ldots, M_k) \leftarrow \mathsf{pad}(M)$
601 $h_0 \leftarrow \mathsf{iv}$
602 for $i = 1, \ldots, k$ :
603     $x \leftarrow R_{\alpha,\rho}(M_i)$
604     $y \leftarrow R_{\alpha,\pi}(h_{i-1} \oplus M_i)$
605     $h_i \leftarrow x \oplus y \oplus h_{i-1}$
606 end for
607 $z \leftarrow R_{\alpha,\pi}(h_k)$
608 $h \leftarrow \mathsf{left}_{l-n}(z \oplus h_k)$
609 return $h$
```

Figure C.3. Game 4 (for $\alpha = 2$, including the boxed statements) and game 5 (for $\alpha = 3$, with the boxed statements removed). In both games, the distinguisher has access to $L_3^{R_\alpha}, R_\alpha$.

Forward Query $R_{4,\pi}(x_1)$

```
000 if $x_1 \in \mathsf{dom}(\mathcal{L}_\pi)$ :
001     return $y_1 = \mathcal{L}_\pi(x_1)$
002 $y_1 \xleftarrow{\$} \mathbb{Z}_2^l \backslash \mathsf{rng}(\mathcal{L}_\pi)$
003 return $\mathcal{L}_\pi(x_1) \leftarrow y_1$
```

Inverse Query $R_{4,\pi}^{-1}(y_1)$

```
200 if $y_1 \in \mathsf{rng}(\mathcal{L}_\pi)$ :
201     return $x_1 = \mathcal{L}_\pi^{-1}(y_1)$
202 $x_1 \xleftarrow{\$} \mathbb{Z}_2^l \backslash \mathsf{dom}(\mathcal{L}_\pi)$
203 return $\mathcal{L}_\pi^{-1}(y_1) \leftarrow x_1$
```

Query $L_3(M)$

```
600 $(M_1, \ldots, M_k) \leftarrow \mathsf{pad}(M)$
601 $h_0 \leftarrow \mathsf{iv}$
602 for $i = 1, \ldots, k$ :
603     $x \leftarrow R_{4,\rho}(M_i)$
604     $y \leftarrow R_{4,\pi}(h_{i-1} \oplus M_i)$
605     $h_i \leftarrow x \oplus y \oplus h_{i-1}$
606 end for
607 $z \leftarrow R_{4,\pi}(h_k)$
608 $h \leftarrow \mathsf{left}_{l-n}(z \oplus h_k)$
609 return $h$
```

Forward Query $R_{4,\rho}(x_2)$

```
100 if $x_2 \in \mathsf{dom}(\mathcal{L}_\rho)$ :
101     return $y_2 = \mathcal{L}_\rho(x_2)$
102 $y_2 \xleftarrow{\$} \mathbb{Z}_2^l \backslash \mathsf{rng}(\mathcal{L}_\rho)$
103 return $\mathcal{L}_\rho(x_2) \leftarrow y_2$
```

Inverse Query $R_{4,\rho}^{-1}(y_2)$

```
300 if $y_2 \in \mathsf{rng}(\mathcal{L}_\rho)$ :
301     return $x_2 = \mathcal{L}_\rho^{-1}(y_2)$
302 $x_2 \xleftarrow{\$} \mathbb{Z}_2^l \backslash \mathsf{dom}(\mathcal{L}_\rho)$
303 return $\mathcal{L}_\rho^{-1}(y_2) \leftarrow x_2$
```

Figure C.4. Game 6. The distinguisher has access to $L_3^{R_4}, R_4$.

Forward Query $R_{5,\pi}(x_1)$

```
000 if $x_1 \in \mathsf{dom}(\mathcal{L}_\pi)$ :
001     return $y_1 = \mathcal{L}_\pi(x_1)$
002 if $x_1 \in \bar{t}(V)$ :
003     $h \xleftarrow{\$} \mathbb{Z}_2^n$
004     $w \xleftarrow{\$} \mathbb{Z}_2^{l-n}$
005     $y_1 \leftarrow x_1 \oplus (h \| w)$
006     if $y_1 \in \Delta_0$ :
007         bad $\leftarrow$ true; GOTO 004
008 else : $y_1 \xleftarrow{\$} \mathbb{Z}_2^l \backslash \mathsf{rng}(\mathcal{L}_\pi)$
009 end if
010 if $y_1 \in \Delta_1$ :
011     bad $\leftarrow$ true; GOTO 002
012 return $\mathcal{L}_\pi(x_1) \leftarrow y_1$
```

Forward Query $R_{5,\rho}(x_2)$

```
100 if $x_2 \in \mathsf{dom}(\mathcal{L}_\rho)$ :
101     return $y_2 = \mathcal{L}_\rho(x_2)$
102 $y_2 \xleftarrow{\$} \mathbb{Z}_2^l \backslash \mathsf{rng}(\mathcal{L}_\rho)$
103 if $y_2 \in \Delta_2$ :
104     bad $\leftarrow$ true; GOTO 102
105 return $\mathcal{L}_\rho(x_2) \leftarrow y_2$
```

Inverse Query $R_{5,\pi}^{-1}(y_1)$

```
200 if $y_1 \in \mathsf{rng}(\mathcal{L}_\pi)$ :
201     return $x_1 = \mathcal{L}_\pi^{-1}(y_1)$
202 $x_1 \xleftarrow{\$} \mathbb{Z}_2^l \backslash \mathsf{dom}(\mathcal{L}_\pi)$
203 if $x_1 \in \Delta_3$ :
204     bad $\leftarrow$ true; GOTO 202
205 return $\mathcal{L}_\pi^{-1}(y_1) \leftarrow x_1$
```

Inverse Query $R_{5,\rho}^{-1}(y_2)$

```
300 if $y_2 \in \mathsf{rng}(\mathcal{L}_\rho)$ :
301     return $x_2 = \mathcal{L}_\rho^{-1}(y_2)$
302 $x_2 \xleftarrow{\$} \mathbb{Z}_2^l \backslash \mathsf{dom}(\mathcal{L}_\rho)$
303 if $x_2 \in \Delta_4$ :
304     bad $\leftarrow$ true; GOTO 302
305 return $\mathcal{L}_\rho^{-1}(y_2) \leftarrow x_2$
```

Query $L_3(M)$

```
600 $(M_1, \ldots, M_k) \leftarrow \mathsf{pad}(M)$
601 $h_0 \leftarrow \mathsf{iv}$
602 for $i = 1, \ldots, k$ :
603     $x \leftarrow R_{5,\rho}(M_i)$
604     $y \leftarrow R_{5,\pi}(h_{i-1} \oplus M_i)$
605     $h_i \leftarrow x \oplus y \oplus h_{i-1}$
606 end for
607 $z \leftarrow R_{5,\pi}(h_k)$
608 $h \leftarrow \mathsf{left}_{l-n}(z \oplus h_k)$
609 return $h$
```

$$
\begin{aligned}
\Delta_0 = \;& \mathsf{rng}(\mathcal{L}_\pi);\\
\Delta_1 = \;& \textstyle\bigcup_{(x_2,y_2)\in\mathcal{L}_\rho} \Big(\{x_1 \oplus x_2 \oplus y_2 \oplus v \mid v \in V \cup (V_{\text{new}} \backslash \{x_1 \oplus x_2 \oplus y_1 \oplus y_2\})\} \cup \\
& \qquad \{x_1 \oplus x_2 \oplus y_2 \oplus x_1' \mid x_1' \in \mathsf{dom}(\mathcal{L}_\pi) \cup \{x_1\}\}\Big),\\
& \text{where } V_{\text{new}} = \{x_1 \oplus x_2', x_1 \oplus x_2' \oplus y_1 \oplus y_2' \mid (x_2', y_2') \in \mathcal{L}_\rho\} \text{ is a multiset;}\\
\Delta_2 = \;& \textstyle\bigcup_{(x_1,y_1)\in\mathcal{L}_\pi} \Big(\{x_1 \oplus x_2 \oplus y_1 \oplus v \mid v \in V \cup (V_{\text{new}} \backslash \{x_1 \oplus x_2 \oplus y_1 \oplus y_2\})\} \cup \\
& \qquad \{x_1 \oplus x_2 \oplus y_1 \oplus x_1' \mid x_1' \in \mathsf{dom}(\mathcal{L}_\pi)\}\Big),\\
& \text{where } V_{\text{new}} = \{x_1' \oplus x_2, x_1' \oplus x_2 \oplus y_1' \oplus y_2 \mid (x_1', y_1') \in \mathcal{L}_\pi\} \text{ is a multiset;}\\
\Delta_3 = \;& \bar{t}(V) \cup \{x_2 \oplus v \mid x_2 \in \mathsf{dom}(\mathcal{L}_\rho), v \in t(V)\};\\
\Delta_4 = \;& \{x_1 \oplus v \mid x_1 \in \mathsf{dom}(\mathcal{L}_\pi), v \in t(V)\}.
\end{aligned}
$$

Figure C.5. Game 7. The distinguisher has access to $L_3^{R_5}, R_5$.

Curriculum Vitae

Bart Mennink was born on November 10, 1985 in Born, The Netherlands. He studied Mathematics and Computer Science at Technische Universiteit Eindhoven, The Netherlands, receiving his Bachelor degree and Master degree (track "Discrete Mathematics and Applications") in Mathematics in 2007 and 2009, respectively, both with the greatest distinction (cum laude). His Master's thesis dealt with encrypted credential schemes and was completed during a nine month internship at Philips Research Labs, Eindhoven, The Netherlands.

In October 2009, Bart joined the research group COSIC (Computer Security and Industrial Cryptography) at the Department of Electrical Engineering of Katholieke Universiteit Leuven, Belgium, as a PhD student. His PhD research was sponsored by a four year fellowship from the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT). The research during his PhD studies lead to over fifteen publications in peer-reviewed international conferences and journals, most notably two publications at the IACR flagship conferences CRYPTO and ASIACRYPT. Together with his co-authors, Bart was awarded with the best paper award at AFRICACRYPT 2012 and at CANS 2012.

List of Publications

International Conferences

[1] Elena Andreeva, Andrey Bogdanov, and Bart Mennink. Towards understanding the known-key security of block ciphers. In *Fast Software Encryption – FSE 2013*, Lecture Notes in Computer Science, Singapore, March 11–13, 2013. Springer, Berlin, Germany. To appear.

[2] Florian Mendel, Bart Mennink, Vincent Rijmen, and Elmar Tischhauser. A simple key-recovery attack on McOE-X. In Josef Pieprzyk, Ahmad-Reza Sadeghi, and Mark Manulis, editors, *CANS 2012: 11th International Conference on Cryptology and Network Security*, volume 7712 of *Lecture Notes in Computer Science*, pages 23–31, Darmstadt, Germany, December 12–14, 2012. Springer, Berlin, Germany. Best paper award.

[3] 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*, volume 7658 of *Lecture Notes in Computer Science*, pages 526–543, Beijing, China, December 2–6, 2012. Springer, Berlin, Germany.

[4] 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*, volume 7417 of *Lecture Notes in Computer Science*, pages 330–347, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Berlin, Germany.

[5] Elena Andreeva, Atul Luykx, and Bart Mennink. Provable security of BLAKE with non-ideal compression function. In Lars R. Knudsen, editor, *SAC 2012: 19th Annual International Workshop on Selected Areas in Cryptography*, volume 7707 of *Lecture Notes in Computer Science*, pages 322–339, Windsor, Ontario, Canada, August 16–17, 2012. Springer, Berlin, Germany.

[6] Elena Andreeva, Bart Mennink, Bart Preneel, and Marjan Škrobot. Security analysis and comparison of the SHA-3 Finalists BLAKE, Grøstl, JH,

Keccak, and Skein. In Aikaterini Mitrokotsa and Serge Vaudenay, editors, *AFRICACRYPT 2012: 5th International Conference on Cryptology in Africa*, volume 7374 of *Lecture Notes in Computer Science*, pages 287–305, Ifrance, Morocco, July 10–12, 2012. Springer, Berlin, Germany. Best paper award.

[7] Elena Andreeva and Bart Mennink. Provable chosen-target-forced-midfix preimage resistance. In Ali Miri and Serge Vaudenay, editors, *SAC 2011: 18th Annual International Workshop on Selected Areas in Cryptography*, volume 7118 of *Lecture Notes in Computer Science*, pages 37–54, Toronto, Ontario, Canada, August 11–12, 2011. Springer, Berlin, Germany.

[8] Jorge Guajardo, Bart Mennink, and Berry Schoenmakers. Anonymous credential schemes with encrypted attributes. In Swee-Huay Heng, Rebecca N. Wright, and Bok-Min Goi, editors, *CANS 2010: 9th International Conference on Cryptology and Network Security*, volume 6467 of *Lecture Notes in Computer Science*, pages 314–333, Kuala Lumpur, Malaysia, December 12–14, 2010. Springer, Berlin, Germany.

[9] Jorge Guajardo and Bart Mennink. On side-channel resistant block cipher usage. In Mike Burmester, Gene Tsudik, Spyros S. Magliveras, and Ivana Ilic, editors, *ISC 2010: 13th International Conference on Information Security*, volume 6531 of *Lecture Notes in Computer Science*, pages 254–268, Boca Raton, FL, USA, October 25–28, 2010. Springer, Berlin, Germany.

[10] 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, *ISC 2010: 13th International Conference on Information Security*, volume 6531 of *Lecture Notes in Computer Science*, pages 39–53, Boca Raton, FL, USA, October 25–28, 2010. Springer, Berlin, Germany.

[11] Elena Andreeva, Bart Mennink, and Bart Preneel. On the indifferentiability of the Grøstl hash function. In Juan A. Garay and Roberto De Prisco, editors, *SCN 2010: 7th International Conference on Security in Communication Networks*, volume 6280 of *Lecture Notes in Computer Science*, pages 88–105, Amalfi, Italy, September 13–15, 2010. Springer, Berlin, Germany.

[12] Jorge Guajardo, Bart Mennink, and Berry Schoenmakers. Modulo reduction for Paillier encryptions and application to secure statistical analysis (extended abstract). In Radu Sion, editor, *FC 2010: 14th International Conference on Financial Cryptography and Data Security*, volume 6052 of *Lecture Notes in Computer Science*, pages 375–382, Tenerife, Canary Islands, Spain, January 25–28, 2010. Springer, Berlin, Germany.

International Journals

[1] Bart Mennink. On the collision and preimage security of MDC-4 in the ideal cipher model. *Designs, Codes and Cryptography*, 2013. To appear.

[2] Elena Andreeva, Bart Mennink, and Bart Preneel. The parazoa family: Generalizing the sponge hash functions. *International Journal of Information Security*, 11(3):149–165, 2012.

[3] Elena Andreeva, Andrey Bogdanov, Bart Mennink, Bart Preneel, and Christian Rechberger. On security arguments of the second round SHA-3 candidates. *International Journal of Information Security*, 11(2):103–120, 2012.

[4] Bart Mennink. Increasing the flexibility of the herding attack. *Information Processing Letters*, 112(3):98–105, 2012.

[5] Elena Andreeva, Bart Mennink, and Bart Preneel. Security properties of domain extenders for cryptographic hash functions. *Journal of Information Processing Systems – JIPS*, 6(4):453–480, 2010.

Workshops

[1] Elena Andreeva, Bart Mennink, Bart Preneel, and Marjan Škrobot. Security analysis and comparison of the SHA-3 Finalists BLAKE, Grøstl, JH, Keccak, and Skein. Bulgarian Cryptography Days 2012 (BulCrypt 2012), 2012.

[2] Elena Andreeva, Bart Mennink, Bart Preneel, and Marjan Škrobot. Security analysis and comparison of the SHA-3 Finalists BLAKE, Grøstl, JH, Keccak, and Skein. NIST's 3rd SHA-3 Candidate Conference 2012, 2012.

[3] Elena Andreeva, Atul Luykx, and Bart Mennink. Provable security of BLAKE with non-ideal compression function. NIST's 3rd SHA-3 Candidate Conference 2012, 2012.

[4] Elena Andreeva, Bart Mennink, and Bart Preneel. The parazoa family: Generalizing the sponge hash functions. ECRYPT II Hash Workshop 2011, 2011.

[5] Elena Andreeva, Bart Mennink, and Bart Preneel. Security reductions of the SHA-3 candidates. NIST's 2nd SHA-3 Candidate Conference 2010, 2010.

[6] Elena Andreeva, Bart Mennink, and Bart Preneel. On the indifferentiability of the Grøstl hash function. NIST's 2nd SHA-3 Candidate Conference 2010, 2010.

[7] Jorge Guajardo and Bart Mennink. Towards side-channel resistant block cipher usage or can we encrypt without side-channel countermeasures? 4th Benelux Workshop on Information and System Security (WISSec 2009), 2009.

[8] Jorge Guajardo, Bart Mennink, and Berry Schoenmakers. Modulo reduction for Paillier encryptions and application to secure statistical analysis. 4th Benelux Workshop on Information and System Security (WISSec 2009), 2009.

[9] Jorge Guajardo, Bart Mennink, and Berry Schoenmakers. Modulo reduction for Paillier encryptions and application to secure statistical analysis. The International Workshop on Signal Processing in the EncryptEd Domain (SPEED 2009), 2009.

Unpublished Manuscripts

[1] Elena Andreeva, Andrey Bogdanov, Yevgeniy Dodis, Bart Mennink, and John P. Steinberger. On the indifferentiability of key-alternating ciphers. Cryptology ePrint Archive, Report 2013/061, 2013.

[2] Atul Luykx, Elena Andreeva, Bart Mennink, and Bart Preneel. Impossibility results for indifferentiability with resets. Cryptology ePrint Archive, Report 2012/644, 2012.

Theses

[1] Bart Mennink. Encrypted certificate schemes and their security and privacy analysis. Master's thesis, Eindhoven University of Technology, Eindhoven, 2009.

[2] Bart Mennink. Puntsgewijze convergentie van Fourierreeksen. Bachelor's thesis, Eindhoven University of Technology, Eindhoven, 2007.

Faculty of Engineering Science
Department of Electrical Engineering (ESAT)
COmputer Security and Industrial Cryptography (COSIC)
Kasteelpark Arenberg 10 — 2446
B-3001 Heverlee (Belgium)