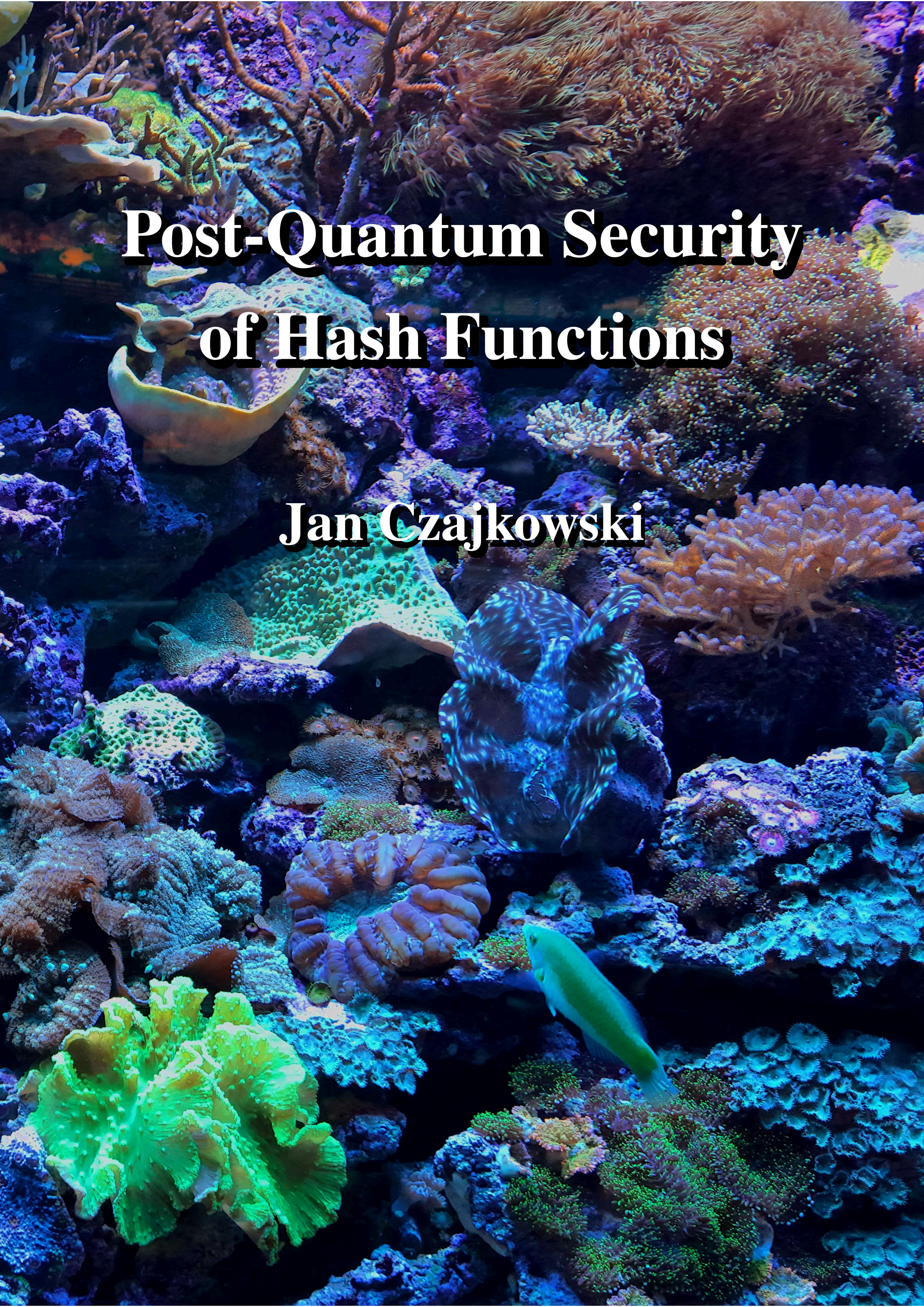
Post-Quantum Security of Hash Functions

Jan Czajkowski



Post-Quantum Security of Hash Functions

Jan Czajkowski

Post-Quantum Security of Hash Functions

ILLC Dissertation Series DS-2022-02

INSTITUTE FOR LOGIC, LANGUAGE AND COMPUTATION

For further information about ILLC-publications, please contact

Institute for Logic, Language and Computation
Universiteit van Amsterdam
Science Park 107
1098 XG Amsterdam
phone: +31-20-525 6051
e-mail: `illc@uva.nl`
homepage: `http://www.illc.uva.nl/`

The investigations were carried out at QuSoft and Centrum Wiskunde & Informatica and supported by the Netherlands Organization for Scientific Research (NWO), in particular by the NWO VIDI grant: Project No. 639.022.519.

Printed and bound by Strefa Xero - Druk i Reklama, Warszawa.

ISBN: 978-90-9035-560-3

Post-Quantum Security of Hash Functions

Academisch Proefschrift

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. ir. K.I.J. Maex
ten overstaan van een door het College voor Promoties
ingestelde commissie,
in het openbaar te verdedigen in de Agnietenkapel
op dinsdag 18 januari 2022, te 15.00 uur

door

Jan Marcin Czajkowski

geboren te Warschau

Promotiecommisie

Promotores:	dr. C. Schaffner	Universiteit van Amsterdam
	prof. dr. H.M. Buhrman	Universiteit van Amsterdam and Centrum Wiskunde & Informatica
Overige leden:	prof. dr. S.O. Fehr	Universiteit Leiden and Centrum Wiskunde & Informatica
	dr. P. Grosso	Universiteit van Amsterdam
	dr. A.T. Hülsing	Technische Universiteit Eindhoven
	prof. dr. D. Unruh	University of Tartu
	dr. M. Walter	Universiteit van Amsterdam
	prof. dr. R.M. de Wolf	Universiteit van Amsterdam and Centrum Wiskunde & Informatica

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

To Ola and Ewa

ACKNOWLEDGMENTS

The last almost five years of my life were extremely important to me. One of the most crucial lessons was how to balance my passion for scientific work and love for my family. I could not finish my PhD without the constant encouragement of my wife Ola. She always believed in me and offered unwavering support. She put up with me coming home after 11 pm just to leave before 9 am the next day and, equally importantly, reminded me to go for holidays.

I also thank my supervisors: Christian Schaffner and Harry Buhrman. My great gratitude goes to Chris. First of all, he employed me when I had almost no experience in cryptography and did not even hide my ignorance. Nonetheless, thanks to his decision I believe I found the subject that now defines my interests, both in research and in life. He always was ready to give me valuable advice on any career and life decisions.

When I started working at UvA, Chris had a project on post-quantum security of the sponge construction ready and waiting for me. A careful reader of the abstract of this thesis will note that almost all results that I present here are about the sponge construction. I am very thankful for the fact that he had such a concrete plan for the beginning of my PhD, thanks to that I had a clear direction and a goal. This made it doable to get into a new field of study and do research in it.

I thank Harry for inviting me to QuSoft, a vibrant institute where I met plenty of very interesting people. It wouldn't be such an active and stimulating place of work without him.

The people I learned the most from, on subjects related to cryptography and scientific work are my collaborators: Chris Schaffner, Andreas Hülsing, Chris Majenz, Leon Groot Bruiderink, Dominique Unruh, Bart Mennink, Sebastian

Zur, and Alex Grilo. I thank all of them and hope to work with them again in new exciting projects.

I also want to thank all my family for their support, from the moment I decided to go to Amsterdam to the very end of my PhD. I am especially grateful for their help in the final months, when they came to Amsterdam to help me take care of my daughter so that I could write a thesis I am proud of.

Working at CWI was an amazing experience, with all the great people working there it was always interesting and fun. There were plenty opportunities to exchange knowledge, but also some time to play table tennis and, most importantly, foosball.

For the most part of my PhD I had the pleasure to share my office with Yfke. Initially in a two-people office and later in a larger office shared with other PhD students. I remember many times that we talked with Yfke about PhD-student problems, or just life, with our feet on the window sills. Thank you for your good advice and understanding.

I also thank the CWI PhD Activity Committee: Giulia, Esteban, Mathé and Mark, for taking me in as a member and organizing great events. In CWI, I finally found a group to play RPG games. The person who started is was Alvaro, but I also thank Willem Jan, Brinn, Wessel, and Jonas for the loads of fun we had playing different games. Among those, from whom I learned the most about how to deal with the hardships of doing a PhD, are Srinivasan, András, and Jeroen. They offered great advice to their younger colleagues which I hope to pass on. It doesn't give justice to the CWI crowd, but I would like to mention and thank Joran, Bas, Isabella, Farokh, Tom, Koen, Suba, Arjan, and Florian for all the time we spent together.

CONTENTS

LIST OF PUBLICATIONS

The content of this dissertation is based on the following articles. The author names are ordered alphabetically, and all authors contributed equally.

[Cza+18] **Post-quantum Security of the Sponge Construction**, with Leon Groot Bruinderink, Andreas Hülsing, Christian Schaffner, and Dominique Unruh, published in the proceedings of *PQCrypto 2018*,

[CHS19] **Quantum Indistinguishability of Random Sponges**, with Andreas Hülsing and Christian Schaffner, published in the proceedings of *CRYPTO 2019*,

[Cza+19] **Quantum Lazy Sampling and Game-Playing Proofs for Quantum Indifferentiability**, with Christian Majenz, Christian Schaffner, and Sebastian Zur, *ArXiv* preprint,

[CG21] **On-State Commutativity of Measurements and Joint Distributions of Their Outcomes**, with Alex B. Grilo, *ArXiv* preprint.

During his PhD, the author has also co-authored the following papers that are not part of this thesis. The first two publications cover subjects that are almost not related to the main topic of this thesis. The last article listed below was not included in the thesis because it was written at the last stages of preparation of the manuscript.

[DCS17] **Adaptive Quantum Metrology under General Markovian Noise** with Rafał Demkowicz-Dobrzański and Pavel Sekatski published in 2017 *Physical Review X* **7**, 041009,

[CPD19] **Many-body effects in quantum metrology** with Rafał Demkowicz-Dobrzański and Krzysztof Pawłowski published in 2019 *New Journal of Physics* **21** 053031.

CHAPTER

1 INTRODUCTION

Chapter contents

The very beginnings of cryptography can already be found in ancient times. It is said that Julius Caesar used a shift cipher[1] to communicate with his generals.

For the most part of history, cryptography was the art of encrypting (enciphering) messages. Even if the techniques got more elaborate—e.g. substitution ciphers more complicated than the Caesar cipher, Vigenère cipher, or the one-time pad—the goal was the same: to hide a message from the unwanted eye. In modern language this part of cryptography is called *private-key* cryptography; To achieve security a private key has to be exchanged between the communicating parties.

Only in recent times cryptography became so widespread, most people on earth use cryptography every day. Whenever we use a secure `https` website, we use cryptography, whenever we use a credit card, we use cryptography, whenever we use an on-line messaging service, we use cryptography. One reason for how widespread cryptography is are the advances in information technology: the Internet, smartphones, etc.. Another reason are the advances in theory of computer science.

In 1976 Whitfield Diffie and Martin Hellman [DH76] published their findings that changed the world. They showed that it is possible for two parties communicating solely over a public channel and using only publicly known techniques to create a secure connection. Meaning, since that time it was no longer necessary to share a secret private key with the recipient of our message. These ideas are the core of what is now called *public-key* cryptography. The actual discovery of public-key cryptography is dated back to 1970, when James H. Ellis wrote a *classified* document discussing the possibility of "non-secret encryption" [Ell70].

Public-key cryptography offers a plethora of applications: public-key encryption (widely used schemes are RSA and Diffie-Hellman), digital signatures (based on the RSA and Diffie-Hellman encryption schemes), identification schemes (the Schnorr identification scheme), and many others. All of these systems are, to some hidden, but to all useful, almost indispensable aspects of everyday life. The schemes we listed allow to securely shop online, update software, access private data online, and many other things. One could say, life without public-key cryptography seems impossible.

This apocalyptic scenario, however, is exactly what comes into the picture when we consider the possibility of developing a large-scale *quantum computer*. Quantum computing is based on a different set of rules, instead of treating information as encoded into bits[2] we now apply the laws of quantum physics. Quantum mechanics allow for encoding in states that are not limited to just

[1]A shift cipher is a type of a *substitution* cipher, where the alphabet is shifted by a fixed number of letters, e.g. "ab"$\rightarrow$"cd"

[2]Variables with two possible values 0 or 1.

two values, but form a continuous spectrum of complex numbers.

In his seminal work from 1994 [Sho94], Peter Shor showed an algorithm that can efficiently[3] break the security of the RSA and Diffie-Hellman encryption and key-exchange schemes and signature schemes based on them. Using Shor's algorithm one can use the publicly available key to break the encryption or forge a signature. The implications of this algorithm are indeed huge, when a quantum computer is developed, most of the current security infrastructure of the internet can be broken.

There is light at the end of the tunnel, though, and it shines from *post-quantum cryptography* [BBD09]. Post-quantum cryptography is resilient to attacks that use quantum (and classical) computers. In case of public-key cryptography it is based on mathematics that is resistant against quantum computers[4] (at least as far as we know).

The field of post-quantum cryptography treats also private-key cryptographic schemes that resist quantum attacks. These are often primitives[5] used together with the systems we mentioned before, like encryption or digital signatures.

This thesis is dedicated to studying post-quantum security of some private-key primitives forming the backbone of the post-quantum secure infrastructure. In what follows we dive into more details of cryptographic constructions that are used to build these primitives and security claims in cryptography.

1.1 Cryptographic Constructions

In this thesis we focus on cryptographic hash functions. At a glance, hash functions are functions that map long inputs (often arbitrarily long) to short outputs. Cryptographic hash functions, that are equipped with some security features[6], are one of the central primitives in cryptography. They are used virtually everywhere: As cryptographically secure checksums to verify integrity of software or data packages, as building block in security protocols, including TLS, SSH, IPSEC, as part of any efficient variable-input-length signature scheme, to build full-fledged hash-based signature schemes, in transformations for CCA-secure[7] encryption, and many more.

[3]In time polynomial in the length of the input.

[4]Examples of mathematical concepts that are (probably) suitable to build post-quantum cryptography include: lattices [Pei16], codes [Sen17], multivariate polynomials [DP17], isogenies [Gal+16], and hash functions [BDS09]

[5]Cryptographic primitives are the basic parts of cryptographic systems, such as hash functions or pseudorandom functions.

[6]One of the most desired security features of a cryptographic hash function is collision resistance. A collision is a pair of inputs that map to the same output. For a collision resistant hash function, it is hard for any adversary to find a collision.

[7]CCA stands for Chosen Ciphertext Attack, more on that can be found in [KL14]

Cryptographic practice almost always requires constructing hash functions out of fixed-input-length primitives. The reason for that is the fact that it is much easier to engineer a secure function with limited domain and codomain. *Cryptographic constructions* are a prescription for building a function out of "smaller" building blocks. By reusing the internal function we get a function with domain and codomain more suitable to our needs. We can construct hash functions, but also functions that do not accept arbitrarily long inputs but have other sought-after features.

The biggest challenge in designing cryptographic constructions is making sure that the security features of the internal function are preserved. Security of constructions is derived from the security of the internal function. Mathematically establishing this derivation is often challenging and is the main topic of this thesis.

One of the most notable examples of cryptographic constructions is the Merkle-Damgård construction [Mer90; Dam90]. It is used in the standardized hash function SHA2 [NIS15]. The construction that we focus on is the sponge construction [Ber+07], prominently used in SHA3 [NIS14].

1.2 Security in Cryptography

The modern approach to cryptography calls for mathematical rigor in all claims of security. The three necessary elements of this rigor are: formal definitions of the claimed security, assumptions that we base the security on, and the adversary model, formalizing the adversarial attacks that the primitive can withstand. First we go over some important assumptions. Bear in mind, however, that the adversaries we discuss in detail later are all quantum. This means that some standard assumptions can be immediately dismissed.

Commonly, assumptions are of number-theoretic nature (these are mainly used in public-key cryptography) or stating that a primitive is in fact an ideal version of itself (more common in private-key cryptography). First of all, the most important cryptographic assumptions are computational, meaning they assume a problem is hard for efficient algorithms. Prominent examples of number-theoretic assumptions are that of hardness of factoring and the discrete-logarithm assumption. They are used in security proofs of RSA and Diffie-Hellman systems respectively. These are also the assumptions that are broken by Shor's algorithm that can be run on a quantum computer. This situation is exactly the fact that gave birth to post-quantum cryptography.

The answer of the community is to make mathematical assumptions resilient against quantum computers. Prominent examples are: learning with errors [Reg09], coding theory [BMV78; AFS05], multivariate polynomials [PG97], and others. It is worth noting that in recent years the National Institute of Standards and Technology (NIST) started a Post-Quantum

Cryptography Standardization process [NIS17] that aims at developing standards for cryptography resilient against quantum computers.

While all widely deployed public-key cryptography is threatened by the rise of quantum computers, hash functions are believed to be only mildly affected. The reason for this effect is twofold: On the one hand, generic quantum attacks achieve at most a square-root speed up compared to their pre-quantum counterparts and can be proven asymptotically optimal [Ben+96; BHT98; Zha15a; HRS16]. On the other hand, there do not exist any dedicated quantum attacks on any specific hash function that perform better than the generic quantum attacks (except, of course, for hash functions based on number theory like, e.g., VSH [CLS06]). Nonetheless, the fact that we use constructions to build new primitives is sometimes a source of quantum attacks, so there is still work to be done in the field of proving security. Throughout this thesis we focus on hash functions that rely on assumptions of ideal primitives. Among the assumptions of an ideal primitive is the Random Oracle Model [BR93], where we assume the hash function or the internal function is a uniformly random function—a primitive that is naturally resistant to all but generic attacks. Interestingly, the Quantum Random-Oracle Model [Bon+11] is an extension of this classic assumption to the quantum world.

We have already mentioned attacks on cryptosystems but not formally described the adversary model. The third part of the security discussion focuses on the capabilities of the adversary, trying to break the security of the cryptosystem. The adversary model is the set of adversaries that should not break the security. First of all the adversary model includes only efficient algorithms[8], as we treat only computational assumptions.

The adversaries in the post-quantum setting[9] are also considered to have access to a large-scale quantum computer. First of all this setting implies that number-theoretic assumptions might not hold, factoring or discrete logarithms are easy when having access to a quantum computer. Second of all, the adversary has quantum access to all public primitives, like hash functions. A post-quantum adversary can just implement the hash function of interest on their quantum machine, the code of all standard hash functions is available on-line[10]. An adversary can use a quantum circuit implementing SHA3 and can thereby query the function in superposition. The adversary could evaluate the underlying sponge construction on the uniform superposition over all messages of a certain length, possibly helping her to, e.g., find a collision.

[8]Query algorithms are assumed to make a limited number of queries, polynomial in the security parameter. The security parameter is usually just the length of the secret key or the output of a hash function.

[9]We mean a situation in which the protocols and primitives that are studied are classical, but the attacker can perform quantum computations.

[10]Note that due to the Kerckhoffs's principle (we refer to, e.g., section 1.2 of [KL14]) this cannot be avoided.

An interesting generalization of the post-quantum setting is the fully quantum setting, where the adversary has quantum access to secretly keyed primitives as well. These are usually considered off-limits (in terms of quantum queries) to post-quantum adversaries. There are attacks that exploit quantum mechanics to attack constructions in this setting [Kap+16; SS17].

There are a number of security notions that we cover in this thesis, starting from collision-resistance, a standard security notion important for all cryptographic hash functions. We go through more advanced notions like pseudorandom functions—functions that look like uniformly random functions to computationally bounded adversaries. The last chapters treat one of the strongest security notions suitable for hash functions: indifferentiability. Under the strong assumption that the internal function is fully random, the notion captures the situation that the internal function is publicly available but the constructed hash function is still random to any efficient adversary. This situation is closest to real life, among the idealized security notions in the literature.

1.3 Plan of the Thesis

In Chapter 2 we go over the preliminary notions that will be important in the rest of the thesis. Next we explore the post-quantum security of the sponge construction. In this thesis we discuss a number of security definitions and prove that the sponge construction fulfills them. In general we cover security notions in the order of increasing security guarantees.

In Chapter 3 we discuss the standard security notions of the sponge construction, that is collision-resistance and collapsingness. We state these results for sponges constructed with one-way functions. We also present two collision-finding algorithms for the sponge construction. This chapter mainly covers the results of [Cza+18].

In Chapter 4 we prove quantum indistinguishability of the sponge construction, a result initially published in [CHS19]. Our proof works for both random functions and permutations, i.e., two classes of internal functions important for the sponge construction.

In Chapter 5 we discuss in detail the conjectured impossibility of the notion of quantum indifferentiability, teh chapter covers the results of [CG21].

After some initial doubts about the validity of quantum indifferentiability it became apparent that the notion is in fact achievable. In Chapter 6 we discuss our findings on the compressed-oracle technique—crucial in discussing quantum indifferentiability. There we also develop a framework for quantum game-playing proofs.

In Chapter 7 we prove a number of results on indifferentiability, including of the sponge construction. We also prove quantum indifferentiability of: the composition of compression functions, the rate $1/3$ construction based on three internal functions, Encrypted Davis Meyer, and its dual. Chapter 6 and Section 7.6 are results from [Cza+19].

In Chapter 8 we analyze a more fine-grained security notion than indifferentiability, that of memory-restricted quantum indifferentiability. We show the importance of the multi-stage setting for quantum indifferentiability by analyzing a quantum version of the counterexample based on the external-storage game that is the first step in the discussion of the problem in classical cryptography.

In Chapter 9 we discuss some open problems that the author encountered during his research. We present open questions and partial results on important issues concerning general quantum indistinguishability of distributions and quantum lazy-sampling of random permutations.

We close with a short concluding chapter.

CHAPTER

2 PRELIMINARIES

Chapter contents

In this chapter we introduce basic concepts that we will use throughout the thesis. In Section 2.1 we describe general mathematical notions. In what follows, we focus on notions from the field of cryptography. In Section 2.2 we describe standard security notions and in Section 2.3 idealized security notions. In Section 2.4 we go over some important proof techniques used in the thesis. Finally, in Section 2.5 we describe important cryptographic constructions.

2.1 Basic Notions

In this section we introduce notation and basic mathematical notions used in the thesis.

2.1.1 Notation

Throughout this thesis we follow some general rules of notation, that we hope help in comprehending the presented material. First of all let us list the different fonts we use to name different mathematical objects. We use

ab small letters to denote parameters and variables,

AB capital letters to denote quantum registers,

$\mathcal{AB}$ capital calligraphic letters to denote sets,

$\mathbf{ab}$ small boldface letters to denote functions,

$\mathbf{AB}$ capital boldface letters to denote arrays and some special functions (functionals),

Ab sans serif letters to denote quantum operators,

$\textsc{Ab}$ small caps letters to denote algorithms,

$\mathfrak{AB}$ capital fraktur letters to denote distributions.

There are of course some exceptions to these rules but we hope they do not introduce misunderstandings and we always explain what a given symbol denotes. In the Symbol Index we list all the important symbols used in this thesis.

We write $[N] := \{0, 1, \ldots, N-1\}$ for the set of size N. Whenever we discuss indices starting from 1 we use $\mathcal{N}_N := \{1, 2, \ldots, N\}$ and when N is clear from the context we just write $\mathcal{N}$. By $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{R}$, and $\mathbb{C}$ we denote the set of natural numbers, integers, reals, and complex numbers respectively. For $\mathcal{S} \subseteq \mathcal{N}$, we denote by $\mathcal{S}(i)$ the i-th element of the set $\mathcal{S}$ in ascending order. For some fixed sets $\mathcal{X}_1, ..., \mathcal{X}_N$, we denote by $\vec{x}$ an element of $\mathcal{X}_1 \times \cdots \mathcal{X}_N$ and for $\mathcal{S} \subseteq \mathcal{N}$ we have $\vec{x}_{\mathcal{S}} := (x_{\mathcal{S}(1)}, \dots, x_{\mathcal{S}(|\mathcal{S}|)})$. We denote the set of all t-element permutations by $\mathcal{I}_{\text{per}}(\{1, 2, \dots, t\})$. We write $\mathcal{Y}^{\mathcal{X}}$ to denote $\{\mathbf{f} : \mathcal{X} \to \mathcal{Y}\}$.

We use the following notation for a set of arbitrary finite-length strings:

$$\mathcal{X}^* = \bigcup_{l \geq 0} \mathcal{X}^l, \tag{2.1}$$

where $\mathcal{X}$ is an arbitrary finite set. For $\mathcal{X} = \{0, 1\}$ we usually denote this set by $\mathcal{M}$.

$\oplus$ denotes the group action in the group $(\mathcal{A}, \oplus)$. In the case of $\mathbb{Z}_2^n$ it is the bitwise XOR. By $\{0, 1\}^n$ we denote the set of all bitstrings of length n. $|m|$ denotes the length of a string and for sets $\mathcal{A}$, $|\mathcal{A}|$ is the cardinality of $\mathcal{A}$. $\|$ denotes concatenation of symbols in a set.

For $\mathbf{P} \in \mathcal{A}^*$, such that $\mathbf{P} = P_1 \| P_2 \| \cdots \| P_{|\mathbf{P}|}$, $|\mathbf{P}|$ denotes the number of symbols in $\mathbf{P}$, $\mathbf{P}_i$ is the i-th symbol of $\mathbf{P}$ and $\lfloor \mathbf{Z} \rfloor_\ell$ are the first ℓ symbols of $\mathbf{Z}$.

For some complex number $c = a + b\mathrm{i}$, we define $\mathfrak{Re}(c) = a$ as its real part. We denote the Euclidean norm of a vector $|\psi\rangle \in \mathbb{C}^d$ by $\| |\psi\rangle \|$.

Let $\operatorname{range} \mathbf{f}$ denote the range of the function $\mathbf{f}$, and $\operatorname{im} \mathbf{f}$ its image (i.e., $\operatorname{im} \mathbf{f}$ contains all values that the function $\mathbf{f}$ actually attains, while $\operatorname{range} \mathbf{f}$ refers to the set into which $\mathbf{f}$ maps according to the declaration of $\mathbf{f}$). The sign function is defined from $\mathbb{R} \to \{-1, 0, 1\}$ as follows:

$$\operatorname{sgn}(x) := \begin{cases} -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \\ 1 & \text{if } x > 0 \end{cases} . \tag{2.2}$$

We call a non-negative function from $\mathbb{N} \to \mathbb{R}$ *negligible* if and only if for every positive integer c, there exists an integer N_c such that for all $x > N_c$, $\mathbf{f}(x) < \frac{1}{x^c}$. We call $\mathbf{f} \leq 1$ *overwhelming* if $1 - \mathbf{f}$ is negligible.

For algorithms A (classical or quantum), we use the notation $(a, b, c) \leftarrow \mathrm{A}(d, e, f)$ to denote that A is executed with inputs d, e, f, and the output triple is assigned to the variables a, b, c. We write A^{H} when an algorithm A has access to the function H as an oracle. By square brackets we also denote (classical or quantum) oracle access to some algorithm. We have $\mathrm{A}[\mathsf{H}] \equiv \mathrm{A}^{\mathsf{H}}$. When discussing adversaries that make quantum or classical queries we sometimes write $\mathrm{A}^{|\mathbf{f}\rangle}$ or $\mathrm{A}^{\mathbf{f}}$ to denote quantum or classical query-access to $\mathbf{f}$ respectively.

We write $\mathbb{P}[C : G]$ to denote the probability that the condition C holds after executing the steps in game G[1]. E.g., $\mathbb{P}[b = 1 : b \leftarrow \mathrm{A}()]$ denotes the probability that the bit b returned by $\mathrm{A}()$ is 1.

When referring to quantum registers, i.e., subsystems of the quantum state of the whole system, we use two types of notation. Let us say we have a quantum register Q holding elements of the Hilbert space $\mathcal{H} = \bigotimes_{x \in \mathcal{S}} (\mathcal{H}_x^X \otimes \mathcal{H}_x^Y)$

[1]More on games in cryptography can be found in Section 2.4.2.

(more details on this can be found in Section 2.1.5). Our Hilbert space is a tensor product of $|\mathcal{S}|$ pairs of $\mathcal{H}_x^X \otimes \mathcal{H}_x^Y$, we assume that there is a natural order in $\mathcal{S}$ and the tensor product is applied in this order. When we want to refer to the i-th register from the left, we write Q_i. When we want to access registers holding all X-parts of Q we write Q^X, with a superscript Q_i^X we access just the X-part of Q_i. Sometimes, however, we want to access the register corresponding to a particular $x \in \mathcal{S}$, then we write $Q(x)$ and similarly the superscript marks the part of $\mathcal{H}_x^X \otimes \mathcal{H}_x^Y$ we want to specifically discuss.

For registers F that hold multiple entries we refer to the particular entries by giving them as argument to F, for example:

$$|\mathbf{f}\rangle_F = \bigotimes_{x \in \mathcal{X}} |\mathbf{f}(x)\rangle_{F(x)}. \tag{2.3}$$

In the algorithms in this chapter we denote quantum and classical databases (that is arrays of input-output pairs) corresponding to different internal functions, by D. Inputs in databases are accessed by referring to D^X, outputs by D^Y.

2.1.2 Distributions

A distribution $\mathfrak{D}$ on a set $\mathcal{X}$ is a function $\mathfrak{D} : \mathcal{X} \to [0,1]$ such that $\sum_{x \in \mathcal{X}} \mathfrak{D}(x) = 1$. We denote sampling x from $\mathcal{X}$ according to $\mathfrak{D}$ by $x \leftarrow \mathfrak{D}$. If $\mathfrak{D}$ is a distribution on $\mathcal{Y}$ then $\mathfrak{D}^{\mathcal{X}}$ denotes a distribution on $\mathcal{Y}^{\mathcal{X}}$ where the output for each input is chosen independently according to $\mathfrak{D}$. By $\overset{\$}{\leftarrow} \mathcal{X}$ we denote sampling uniformly at random from the set $\mathcal{X}$.

2.1.3 Model of Computation

An efficient algorithm is a procedure with runtime that is polynomial in the length of its input. When the input is left implicit, we just assume efficient algorithms run in time polynomial in the security parameter k. Quantum efficient algorithms run in *quantum polynomial-time*, this means the number of quantum basic operations[2] is limited to a polynomial number in the input length. In results with asymptotic statements (referring, e.g., to quantum-polynomial-time adversaries), we assume a security parameter that is implicitly provided to all adversaries, and that all parameters and functions may implicitly depend on.

Often the only important parameter of the adversary's interaction with the analyzed cryptographic protocol is the number of queries that she makes. Then we limit the adversary's computational capabilities by just assuming she makes a polynomial number of queries but her computation time is not limited.

[2]Basic operations are just quantum gates from a predefined set of quantum operations, more on this topic can be found in [NC10].

2.1.4 Quantifying Security

In general, throughout this thesis we state security of cryptographic systems by indistinguishability of two worlds. Usually they are the real and the ideal world. The former is the actual protocol we analyze and the latter the system we hope to simulate with our protocol. This is a common approach to security in cryptography. The actual claims are made on the difference of probabilities that an adversary or environment outputs 1 when interacting with one world or the other. The final result is a bound on the difference of probabilities.

By stating results in this way we can prove *average-case* security. In the average-case approach we average over random variables, such as the key. This approach can be compared to worst-case security where we state the number of queries (made to the analyzed primitive) necessary to achieve a constant difference in probabilities. In general the average-case approach is much more natural in cryptography than the worst-case approach.

2.1.5 Quantum Computing and the Quantum Threat Model

With the increased interest and efforts in the field of quantum computing, it is evident that cryptographic threat models should allow attackers to carry out quantum computations which defines the research area of post-quantum cryptography [Ber09]. An attacker of the future can run Shor's algorithm [Sho94] on a quantum machine to break the security of schemes based on RSA or discrete logarithms. However, conventional security proofs in the random-oracle model might also break down in the light of quantum attackers who are allowed to ask queries in superposition [Bon+11] (the random oracle models the hash function, which in turn can be implemented on the adversary's quantum machine). Furthermore, tasks like preimage or collision finding can be sped up using quantum search algorithms [Gro96; Amb07].

We assume the reader is familiar with the usual notation in quantum computation, but we give a very short introduction here. A quantum system A is a complex finite-dimensional Hilbert space $\mathcal{H}$, together with an inner product $\langle\cdot|\cdot\rangle$.The *pure* state of a quantum system is given by a vector $|\Psi\rangle$ of unit norm ($\langle\Psi|\Psi\rangle = 1$). A joint system of $\mathcal{H}_1$ and $\mathcal{H}_2$ is denoted by $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2$, with elements $|\Psi\rangle = |\Psi_1\rangle|\Psi_2\rangle$ for $|\Psi_1\rangle \in \mathcal{H}_1, |\Psi_2\rangle \in \mathcal{H}_2$. More generally, quantum states are represented by positive semi-definite operators with unit trace, i.e., $\rho \succeq 0, \mathrm{Tr}(\rho) = 1$, called *density operators*. We denote the set of all density operators by $\mathcal{D}(\mathcal{H})$. In the language of density operators, pure states are rank-one density operators. *Mixed* states are linear combinations of pure states. Operations on quantum states are represented by unitary operations U or more generally by Completely Positive Trace Preserving maps. In this thesis we focus on the former. A unitary transformation U over a d-dimensional Hilbert space $\mathcal{H}$ is a $d \times d$ matrix U such that $\mathsf{U}\mathsf{U}^\dagger = \mathbb{1}_d$, where $\mathsf{U}^\dagger$ represents the conjugate transpose. We assume familiarity with these concepts throughout the technical parts of this thesis. For an introduction, we refer the reader to a textbook on quantum computation or quantum information such as [NC10].

The quantum threat model we consider allows the adversary to query oracles in superposition. Oracles are modeled as unitary operators $\mathsf{U}_{\mathbf{h}}$ acting on computational basis states as follows

$$\mathsf{U}_{\mathbf{h}}|x, y\rangle = |x, y \oplus \mathbf{h}(x)\rangle. \tag{2.4}$$

The adversary is considered to have access to a fault-tolerant (perfect) quantum computer. More details on quantum accessible oracles can be found in Section 6.2. An oracle algorithm $A^{\mathbf{h}}$ can evaluate the unitary $\mathsf{U}_{\mathbf{h}}$ in a single step. Any quantum algorithm making q queries can then be written as a final transformation $\mathsf{U}_q\mathsf{U}_{\mathbf{h}} \cdots \mathsf{U}_1\mathsf{U}_{\mathbf{h}}\mathsf{U}_0$ for unitaries U_i applied between oracle queries $\mathsf{U}_{\mathbf{h}}$.

Let us define the *Quantum Fourier Transform* (QFT), a unitary change of basis that we will make heavy use of. For $N \in \mathbb{N}_{>0}$ and $x, \xi \in [N] = \mathbb{Z}_N$ the transform is defined as

$$\mathsf{QFT}_N|x\rangle := \frac{1}{\sqrt{N}} \sum_{\xi \in [N]} \omega_N^{\xi \cdot x}|\xi\rangle, \tag{2.5}$$

where $\omega_N := e^{\frac{2\pi \mathrm{i}}{N}}$ is the N-th root of unity. An important identity for some calculations is

$$\sum_{\xi \in [N]} \omega_N^{x \cdot \xi} \cdot \bar{\omega}_N^{x' \cdot \xi} = N\delta_{x,x'}, \tag{2.6}$$

where $\bar{\omega}_N = e^{-\frac{2\pi i}{N}}$ is the complex conjugate of ω_N and $\delta_{x,x'}$ is the Kronecker delta function.

If we talk about n qubits, the identity on their Hilbert space is denoted by $\mathbb{1}_n$, we also use this notation to denote the dimension of the identity operator, the actual meaning will be clear from the context. We write U^A to denote that we act with U on register A.

2.1.5.1 Quantum Measurements

A projective measurement is specified by a family of projectors $\{\mathsf{P}_i\}$ that are mutually orthogonal and sum up to $\mathbb{1}_d$, one projector P_i for each possible measurement outcome i. For any quantum state $|\Psi\rangle$, let $\langle\Psi|$ denote the adjoint of $|\Psi\rangle$. In particular, $|\Psi\rangle\langle\Psi|$ denotes the orthogonal projector onto $|\Psi\rangle$.

To describe general measurements we are going to introduce the notion of Positive Operator Valued Measure (POVM). The only requirement of POVMs is that they consist of positive operators and sum up to the identity operator so that measurement probabilities sum to one. More formally, a POVM with set of outcomes $\mathcal{X}$ is described by a set of operators $\mathcal{M} = \{\mathsf{Q}^x\}_{x \in \mathcal{X}}$, where $\forall x \in \mathcal{X}$: $\mathsf{Q}^x \succeq 0$, $\sum_x \mathsf{Q}^x = \mathbb{1}$.

We denote the probability of getting the outcome x when measuring ρ with the measurement $\mathcal{M}$ by $\mathbb{P}[x \leftarrow \mathcal{M}(\rho)] := \mathrm{Tr}(\mathsf{Q}^x\rho)$. To describe the post measurement state, we can write down operators of $\mathcal{M}$ as products of linear operators on $\mathcal{H}$ (denoted by $\mathcal{L}(\mathcal{H})$), $\mathsf{Q}^x = \mathsf{A}^{x\dagger}\mathsf{A}^x$ (which is always possible since $\mathsf{Q}^x \succeq 0$), where $\mathsf{A}^x \in \mathcal{L}(\mathcal{H})$. The decomposition of Q^x into the square of A^x is not unique, note that we can produce the same Q^x by multiplying A^x with a unitary operator. This ambiguity, however, does not change the probability of getting the outcome x, moreover we usually assume A^x to be positive. The post-measurement state when the outcome of $\mathcal{M}$ on ρ is x is given by

$$\rho_x := \frac{\mathsf{A}^x\rho\mathsf{A}^{x\dagger}}{\mathrm{Tr}(\mathsf{Q}^x\rho)}. \tag{2.7}$$

The operator A^x is called the *square root*[3] of Q^x.

For a sequence of measurements $(\mathcal{M}_i)_i$, we can define the conditional probability of measuring x_{i_2} with $\mathcal{M}_j$ after measuring x_{i_1} with $\mathcal{M}_i$ in the following way

$$\mathbb{P}[x_{i_2} \leftarrow \mathcal{M}_j(\rho_{x_{i_1}})] := \mathrm{Tr}\left(\mathsf{Q}_j^{x_{i_2}} \frac{\mathsf{A}_i^{x_{i_1}}\rho\mathsf{A}_i^{\dagger x_{i_1}}}{\mathrm{Tr}\mathsf{Q}_i^{x_{i_1}}\rho}\right). \tag{2.8}$$

It is important to note that with conditional probabilities defined as above, in general the order of measurements matters for the outcome, this is the main subject of Chapter 5.

2.1.6 Entropy

The notion of *entropy* is one of the most basic notions in physics and information theory. In this thesis, in Chapter 8, we use it to quantify the number of bits that can be held by a quantum memory register. In this section we provide a limited number of definitions that will be useful in that part of this work.

When an adversary has access to a quantum state ρ_x^B that depends on a classical random variable X the situation is described by a *classical-quantum* (c-q) state:

$$\rho = \rho^{XB} := \sum_x \mathbb{P}_X[x \leftarrow X]|x\rangle_X\langle x| \otimes \rho_x^B, \tag{2.9}$$

where $\{|x\rangle\}_x$ is a family of mutually orthogonal vectors representing the classical values of X. If ρ_x^B is a trivial state in $\mathbb{C}$ we call the state *classical*, such a state is just a description of a classical random variable.

The amount of information held by a quantum state is described by quantum entropy. Min and max entropy of a quantum state is defined as

$$\mathbf{H}_{\min}(\rho) := -\log(\lambda_{\max}(\rho)), \tag{2.10}$$

[3]When we talk about general quantum maps, (not-necessarily positive) square-root operators of POVM elements are often referred to as Kraus operators.

$$\mathbf{H}_{\max}(\rho) := \log(\text{rank}(\rho)), \tag{2.11}$$

where $\lambda_{\max}(\rho)$ is the maximal eigenvalue of ρ. Max entropy is the logarithm of the rank of ρ, note that rank$(\rho) \leq \dim(\rho)$, where $\dim$ is the dimension of the space ρ acts on. Whenever ρ is a classical state then we write $\mathbf{H}_{\min}(X)$. A good reference for this and the following definitions can be found in [Ren08; Sch07].

Definition 2.1 (Definitions 3.1.1 and 3.1.2 in [Ren08]). *Let $\rho^{AB} \in \mathcal{D}(\mathcal{H}_A \otimes \mathcal{H}_B)$ and $\sigma^B \in \mathcal{D}(\mathcal{H}_B)$. The min-entropy of ρ^{AB} relative to σ^B is*

$$\mathbf{H}_{\min}(\rho^{AB} \mid \sigma^B) := -\log \lambda, \tag{2.12}$$

where λ is the minimum real number such that $\lambda \cdot \mathbb{1}^A \otimes \sigma^B - \rho^{AB}$ is non-negative.

The min-entropy of ρ^{AB}given $\mathcal{H}^B$ is

$$\mathbf{H}_{\min}(\rho^{AB} \mid B) := \sup_{\sigma^B} \mathbf{H}_{\min}(\rho^{AB} \mid \sigma^B), \tag{2.13}$$

where the supremum ranges over all $\sigma^B \in \mathcal{D}(\mathcal{H}_B)$.

For such a c-q state the conditional min entropy is written as just $\mathbf{H}_{\min}(X \mid B)_\rho$

Now we need a couple of lemmas. We present versions of lemmas simplified to our needs. Below we skip one quantum register and omit the smooth entropies.

Lemma 2.2 (Lemma 3.2.9 in [Ren08]). *Let $\rho^{XB} \in \mathcal{D}(\mathcal{H}_X \otimes \mathcal{H}_B)$ and $\sigma^B \in \mathcal{D}(\mathcal{H}_B)$ be the fully mixed state on the image of ρ^B. Then*

$$\mathbf{H}_{\min}(\rho^{XB}) - \mathbf{H}_{\max}(\rho^B) \leq \mathbf{H}_{\min}(\rho^{XB} \mid \sigma^B). \tag{2.14}$$

Additionally we need a lemma that dropping a quantum register cannot increase the min-entropy.

Lemma 2.3 (Lemma 2.20 in [Sch07]). *Let $\rho^{XB} \in \mathcal{D}(\mathcal{H}_X \otimes \mathcal{H}_B)$ be a c-q state. Then*

$$\mathbf{H}_{\min}(\rho^{XB}) \geq \mathbf{H}_{\min}(X). \tag{2.15}$$

Lemma 2.4. *If $\mathcal{H}_B = (\mathbb{C}^2)^{\otimes n}$ and $\rho^{XB} \in \mathcal{D}(\mathcal{H}_X \otimes \mathcal{H}_B)$ is a c-q state, then*

$$\mathbf{H}_{\min}(X \mid B) \geq \mathbf{H}_{\min}(X) - n. \tag{2.16}$$

Proof. First note that from Definition 2.1 we have $\mathbf{H}_{\min}(\rho^{XB} \mid \sigma^B) \leq \mathbf{H}_{\min}(X \mid B)$ for any state σ^B. Then from Lemma 2.2 we have

$$\mathbf{H}_{\min}(\rho^{XB}) - \mathbf{H}_{\max}(\rho^B) \leq \mathbf{H}_{\min}(X \mid B). \tag{2.17}$$

We can further bound $\mathbf{H}_{\min}(X) \leq \mathbf{H}_{\min}(\rho^{XB})$ as in Lemma 2.3, we bound $\mathbf{H}_{\max}(\rho^B) \leq n$ using Equation 2.11 and the comment below it. □

2.2 Standard Security Notions

By standard security we mean notions that focus on the hardness of specific tasks, not assuming specific probability distributions for the primitives. These tasks are possible adversarial attacks, often it is sufficient to exclude a specific attack to achieve security of a cryptographic system. Notions of pseudorandomness are also included in this class as they avoid the assumption that a primitive is truly random—instead it just "looks" random.

An interesting framework that captures standard notions of security (and beyond) is the framework of Universally Composable (UC) security [Can01]. The UC framework is used to define security that holds even when a protocol is used in larger systems. For example considering many instances of the same cryptosystem operating in the same moment can make new attacks possible. In the UC framework we define the ideal functionality and show that our protocol is indistinguishable from the ideal protocol for any adversaries and environments from a given set.

In the rest of this section, we analyze the notions of pseudorandom functions, preimage-resistance, collision-resistance, and collapsingness.

2.2.1 Pseudorandom Functions and Permutations

A common primitive that we want to use in cryptography is a random function, i.e. a function that outputs a random output on every input. However, it is highly inefficient to sample a random function: We would basically need to store a table of all input-output pairs. Thankfully, we can make use of the fact that we often consider computationally bounded adversaries. We decrease the randomness specifying the function, instead of sampling all outputs at random we just sample a random *key*. Pseudorandom functions (PRF) are families of keyed functions, where the key is $k \in \mathcal{K}_n$. The feature that justifies using a PRF instead of a fully random function—called pseudorandomness—is that no efficient adversary can distinguish between a PRF and a random function without knowing the secret key. The security parameter is $n := \lfloor \log |\mathcal{K}_n| \rfloor$.

Following [Zha12], let us define *quantum-secure* pseudorandom functions and pseudorandom permutations (PRPs). They are the pseudorandom primitives that are secure even when the adversary makes quantum queries.

Definition 2.5 (Quantum-secure PRF/PRP). *Say $\{\mathbf{f} : \mathcal{K}_n \times \mathcal{S} \to \mathcal{S}\}_n$ is a family of functions (permutations) indexed by $n \in \mathbb{N}$ with $\lfloor \log |\mathcal{K}_n| \rfloor = n$. This family is a quantum-secure pseudorandom function (permutation) if for every quantum algorithm running in polynomial time (hence making at most a polynomial number of quantum queries to its oracles), there is a negligible function ϵ^{PR} such that*

$$\left| \underset{k \xleftarrow{\$} \mathcal{K}_n}{\mathbb{P}} \left[\mathrm{A}^{|\mathbf{f}_k\rangle}(1^n) = 1 \right] - \underset{\mathbf{g} \xleftarrow{\$} \mathcal{S}^{\mathcal{S}}}{\mathbb{P}} \left[\mathrm{A}^{|\mathbf{g}\rangle}(1^n) = 1 \right] \right| \leq \epsilon^{\mathrm{PR}}(n), \tag{2.18}$$

where $\mathbf{g}$ *is sampled uniformly from the set of functions (permutations) from* $\mathcal{S}$ *to* $\mathcal{S}$*. Below, we refer to* ϵ^{PR} *as advantage.*

In [Zha12] Zhandry showed how to construct a quantum-secure PRF and proved that indeed it retains its security when the adversary is quantum. Classical PRFs are analyzed in much depth in [KL14].

2.2.2 Preimage and Collision Resistance

A preimage of a particular output y of a function $\mathbf{h}$ is an input x that yields $\mathbf{h}(x) = y$. Collision is a pair of distinct inputs to $\mathbf{h}$ that give the same output. A common feature of secure hash functions is the hardness of finding any collisions or preimages of given values. Below we provide formal definitions of security established by the resistance of a hash function against attacks finding these sets.

Throughout this thesis we analyze mainly randomness of functions that comes from external random functions. Having this in mind we present definitions different from the classic ones. Classic definitions of preimage-resistance and collision-resistance [KL14], for hash functions that are not based on oracle access to a random primitive, state that it is hard to find a preimage or a collision for a uniformly random key. In the following we consider families of distributions $\mathfrak{D}_n$ indexed by $n \in \mathbb{N}$.

There are several variants of preimage-resistance, we focus on preimages of a fixed value.

Definition 2.6 (Zero-preimage-resistance)**.** *For a random* H *distributed according to* $\mathfrak{D}_n$*, we call a function* $\mathbf{h}^{\mathsf{H}} : \mathcal{X} \to \mathcal{Y}$ zero-preimage-resistant *if and only if for any quantum-polynomial-time adversary* A^{H} *making quantum queries, there is a negligible function* ϵ *such that*

$$\underset{\mathsf{H}\leftarrow\mathfrak{D}}{\mathbb{P}}\left[\mathbf{h}^{\mathsf{H}}(x) = 0 : x \leftarrow \mathrm{A}^{\mathsf{H},\mathbf{h}}(1^n)\right] < \epsilon(n). \tag{2.19}$$

One of the most important properties of a hash function $\mathbf{h}$ is collision-resistance. That is, it is infeasible to find $x \neq x'$ with $\mathbf{h}(x) = \mathbf{h}(x')$. Intuitively, collision-resistance guarantees some kind of computational injectivity—given $\mathbf{h}(x)$, the value x is effectively determined. Of course, information-theoretically, x is not determined, but in many situations, we can treat the preimage x as unique, because we will never see another value with the same hash. A formal definition is

Definition 2.7 (Collision resistance)**.** *For a random* H *distributed according to* $\mathfrak{D}_n$*, we call a function* $\mathbf{h}^{\mathsf{H}} : \mathcal{X} \to \mathcal{Y}$ collision-resistant *if and only if for any quantum-polynomial-time adversary* A^{H} *making quantum queries, there is a negligible function* ϵ *such that*

$$\underset{\mathsf{H}\leftarrow\mathfrak{D}}{\mathbb{P}}\left[\mathbf{h}^{\mathsf{H}}(x) = \mathbf{h}^{\mathsf{H}}(x') \wedge x \neq x' : (x, x') \leftarrow \mathrm{A}^{\mathsf{H},\mathbf{h}}(1^n)\right] < \epsilon(n). \tag{2.20}$$

We say that $\mathbf{h}$ *has a collision in* $\mathcal{M}$ if the image of $\mathbf{h}$ under $\mathcal{M}$ is smaller than $\mathcal{M}$

$$|\mathbf{h}(\mathcal{M})| < |\mathcal{M}|\,. \tag{2.21}$$

For uniformly random functions $\mathbf{h} : \mathcal{X} \to \mathcal{Y}$ we have explicit bounds on the probability of finding preimages and collisions. For classical adversaries, $\mathbf{h}$ is zero-preimage-resistant with $\epsilon = \frac{q}{|\mathcal{Y}|}$ and collision-resistant with $\epsilon = \frac{(q-1)q}{2|\mathcal{Y}|}$ [KL14], where q is the maximal number of queries the adversary can make.

Quantum access to $\mathbf{h}$ helps the adversary in finding a preimage or a collision. A random function is preimage-resistant against quantum adversaries with $\epsilon = 8\frac{(2q+1)^2}{|\mathcal{Y}|}$. This result can be achieved with a reduction to finding the 1-preimage in a Boolean function that outputs 1 with probability $\gamma = \frac{1}{|\mathcal{Y}|}$. Every query to $\mathbf{h}$ is simulated as 0 if the Boolean function outputs 1 and the output of a random function $\mathbf{h}' : \mathcal{X} \to \mathcal{Y} \setminus \{0\}$ otherwise. A query to $\mathbf{h}$ is calculated using two queries to the Boolean function (one to compute the output and one to uncompute it). The bound comes from Theorem 1 in [HRS16], which shows that a $2q$-query quantum adversary can find a 1-preimage in $\mathbf{h}$ with probability at most $8\gamma(2q+1)^2$.

The success probability of a collision-finding algorithm for a uniformly random $\mathbf{h} : \mathcal{S} \to \mathcal{Y}$ is at most $\frac{\pi^2}{3}\frac{(q+2)^3}{|\mathcal{Y}|}$, as proved by Zhandry in Theorem 7 in [Zha15a].

2.2.3 Collapsing Hash Functions

Collapsingness is a security notion defined in [Unr16b]; It is a purely quantum notion strengthening collision-resistance. It was developed to capture the required feature of hash functions used in cryptographic commitment protocols.

Collision-resistant hashes can be used to extend the message space of signature schemes (by signing the hash of the message), or to create commitment schemes (e.g., sending $\mathbf{h}(x\|r)$ for random r commits us to x; we cannot change our mind about x because we cannot find another preimage).

In the post-quantum setting, it was shown by Unruh [Unr16b] that collision-resistance is weaker than expected: For example, the commitment scheme sketched in the previous paragraph is not binding: it is possible for an attacker to send a hash h, then to be given a value x, and then to send a random value r such that $\mathbf{h}(x\|r) = h$, thus opening the commitment to any desired value—even if $\mathbf{h}$ is collision-resistant against quantum adversaries[4].

[4]More precisely, [Unr16b] shows that relative to certain oracles, a collision-resistant hash function exists that allows such attacks. In particular, this means that there cannot be a relativizing proof that the commitment scheme is binding assuming a collision-resistant hash function. In a recent paper Zhandry [Zha19b] improved this result by presenting a standard model instantiation of a collision-resistant function that also allows attacks from [Unr16b].

This example contradicts the intuitive requirement that $\mathbf{h}(x)$ determines x.

Fortunately, Unruh [Unr16b] also presented a strengthened security definition for post-quantum secure hash functions: collapsing hash functions. Roughly speaking, a hash function is collapsing if, given a superposition of values m, measuring $\mathbf{h}(m)$ has the same effect as measuring m (at least from the point of view of a computationally limited observer). Collapsing hash functions serve as a drop-in replacement for collision-resistant ones in the post-quantum setting: Unruh showed that several natural classical commitment schemes (namely the scheme sketched above, and the statistically-hiding schemes from [HM96]) become post-quantum secure when using a collapsing hash function instead of a collision-resistant one. The collapsing property also directly implies collision-resistance.

In light of these results, it is desirable to find hash functions that are collapsing. Unruh [Unr16b] showed that a random oracle is collapsing. That is, a hash function $\mathbf{h}$ is collapsing when it is a random oracle. However, this example has little relevance for real-world hash functions: A practical hash function is typically constructed by iteratively applying some elementary building block (e.g., a "compression function") in order to hash large messages. So even if we are willing to model the elementary building block as a random oracle, the overall hash-function construction should arguably not be modeled as a random oracle[5].

For hash functions based on the Merkle-Damgård construction (such as SHA2 [Nat15]), Unruh [Unr16a] showed: If the compression function is collapsing, so is the hash function resulting from the Merkle-Damgård construction. In particular, if we model the compression function as a random oracle (as is commonly done in the analysis of practical hash functions), we have that hash functions based on the Merkle-Damgård construction are collapsing and thus suitable for use in the post-quantum setting.

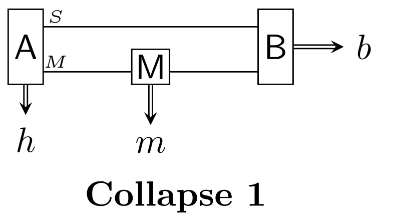
As mentioned above, intuitively, we wish that $\mathbf{h}(m)$ uniquely identifies m in some sense. In the classical setting, this wish naturally leads to the requirement that it is hard to find $m \neq m'$ with $\mathbf{h}(m) = \mathbf{h}(m')$. Then we can treat $\mathbf{h}(m)$ as if it had only a single preimage (even though, of course, a compressing $\mathbf{h}$ will have many preimages, we just cannot find them). In the quantum setting, there is another interpretation of the requirement that $\mathbf{h}(m)$ identifies m. Namely, if we are given a register M that contains a superposition of many values m, then measuring $\mathbf{h}(m)$ on that register should—intuitively—fully determine m. That is, the effect on the register M should be the same, no matter whether we measure just the hash $\mathbf{h}(m)$ or the whole message m. One can see that for any compressing function $\mathbf{h}$, it is impossible that measuring $\mathbf{h}(m)$

[5]For example, hash functions using the Merkle-Damgård construction are not well modeled as a random oracle. If we use MAC$(k, m) := \mathbf{h}(k\|m)$ as a message authentication code (MAC) with key k, we have that this MAC is secure (unforgeable) when $\mathbf{h}$ is a random oracle, but easily broken when $\mathbf{h}$ is a hash function built using the Merkle-Damgård construction.

and m has information-theoretically the same effect on the state [6]. However, what we can hope for is that for a computationally limited adversary, the two situations are indistinguishable. In other words, we require that no quantum-polynomial-time adversary can distinguish whether we measure $\mathbf{h}(m)$ or m. This property is useful in proofs, because we can replace $\mathbf{h}(m)$-measurements by m-measurements and vice versa.

We can slightly simplify this condition if we require that the register M already contains a superposition of values m that all have the same hash $\mathbf{h}(m)$. In this case, measuring $\mathbf{h}(m)$ has no effect on the state, so we can state the requirement as: If M contains a superposition of messages m with the same $\mathbf{h}(m) = h$, then no quantum-polynomial-time adversary can distinguish whether we measure M in the computational basis, or whether we do not measure it at all.

Or slightly more formally: We let the adversary A produce a register M and a hash value h (subject to the promise that measuring M would lead to an m with $\mathbf{h}(m) = h$). The adversary additionally keeps an internal state in register S. Then we either measure M in the computational basis (**Collapse 1**, depicted in Figure 2.1), or we do not perform any such measurement (**Collapse 2**, depicted in Figure 2.1). Finally, we give registers S (the internal state) and M (the potentially measured message register) to the adversary's second part B. We call $\mathbf{h}$ *collapsing* if no quantum-polynomial-time (A, B) can distinguish **Collapse 1** and **Collapse 2**.



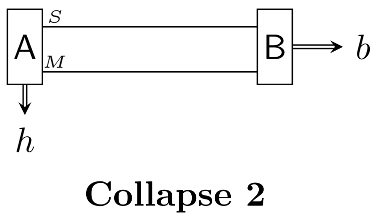


Figure 2.1: Games from the definition of collapsing hash functions. M represents a measurement in the computational basis. (A, B) is assumed to satisfy the property that M always returns m with $\mathbf{h}(m) = h$. A function is collapsing if the probability of $b = 1$ is negligibly close in both games.

This idea is formalized by the following definition. For quantum algorithms A, B with quantum access to $\mathbf{h}$, consider the following games:

$$\textbf{Collapse 1}: \quad (S, M, h) \leftarrow \mathsf{A}^{\mathbf{h}}(),\ m \leftarrow \mathsf{M}(M),\ b \leftarrow \mathsf{B}^{\mathbf{h}}(S, M), \tag{2.22}$$

$$\textbf{Collapse 2}: \quad (S, M, h) \leftarrow \mathsf{A}^{\mathbf{h}}(), \qquad b \leftarrow \mathsf{B}^{\mathbf{h}}(S, M). \tag{2.23}$$

[6]E.g., M could contain $\sum_m 2^{-|m|/2}|m\rangle$. Then measuring $\mathbf{h}(m)$ will lead to the state $\sum_{m \text{ s.t. } \mathbf{h}(m)=h} \frac{1}{\sqrt{|\mathbf{h}^{-1}(h)|}}|m\rangle$ which is almost orthogonal for large $|\mathbf{h}^{-1}(h)|$ to the state $|m\rangle$ we get when measuring m.

Here S, M are quantum registers. $\mathsf{M}(M)$ is a measurement of M in the computational basis. The intuitive meaning of the above games is that part A of the adversary prepares a quantum register M that holds a superposition of inputs to $\mathbf{h}$ that all map to h. Then she sends M along with the side information S to B. The task of the second part of the adversary is to decide whether measurement M of the register M occurred or not. We call an adversary (A, B) *valid* if and only if $\mathbb{P}[\mathbf{h}(m) = h] = 1$ when we run $(S, M, h) \leftarrow \mathrm{A}^{\mathbf{h}}()$ in **Collapse 1** from Equation (2.22) and measure M in the computational basis as m.

Definition 2.8 (Collapsing [Unr16b])**.** *A function* $\mathbf{h}$ *is* collapsing *with advantage* ϵ *if for any valid quantum-polynomial-time adversary* (A, B)

$$|\mathbb{P}[b = 1 : \textbf{Collapse 1}] - \mathbb{P}[b = 1 : \textbf{Collapse 2}]| < \epsilon. \tag{2.24}$$

We say that $\mathbf{h}$ *is collapsing if the* collapsing-advantage ϵ *is negligible.*

A more in-depth analysis of this security notion can be found in [Unr16b; Unr16a; Cza+18; Feh18].

One can achieve tighter results of security by directly analyzing the security of t parallel evaluations of the hash function (see [Unr16a]). For a set $\mathcal{M}$, we call an adversary (A, B) *t-valid on* $\mathcal{M}$ *for* $\mathbf{h}^{\mathsf{H}}$ if and only if $\Pr[\forall i\, \mathbf{h}^{\mathsf{H}}(m_i) = h_i \wedge m_i \in \mathcal{M}] = 1$ when we run $(S, M_1, \ldots, M_t, h_1, \ldots, h_t) \leftarrow \mathrm{A}^{\mathsf{H}}()$ and measure all M_i in the computational basis as m_i. If we omit "on $\mathcal{M}$", we assume $\mathcal{M}$ to be the domain of $\mathbf{h}^{\mathsf{H}}$. This definition is from [Unr16a], with the only difference that now adversaries and hash functions may depend on an oracle H, instead of depending on a public parameter. We do not focus on this generalization but in [Cza+18] we treat in more details collapsingness for parallel queries to $\mathbf{h}$.

2.2.3.1 Miscellaneous Facts

The following properties of collapsing hash functions will be useful in this thesis. We present a number of results from [Unr16b], note however that they were originally proven in a setting without oracle H, but all proofs from [Unr16b] relativize. All results here hold both if runtime is measured in computation steps, and when time is measured in the number oracle queries.

If $\mathbf{h}^{\mathsf{H}}$ is injective, then $\mathbf{h}^{\mathsf{H}}$ is collapsing:

Lemma 2.9 (Lemma 24 in [Unr16b])**.** *If* $\mathbf{h}^{\mathsf{H}}$ *is injective, and* $(\mathrm{A}^{\mathsf{H}}, \mathrm{B}^{\mathsf{H}})$ *is a valid adversary with collapsing-advantage* ε *against* $\mathbf{h}^{\mathsf{H}}$*, then* $\epsilon = 0$.

If $\mathrm{g}^{\mathsf{H}} \circ \mathbf{h}^{\mathsf{H}}$ is collapsing, and g^{H} is quantum-polynomial-time computable, then $\mathbf{h}^{\mathsf{H}}$ is collapsing:

Lemma 2.10 (Lemma 12 in [Cza+18])**.** *Fix oracle functions* g^{H} *and* $\mathbf{h}^{\mathsf{H}}$*. Let* (A, B) *be a valid adversary against some function* $\mathbf{h}^{\mathsf{H}}$ *with runtime* τ *and collapsing-advantage*

ε against $\mathbf{h}^{\mathsf{H}}$. Then there is an adversary $(\mathrm{A}', \mathrm{B}')$ that is valid for $\mathbf{g}^{\mathsf{H}} \circ \mathbf{h}^{\mathsf{H}}$, has runtime $\tau + \tau_{\mathbf{g}}$, and collapsing-advantage ε against $\mathbf{g}^{\mathsf{H}} \circ \mathbf{h}^{\mathsf{H}}$. Here $\tau_{\mathbf{g}}$ is the time required for computing $\mathbf{g}^{\mathsf{H}}$.

If $\mathbf{g}^{\mathsf{H}}$ and $\mathbf{h}^{\mathsf{H}}$ are collapsing, and $\mathbf{h}^{\mathsf{H}}$ is quantum-polynomial-time computable, then $\mathbf{g}^{\mathsf{H}} \circ \mathbf{h}^{\mathsf{H}}$ is collapsing:

Lemma 2.11 (Lemma 37 in [Unr16a]). *Fix oracle functions $\mathbf{g}^{\mathsf{H}}$ and $\mathbf{h}^{\mathsf{H}}$. If there is a τ-time adversary (A, B), valid for $\mathbf{g}^{\mathsf{H}} \circ \mathbf{h}^{\mathsf{H}}$, with collapsing-advantage ε against $\mathbf{g}^{\mathsf{H}} \circ \mathbf{h}^{\mathsf{H}}$, then there are:*

- *a $(\tau + O(\tau_{\mathbf{h}}))$-time adversary $(\mathrm{A}', \mathrm{B}')$, valid for $\mathbf{g}^{\mathsf{H}}$ on $\operatorname{im} \mathbf{h}^{\mathsf{H}}$, with some collapsing-advantage ε' against $\mathbf{g}^{\mathsf{H}}$,*
- *a $(\tau + O(\tau_{\mathbf{h}}))$-time adversary $(\mathrm{A}'', \mathrm{B}'')$, valid for $\mathbf{h}^{\mathsf{H}}$, with some collapsing-advantage ε'' against $\mathbf{h}^{\mathsf{H}}$.*

such that $\varepsilon \leq \varepsilon' + \varepsilon''$. Here $\tau_{\mathbf{h}}$ is an upper bound on the time for evaluating $\mathbf{h}^{\mathsf{H}}$ (on the messages that A outputs on the registers M_j).

In [Unr16b], this lemma had an additional $O(\ell_{mid})$ in the runtime of $(\mathrm{A}'', \mathrm{B}'')$ where ℓ_{mid} denotes the length of the output of $\mathbf{h}^{\mathsf{H}}$. Since this is always dominated by $O(\tau_{\mathbf{h}})$, we omit this term here.

It was shown in [Unr16b] that if H is a random oracle then is it collapsing:

Lemma 2.12 (Lemma 37 in [Unr16b]). *Let $\mathsf{H} : \mathcal{X} \to \mathcal{Y}$ be a random oracle, then any valid adversary $(\mathrm{A}^{\mathsf{H}}, \mathrm{B}^{\mathsf{H}})$ making q quantum queries to H has collapsing-advantage $\varepsilon \in O\left(\sqrt{\frac{q^3}{|\mathcal{Y}|}}\right)$.*

2.3 Idealized Security Notions

When discussing idealized security we assume some primitive to be ideal, which most commonly means fully random. The reason for introducing such a model is to simplify the cryptographic primitives enough so that it is possible to rigorously prove security without going into the details of actual implementations.

Assuming ideal functions in cryptography brings our focus to generic attacks. Generic attacks do not exploit implementation flaws, instead just focus on the anticipated functionality and try to find flaws in it. In the case of cryptographic constructions, we often assume the internal function to be ideal. This scenario allows us to analyze the construction itself.

A framework that is interesting in the discussion of idealized security is the framework of Abstract Cryptography (AC). Developed by Ueli Maurer and Renato Renner [Mau10; Mau11; MR11], it forms a kind of a dual framework to the UC framework. Whereas the latter captures security of interacting systems by carefully defining the notion of Interacting Turing Machines that exchange tapes to communicate, the former is oblivious to the computation model and focuses on the algebraic relations of the involved protocols. One can say UC is a bottom-up and AC a top-to-bottom approach. We mention AC because indifferentiability is a simple notion well placed in the more general AC framework.

In the following we introduce the random-oracle model and its generalizations. Next, in sections 2.3.2 and 2.3.3 we discuss two security notions that will be the among the main subjects of this thesis, indistinguishability and indifferentiability respectively. All these notions are best suited to formalize the security of hash functions.

2.3.1 Random-Oracle Model

The best known idealized security notion is the Random-Oracle Model (ROM). In the ROM one assumes that publicly accessible hash functions are random [BR93]. This is a very useful assumption as it simplifies proofs, and also cryptographic constructions designed with the ROM in mind are more efficient. It might be possible to prove that the randomness assumption is false, but good cryptographic hash functions indeed behave as if they were random.

A generalization of the ROM to the quantum world was introduced in [Bon+11] under the name of the quantum-accessible random oracle model, we, however, follow the simplified name of the Quantum Random-Oracle Model (QROM). The assumption of this model is that an adversary can access the random oracle in superposition. An interesting separation of the classical and quantum models has been recently shown in [YZ20]. Another treatment of the classical and quantum models can be found in [Gri+20].

We formally define a random oracle by the following distribution:

Definition 2.13 (Random Oracle). *A random oracle is sampled from a distribution $\mathfrak{R}$ on functions from $\mathcal{M} \times \mathbb{N}$ to $\mathcal{M}$, where $\mathcal{M} := \mathcal{A}^*$. We define $\mathbf{h} \leftarrow \mathfrak{R}$ as follows:*

- *Choose $\mathbf{g}$ uniformly at random from $\{\mathbf{g} : \mathcal{M} \to \mathcal{M}\}$.*
- *For each $(x, \ell) \in \mathcal{M} \times \mathbb{N}$ set $\mathbf{h}(x, \ell) := \lfloor \mathbf{g}(x) \rfloor_\ell$, that is, output the first ℓ characters of the output of $\mathbf{g}$.*

We denote a random oracle, that provides access to a function distributed according to $\mathfrak{R}$, by R. We often omit the second input to the random oracle, assuming that all outputs have the same length.

2.3.2 Indistinguishability

Indistinguishability is an computational notion of distance between two distributions. By computational, we mean that no adversary is able to distinguish the distributions, not that they are necessarily close in statistical distance. In this thesis we focus on distributions of functions, and the model we have in mind is such that the adversary can make queries to the function sampled according to one distribution or the other.

When discussing cryptographic constructions, indistinguishability from a random oracle is a security notion for the constructed hash functions. We do not, however, give the adversary access to the internal functions. Hence, the notion is not perfectly suited to capture the real world. For keyed primitives, though, it is much more useful, as we illustrate in Chapter 4.

By classical indistinguishability we mean a feature of two distributions that are hard to distinguish if only polynomially many classical queries are allowed. Technically we consider families of distributions indexed by the security parameter $n \in \mathbb{N}$, the adversary knows the security parameter through the input 1^n that she is provided. For the sake of simplicity, we omit these details in the following definitions. The mentioned polynomial is evaluated on the security parameter. Note however that we have not yet specified this parameter. For now though we leave it implicit, the security parameter will be specified for the particular construction we are going to analyze. A common versions of the following definitions assume the adversary to be computationally bounded (so not only the number of queries is bounded but also A's runtime). We, however, use stronger definitions in this thesis, as we do not analyze distributions related to hardness of computational problems.

Definition 2.14 (Classical Indistinguishability)**.** *Two distributions $\mathfrak{D}_1$ and $\mathfrak{D}_2$ over a set $\mathcal{Y}^{\mathcal{X}}$ are computationally classically indistinguishable if no quantum algorithm* A *can distinguish $\mathfrak{D}_1$ from $\mathfrak{D}_2$ using a polynomial number of classical queries. That is, for all* A*, there is a negligible function ϵ such that*

$$\left|\underset{\mathrm{g}\leftarrow\mathfrak{D}_1}{\mathbb{P}}\left[b=1 : b \leftarrow \mathrm{A}^{\mathrm{g}}\right] - \underset{\mathrm{g}\leftarrow\mathfrak{D}_2}{\mathbb{P}}\left[b=1 : b \leftarrow \mathrm{A}^{\mathrm{g}}\right]\right| \leq \epsilon. \tag{2.25}$$

We write A^{g} to denote that adversary A has classical oracle access to g.

We will use the following generalization of the above definition to specify a notion valid for quantum adversaries.

Definition 2.15 (Quantum Indistinguishability [Zha12])**.** *Two distributions $\mathfrak{D}_1$ and $\mathfrak{D}_2$ over a set $\mathcal{Y}^{\mathcal{X}}$ are computationally quantumly indistinguishable if no quantum algorithm* A *can distinguish $\mathfrak{D}_1$ from $\mathfrak{D}_2$ using a polynomial number of quantum queries. That is, for all* A*, there is a negligible function ϵ such that*

$$\left|\underset{\mathrm{g}\leftarrow\mathfrak{D}_1}{\mathbb{P}}\left[b=1 : b \leftarrow \mathrm{A}^{|\mathrm{g}\rangle}\right] - \underset{\mathrm{g}\leftarrow\mathfrak{D}_2}{\mathbb{P}}\left[b=1 : b \leftarrow \mathrm{A}^{|\mathrm{g}\rangle}\right]\right| \leq \epsilon. \tag{2.26}$$

We write $\mathrm{A}^{|\mathrm{g}\rangle}$ to denote that adversary A has quantum oracle access to g, i.e. she can query g on a superposition of inputs.

In this thesis we often focus on indistinguishability from a random oracle. The first distribution is the one analyzed (e.g. the distribution of the cryptographic construction of interest) and the other is the uniform distribution over the set of all functions from $\mathcal{X}$ to $\mathcal{Y}$.

2.3.3 Indifferentiability

An important generalization of indistinguishability is *indifferentiability*. The main difference between these notions is the fact that we give the adversary access to the internal function of the cryptographic construction. This one change, although formally quite impactful, makes the security notion applicable to a wide range of realistic scenarios.

The notion was introduced by Maurer, Renner, and Holenstein in [MRH04]. Indifferentiability is one of the strongest security notions possible assuming that the internal function is random, while giving the distinguisher realistic capabilities in terms of access to public primitives. In the case of hash functions like SHA-2 [NIS15] and SHA-3 [NIS14], the specification of the internal function is in fact freely available.

Indifferentiability captures indistinguishability of two worlds: the real and the ideal world. The former is the real construction using an ideal primitive. The latter is the ideal version of the construction (often the random oracle) and a *simulator*. The simulator is an algorithm that simulates the public primitives (e.g. the internal functions) to the adversary. Assuming the internal functions to be ideal is the standard way of treating primitives in this class of security notions, showing that the constructed function is close to ideal is our goal.

The importance of indifferentiability lies in the fact that it captures the real applications of the constructions but also the fact that it guarantees composability. A system that is secure according to a composable security notion can be used in a larger system, without losing security guarantees.

Many constructions have been proven indifferentiable from a random oracle: Merkle-Damgård construction [Cor+05], sponge construction [Ber+08], Feistel networks [DS16], an many others [KM07; LLG12; Can+12; Bal14; GL16; Dai+17].

In recent years it was shown that the notion can be generalized to the quantum world. Although at first Carstens, Ebrahimi, Tabia, and Unruh gave arguments for the impossibility of proving quantum indifferentiability [Car+18], Zhandry later showed that it is indeed possible [Zha19a]. Zhandry developed a technique that allows to prove indifferentiability for the Merkle-Damgård construction. His result does not contradict the result of [Car+18], as it handles the *imperfect case*, albeit with a negligible error. We have also proven indifferentiability of the sponge construction in [Cza+19]. More details on [Car+18] can be found in Chapter 5.

An interesting observation has been made by Ristenpart, Shacham, and Shrimpton in [RSS11] on the applicability of indifferentiability to the setting of multiple interacting but distinct adversaries. They observed that the regular indifferentiability notion might not be enough in this setting, we cover this topic in more detail in Section 2.3.3.2 and prove some new results in Chapter 8. Since the publication of [RSS11] many generalizations of the notion have been made to mitigate the problem by specifying the computational capabilities of the simulators.

2.3.3.1 Regular Indifferentiability

Access to the publicly known internal function and the hash function constructed from it is handled by *interfaces*. An interface to a system is an access structure defined by the format of inputs and expected outputs. Let us illustrate this definition by an example, let the system C under consideration be a hash function $\mathsf{H}_{\mathbf{f}} : \{0,1\}^* \to \{0,1\}^n$, constructed using a function $\mathbf{f} : \{0,1\}^n \to \{0,1\}^n$. Then the *private* interface of the system accepts finite-length strings as inputs and outputs n-bit long strings. Outputs from the private interface are generated by the hash function, so we can write (slightly abusing notation) $\mathsf{C}^{\mathrm{priv}} = \mathsf{H}_{\mathbf{f}}$. The *public* interface accepts n-bit long strings and outputs n-bit strings as well. We have that $\mathsf{C}^{\mathrm{pub}} = \mathbf{f}$. The motivation for calling the interfaces private and public is that the internal function $\mathbf{f}$ is usually a publicly specified function (like in e.g. SHA-3 [NIS14]). The construction is run locally by the distinguisher.

A cryptographic system is an algorithm possibly calling subroutines. In this thesis we understand it as a cryptographic construction.

The following definitions and Theorem 2.18 are the rephrased versions of definitions and theorems from [MRH04; Cor+05]. We also make explicit the fact that the definitions are independent of the threat model we consider—whether it is the classical model or the quantum model. To expose those two cases we write "classical or quantum" next to algorithms that can be classical or quantum machines; Communication between algorithms (systems, adversaries, and environments) can also be of two types, where quantum communication will involve quantum states (consisting of superpositions of inputs).

Definition 2.16 (Indifferentiability, Definition 3 in [MRH04], rephrased). *A cryptographic (classical or quantum) system* C *is* (q,ϵ)-indifferentiable *from* R, *if there is an efficient (classical or quantum)* simulator S *such that for any efficient (classical or quantum)* distinguisher D *with binary output (0 or 1) that makes at most* q *(classical or quantum) queries there is a negligible function* $\epsilon' \in O(\epsilon)$ *such that the advantage*

$$\left|\mathbb{P}\left[b=1 : b \leftarrow \mathrm{D}[\mathsf{C}_k^{\mathrm{priv}}[\mathsf{C}_k^{\mathrm{pub}}], \mathsf{C}_k^{\mathrm{pub}}]\right] - \mathbb{P}\left[b=1 : b \leftarrow \mathrm{D}[\mathrm{R}_k^{\mathrm{priv}}, \mathsf{S}[\mathrm{R}_k^{\mathrm{pub}}]]\right]\right| \leq \epsilon'(k), \tag{2.27}$$

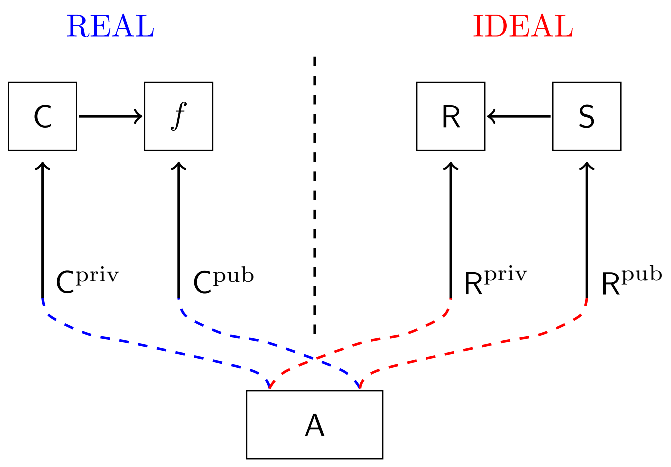


Figure 2.2: A schematic representation of the notion of indifferentiability, Definition 2.16. Arrows denote "access to" the pointed system.

where k is the security parameter.

It is important to note that if R is the random oracle (which is often the case), then both its interfaces are the same and output random outputs of appropriate given length. The definitions are still valid and the theorem below holds also if we interpret efficiency in terms of queries made by the algorithms. Note that then we can allow the algorithms to be unbounded with respect to runtime, the distinction between quantum and classical queries is still of crucial importance though. In Figure 2.2 we present a scheme of the situation captured by Definition 2.16.

Indifferentiability is a strong notion of security mainly because if fulfilled it guarantees composability of the secure cryptosystem. The "as secure as" relation is important in proving composability:

Definition 2.17 (As secure as, Definition 1 in [MRH04], rephrased)**.** *A cryptographic (classical or quantum) system* C *is said to be* as secure as C′ *if for all efficient (classical or quantum) environments* Env *the following holds: For any efficient (classical or quantum) attacker* A *accessing* C *there exists another efficient (classical or quantum) attacker* A′ *accessing* C′ *such that the difference between the probability distributions of the binary outputs of* Env[C, A] *and* Env[C′, A′] *is negligible, i.e.*

$$|\mathbb{P}\left[b = 1 : b \leftarrow \textsc{Env}[\mathsf{C}, \mathsf{A}]\right] - \mathbb{P}\left[b = 1 : b \leftarrow \textsc{Env}[\mathsf{C}', \mathsf{A}']\right]| \leq \epsilon(k)\,, \tag{2.28}$$

where ϵ is a negligible function.

In the following we say that a cryptosystem T is *compatible* with C if the interfaces for interaction of T with C are matching.

Theorem 2.18 (Composability, Theorem 1 in [MRH04], rephrased)**.** *Let* T *range over* (*classical or quantum*) *cryptosystems compatible with* C *and* R*, then* C *is* (q, ε)-*indifferentiable from* R *if and only if for all* T, T[C] *is as secure as* T[R].

2.3.3.2 Resource-Restricted Indifferentiability

In [RSS11] the authors identify a problem with the regular notion of indifferentiability. They show that the original definition does not guarantee security for games that have multiple stages. A solution the authors of [RSS11] propose is reset indifferentiability. This notion allows the distinguisher to reset the simulator to her initial state. In [Dem+13] the authors propose an interesting alternative to regular indifferentiability: memory-aware indifferentiability where we explicitly limit the memory of the simulators. It is also shown in [RSS11; Dem+13] that reset indifferentiability is equivalent to regular indifferentiability with stateless simulators. It is proved in [BBM13] that multi-stage indifferentiability (indifferentiability with stateless simulators) is equivalent to reset indifferentiability. Moreover allowing just one reset is enough to capture the security guaranteed by reset indifferentiability.

By resource-restricted indifferentiability we mean in general the notion of indifferentiability where we restrict the simulator in some way. The most important restriction is by allowing only some number of (quantum) bits of internal memory. In the classical case the simulator has just the t classical bits of memory. In the quantum case these bits are quantum, but S does not have any additional classical memory. Moreover, any randomness generated by the simulator has to be stored in her memory.

Definition 2.19 (Memory-Restricted Indifferentiability)**.** *A cryptographic* (*classical or quantum*) *system* C *is* (q, ϵ)-indifferentiable *from* R *in the presence of* t (*quantum*) *bits of adversarial memory, if there is an efficient* (*classical or quantum*) simulator S *with memory restricted to* t (*quantum*) *bits such that for any efficient* (*classical or quantum*) distinguisher D *with binary output* (*0 or 1*) *that makes at most* q (*classical or quantum*) *queries there is a negligible function* $\epsilon' \in O(\epsilon)$ *such that the advantage*

$$\left|\mathbb{P}\left[b = 1 : b \leftarrow \mathsf{D}[\mathsf{C}_k^{\mathrm{priv}}[\mathsf{C}_k^{\mathrm{pub}}], \mathsf{C}_k^{\mathrm{pub}}]\right] - \mathbb{P}\left[b = 1 : b \leftarrow \mathsf{D}[\mathsf{R}_k^{\mathrm{priv}}, \mathsf{S}[\mathsf{R}_k^{\mathrm{pub}}]]\right]\right| \leq \epsilon'(k), \tag{2.29}$$

where k *is the security parameter.*

Stateless indifferentiability is just memory-restricted indifferentiability when we give the simulator $t = 0$ bits of memory.

In [JM18] the authors propose a solution to the problem of multi-stage games that is done in the spirit of abstract cryptography [MR11; Mau11]. They propose a security notion that combines indifferentiability with Universal Computational Extractors [BHK13; BHK14]. This notion captures a similar level of security as indifferentiability but in a limited *context*. A context is understood as the scenarios in which security is valid.

2.4 Useful Proof Techniques

In this section we describe three techniques and frameworks that are especially useful in the reminder of the thesis. We focus on the polynomial method (Section 2.4.1), the game-playing framework (Section 2.4.2), and the one-way to hiding lemma (Section 2.4.3). The second framework is a standard classical approach to proving security of a variety of cryptosystems. The other two are useful methods in quantum cryptography.

2.4.1 The Polynomial Method

The polynomial method uses the rich mathematical structure of polynomials to prove closeness of probability distributions. The polynomial method was first developed by Nisan and Szegedy [NS94] and then extended to bound the query complexity of quantum algorithms in [Bea+01]. Later, a similar approach was applied to average case complexity, more useful in cryptography [Zha15b; Zha12; SY17].

In this section we describe the proof technique—based on approximating polynomials—that proves useful when dealing with notions like quantum indistinguishability.

Theorem 2.20 (Theorem 3.1 in [Zha15b]). *Let* A *be a quantum algorithm making* q *quantum queries to an oracle* $\mathbf{h} : \mathcal{X} \to \mathcal{Y}$*. If we draw* $\mathbf{h}$ *from some distribution* $\mathfrak{D}$*, then the quantity* $\mathbb{P}_{\mathbf{h}\leftarrow\mathfrak{D}}[b = 1 : b \leftarrow \mathrm{A}^{|\mathbf{h}\rangle}]$ *is a linear combination of the quantities* $\mathbb{P}_{\mathbf{h}\leftarrow\mathfrak{D}}[\forall i \in [2q] : \mathbf{h}(X^i) = Y^i]$*, where* $\forall i \in [2q] : (X^i, Y^i) \in \mathcal{X} \times \mathcal{Y}$*. Moreover the coefficients of the linear combination depend only on* A *and not* $\mathbf{h}$*.*

The intuition behind the above theorem is that with q queries the amplitudes of the quantum state of the algorithm depend on at most q input-output pairs. The probability of any outcome is a linear combination of squares of amplitudes, that is why we have $2q$ input-output pairs in the probability function. Finally as the probability of any measurement depends on just $2q$ input-output pairs the same holds for the algorithm's output probability. All the information about $\mathbf{h}$ comes from the queries A made.

We use the above theorem together with statements about approximating polynomials to connect the probability of some input-output behavior of a function from a given distribution with the probability of the adversary distinguishing two distributions.

Theorem 2.21 (Theorem 7.3 in [Zha12]). *Fix* q*, and let* $\mathfrak{F}_t$ *be a family of distributions on* $\mathcal{Y}^{\mathcal{X}}$ *indexed by* $t \in \mathbb{Z}^+ \cup \{\infty\}$*. Suppose there is an integer* d *such that for every* $2q$ *pairs* $\forall i \in [2q] : (X^i, Y^i) \in \mathcal{X} \times \mathcal{Y}$*, the function* $\mathbf{p}(1/t) = \mathbb{P}_{\mathbf{h}\leftarrow\mathfrak{F}_t}[\forall i \in [2q] : \mathbf{h}(X^i) = Y^i]$ *is a polynomial of degree at most* d *in* $1/t$

(for $t = \infty$ the inverse is 0). Then for any quantum algorithm A *making at most q quantum queries, the output distribution under $\mathfrak{F}_t$ and $\mathfrak{F}_\infty$ are $\pi^2 d^3/3t$-close:*

$$\left| \underset{\mathbf{h} \leftarrow \mathfrak{F}_t}{\mathbb{P}} \left[b = 1 : b \leftarrow \mathrm{A}^{|\mathbf{h}\rangle} \right] - \underset{\mathbf{h} \leftarrow \mathfrak{F}_\infty}{\mathbb{P}} \left[b = 1 : b \leftarrow \mathrm{A}^{|\mathbf{h}\rangle} \right] \right| < \frac{\pi^2 d^3}{6t}. \tag{2.30}$$

This theorem is an average-case version of the polynomial method often used in complexity theory. If the polynomial approximating the ideal behavior of $\mathbf{h} \leftarrow \mathfrak{F}_\infty$ is of low degree the distance between the output distributions must be small.

2.4.2 Game-Playing Proofs

The modern approach to cryptography relies on mathematical rigor: Trust in a given cryptosystem is mainly established by proving that, given a set of assumptions, it fulfills a security definition formalizing real-world security needs. Apart from the definition of security, the mentioned assumptions include the threat model, specifying the type of adversaries we want to be protected against. One way of formalizing the above notions is via *games*, i.e. programs interacting with the adversaries and outputting a result signifying whether there has been a breach of security or not. Adversaries in this picture are also modeled as programs, or more formally as interacting Turing machines.

The framework of game-playing proofs introduced by Bellare and Rogaway in [BR06]—modeling security arguments as games, played by the adversaries—is especially useful because it makes proofs easier to verify. Probabilistic considerations might become quite involved when talking about complex systems and their interactions; the structure imposed by games, however, simplifies them. In the game-playing framework, randomness can be, for example, considered to be sampled on the fly, making conditional events easier to analyze. A great example of that technique is given in the proof of the PRP/PRF switching lemma in [BR06].

Many proofs of security in cryptography follow the Game-Playing framework, proposed in [BR06]. It is a very powerful technique as cryptographic security proofs tend to be simpler to follow and formulate in this framework. The central idea of this approach are *identical-until-bad* games. Say games G and H are two programs that are syntactically identical except for code that follows after setting a flag Bad to one, then we call those games identical-until-bad. Usually in cryptographic proofs G and H will represent two functions that an adversary A will have oracle access to. In the following we denote the situation when A interacts with H by A^{H}. Then we can say the following about the adversary's view.

Lemma 2.22 (Fundamental lemma of game-playing, Lemma 2 of [BR06]). *Let* G *and* H *be identical-until-bad games and let* A *be an adversary that outputs a bit* b. *Then*

$$\left|\mathbb{P}[b=1 : b \leftarrow \mathrm{A}^{\mathsf{H}}] - \mathbb{P}[b=1 : b \leftarrow \mathrm{A}^{\mathsf{G}}]\right| \leq \mathbb{P}[\mathrm{Bad}=1 : \mathrm{A}^{\mathsf{G}}]. \tag{2.31}$$

2.4.3 One-Way to Hiding Lemma

The One-way To Hiding (O2H) lemma is the quantum counterpart of the fundamental game-playing lemma (Lemma 2.22). One of the common use-cases of the game-playing framework is reprogramming random oracles. By that we mean changing the output of a random oracle to some chosen value on some input. The original O2H lemma was used in that exact way. A new formulation of the lemma generalized modifying the random oracle to a set $\mathcal{S}$ of inputs that differ in the two oracles that are being distinguished by the adversary.

The original O2H lemma, Lemma 5 in [Unr14], is phrased in terms of reprogramming a random oracle. The statement is that the distinguishing advantage of algorithms given as input $\mathbf{h}(x)$ and some uniformly random y is bounded by $2q\sqrt{P_{\mathrm{B}}}$. The quantity P_{B} is the probability that for a randomly chosen query (i.e. one of the q queries the adversary makes) A queries x (when her query register is measured in the computational basis).

The new formulation is based on the idea of punctured oracles: quantum accessible oracles that include a measurement of the adversary's query register after every query that checks if $x \in \mathcal{S}$ is measured. If the check succeeds, i.e. the measurement detects that the adversary queried $x \in \mathcal{S}$, then the measurement outputs 1. Such oracles are denoted by $\mathbf{h}\setminus\mathcal{S}$. By Find we denote the event that any of these measurements outputs 1. The statement of this new formulation of the O2H lemma is as follows:

Theorem 2.23 (Theorem 1 in [AHU19]). *Let* $\mathcal{S} \subseteq \mathcal{X}$ *be random. Let* $\mathbf{g}, \mathbf{h} : \mathcal{X} \to \mathcal{Y}$ *be random functions satisfying* $\forall x \notin \mathcal{S} : \mathbf{g}(x) = \mathbf{h}(x)$. *Let* z *be a random bitstring.* ($\mathcal{S}$, $\mathbf{g}$, $\mathbf{h}$, z *may have arbitrary joint distribution.*) *Let* A *be an oracle algorithm* (*not necessarily unitary*) *of query depth* d, *then*

$$\left|\mathbb{P}\left[b=1 : b \leftarrow \mathrm{A}^{\mathbf{h}}(z)\right] - \mathbb{P}\left[b=1 : b \leftarrow \mathrm{A}^{\mathbf{g}}(z)\right]\right| \leq 2\sqrt{(d+1)\mathbb{P}\left[\mathrm{Find} : \mathrm{A}^{\mathbf{h}\setminus\mathcal{S}}(z)\right]}, \tag{2.32}$$

$$\left|\mathbb{P}\left[b=1 : b \leftarrow \mathrm{A}^{\mathbf{h}}(z)\right] - \mathbb{P}\left[b=1 : b \leftarrow \mathrm{A}^{\mathbf{h}\setminus\mathcal{S}}(z)\right]\right| \leq \sqrt{(d+1)\mathbb{P}\left[\mathrm{Find} : \mathrm{A}^{\mathbf{h}\setminus\mathcal{S}}(z)\right]}. \tag{2.33}$$

More results can be found in the original paper. Another interesting generalization of the O2H lemma can be found in [Bin+19]. In [Cza+19] we generalize the technique to include the results of [Zha19a], Chapter 6 is dedicated to this generalization.

An interesting lemma that we are going to use in this thesis gives a bound on probability of finding.

Lemma 2.24 (Corollary 1 in [AHU19]). *Suppose* $\mathbf{h}$, $\mathcal{S}$, *and* z *are independent, and that* A *is a* q*-query algorithm. Let* $P_{\max} := \max_x \mathbb{P}\left[x \in \mathcal{S}\right]$. *Then*

$$\mathbb{P}\left[\text{Find} : \mathrm{A}^{\mathbf{h}\setminus\mathcal{S}}(z)\right] \leq 4q \cdot P_{\max}. \tag{2.34}$$

The statement in [AHU19] considers semi-classical oracles, oracles that only answer if $x \in \mathcal{S}$, but Lemma 2.24 is a straight forward corollary of that. Note that A with access to the semi-classical oracle can easily simulate $\mathbf{h} \setminus\mathcal{S}$ by using just two queries to the semi-classical oracle.

We also state a generalization of Lemma 2.24 valid for adversaries that get quantum input.

Lemma 2.25. *Let* A *be any quantum oracle algorithm making at most* q *quantum queries to a punctured oracle with domain* $\mathcal{X}$. *Let* $\mathcal{S} \subseteq \mathcal{X}$ *and* $|\mathrm{st}\rangle \in \mathbb{C}^{2n}$. ($\mathcal{S}$ *and* $|\mathrm{st}\rangle$ *may have arbitrary joint distribution independent from* $\mathbf{h}$.) *Let* B *be an algorithm that on input* $|\mathrm{st}\rangle$ *chooses* $i \overset{\$}{\leftarrow} \{1, \ldots, q\}$ *; runs* $\mathrm{A}^{\mathbf{h}}$ *until (just before) the* i*-th query; then measures the query input register in the computational basis and outputs the measurement output* x.

Then

$$\mathbb{P}\left[\text{Find} : \mathrm{A}^{\mathbf{h}\setminus\mathcal{S}}(|\mathrm{st}\rangle)\right] \leq 4q \cdot P_{\text{guess}}, \tag{2.35}$$

where P_{guess} *is the the guessing probability of* $\mathcal{S}$ *(treated as a random variable) given the quantum side-information* $|\mathrm{st}\rangle$.

Proof. Theorem 2 in [AHU19] states

$$\mathbb{P}\left[\text{Find} : \mathrm{A}^{\mathbf{h}\setminus\mathcal{S}}(z)\right] \leq 4q \cdot \mathbb{P}\left[\mathcal{S} \cap \mathcal{T} = \emptyset : \mathcal{T} \leftarrow \mathrm{B}(z)\right] \tag{2.36}$$

for a random z (with arbitrary joint distribution). Following the proof of this theorem one can easily generalize the statement to quantum inputs. The proof of Theorem 2 in [AHU19] does not depend on the sort of input the algorithms have, just that B can simulate A.

The next step is the observation (made in the proof of Lemma 2.24 in [AHU19]) that the probability $\mathbb{P}\left[\mathcal{S} \cap \mathcal{T} = \emptyset : \mathcal{T} \leftarrow \mathrm{B}(|\mathrm{st}\rangle)\right]$ is bounded by the probability that B outputs an element of $\mathcal{S}$:

$$\mathbb{P}\left[\mathcal{S} \cap \mathcal{T} = \emptyset : \mathcal{T} \leftarrow \mathrm{B}(|\mathrm{st}\rangle)\right] \leq \mathbb{P}\left[x \in \mathcal{S} : x \leftarrow \mathrm{B}(|\mathrm{st}\rangle)\right] \leq P_{\text{guess}} \tag{2.37}$$

The final bound comes from introducing P_{guess}. □

2.5 Cryptographic Constructions

One of the most versatile cryptographic primitives are cryptographic hash functions. A hash function is supposed to map arbitrary-length inputs to some fixed-length output. Moreover, for it to be useful in cryptography, it should also posses other features, e.g. collision-resistance. A hash function is said to be collision-resistant if no probabilistic polynomial time adversary can find two inputs mapping to the same output with non-negligible probability.

More concretely, a *hash function* is a function $\mathbf{h}^{\mathsf{H}} : \mathcal{X} \to \mathcal{Y}$ for some range $\mathcal{X}$ and domain $\mathcal{Y}$. Typically, $\mathcal{Y}$ consists of fixed-length strings, and $\mathcal{X}$ consists of fixed-length strings or arbitrary-length strings, e.g. $\mathcal{A}^*$. $\mathbf{h}$ can depend on an oracle H. Typically, H will be a random function, a random permutation, or simply be missing if we are in the standard model. Unless specified otherwise, we make no assumptions about the distribution of H.

It is rather hard to design a function that maps arbitrary-length strings and is collision resistant. It is possible though to make a good compression function. That is a function that takes as input only fixed-length strings and maps them to shorter strings while being collision-resistant. But is having such a "small" function sufficient? The answer is yes. The reason is the existence of constructions that apply the "small" function many times in a way that one can input arbitrary-length strings and still get the desired behavior, e.g. collision-resistance.

Among the most famous and widely used constructions for hash functions is the Merkle-Damgård construction [Mer90; Dam90] used in the SHA-1[7] and SHA-2 standardized hash functions [NIS15].

In this section we also discuss constructions for functions with fixed-length inputs. A common type of constructions in cryptography build a one-way function out of permutations. The reason for why such conversion is popular is that in practice it is easier to build a secure (colloquially saying: behaving randomly) permutation.

2.5.1 Composition of Compression Functions

The composition compression function is used in the proof of indifferentiability of the Merkle-Damgård construction in Lemma 3.2 in [Cor+05]. It is a simple compression function that can be used to exemplify proof techniques, for example for indifferentiability. The composition of compression functions is defined as

$$\textsc{Comp}_{\mathbf{h}_1,\mathbf{h}_2}(x_1, x_2) := \mathbf{h}_2(\mathbf{h}_1(x_1), x_2). \tag{2.38}$$

[7]This is no longer considered a secure hash function, note the first collision found in 2017 [Ste+17].

In [Cor+05] the authors provide a sketch of the indifferentiability proof of Comp.

This construction is used by us as an example of our methods for proving quantum indifferentiability. Moreover the construction is used in the External Storage Game, discussed in Chapter 8. In Figure 2.3 we present a scheme of the construction.

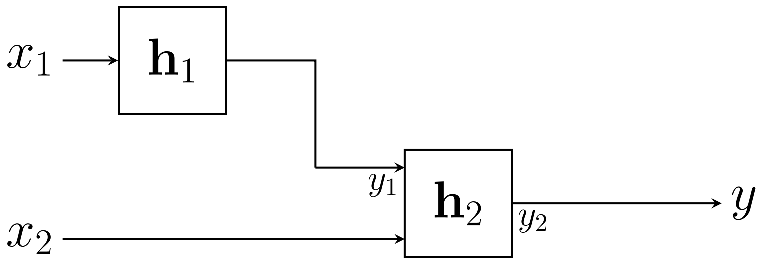


Figure 2.3: A schematic representation of composition of two hash functions $\text{Comp}_{\mathbf{h}_1,\mathbf{h}_2}(x_1, x_2) = y$.

2.5.2 The Sponge Construction

Another important construction is the sponge construction [Ber+07], underlying for example the current international hash function standard SHA-3 [NIS14], but also other hash functions such as Quark [Aum+10], Photon [GPP11], Spongent [Bog+13], Gluon [Ber+12a], KangarooTwelve [Ber+18], or the extendable-output function SHAKE [NIS14]. The sponge construction builds a hash function $\mathbf{h}$ from an internal function[8] $\mathbf{f}$.

The sponge construction has been proven indifferentiable from a random oracle [Ber+08]. From indifferentiability, most other security guarantees can been derived. Quantum security of the sponge construction has been proven in [Cza+18; CHS19; Cza+19], these results are presented in detail in chapters 3, 4, and 7.

The sponge construction is an extendable-output function that maps arbitrary-length inputs to outputs of a length specified by an additional input. The construction operates on states $s \in \mathcal{A} \times \mathcal{C}$, where $\mathcal{A}$ and $\mathcal{C}$ are finite sets. The parameter $r := \lfloor \log_2(|\mathcal{A}|) \rfloor$ is called the *rate* and the parameter $c := \lfloor \log_2(|\mathcal{C}|) \rfloor$ is called the *capacity*. The part in $\mathcal{A}$ of the state is called the *outer* part or outer state, the remaining part in $\mathcal{C}$ is called the *inner* part or inner state. The sponge uses an internal function $\mathbf{f} : \mathcal{A} \times \mathcal{C} \to \mathcal{A} \times \mathcal{C}$. To process a message consisting of several symbols (elements of $\mathcal{A}$), the sponge alternates between mixing a new message block into the outer state and applying $\mathbf{f}$, as

[8] $\mathbf{f}$ is not called a compression function, since the domain and range of $\mathbf{f}$ are identical.

shown in Figure 2.4. When all message blocks are processed (i.e. *absorbed* into the internal state) the sponge can be *squeezed* to produce ℓ outputs in $\mathcal{A}$ by alternating between applying $\mathbf{f}$ and outputting the outer state. We write $\textsc{Sponge}_{\mathbf{f}}$ for the sponge using $\mathbf{f}$ as internal function. The function constructed in that way behaves as follows, $\textsc{Sponge}_{\mathbf{f}} : \mathcal{A}^* \times \mathbb{N} \to \mathcal{A}^*$. We call the sponge construction using some $\mathbf{f}$ simply the Sponge.

Technically the sponge construction is defined together with a padding function $\textsc{pad} : \mathcal{M} \to \mathcal{A}^*$. In our discussion padding does not play an important role so we often omit it. The requirement on $\textsc{pad}$ put in [Ber+07] is that the mapping $\textsc{pad}(m)$ must be injective, and must be such that $|\textsc{pad}(m)| \geq 1$ and that the last character of $\textsc{pad}(m)$ is never 0 (this ensures injectivity for inputs of different lengths).

In Figure 2.4 we present a scheme of the sponge construction evaluated on input $\mathbf{M} = (M_1 \| M_2 \| M_3)$.

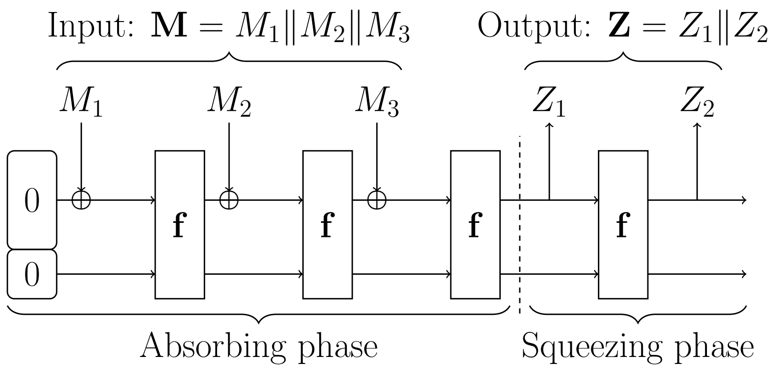

Input: $\mathbf{M} = M_1\|M_2\|M_3$
Output: $\mathbf{Z} = Z_1\|Z_2$
M_1
M_2
M_3
Z_1
Z_2
0
0
f
f
f
f
Absorbing phase
Squeezing phase

Figure 2.4: A schematic representation of the sponge construction: $\textsc{Sponge}_{\mathbf{f}}(M_1\|M_2\|M_3, 2) = Z_1\|Z_2$.

In Algorithm 2.1 we present the definition of $\textsc{Sponge}$. We assume that the set of outer states $\mathcal{A}$ is a finite Abelian group. The action of this group is denoted by $\oplus$. The initial value of the state in the construction is $(0,0) \in \mathcal{A} \times \mathcal{C}$, $0 \in \mathcal{A}$ is the neutral element and $0 \in \mathcal{C}$ is an arbitrary fixed element of $\mathcal{C}$. In the following we denote the part of the entire state S in $\mathcal{A}$ by $\bar{S}$ and the part in $\mathcal{C}$ by $\hat{S}$. To denote the internal function with output limited to the part in $\mathcal{A}$ and $\mathcal{C}$ we use the same notation as for states, $\bar{\mathbf{f}}$ and $\hat{\mathbf{f}}$ respectively. We often write $\mathcal{S} := \mathcal{A} \times \mathcal{C}$ to denote the combined set of states. We also write $\textsc{Sponge}_{\mathbf{f}}[\textsc{pad}, \mathcal{A}, \mathcal{C}, \ell]$ to denote $\textsc{Sponge}_{\mathbf{f}}[\textsc{pad}, \mathcal{A}, \mathcal{C}]$ that takes arguments in $\mathcal{M}$ and always outputs strings of length ℓ.

The absorbing phase of the sponge construction is defined in Algorithm 2.2.

Note that we use a general formulation of the construction, using any finite sets for $\mathcal{A}$ and $\mathcal{C}$. This is the way sponges are defined in [Ber+07]. In most of

Algorithm 2.1 $\textsc{Sponge}_{\mathbf{f}}[\textsc{pad}, \mathcal{A}, \mathcal{C}]$

```
    input: $\mathbf{M} \in \mathcal{M}, \ell \geq 0$
    output: $\mathbf{Z} \in \mathcal{A}^{\ell}$
 1: $\mathbf{P} := \textsc{pad}(\mathbf{M})$
 2: $S := (0,0) \in \mathcal{A} \times \mathcal{C}$                ▷ Initial Value
 3: for $i = 1$ to $|\mathbf{P}|$ do                              ▷ Absorbing phase
 4:     $S := (\bar{S} \oplus P_i, \hat{S})$
 5:     $S := \mathbf{f}(S)$
 6: $\mathbf{Z} := \bar{S}$                                       ▷ Squeezing phase
 7: while $|\mathbf{Z}| < \ell$ do
 8:     $S := \mathbf{f}(S)$
 9:     $\mathbf{Z} := \mathbf{Z} \| \bar{S}$
10: return $\lfloor \mathbf{Z} \rfloor_{\ell}$
```

Algorithm 2.2 $\textsc{Absorb}_{\mathbf{f}}$

```
    input: $\mathbf{P} \in \mathcal{A}^*$
    output: $S \in \mathcal{A} \times \mathcal{C}$
 1: $S := (0,0) \in \mathcal{A} \times \mathcal{C}$                ▷ Initial Value
 2: for $i = 1$ to $|\mathbf{P}|$ do
 3:     $S := (\bar{S} \oplus P_i, \hat{S})$
 4:     $S := \mathbf{f}(S)$
 5: return $S$
```

the other literature, e.g. [Ber+11b; NIS14], the sponge construction is defined on $\mathcal{A} = \{0,1\}^r$ and $\mathcal{C} = \{0,1\}^c$. All our results also work for $\textsc{Sponge}$ defined with bit-strings and bitwise XOR, as specified in [NIS14]. In Algorithm 2.1 we use $\lfloor \mathbf{Z} \rfloor_{\ell}$, for a general $\mathcal{A}$ this operation does not do anything; For $\mathcal{A} = \mathcal{A}_0^n$ (where $\mathcal{A}_0$ is some finite set), however, $\lfloor \mathbf{Z} \rfloor_{\ell}$ denotes the first ℓ symbols in $\mathbf{Z}$. For example when $\mathcal{A} = \{0,1\}^r$ then $\textsc{Sponge}$ outputs the first ℓ bits—it is important to interpret this notation like that and not as the first ℓ blocks of r bits.

For sponges defined on bit-strings the commonly used padding is of the form $\textsc{pad}(\mathbf{M}) = \mathbf{M} \| \widetilde{\textsc{pad}}(|\mathbf{M}| - \lfloor |\mathbf{M}| \rfloor)$, where $\widetilde{\textsc{pad}}$ denotes the function outputting a string, such that the length of the padded message is a multiple of r. For instance, the padding used in Keccak [NIS14] is $\mathbf{M} \| \textsc{pad}(\mathbf{M}) = \mathbf{M} \| 10^*1$ which appends to message $\mathbf{M}$ the bitstring 10^*1 with a suitable number of 0's (possibly none) such that the padded message is a multiple of r. Note that attaching a binary encoding of the message length $|\mathbf{M}|$ is not possible according to this assumption.

2.5.2.1 Sponge Graph

An important feature of the sponge construction is that one can associate to the internal function **f** a graph $G = (\mathcal{V}, \mathcal{E})$ [Ber+07]. It is called the *sponge graph;* The set of nodes $\mathcal{V} := \mathcal{A} \times \mathcal{C}$ corresponds to all possible states of the sponge. A directed edge connects any two nodes (s, t) whenever $\mathbf{f}(s) = t$, hence there are $|\mathcal{A} \times \mathcal{C}|$ edges in $\mathcal{E}$. From each node starts exactly one edge. We group the nodes with the same inner-part values into *supernodes*, so that we have $|\mathcal{C}|$ supernodes and each such supernode consists of $|\mathcal{A}|$ nodes. Edges between nodes are also edges between supernodes.

Whenever the adversary queries Sponge, she starts at the $(0, 0)$ node. This node is called the *root*. Next the first character (or block of characters, depending on $\mathcal{A}$) p_1 in the padded message $p = \textsc{pad}(m)$ is added to the outer part of the state and queried to the internal function $\mathbf{f}(p_1, 0) = s_2$. The node s_2 is the node in the edge $((p_1, 0), s_2) \in \mathcal{E}$. The same operation is repeated for all characters in p, concluding the absorbing phase. When Sponge starts generating output, we no longer modify the state, or just add 0 to the outer part. Note that knowing just p and G we can get to the last node traversed by $\textsc{Sponge}_{\mathbf{f}}(m)$. This leads us to the definition of a *sponge path*.

Definition 2.26 (Sponge Path, Definition 3 in [Ber+08])**.** *First, the empty string is a sponge path to the node $(0, 0)$. Then, if p is a sponge path to node $s = (\bar{s}, \hat{s})$ and there is an edge $((\bar{s} + a, \hat{s}), t)$ in the sponge graph G, $p' = p \| a$ is a sponge path to node t.*

Given the above definition, let us say that if p is a sponge path to s, then we define a function

$$\mathsf{SpPath}(s, G) := p. \tag{2.39}$$

The output of the above function is the input to the construction $\textsc{Sponge}_{\mathbf{f}}(., \ell = 1)$ (so the sponge that outputs a single symbol from $\mathcal{A}$, or r bits if $\mathcal{A} = \{0, 1\}^r$) that yields the output $\bar{s}$.

When we talk about the simulator in a proof of indifferentiability, we define the *simulator graph*. The graph kept by the simulator differs from the sponge graph discussed above by the number of edges in it. As the simulator lazy samples the internal function **f**, the set of edges $\mathcal{E}$ grows by at most one edge per one adversary's query. Other than that, all definitions and features of the sponge graph hold for the simulator graph as well. We refer to the simulator graph G as just the (sponge) graph whenever it is clear from context.

A supernode is called *rooted* if there is a path (that is just a set of connected edges) leading to it that starts at the root (the 0-supernode). The set $\mathcal{R}$ is the set of all rooted supernodes in G. By $\mathcal{U}$ we denote the set of supernodes containing a node with an outgoing edge.

A simulator graph is called *saturated* if $\mathcal{R} \cup \mathcal{U} = \mathcal{C}$. It means that for every inner state in $\mathcal{C}$ there is a path in G that leads to it from 0 (the root) or leads from it to another node. Saturation will be important in the proof of indifferentiability as the simulator wants to pick outputs of **f** without colliding inner parts (so not in $\mathcal{R}$) and making the path leading from 0 to the output longer by just one edge (so not in $\mathcal{U}$).

2.5.2.2 Keyed Sponges

Sponges can be keyed in several ways. For example, the state can be initialized with the key, referred to as root-keyed sponge in [And+15]. Another option is to just apply the sponge on the concatenation of key and message. This was called the keyed sponge in [Ber+11a] and the outer-keyed sponge in [And+15]. The last and for us most relevant concept is keying the sponge by replacing **f** with a keyed function $\mathbf{f}_K$. For the special case of $\mathbf{f}_K$ being a single-key Even-Mansour construction this was called E-M keyed sponge construction in [Cha+12] and later the inner-keyed sponge in [And+15]. We refer to the general case for any keyed function $\mathbf{f}_K$ as keyed-internal-function sponge.

2.5.2.3 Sponge Collisions

An *inner-collision* is a pair of input messages $m_1, m_2 \neq m_1$ such that

$$\widehat{\text{Absorb}}(\text{pad}(m_1)) = \widehat{\text{Absorb}}(\text{pad}(m_2)). \tag{2.40}$$

Let us assume $\mathcal{A} = \mathcal{A}_0^n$ for some finite set $\mathcal{A}_0$. With a padding rule that depends only on $|m| - \lfloor |m|/n \rfloor$ and works by just appending a string to m, it is possible to construct a collision in Sponge given an inner collision.

Claim 2.27. *Given the above assumptions about* pad, *from an inner collision one can construct a full collision of arbitrary length.*

Proof. We have m_1 and $m_2 \neq m_1$ such that $\widehat{\text{Absorb}}(\text{pad}(m_1))$ $= \widehat{\text{Absorb}}(\text{pad}(m_2))$. Then we define $m_3 := \text{pad}(m_2)\|$ $\Big(\overline{\text{Absorb}}(\text{pad}(m_2)) \oplus \overline{\text{Absorb}}(\text{pad}(m_1))\Big)$. When adding the last block to the state, we modify it to $\overline{\text{Absorb}}(\text{pad}(m_1))\|\widehat{\text{Absorb}}(\text{pad}(m_2))$. Thanks to our assumption the state is then the same as when evaluating $\text{pad}(m_1)$. The full state after absorbing m_3 is the same as after absorbing $\text{pad}(m_1)\|0^n$. The second padded message we define is $m_3' = \text{pad}(m_1)\|0^n$. The output-collision is (m_3', m_3) and it works for any length output. Note that m_3 and m_3' are distinct, the padding rule is injective and $m_1 \neq m_2$. Moreover m_3 and m_3' have length being a multiple of n so pad will just append an identical full block to both messages, preserving the collision. □

2.5.3 Rate-1/3 Compression Function

This construction (that we denote by RATE-1/3) of a compression function has been first defined in [SS08]. The authors explore constructing a compression function out of three functions. They also prove that if the internal functions are random, then the constructed hash function is collision resistant. Also note [MT07], for a discussion on security beyond the birthday-bound[9] barrier for other constructions based on random functions.

The construction is defined as follows:

$$\text{RATE-1/3}_{\mathbf{f}_1,\mathbf{f}_2,\mathbf{f}_3}(x_1, x_2) := \mathbf{f}_3\left(\mathbf{f}_1(x_1) \oplus \mathbf{f}_2(x_2)\right) \oplus \mathbf{f}_1(x_1), \tag{2.41}$$

where $\mathbf{f}_1 : \mathcal{X}_1 \to \mathcal{Y}$, $\mathbf{f}_2 : \mathcal{X}_2 \to \mathcal{Y}$, $\mathbf{f}_3 : \mathcal{X}_3 \to \mathcal{Y}$, and $\mathcal{X}_3 := \mathcal{Y}$, $\mathcal{Y}$ is an Abelian group. Without loss of generality for all results in this thesis concerning RATE-1/3 we can set $\mathcal{Y} = \{0,1\}^n$ and $\oplus$ is the bitwise XOR. When referring to RATE-1/3, $N = 2^n$.

In Figure 2.5 we present the scheme of the construction.

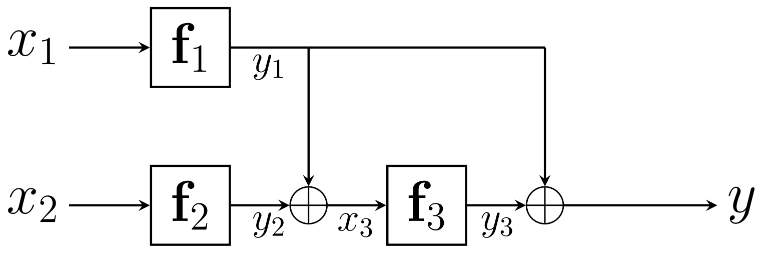


Figure 2.5: A schematic representation of the rate-1/3 hash functions $\text{RATE-1/3}_{\mathbf{f}_1,\mathbf{f}_2,\mathbf{f}_3}(x_1, x_2) = y$.

2.5.4 Encrypted Davis-Meyer Construction

The Encrypted Davis-Mayer (EDM) construction was introduced in [CS16] and further analyzed in [DHT17; MN17a]. The construction can be used to define a secure Message Authentication Code (MAC) or a PRF. Again, the advantage of constructing a function from permutations comes from practical reasons.

The EDM construction is defined as

$$\text{EDM}_{\boldsymbol{\pi}_1,\boldsymbol{\pi}_2}(x) := \boldsymbol{\pi}_2\left(\boldsymbol{\pi}_1(x) \oplus x\right), \tag{2.42}$$

where $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2$ are permutations $\mathcal{X} \to \mathcal{X}$, where $\mathcal{X} = \{0,1\}^n$. In Figure 2.6 we present a scheme of the construction.

[9]The birthday-bound is the bound on generic collision-finding algorithms.

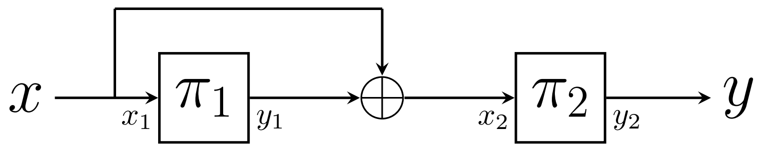


Figure 2.6: A scheme of the EDM construction.

In [CS16] the authors prove MAC and PRF security of the construction based on PRPs. The definition of MAC security can be found in [KL14], it is not the subject of this thesis, though. In [MN17a] the authors prove indistinguishability from a random function of the construction based on random permutations. In Chapter 7 we prove that EDM is classically indifferentiable from a random oracle when instantiated with random permutations and quantumly indifferentiable from a random oracle when instantiated with one-way random permutations (the adversary is given only forward access).

2.5.5 Encrypted Davis-Meyer Dual Construction

A construction very similar to EDM was defined in [MN17a]. The Encrypted Davis-Mayer Dual (EDMD) construction is also a design of a function using two permutations. The construction is defined as

$$\mathrm{EDMD}_{\boldsymbol{\pi}_1,\boldsymbol{\pi}_2}(x) := \boldsymbol{\pi}_2\left(\boldsymbol{\pi}_1(x)\right) \oplus \boldsymbol{\pi}_1(x), \tag{2.43}$$

where $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2$ are permutations $\mathcal{X} \to \mathcal{X}$, where $\mathcal{X} = \{0,1\}^n$. In Figure 2.7 we present a scheme of the construction.

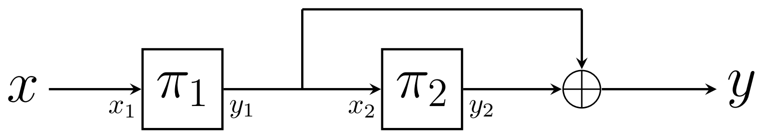


Figure 2.7: A scheme of the EDMD construction.

In [MN17a] the authors prove indistinguishability from a random function of the construction based on random permutations. In [MN17b] the authors show an interesting instantiation of EDMD. In Chapter 7 we prove that EDMD is classically indifferentiable from a random oracle when instantiated with random permutations and quantumly indifferentiable from a random oracle when instantiated with one-way random permutations (the adversary is given only forward access).

CHAPTER

3

STANDARD SECURITY OF SPONGES

Chapter contents

This chapter is dedicated to standard security notions and security proofs that relate them. We analyze collision resistance and collapsingness of the sponge construction. The assumptions we make are also limited to collision resistance, collapsingness, and zero-preimage resistance.

We show what these results mean for ideal primitives—random functions—and provide analogous statements that use stronger results, namely quantum indifferentiability.

We also present two quantum algorithms for collision search. One algorithm finds collisions in random sponges and the other in any functions (but only given access to an external random oracle).

3.1 Introduction

In the classical setting, we know that the sponge construction is collision-resistant if the internal function $\mathbf{f}$ is modeled as a random function or a random permutation [Ber+08]. Bertoni et al. [Ber+08] shows that the sponge construction is indifferentiable from a random oracle *in the classical setting*. Together with the fact that the random oracle is collision-resistant, collision-resistance of the sponge construction follows. However, the proof in [Ber+08] does not carry over to the post-quantum setting: their proof relies on the fact that queries performed by the adversary to the internal function are classical (i.e., not in superposition between different values). As first argued in [Bon+11], random oracles and related objects should be modeled as functions that can be queried in superposition of different inputs. Thus, we do not know whether the sponge construction (and thus hash functions like SHA-3) is collapsing (or at least collision-resistant in the post-quantum setting).

In Section 3.2 we prove that if the internal function $\mathbf{f}$ is collision-resistant when restricted to the outer and inner part of its output and it is hard to find a zero-preimage of $\mathbf{f}$ (restricted to the inner part of its output), then the sponge construction is collision-resistant. This result was first presented in [Cza+18]. In Chapter 7 we state that the sponge construction is indifferentiable from a random oracle if $\mathbf{f}$ is a random function; We give bounds that come from the fact that quantum collision-resistance is implied by quantum indifferentiability and compare them to the direct approach.

Then, in Section 3.3, we show a quantum algorithm for finding collisions in a random sponge. We also state the result of [Cza+18] where we show an algorithm that finds collisions in any function (given access to a random oracle), in particular in the sponge construction. The number of quantum queries to $\mathbf{f}$ asymptotically matches our bounds for collision resistance.

At the end of the chapter, in Section 3.4, we state results about collapsingness from [Cza+18]. If the internal function $\mathbf{f}$ is collapsing when restricted to the outer and inner part of its output, respectively, and if it is hard to find a zero-preimage of $\mathbf{f}$ (restricted to the inner part of its output), then the sponge construction is collapsing. If the internal function $\mathbf{f}$ is a random function, then the sponge construction is collapsing. We show this result directly and via indifferentiability.

It should be stressed that we *do not* show that the sponge construction is collapsing (or even collision-resistant) if the internal function $\mathbf{f}$ is an *efficiently invertible* random permutation. In this case, it is trivial to find zero-preimages by applying the inverse permutation to 0. This means that the presented result cannot be directly used to show the security of, say, SHA-3, because SHA-3 uses an efficiently invertible permutation as internal function. Our results apply to hash functions where the internal function is not (efficiently) invertible, e.g., Gluon [Ber+12a]. It seems that this limitation is just a residue of our technique. We do believe, however, that the results hold for permutations too, note for example that sponges are indistinguishable from random oracles for functions and permutations[1].

The special role of the $0 \in \mathcal{C}$ comes from the fact that if the sponge state is of the form $(a, 0)$ at some point (except for the initial value), then it is possible to construct an inner-collision. In turn, an inner-collision can be easily transformed into a full collision in Sponge, which breaks the security of the construction. For the preimage of zero to be useful, though, it has to lie in a sponge path. Intuitively speaking, the chances of an adversary finding a sponge path that leads to $0 \in \mathcal{C}$ are extremely low. This fact is difficult to formally incorporate into our proofs without solving a much harder task of proving quantum indifferentiability of Sponge.

3.2 Collision-resistance of the sponge construction

In this section we state our result concerning collision-resistance of the sponge construction. We motivate our statement with Lemma 3.2 connecting attacks on some features of the internal function with collision-resistance of the overall construction. Those features are collision-resistance of $\hat{\mathbf{f}}$, collision-resistance of $\bar{\mathbf{f}}$, as in Definition 2.7, and zero-preimage-resistance of $\hat{\mathbf{f}}$, Definition 2.6.

Before we proceed, let us discuss our choice of sponge parameters. We state our results in terms of the general sets $\mathcal{A}$ and $\mathcal{C}$. The group of outer states is a Cartesian product: $\mathcal{A} = \mathcal{A}_0^n$ with action $\oplus : \mathcal{A} \times \mathcal{A} \to \mathcal{A}$ defined as $a \oplus b := a_1 \oplus_0 b_1 \oplus_0 \cdots \oplus_0 a_n \oplus_0 b_n$, where $\oplus_0$ is the action in the group $\mathcal{A}_0$. We chose such parameters to make our result the most general. An example of parameters that is widely used is $\{0, 1\}^r$. When discussing the last $\mathbf{f}$ in an evaluation of $\text{Sponge}_{\mathbf{f}}$

[1]As proven in [CHS19] and Chapter 4.

we write:

$$\bar{\mathbf{f}}_{\min} := \lfloor \bar{\mathbf{f}} \rfloor_{\min\{\ell,n\}} \tag{3.1}$$

to denote the function outputting the first $\min\{\ell, n\}$ symbols in the outer part of the output.

Here we state the reduction between features of $\mathbf{f}$ and collision-resistance of the sponge construction.

Theorem 3.1. *Assume a quantum τ-time adversary* B *with quantum access to* $\mathbf{f}$ *that finds a collision in* $\textsc{Sponge}_{\mathbf{f}}[\textsc{pad}, \mathcal{A}, \mathcal{C}, \ell]$ *with probability* ε. *Then there exist:*

- *a* $\left(\tau + 2\tau_{\textsc{pad}} + O\left((4T_m + 2\lceil \frac{\ell}{n} \rceil)\tau_{\mathbf{f}}\right)\right)$*-time adversary* A_1 *that finds a collision in* $\hat{\mathbf{f}}$ *with probability* $\hat{\varepsilon}_{\text{coll}}$,
- *a* $\left(\tau + 2\tau_{\textsc{pad}} + O\left((4T_m + 2\lceil \frac{\ell}{n} \rceil)\tau_{\mathbf{f}}\right)\right)$*-time adversary* A_2 *that finds a pre-image of* 0 *under* $\hat{\mathbf{f}}$ *with probability* $\hat{\varepsilon}_{\text{zero}}$,
- *and a* $\left(\tau + 2\tau_{\textsc{pad}} + O\left((4T_m + 2\lceil \frac{\ell}{n} \rceil)\tau_{\mathbf{f}}\right)\right)$*-time adversary* A_3 *that finds a collision in* $\bar{\mathbf{f}}_{\min}$ *with probability* $\bar{\varepsilon}_{\text{coll}}$,

such that $\varepsilon \leq \hat{\varepsilon}_{\text{coll}} + \hat{\varepsilon}_{\text{zero}} + \bar{\varepsilon}_{\text{coll}}$. T_m *denotes an upper bound on the number of blocks in the padding of the input messages,* $\tau_{\mathbf{f}}$ *is the time required for a single classical invocation of* $\mathbf{f}$ *and* $\tau_{\textsc{pad}}$ *is the time required for one invocation of* $\textsc{pad}$. *The same bounds hold when measuring the time in number of oracle queries.*

Before proving Theorem 3.1 we present the lemma relating the output of a sponge-collision-finder with collisions and pre-images of $\mathbf{f}$.

Lemma 3.2. *Assume that* $\textsc{pad}$ *is injective. There is a deterministic polynomial-time oracle algorithm* A *such that for any* $m \neq m'$ *with* $\textsc{Sponge}_{\mathbf{f}}(m)$
$= \textsc{Sponge}_{\mathbf{f}}(m')$, $\mathrm{A}^{\mathbf{f}}(m, m')$ *outputs one of the following:*

- $(\text{inner}, (s, s'))$ *where* (s, s') *is a collision of* $\hat{\mathbf{f}}$,
- (zero, s) *where* s *is a zero-preimage of* $\hat{\mathbf{f}}$,
- *or* $(\text{outer}, (s, s'))$ *where* (s, s') *is a collision of* $\bar{\mathbf{f}}_{\min}$.

The runtime of the algorithm is at most $2\tau_{\textsc{pad}} + O(T_m \tau_{\mathbf{f}})$, *where* T_m *denotes an upper bound on the number of blocks in the padded input messages,* $\tau_{\mathbf{f}}$ *is the time required for a single classical invocation of* $\mathbf{f}$ *and* $\tau_{\textsc{pad}}$ *is the time of computing* $\textsc{pad}$.

Proof. Algorithm A starts by computing the first inner-state of the squeezing phase on input of the two colliding messages, i.e., it evaluates $\textsc{Absorb} \circ \textsc{pad}$, defined in Algorithm 2.2. We will denote the states traversed during this calculation by s_i and s_i' for m and m', respectively. As our analysis starts with the final state of this computation and revisits the intermediate states in backwards direction, we denote by s_0 the final state, whose outer part is output (for $\ell < n$ only the first ℓ symbols), by s_{-1} the state just before the last application of $\mathbf{f}$ and so on. Using $p := \textsc{pad}(m)$ and $p' := \textsc{pad}(m')$, the intermediate states s_{-i} for $1 \leq i \leq |p|_n - 1$, where $|p|_n$ denotes the number of n-symbol blocks in p, are defined by $s_{-i} := \bar{\mathbf{f}}(s_{-i-1}) \oplus p_{|p|_n+1-i} \| \hat{\mathbf{f}}(s_{-i-1})$, $s_0 := \mathbf{f}(s_{-1})$ and $s_{-|p|_n} := p_1 \| 0$. As m and m' collide per assumption, we have $\lfloor s_0 \rfloor_{\min\{\ell,n\}} = \lfloor s'_0 \rfloor_{\min\{\ell,n\}}$.

1. Algorithm A first checks if s_{-1} or s'_{-1} are a preimage of $0 \in \mathcal{C}$, or form a collision under $\bar{\mathbf{f}}_{\min}$. If the inner part of s_0 (or s'_0) is 0, s_{-1} (s'_{-1}) is a preimage of 0 under $\hat{\mathbf{f}}$ and A outputs (zero, s_{-1}) ((zero, s'_{-1}), respectively). If $s_{-1} \neq s'_{-1}$, A outputs (outer, s_{-1}, s'_{-1}). These two states form a collision under $\bar{\mathbf{f}}_{\min}$ because they are the inputs to the last $\mathbf{f}$ in Sponge and $\lfloor s_0 \rfloor_{\min\{\ell,n\}} = \lfloor s'_0 \rfloor_{\min\{\ell,n\}}$. Otherwise, $s_{-1} = s'_{-1}$ and there are no preimages of zero.

2. If not done yet, $s_{-1} = s'_{-1}$ and A checks for a preimage of $0 \in \mathcal{C}$ or a collision in $\hat{\mathbf{f}}$. If $\hat{s}_{-1} = 0$, A found a preimage of 0. This is true as if both messages ended here then $s_{-1} = s'_{-1}$ would imply that $p = p'$ (and so $m = m'$) which contradicts the assumptions of the lemma. Hence, at least one message must be longer. Assuming the longer message is m, A outputs (zero, s_{-2}) (or (zero, s'_{-2}) if it was m'). Next the algorithm checks if $p_{-1} = p'_{-1}$, where we follow a similar notation for message blocks as for the states. The last block of the input is denoted by p_{-1}. If $p_{-1} \neq p'_{-1}$, A outputs (inner, s_{-2}, s'_{-2}). This is a collision of $\hat{\mathbf{f}}$ because $p_{-1} \neq p'_{-1}$ but $s_{-1} = s'_{-1}$. Thus $\mathbf{f}(s_{-2}) \neq \mathbf{f}(s'_{-2})$ which in turn implies $s_{-2} \neq s'_{-2}$ while $\hat{\mathbf{f}}(s_{-2}) = \hat{\mathbf{f}}(s'_{-2})$. We can be certain that there are at least two applications of $\mathbf{f}$ both in Sponge(m) and Sponge(m') because the right half of $s_{-1} = \hat{s}_{-1}$ is not 0.

3. If $p_{-1} = p'_{-1}$ we end up in the same situation as before but now for $i = 2$. Namely we have that $s_{-2} = s'_{-2}$ and the algorithm performs the same checks as before but for a bigger i. Repeat Step 2 for all $2 \leq i \leq \min\{|p|_n, |p'|_n\}$.

If the iteration ends without success, this especially means that no collision was found but at least one message was fully processed. In this case A outputs a preimage of 0 under $\hat{\mathbf{f}}$. That is because no collisions means that all compared message blocks are the same but the two messages are different per assumption. Hence, they must have different lengths. With different length messages that traverse the same state values at the point of $i = \min\{|p|_n, |p'|_n\}$ the inner part of both states is 0, so the algorithm will output (zero, $s'_{-|p|_n-1}$) (assuming $|p|_n < |p'|_n$). □

Proof of Theorem 3.1. We reduce the problem of finding collisions in Sponge to attacks on the internal function. Adversaries A_1, A_2, and A_3—that have quantum access to **f**—run B and then classically compute Sponge on the outputs of the collision finder. If that in fact is a collision they run algorithm A from Lemma 3.2 and output the last register of its output. Otherwise the algorithms outputs $\perp$. Note that the runtime of the described adversaries agrees with the claim. That allows us to write

$$\begin{aligned}
&\mathbb{P}[m \neq m' \wedge \textsc{Sponge}(m) = \textsc{Sponge}(m') : (m, m') \leftarrow \mathrm{B}] = \\
&\mathbb{P}\left[\left(s \neq s' \wedge \hat{\mathbf{f}}(s) = \hat{\mathbf{f}}(s') : (s, s') \leftarrow \mathrm{A}_1\right)\right. \\
&\left.\vee \left(s \in \hat{\mathbf{f}}^{-1}(0) : s \leftarrow \mathrm{A}_2\right) \vee \left(s \neq s' \wedge \bar{\mathbf{f}}_{\min}(s) = \bar{\mathbf{f}}_{\min}(s') : (s, s') \leftarrow \mathrm{A}_3\right)\right] \\
&\leq \hat{\varepsilon}_{\text{coll}} + \hat{\varepsilon}_{\text{zero}} + \bar{\varepsilon}_{\text{coll}},
\end{aligned} \tag{3.2}$$

where the inequality comes from the union bound. □

Note that the same bounds hold when measuring the time in number of oracle queries.

It is true that for $\ell > n$ our bound seems to be not optimal but our reductionist approach is not well suited to deal with consecutive applications of **f**.

3.2.1 Collision Resistance of Random Sponges

Let us now analyze the case of a random sponge. The success probability of a generic collision-finding algorithm in $\mathbf{h} : \mathcal{A} \times \mathcal{C} \to \mathcal{C}$ is at most $\frac{\pi^2}{3}\frac{(q+2)^3}{|\mathcal{C}|}$, as proven by Zhandry in Theorem 7 in [Zha15a]. To use Zhandry's results we need to make sure the distribution of $\hat{\mathbf{f}}$ is in fact uniform.

Lemma 3.3. *If* $\mathbf{f} : \mathcal{A} \times \mathcal{C} \to \mathcal{A} \times \mathcal{C}$ *is a random function and* A *that looks for collisions in* $\hat{\mathbf{f}}$ *makes at most* q *quantum queries to* **f** *and has success probability* $\hat{\varepsilon}_{\text{coll}}$*, then* $\hat{\varepsilon}_{\text{coll}} \leq \frac{\pi^2}{3}\frac{(q+2)^3}{|\mathcal{C}|}$.

Proof. Let $\mathbf{h} : \mathcal{A} \times \mathcal{C} \to \mathcal{C}$ be a uniformly random function. Let A′ perform the following steps: It picks a uniformly random function $\mathbf{g} : \mathcal{A} \times \mathcal{C} \to \mathcal{A}$ (we do not care about the runtime of A′, so it is not a problem that picking **g** takes exponential time). Then it simulates A where **f**′ is the function defined by $\mathbf{f}'(x) := \mathbf{g}(x) \| \mathbf{h}(x)$. Note that an oracle query to **f**′ can be implemented using a single oracle query to **h**. So A′ still performs q oracle queries.

The joint distribution of **f**′ and **h** is the same as that of **f** and $\hat{\mathbf{f}}$. Thus the advantage of A′ against **h** is the same as the advantage $\hat{\varepsilon}_{\text{coll}}$ of A against $\hat{\mathbf{f}}$. Thus we can use results proven in [Zha15a], $\hat{\varepsilon}_{\text{coll}} \leq \frac{\pi^2}{3}\frac{(q+2)^3}{|\mathcal{C}|}$. □

Similarly for $\bar{\mathbf{f}}_{\min}$ and $\bar{\mathbf{f}}$ it holds that the advantage of a q-query collision finder is $\bar{\varepsilon}_{\text{coll}} \leq \frac{\pi^2}{3}\frac{(q+2)^3}{|\mathcal{A}_0|^\ell}$ and $\frac{\pi^2}{3}\frac{(q+2)^3}{|\mathcal{A}|} = \frac{\pi^2}{3}\frac{(q+2)^3}{|\mathcal{A}_0|^n}$, respectively.

Zero-pre-image finding in a random function has probability of success at most $\hat{\varepsilon}_{\text{zero}} \leq 2^5(q+1)^2\,|\mathcal{C}|^{-1}$ as shown next.

Lemma 3.4. *Let* $\mathbf{f} : \mathcal{A} \times \mathcal{C} \to \mathcal{A} \times \mathcal{C}$ *be a random function. Then a* q*-query adversary finds a zero-preimage of* $\hat{\mathbf{f}}$ *with probability* $\leq \frac{2^5(q+1)^2}{|\mathcal{C}|}$.

Proof. Assume an algorithm $\mathrm{A}^{\mathbf{f}}$ that finds a zero-preimage of $\hat{\mathbf{f}}$ with some probability $\hat{\varepsilon}_{\text{zero}}$. Let $\mathbf{h} : \mathcal{A} \times \mathcal{C} \to \{0,1\}$ be a random function where each $\mathbf{h}(z)$ is independently chosen, with $\mathbb{P}[\mathbf{h}(z) = 1] = \gamma := |\mathcal{C}|^{-1}$. Let $\mathrm{B}^{\mathbf{h}}$ be the following algorithm: It picks functions $\mathbf{f}_1, \mathbf{f}_0 : \mathcal{A} \times \mathcal{C} \to \mathcal{A} \times \mathcal{C}$, where each $\mathbf{f}_0(x)$ is independently and uniformly chosen from $\mathcal{A} \times (\mathcal{C} \setminus \{0\})$ and each $\mathbf{f}_1(x)$ is independently and uniformly chosen from $\mathcal{A} \times \{0\}$. Define $\mathbf{f}'(x) := \mathbf{f}_{\mathbf{h}(x)}(x)$. Then $\mathrm{B}^{\mathbf{h}}$ runs $x \leftarrow \mathrm{A}^{\mathbf{f}'}$ and returns x. A query to $\mathbf{f}'$ can be implemented using 2 queries to $\mathbf{h}$. Thus $\mathrm{B}^{\mathbf{h}}$ performs $\leq 2q$ queries to $\mathbf{h}$. (We do not care about the runtime of $\mathrm{B}^{\mathbf{h}}$. Thus the exponential time required for implementing $\mathbf{f}_0, \mathbf{f}_1$ is not a problem.)

Note that $\mathbf{f}' : \mathcal{A} \times \mathcal{C} \to \mathcal{A} \times \mathcal{C}$ is a uniformly random function (for $\mathbf{h}$ distributed as described above). Thus $\mathrm{A}^{\mathbf{f}'}$ finds a zero-preimage of $\hat{\mathbf{f}}$ with probability $\hat{\varepsilon}_{\text{zero}}$, i.e., an x with $\hat{\mathbf{f}}(x) = 0$. Such an x then satisfies $\mathbf{h}(x) = 1$. Thus $\mathrm{B}^{\mathbf{h}}$ finds a 1-preimage of $\mathbf{h}$ with probability $\hat{\varepsilon}_{\text{zero}}$. [HRS16, Theorem 1] shows that a $2q$-query adversary can find a 1-preimage in $\mathbf{h}$ with probability at most $8\gamma(2q+1)^2$. Thus $\hat{\varepsilon}_{\text{zero}} \leq 8\gamma(2q+1)^2 \leq 2^5(q+1)^2\,|\mathcal{C}|^{-1}$. □

We are now ready to bound the probability to find collisions in a random Sponge. By coll $\leftarrow \mathrm{B}^{\mathbf{f}}$ we denote $m \neq m' \wedge \textsc{Sponge}(m) = \textsc{Sponge}(m') : (m, m') \leftarrow \mathrm{B}^{\mathbf{f}}$.

Corollary 3.5. *For any quantum adversary* B *finding collisions in* $\textsc{Sponge}_{\mathbf{f}}[\textsc{pad}, \mathcal{C}, \mathcal{A}_0^n]$ *with random* $\mathbf{f}$*, we have that*

$$\mathbb{P}[\text{coll} \leftarrow \mathrm{B}^{\mathbf{f}}] \leq \frac{\pi^2}{3}(q+2)^3 \max\{|\mathcal{C}|^{-1}, |\mathcal{A}|^{-1}, |\mathcal{A}_0|^{-\ell}\}. \tag{3.3}$$

The fact that we have a $|\mathcal{A}|^{-1} = |\mathcal{A}_0|^{-n}$ term in the above bound is a result of restricting $\bar{\mathbf{f}}_{\min}$ to $\bar{\mathbf{f}}$ in the case of $\ell > n$.

One possible approach to filling this gap between $1/|\mathcal{A}|$ and $1/|\mathcal{A}_0|^\ell$ for $\ell > n$ is through quantum indifferentiability of the sponge construction. In Chapter 7 we prove quantum indifferentiability of sponges with uniformly random $\mathbf{f}$, the bound that we get is then:

Theorem 3.6. *For any quantum adversary* B *finding collisions in* $\textsc{Sponge}_{\mathbf{f}}[\textsc{pad}, \mathcal{C}, \mathcal{A}_0^n]$ *with random* $\mathbf{f}$ *is bounded as*

$$\mathbb{P}[\text{coll} \leftarrow \mathrm{B}^{\mathbf{f}}] \leq \frac{\pi^2}{3} \frac{(q+2)^3}{|\mathcal{A}_0|^\ell} + 56 \frac{q(q+1)}{|\mathcal{C}|} + \sqrt{7 \frac{q(q+1)^2}{|\mathcal{C}|}}. \tag{3.4}$$

Proof. Given that $\textsc{Sponge}_{\mathbf{f}}$ is quantumly indifferentiable we can write

$$\mathbb{P}\left[\text{coll} \leftarrow \mathrm{B}'[\textsc{Sponge}_{\mathbf{f}}, \mathbf{f}]\right] \leq \varepsilon_I + \underset{\mathbf{h} \leftarrow \mathfrak{R}}{\mathbb{P}}\left[\text{coll} \leftarrow \mathrm{B}'[\mathbf{h}, \mathrm{S}]\right] \tag{3.5}$$

$$\leq \varepsilon_I + \frac{\pi^2}{3} \frac{(q+2)^3}{|\mathcal{A}_0|^\ell} \tag{3.6}$$

where B′ simulates B but is given access to $\textsc{Sponge}_{\mathbf{f}}$ and $\mathbf{f}$. We denote the indifferentiability advantage by ε_I. From Theorem 7.9 we know that $\varepsilon_I = 56 \frac{q(q+1)}{|\mathcal{C}|} + \sqrt{7 \frac{q(q+1)^2}{|\mathcal{C}|}}$. The inequality follows from using an indifferentiability distinguisher that outputs 1 if B′ outputs a collision. □

3.3 Quantum Collision-Finding Algorithms

In the following we present two quantum collision-finding algorithms against the sponge construction. The main contribution of this section is an algorithm that outputs collisions in random sponges with high probability. We also present the result from [Cza+18] where the collision can be found in any function $\mathbf{h}$ but with the use of an external random oracle. We focus on the standard parameters of Sponge: we set $\mathcal{A} = \{0,1\}^r$ and $\mathcal{C} = \{0,1\}^c$. We also put requirements on pad so that the padding just adds bits to the end of the message.

The general working of both attacks is to select a suitable function g, run a collision-finding algorithm by Ambainis [Amb07] to obtain a collision for g, and finally turn this collision into a collision for the target sponge. The *suitable* function in this context refers to the function giving a result that is optimal with respect to the number of queries to the internal function. We make a case distinction whether the length of the required collision ℓ is smaller or bigger than the capacity c of the sponge; This distinction in the algorithm that does not use a random oracle is done against $\mathbf{b}(c)$, an almost linear function of c defined in Equation (3.7). In the case $\ell < \mathbf{b}(c)$, we simply search for an output collision in Sponge. In the other case $\ell \geq \mathbf{b}(c)$, it is more efficient to search for an inner-collision, as an inner-collision is a collision in a function with c bits of output and can be extended to arbitrary-length output collision.

In this section, we assume that the padding function is of the form $\textsc{pad}(m) = m \| \widetilde{\textsc{pad}}(|m| - \lfloor |m|/r \rfloor)$, where $\widetilde{\textsc{pad}}$ denotes the function outputting a bitstring, such that the length of the padded message is a multiple of r. For instance, the padding used in Keccak is $\textsc{pad}(m) = m\|10^*1$ which appends to message m the bitstring 10^*1 with a suitable number of 0's (possibly none) such that the padded message is a multiple of r [NIS14]. Note that attaching a binary encoding of the message length $|m|$ is not possible according to this assumption. Say that $n_{\textsc{pad}}$ is the minimal number of bits appended by $\textsc{pad}$ (i.e. two in the example above) and we will assume that $0 < n_{\textsc{pad}} < r$.

Our attacks make heavy use of the following quantum algorithm by Ambainis [Amb07].

Theorem 3.7. ([*Amb07*] *Theorem 3*) *There exists a constant* k_{coll} *and a quantum algorithm* Coll *that finds a collision in any* $\mathrm{g} : \mathcal{X} \to \mathcal{Y}$ *that has at least one collision.* Coll *makes* $k_{\text{coll}} \cdot |\mathcal{X}|^{2/3}$ *quantum queries to* g *and finds a collision with probability at least* $15/16$.

We note that [Amb07] also gives guarantees on the actual quantum running time and memory requirements of the quantum collision-finding algorithm. Concretely, there exist (small) constants $k'_{\text{coll}}, k''_{\text{coll}}$ such that the running time and quantum memory is at most $k'_{\text{coll}} \cdot |\mathcal{X}|^{2/3} \cdot \log^{k''_{\text{coll}}}(|\mathcal{X}| + |\mathcal{Y}|)$. Therefore, all our results of this section which are stated in terms of query complexity also yield guarantees on the running time and memory, incurring the same blowup by a poly-logarithmic factor in the number of queries.

3.3.1 Quantum Collision Finding Without a Random Oracle

In this section we present a collision-finding algorithm for random sponges that does not use an external independent random oracle. The algorithm has quantum access to the internal function f. As we already mentioned we make heavy use of algorithm Coll from Theorem 3.7 for the crucial algorithmic part of the proof. Our contributions lie mostly in designing the appropriate function g that is queried by Coll and then proving that g has a collision in set $\mathcal{M}$—the key assumption of Theorem 3.7; We define $\mathcal{M}$ below in Definition 3.9. Note that in Equation (2.21) we specify what we mean by having a collision in a set.

Generic quantum collision-finding algorithms require some structure in the function in which we look for collisions. In [BHT98; Amb07; Zha15a] the function has to be two-to-one or have at least one collision, in [BES18] the distribution of outputs has to be independent (which is not the case for random sponges). We are not aware of any other generic collision-finding algorithms so we use the results of [Amb07] and prove that the assumption is indeed fulfilled by random sponges.

We first state the main theorem of this section. Later we state and prove two lemmas important for proving the main result. We finish with the proof of Theorem 3.8.

Theorem 3.8. *Let* $\mathbf{f}$ *be a random internal function (function or permutation). There exists a quantum algorithm* SPONGE-COLL *making* q *quantum queries to* $\mathbf{f}$ *that outputs a collision of length* ℓ *in* $\text{SPONGE}_{\mathbf{f}}[\text{PAD}, \{0,1\}^r, \{0,1\}^c]$*, i.e. two messages* $m \neq m'$ *such that* $\text{SPONGE}_{\mathbf{f}}(m, \ell) = \text{SPONGE}_{\mathbf{f}}(m', \ell)$*, with error independent of the sponge parameters.*

The number of queries depends on the sponge parameters, the distinction between which is done with

$$\mathbf{b}(c) := \frac{r - n_{\text{PAD}}}{r - n_{\text{PAD}} + 2} c - 2\log_2\left(\left\lceil \frac{c}{r} \right\rceil\right) - 2(r - n_{\text{PAD}}) - 5. \tag{3.7}$$

Depending on $\mathbf{b}(c)$ *we have two regimes of* SPONGE-COLL*:*

- *If* $\ell < \mathbf{b}(c)$*, then the number of queries to* $\mathbf{f}$ *is at most* $3k_{\text{coll}} \cdot \frac{3\ell + 2(r-n_{\text{PAD}})+6}{4(r-n_{\text{PAD}})} 2^{n'/3}$*, where* $n' = \frac{r-n_{\text{PAD}}+1}{r-n_{\text{PAD}}}\ell + 2(r - n_{\text{PAD}} + 1) + 6\frac{r-n_{\text{PAD}}+1}{r-n_{\text{PAD}}}$.
- *If* $\ell \geq \mathbf{b}(c)$*, then the number of queries to* $\mathbf{f}$ *is at most* $3k_{\text{coll}} \cdot \frac{c + 2(r-n_{\text{PAD}})+6}{4(r-n_{\text{PAD}})} 2^{n'/3}$*, where* $n' = \frac{r-n_{\text{PAD}}+1}{r-n_{\text{PAD}}}c + 2(r - n_{\text{PAD}} + 1) + 6\frac{r-n_{\text{PAD}}+1}{r-n_{\text{PAD}}}$.

Proof idea. We prove our statement by constructing a collision-finding algorithm SPONGE-COLL and proving how many queries it makes. The main ingredient of SPONGE-COLL is Ambainis' collision-finding algorithm COLL, as stated in Theorem 3.7. Both query complexity and probability of error of SPONGE-COLL are inherited from this subroutine. The query complexity is derived from the size of the domain of the function g : $\mathcal{X} \to \mathcal{Y}$ in which we are looking for collisions.

The two sources of collisions in $\text{SPONGE}_{\mathbf{f}}$ are inner-collisions and output-collisions, we defined them in detail in Section 2.5.2.3. We distinguish two regimes, divided by $\mathbf{b}(c)$. Note that for long outputs, $\ell \geq \mathbf{b}(c)$ it is easier to find an inner-collision and use it to construct an output collision; In the other case, $\ell < \mathbf{b}(c)$, it is more reasonable to go directly for an output collision. These two cases are the two choices of g: the inner part of the absorbing phase and a regular sponge respectively.

The assumption one makes in order to use Ambainis' algorithm is that there exists at least one collision in g. In case of a random function g$'$: $\mathcal{X} \to \mathcal{Y}$ it can be argued that such a collision exists for $|\mathcal{X}| = |\mathcal{Y}|^{1/2}$ [Zha15a]. In our case this function is a random sponge, i.e. SPONGE constructed using a random internal function.

The main task that needs to be solved to argue that there is a collision in g, is to construct the domain $\mathcal{X}$ of the right size and a structure, which allows to deal with the padding function in SPONGE. Then we define the algorithm SPONGE-COLL, using Ambainis' algorithm as a black box. In order to show that there exists at least one collision in sponge with the domain $\mathcal{X}$ and codomain defined by the parameters of SPONGE (rate r, capacity c, and output length ℓ), we use the fact that the internal function is random, and we calculate the probability of getting a collision if the whole domain of g was sampled. We assume that every new input to f yields an independently distributed output (the distribution depends on whether our internal function is a random function or a random permutation).

We prove the assumption of g having at least one collision in the two cases ($\ell \geq \mathbf{b}(c)$ and $\ell < \mathbf{b}(c)$) using Lemma 3.11 and Lemma 3.12. The last ingredient of the proof is calculating the query complexity—in terms of queries to f—of algorithm Sponge-Coll. □

We follow the plan sketched in the proof idea above. First we define the domain of g. Remember that in this section, n_{PAD} denotes the minimal number of bits appended by PAD, and we assumed that $0 < n_{\text{PAD}} < r$. Moreover we require that the padding rule is injective, does not output messages ending with 0, and works by just appending bits to the message.

We construct domain $\mathcal{X}$ in such a way that querying its elements to Sponge in the right order makes just a single fresh query to f in the absorbing phase. We define a set of inputs to Sponge and denote it by $\mathcal{M}$. The intuition is that short messages in $\mathcal{M}$ are prefixes of longer messages in $\mathcal{M}$. We need to be careful though to include the padding function from Sponge.

Definition 3.9. *Let us define* $\mathcal{M}(1) := \bigcup_{i=0}^{r-n_{\text{PAD}}} \{0,1\}^i$. *For some parameter* n' *to be specified in algorithm* Sponge-Coll *below and* $k := \left\lfloor \frac{n'}{2(r-n_{\text{PAD}}+1)} \right\rfloor$, *we define*

$$\mathcal{M}(j) := \{\text{PAD}(m_1)\| \cdots \|\text{PAD}(m_{j-1})\|m_j : m_1, \ldots, m_j \in \mathcal{M}(1)\} \text{ and} \tag{3.8}$$

$$\mathcal{M}_{n'} := \bigcup_{j=1}^{k} \mathcal{M}(j). \tag{3.9}$$

The domain of g is going to be defined as $\mathcal{X} = \mathcal{M}_{n'}$ for an appropriately chosen n'. We omit the subscript n' when it is clear from the context or not necessary for the argument.

The size of the set defined above can be easily bounded. First of all we have

$$|\mathcal{M}_{n'}| = \frac{2^{r-n_{\text{PAD}}+1} - 1}{2(2^{r-n_{\text{PAD}}} - 1)} \left((2^{r-n_{\text{PAD}}+1} - 1)^k - 1 \right), \tag{3.10}$$

which can be bounded as follows:

$$|\mathcal{M}_{n'}| \leq \frac{3}{2} 2^{(r-n_{\text{PAD}}+1)k} \leq \frac{3}{2} 2^{n'/2}, \tag{3.11}$$

$$|\mathcal{M}_{n'}| \geq \frac{1}{2} 2^{(r-n_{\text{PAD}})k} \geq \frac{1}{2} \frac{1}{2^{k+r-n_{\text{PAD}}}} 2^{n'/2}. \tag{3.12}$$

The special feature of the set $\mathcal{M}$ is that for every $m' \in \mathcal{M} : |m'| > r$ there exists $m \in \mathcal{M} : |\text{PAD}(m)| < |m'|$ such that $\text{PAD}(m)$ is a prefix of m'. Moreover we can choose this m so that $|\text{PAD}(m)| = r\lfloor \frac{|m'|}{r} \rfloor$, this statement is proven in the lemma below.

Lemma 3.10. *$\mathcal{M}$ is defined as in Definition 3.9, $n_{\text{PAD}} < r$, then*

$$\forall m' \in \mathcal{M} : |m'| > r \; \exists m \in \mathcal{M} : \\ |\text{PAD}(m)| = r \lfloor |m'|/r \rfloor \wedge \text{PAD}(m) \text{ is a prefix of } m'. \tag{3.13}$$

Proof. Note that $|m'| > r$ implies that the number of blocks of m' is $k > 1$ and in the definition of $\mathcal{M}$ there are at least two subsets $\mathcal{M}(j)$ constituting the set. From the definition of $\mathcal{M}$ and the fact that $m' \in \mathcal{M}$ we have that $m' = \text{PAD}(m_1)\|\cdots\|\text{PAD}(m_n)\|a$, where $n = \lfloor \frac{|m'|}{r} \rfloor$ and $a \in \mathcal{M}(1)$. Note that $\text{PAD}(m_1)\|\cdots\|\text{PAD}(m_n) = \text{PAD}(m_1)\|\cdots\|\text{PAD}(m_{n-1})\|b\|\widetilde{\text{PAD}}(b) = \text{PAD}(m'')$, where $m'' = \text{PAD}(m_1)\|\cdots\|\text{PAD}(m_{n-1})\|b$ and $b \in \mathcal{M}(1)$. From the assumptions we know that $n \geq 1$, together with $|b| < r - n_{pad}$ we have that $m'' \in \mathcal{M}$. Now we see that $|\text{PAD}(m'')| = |\text{PAD}(m_1)\|\cdots\|\text{PAD}(m_n)| = r \left\lfloor \frac{|m'|}{r} \right\rfloor$ and $\text{PAD}(m'')$ is a prefix of m'. □

Let us consider the following order relation on the set $\mathcal{M}$: for $m, m' \in \mathcal{M}$ we say that $m \prec m'$ if $|m| < |m'|$ or in case that $|m| = |m'|$, if m is prior to m' in lexicographical order. Then a corollary of Lemma 3.10 is that if all elements $m_1 \prec m_2 \prec \ldots \prec m_{|\mathcal{M}|}$ of $\mathcal{M}$ are queried in this order then each query to Absorb $\circ$ PAD makes exactly one fresh query to $\mathbf{f}$.

Let us provide an example of $\mathcal{M}$ for a simple padding that appends a single 1 bit followed by the minimal number of 0's and another 1 such that the total length of the message is a multiple of r. In this case $n_{\text{PAD}} = 2$ and $\mathcal{M}(1)$ is the set of all strings of length up to $r - 2$ bits. Padding $m \in \mathcal{M}(1)$ prolongs it to r-bits, so $\mathcal{M}(2)$ is the set of all strings of length at least r and up to $2r - 2$ with the exception of strings starting with the string 0^r. The whole domain $\mathcal{M}$ is the set of strings composed of up to k blocks.

Before we proceed with the proof we state two important lemmas providing bounds on the probability of collisions of Sponge in $\mathcal{M}$.

Lemma 3.11. *We set $M := |\mathcal{M}_{n'}|\,(j_{\max}+1)$, where $j_{\max} = 0$ or $j_{\max} = \left\lfloor \frac{\ell}{r} \right\rfloor$ depending on whether $\ell \geq \mathbf{b}(c)$ or $\ell < \mathbf{b}(c)$ respectively. Below $\mathcal{S} = \{0,1\}^r \times \{0,1\}^c$ and $\mathcal{I}_{\text{per}}(\mathcal{S})$ denotes the set of bijections $\mathbf{f} : \mathcal{S} \to \mathcal{S}$.*

The probability that Sponge$_{\mathbf{f}}[\text{PAD}, \{0,1\}^r, \{0,1\}^c, \ell]$ with a random internal function has an inner-collision in $\mathcal{M}_{n'}$ defined in Definition 3.9 is bounded as follows

(*i*) *when instantiated with a random function:*

$$\underset{\mathbf{f} \xleftarrow{\$} \mathcal{S}^{\mathcal{S}}}{\mathbb{P}} \left[\text{Sponge}_{\mathbf{f}} \text{ has an inner-collision in } \mathcal{M}_{n'}\right] \leq \frac{M(M+1)}{2^{c+1}}, \tag{3.14}$$

$$\underset{\mathbf{f} \xleftarrow{\$} \mathcal{S}^{\mathcal{S}}}{\mathbb{P}} \left[\text{Sponge}_{\mathbf{f}} \text{ has an inner-collision in } \mathcal{M}_{n'}\right] \geq \frac{M(M+1)}{2^{c+2}}, \tag{3.15}$$

(*ii*) *when instantiated with a random permutation:*

$$\underset{\mathbf{f} \xleftarrow{\$} \mathcal{I}_{\text{per}}(\mathcal{S})}{\mathbb{P}} \left[\text{SPONGE}_{\mathbf{f}} \text{ has an inner-collision in } \mathcal{M}_{n'}\right] \leq \frac{2^r M(M+1) - (M-1)M}{2^{r+c+1} - 2M}, \tag{3.16}$$

$$\underset{\mathbf{f} \xleftarrow{\$} \mathcal{I}_{\text{per}}(\mathcal{S})}{\mathbb{P}} \left[\text{SPONGE}_{\mathbf{f}} \text{ has an inner-collision in } \mathcal{M}_{n'}\right] \geq \frac{M(M+1)}{2^{c+2}} - \frac{(M-1)M}{2^{r+c+2}}. \tag{3.17}$$

Proof. The main part of the proof are algorithms INNER-COLL and OUTPUT-COLL defined in Algorithm 3.1. The two algorithms are just loops over the whole set $\mathcal{M}$ evaluating SPONGE$_{\mathbf{f}}$ on all elements of $\mathcal{M}$. Set Σ is the set of all traversed states—that is all states output by $\mathbf{f}$ in all evaluations of the internal function. By $\widehat{\Sigma}$ we denote the set of inner parts of elements of Σ. Set Ω is the set of all traversed outputs of length ℓ. Throughout the proof we treat $\mathcal{M}$, r, c, and ℓ as global constants. In this proof we only use INNER-COLL, the second algorithm is used in the proof of Lemma 3.12.

Algorithm 3.1 $\boxed{\text{INNER-COLL}}$, $\boxed{\text{OUTPUT-COLL}}$

```
    Output: $b \in \{0,1\}$
 1: $\Sigma := \emptyset, \Omega := \emptyset$
 2: if $\ell < \mathrm{b}(c)$ then $j_{\max} := \lfloor \frac{\ell}{r} \rfloor$
 3: else $j_{\max} := 0$
 4: for $i = 1, \ldots, |\mathcal{M}|$ do
 5:     for $j = 0, \ldots, j_{\max}$ do
 6:         $s := \underbrace{\mathbf{f} \circ \mathbf{f} \circ \cdots \circ \mathbf{f}}_{j \text{ times}} \circ \text{ABSORB}_{\mathbf{f}} \circ \text{PAD}(m_i)$    ▷ $m_1 \prec \cdots \prec m_i \prec \cdots \prec m_{|\mathcal{M}|}$
 7:         if $\hat{s} \in \widehat{\Sigma} \cup \{0^c\}$ then    ▷ Inner-collision
 8:             $\boxed{\textbf{return } \text{b=1}}$    ▷ set random variable $C^i_j = 1$
 9:         Append $s$ to $\Sigma$
10:     $z := \text{SPONGE}_{\mathbf{f}}(\ell, m_i)$
11:     if $z \in \Omega$ then    ▷ Output-collision
12:         $\boxed{\textbf{return } \text{b=1}}$
13:     Append $z$ to $\Omega$
14: return $b = 0$
```

With the above algorithm defined we connect its output to the existence of collisions in SPONGE:

$$\mathbb{P}\left[\text{SPONGE}_{\mathbf{f}} \text{ has an inner-collision in } \mathcal{M}\right] = \mathbb{P}\left[1 \leftarrow \text{INNER-COLL}\right]. \tag{3.18}$$

The probability on the left- and the right-hand sides are over sampling of a random f, there is no internal randomness of Inner-Coll, moreover algorithm Inner-Coll is just a loop over all states traversed when evaluating the sponge on all elements of $\mathcal{M}$, i.e. Sponge$(\mathcal{M})$; For this reason the events are a different way of stating the same fact: when evaluating Sponge$(\mathcal{M})$ at least one inner-state is 0^c or some other traversed inner-state. The only difference is the notion of time introduced by Inner-Coll, it does not change the event but makes our task of evaluating the probability easier. The two bounds we want to calculate in this proof are an upper and a lower bound on $\mathbb{P}\left[1 \leftarrow \text{Inner-Coll}\right]$. Note however that $\mathbb{P}\left[0 \leftarrow \text{Inner-Coll}\right] = 1 - \mathbb{P}\left[1 \leftarrow \text{Inner-Coll}\right]$, the two bounds are lower bounds on $\mathbb{P}\left[0 \leftarrow \text{Inner-Coll}\right]$ and $\mathbb{P}\left[1 \leftarrow \text{Inner-Coll}\right]$.

For now we do not specify the distribution of f and expand the event $b \leftarrow$ Inner-Coll in terms of more concise events. For all $1 \leq i \leq |\mathcal{M}|$ and $0 \leq j \leq j_{\max}$ we define a random variable $C^i_j \in \{0,1\}$. The variable $C^i_j = 1$ if in Algorithm 3.1 for indices i and j condition $\hat{s} \in \widehat{\Sigma} \cup \{0^c\}$ is fulfilled. Note that if $1 \leftarrow$ Inner-Coll, it must be that some $C^i_j = 1$ while all previous $C^{i'}_{j'}$ are 0. By previous events we mean such that $(i',j') \prec (i,j)$, which denotes indices (i',j') that appear prior to (i,j) in Inner-Coll's loops, that is $i' < i$ or $j' < j$ if $i' = i$. Note that the events are disjoint and we have equality in the second equality below:

$$\mathbb{P}[1 \leftarrow \text{Inner-Coll}] = \mathbb{P}\left[\bigvee_{(i,j)\in[|\mathcal{M}|]\times\{0,\ldots,j_{\max}\}} \left(1 \leftarrow C^i_j \bigwedge_{(i',j')\prec(i,j)} 0 \leftarrow C^{i'}_{j'}\right)\right] \tag{3.19}$$

$$= \sum_{(i,j)\in[|\mathcal{M}|]\times\{0,\ldots,j_{\max}\}} \mathbb{P}\left[1 \leftarrow C^i_j \bigwedge_{(i',j')\prec(i,j)} 0 \leftarrow C^{i'}_{j'}\right]. \tag{3.20}$$

Note that we mention explicitly only the events that are no later than (i,j). Given that $1 \leftarrow C^i_j$ what happens after that is of no consequence for the probability of $1 \leftarrow$ Inner-Coll so we leave it implicit.

For the second probability $\mathbb{P}\left[0 \leftarrow \text{Inner-Coll}\right]$ we have

$$\mathbb{P}[0 \leftarrow \text{Inner-Coll}] = \mathbb{P}\left[\bigwedge_{(i,j)\in[M]\times\{0,\ldots,j_{\max}\}} 0 \leftarrow C^i_j\right]. \tag{3.21}$$

Due to the padding rule (as defined in Section 2.5.2) and the construction of $\mathcal{M}$, for every pair (i,j) in Line 6 of Algorithm 3.1 only one new edge is created in the sponge graph; All but the last query to f had already been done in previous steps of Inner-Coll. Note that we can interpret the computation of the j-th output block of message $p = \text{pad}(m_i)$ as an evaluation of Absorb on message $p\|0^{(j-1)r}$, a message that has not been processed before (note that $0^r \notin \mathcal{M}(1)$).

Next we evaluate the probability of $b \leftarrow C^i_j$:

$$\begin{aligned}&\mathbb{P}\left[b \leftarrow C^i_j \bigwedge_{(i',j')\prec(i,j)} 0 \leftarrow C^{i'}_{j'}\right]\\ &= \mathbb{P}\left[b \leftarrow C^i_j \mid \bigwedge_{(i',j')\prec(i,j)} 0 \leftarrow C^{i'}_{j'}\right] \mathbb{P}\left[\bigwedge_{(i',j')\prec(i,j)} 0 \leftarrow C^{i'}_{j'}\right]. \qquad (3.22)\end{aligned}$$

The correct interpretation of the condition $\bigwedge_{(i',j')\prec(i,j)} 0 \leftarrow C^{i'}_{j'}$ is that all inner states output by $\mathbf{f}$ in the evaluations of Sponge in Inner-Coll up to (i,j) are distinct and different from 0^c. So the condition only puts restrictions on $\mathbf{f}$ and possibly changes the distribution of its outputs.

To calculate the probability that the condition in Equation (3.22) holds we split it into a product of conditional probabilities:

$$\mathbb{P}\left[\bigwedge_{(i',j')\prec(i,j)} 0 \leftarrow C^{i'}_{j'}\right] = \prod_{(i',j')\prec(i,j)} \mathbb{P}\left[0 \leftarrow C^{i'}_{j'} \mid \bigwedge_{(i'',j'')\prec(i',j')} 0 \leftarrow C^{i''}_{j''}\right]. \qquad (3.23)$$

Let us give an example of the correct interpretation of the above conditions, that we already explained above. For $j_{\max} = 0$ and $i = 2$ we have:

$$\begin{aligned}&\mathbb{P}\left[0 \leftarrow C^2_0 \mid 0 \leftarrow C^1_0\right]\mathbb{P}\left[0 \leftarrow C^1_0\right]\\ &= \mathbb{P}\left[\mathbf{f}(p_2\|0^c) = s_2 : \hat{s}_2 \notin \{0^c, \hat{s}_1\} \mid \mathbf{f}(p_1\|0^c) = s_1, \hat{s}_1 \neq 0^c\right]\\ &\cdot \mathbb{P}\left[\mathbf{f}(p_1\|0^c) = s_1 : \hat{s}_1 \neq 0^c\right], \qquad (3.24)\end{aligned}$$

where $p_i := \textsc{pad}(m_i)$. Coming back to how we defined $\mathcal{M}$, with the above conditions for each (i,j) we not only make just one additional query to $\mathbf{f}$ but we make one additional fresh query to $\mathbf{f}$.

Considering the above interpretation, $\mathbb{P}\left[b \leftarrow C^i_j \mid \bigwedge_{(i',j')\prec(i,j)} 0 \leftarrow C^{i'}_{j'}\right]$ is just the probability of an inner-collision, or lack of it, given that all the traversed inner-states are distinct and not 0^c. Note that this is quite a similar situation to the discussion in Chapter 4, where we analyze sets of queries to Sponge, from the perspective of states being unique.

Now we introduce the two cases: $\mathbf{f} \xleftarrow{\$} \mathcal{S}^{\mathcal{S}}$ and $\mathbf{f} \xleftarrow{\$} \mathcal{I}_{\text{per}}(\mathcal{S})$. By $\overset{\text{Fun}}{=}$ and $\overset{\text{Per}}{=}$ we denote the relation (in this case equality) valid when $\mathbf{f}$ is a random function or permutation respectively. We start with bounds on the probability of there being collisions:

$$\mathbb{P}\left[1 \leftarrow \textsc{Inner-Coll}\right] \leq \sum_{(i,j)\in[|\mathcal{M}|]\times\{0,\dots,j_{\max}\}} \mathbb{P}\left[1 \leftarrow C^i_j \mid \bigwedge_{(i',j')\prec(i,j)} 0 \leftarrow C^{i'}_{j'}\right] \qquad (3.25)$$

$$\overset{\text{Fun}}{=} \sum_{k=1}^{|\mathcal{M}|(j_{\max}+1)} \frac{k}{2^c} = \frac{|\mathcal{M}|(j_{\max}+1)(|\mathcal{M}|(j_{\max}+1)+1)}{2^{c+1}}. \qquad (3.26)$$

$$\overset{\text{Per}}{=} \sum_{k=1}^{|\mathcal{M}|(j_{\max}+1)} \frac{(2^r-1)(k-1)+2^r}{2^{r+c}-(k-1)} \le \sum_{k=1}^{|\mathcal{M}|(j_{\max}+1)} \frac{2^r k-(k-1)}{2^{r+c}-|\mathcal{M}|(j_{\max}+1)} \tag{3.27}$$

$$\le \frac{|\mathcal{M}|(j_{\max}+1)\left(2^r(|\mathcal{M}|(j_{\max}+1)+1)-(|\mathcal{M}|(j_{\max}+1)-1)\right)}{2^{r+c+1}-2|\mathcal{M}|(j_{\max}+1)}\,. \tag{3.28}$$

In the first inequality we bound the probability of the condition by 1. The probability that a uniformly sampled c-bit string matches one of the $k-1$ previous right states is $\frac{k}{2^c}$ in case f is a random function. For permutations we have 2^r states with inner part 0^c and 2^r-1 states for every traversed inner-state.

To upper bound the probability of no collisions $\mathbb{P}\left[0 \leftarrow C_{j'}^{i'} \mid \bigwedge_{(i'',j'')\prec(i',j')} 0 \leftarrow C_{j''}^{i''}\right]$ we first calculate the conditional probability of a single $0 \leftarrow C_j^i$ for random functions $\overset{\text{Fun}}{=} \left(1-\frac{k}{2^n}\right)$ and random permutations $\overset{\text{Per}}{=} \left(1-\frac{(2^r-1)(k-1)+2^r}{2^{r+c}-(k-1)}\right)$. The total conditional probability is

$$\mathbb{P}\left[0 \leftarrow \textsc{Inner-Coll}\right] = \prod_{(i,j)\in[|\mathcal{M}|]\times\{0,\dots,j_{\max}\}} \mathbb{P}\left[0 \leftarrow C_j^i \mid \bigwedge_{(i',j')\prec(i,j)} 0 \leftarrow C_{j'}^{i'}\right] \tag{3.29}$$

$$\overset{\text{Fun}}{=} \prod_{k=1}^{|\mathcal{M}|(j_{\max}+1)} \left(1-\frac{k}{2^c}\right). \tag{3.30}$$

$$\overset{\text{Per}}{=} \prod_{k=1}^{|\mathcal{M}|(j_{\max}+1)} \left(1-\frac{(2^r-1)(k-1)+2^r}{2^{r+c}-(k-1)}\right). \tag{3.31}$$

To calculate a more useful bound we use two important facts:

$$\forall x \in \mathbb{R} : 1-x \le e^{-x}, \tag{3.32}$$

$$\forall x \in [0,1] : 1-e^{-x} \ge \frac{x}{2}. \tag{3.33}$$

With these we have $\prod_k(1-x(k)) \le \prod_k \exp(-x(k)) = \exp(-\sum_k x(k))$. For the final bound we can apply the second inequality from above: $1-\exp(-\sum_k x(k)) \ge \frac{1}{2}\sum_k x(k)$.

In the case of a random function we use the above statements to derive that

$$\begin{aligned}\mathbb{P}[1 \leftarrow \textsc{Inner-Coll}] &= 1-\mathbb{P}[0 \leftarrow \textsc{Inner-Coll}] \\ &\overset{\text{Fun}}{\ge} \frac{|\mathcal{M}|(j_{\max}+1)(|\mathcal{M}|(j_{\max}+1)+1)}{2^{c+2}}.\end{aligned} \tag{3.34}$$

In the case of a random permutation we use the same techniques as above to obtain

$$\mathbb{P}[1 \leftarrow \textsc{Inner-Coll}] \overset{\text{Per}}{\ge} \frac{1}{2} \sum_{k=1}^{|\mathcal{M}|(j_{\max}+1)} \frac{2^r k-(k-1)}{2^{r+c}-(k-1)} \ge \sum_{k=1}^{|\mathcal{M}|(j_{\max}+1)} \frac{2^r k-(k-1)}{2^{r+c+1}}$$

$$= \frac{|\mathcal{M}|\,(j_{\max}+1)(|\mathcal{M}|\,(j_{\max}+1)+1)}{2^{c+2}} - \frac{(|\mathcal{M}|\,(j_{\max}+1)-1)\,|\mathcal{M}|\,(j_{\max}+1)}{2^{r+c+2}}. \tag{3.35}$$

In both lower bounds we assume that the sum is smaller than 1, as required in Inequality (3.33). □

The second lemma states lower bounds on the probability of output-collisions.

Lemma 3.12. *The probability that* $\textsc{Sponge}_{\mathbf{f}}[\textsc{pad}, \{0,1\}^r, \{0,1\}^c, \ell]$ *for a random internal function has an output-collision in* $\mathcal{M}_{n'}$*, with size denoted by* $M := |\mathcal{M}_{n'}|$*, defined in Definition 3.9, is bounded as follows*

(*i*) *when instantiated with a random function:*

$$\begin{aligned}&\underset{\mathbf{f}\xleftarrow{\$}\mathcal{S}^{\mathcal{S}}}{\mathbb{P}}\left[\textsc{Sponge}_{\mathbf{f}} \text{ has a collision in } \mathcal{M}_{n'}\right]\\ &\geq \frac{(M-1)M}{2^{\ell+2}}\left(1-\frac{M(j_{\max}+1)(M(j_{\max}+1)+1)}{2^{c+1}}\right),\end{aligned} \tag{3.36}$$

(*ii*) *when instantiated with a random permutation:*

$$\begin{aligned}&\underset{\mathbf{f}\xleftarrow{\$}\mathcal{I}_{\text{per}}(\mathcal{S})}{\mathbb{P}}\left[\textsc{Sponge}_{\mathbf{f}} \text{ has a collision in } \mathcal{M}_{n'}\right]\\ &\geq \frac{(M-1)M}{2^{\ell+2}}\left(1-\frac{2^r M(j_{\max}+1)(M(j_{\max}+1)+1)}{2^{r+c+1}-2M(j_{\max}+1)}\right.\\ &\left.+\frac{(M(j_{\max}+1)-1)M(j_{\max}+1)}{2^{r+c+1}-2M(j_{\max}+1)}\right),\end{aligned} \tag{3.37}$$

where $j_{\max} = 0$ *or* $j_{\max} = \lfloor\frac{\ell}{r}\rfloor$ *depending on whether* $\ell \geq \mathbf{b}(c)$ *or* $\ell < \mathbf{b}(c)$ *respectively. Above* $\mathcal{S} = \{0,1\}^r \times \{0,1\}^c$*,* $\mathcal{I}_{\text{per}}(\mathcal{S})$ *denotes the set of bijections* $\mathbf{f} : \mathcal{S} \to \mathcal{S}$*.*

Proof. The intuition for the proof is as follows: Having traversed only distinct inner-states at any point of the loop implies that the output of Sponge is uniformly random (Theorem 1 in [Ber+07]), which allows us to use birthday bounds. In more detail we have:

$$\begin{aligned}&\mathbb{P}\left[\textsc{Sponge} \text{ has a collision in } \mathcal{M}\right]\\ &= \mathbb{P}\left[\textsc{Sponge} \text{ has a collision in } \mathcal{M} \wedge \textsc{Sponge} \text{ has an inner-collision in } \mathcal{M}\right]\\ &+ \mathbb{P}\left[\textsc{Sponge} \text{ has a collision in } \mathcal{M} \wedge \textsc{Sponge} \text{ has no inner-collisions in } \mathcal{M}\right]\\ &\geq \mathbb{P}\left[\textsc{Sponge} \text{ has a collision in } \mathcal{M} \mid \textsc{Sponge} \text{ has no inner-collisions in } \mathcal{M}\right]\\ &\cdot \mathbb{P}\left[\textsc{Sponge} \text{ has no inner-collisions in } \mathcal{M}\right]\end{aligned} \tag{3.38}$$

For bounding $\mathbb{P}\left[\text{Sponge has no inner-collisions in } \mathcal{M}\right]$ we use Lemma 3.11.

Similarly to the previous lemma we have $\mathbb{P}\left[\text{Sponge has a collision in } \mathcal{M}\right] = \mathbb{P}\left[1 \leftarrow \text{Output-Coll}\right]$. Conditioning on "Sponge has no inner-collisions in $\mathcal{M}$" is also done through the algorithm Inner-Coll and more specifically through the random variables C^i_j defined in the proof of Lemma 3.11. As we already explained in Equation (3.24) and in the comments above it, we interpret $\bigwedge_{i,j} 0 \leftarrow C^i_j$ by the appropriate constraints on $\mathbf{f}$.

Conditioning on "Sponge has no inner-collisions in $\mathcal{M}$" changes the overall distribution of $\mathbf{f}$. For random functions though, outputs are distributed just uniformly at random. The bound that we get in the case of random functions is

$$\mathbb{P}\left[1 \leftarrow \text{Output-Coll} \mid \bigwedge_{(i,j)\in[M]\times\{0,\ldots,j_{\max}\}} 0 \leftarrow C^i_j\right] \overset{\text{Fun}}{=} 1 - \prod_{i=1}^{|\mathcal{M}|}\left(1 - \frac{i-1}{2^\ell}\right) \tag{3.39}$$

$$\geq \frac{(|\mathcal{M}|-1)\,|\mathcal{M}|}{2^{\ell+2}} \tag{3.40}$$

where we use the same techniques as in the proof of Lemma 3.11. The bound on inner-collisions comes from Equation (3.26).

For random permutations, outer and inner states are not sampled independently. To calculate the probability of output-collisions we need to analyze the conditional distribution of the outer part of the outputs of $\mathbf{f}$. We can think of sampling outputs of a random permutation $\mathbf{f} : \mathcal{A}\times\mathcal{C} \to \mathcal{A}\times\mathcal{C}$ as a two-step process: first sampling the inner part of the output and then the outer part. By $\mathcal{D}$ we denote the set of outputs of previous queries. In the language of the sponge graph, $\mathcal{D}$ is the set of nodes with incoming edges. By $\overline{\mathcal{D}}(\hat{s})$ we denote the set of nodes in the supernode $\hat{s}$ with an incoming edge. We first sample uniformly from $\mathcal{A} \times \mathcal{C} \setminus \mathcal{D}$ but then discard the outer state. The value of the inner state $\hat{s}$ is then effectively sampled from $\mathcal{C}$ with weights $\frac{|\mathcal{A}\setminus\overline{\mathcal{D}}(\hat{s})|}{|\mathcal{A}\times\mathcal{C}\setminus\mathcal{D}|}$. To sample the outer state we just sample uniformly from $\mathcal{A} \setminus \overline{\mathcal{D}}(\hat{s})$.

Given the above sampling, note that the conditional probability is such that the inner part of the state of the outcome is distinct from 0^c and any other output inner-states. So for the output inner-state $\hat{s}$ there are no incoming edges in $\overline{\mathcal{D}}(\hat{s})$ and the outer state is sampled uniformly at random. Hence, for random permutations the probability of output-collisions, given no inner-collisions is

$$\mathbb{P}\left[1 \leftarrow \text{Output-Coll} \mid \bigwedge_{(i,j)\in[M]\times\{0,\ldots,j_{\max}\}} 0 \leftarrow C^i_j\right] \overset{\text{Per}}{\geq} \frac{(|\mathcal{M}|-1)\,|\mathcal{M}|}{2^{\ell+2}} \tag{3.41}$$

and the bound on inner-collisions comes from Equation (3.28). □

With these bounds we can proceed with the proof of the main theorem of this section.

Proof of Theorem 3.8. First we state the quantum collision-finding algorithm:

Algorithm 3.2 Sponge-Coll

```
    Output: $m \neq m'$ such that $\textsc{Sponge}_\mathbf{f}(\ell, m) = \textsc{Sponge}_\mathbf{f}(\ell, m')$ or "fail"
 1: if $\ell < \mathrm{b}(c)$ then
 2:     Set $n' := \frac{r-n_{\text{PAD}}+1}{r-n_{\text{PAD}}}\ell + 2(r - n_{\text{PAD}}) + 6\frac{r-n_{\text{PAD}}+1}{r-n_{\text{PAD}}}$, define $\mathrm{g} : \mathcal{M}_{n'} \to \{0,1\}^\ell$ as $\mathrm{g} := \textsc{Sponge}_\mathbf{f}$
 3: else if $\ell \geq \mathrm{b}(c)$ then
 4:     Set $n' := \frac{r-n_{\text{PAD}}+1}{r-n_{\text{PAD}}}c + 2(r - n_{\text{PAD}}) + 6\frac{r-n_{\text{PAD}}+1}{r-n_{\text{PAD}}}$, define $\mathrm{g} : \mathcal{M}_{n'} \to \{0,1\}^c$ as $\mathrm{g} := \widehat{\textsc{Absorb}} \circ \textsc{pad}$
 5: Run algorithm Coll from Theorem 3.7 on g, making $k_{\text{coll}} \cdot |\mathcal{M}_{n'}|^{2/3}$ queries.
 6: if $(m, m') \leftarrow \textsc{Coll}$ then
 7:     if $\ell < \mathrm{b}(c)$ then
 8:         return $(m, m')$
 9:     else if $\ell \geq \mathrm{b}(c)$ then
10:         return an output collision constructed from $(m, m')$ according to the proof of Claim 2.27
```

The probability of success of Algorithm 3.2 comes from the probability of success of Coll, which in turn is guaranteed to be high if there is a collision in the domain of g, Theorem 3.7 does not make any assumptions about other properties of g. If Ambainis' algorithm succeeds in outputting a collision then Sponge-Coll outputs a collision:

$$\mathbb{P}\left[\text{collision} \leftarrow \textsc{sponge-coll}\right] \geq \mathbb{P}\left[\text{collision} \leftarrow \textsc{Coll}\right] \tag{3.42}$$

$$= \mathbb{P}\left[\text{collision} \leftarrow \textsc{Coll} \wedge \mathrm{g} \text{ has a collision}\right]$$

$$+ \mathbb{P}\left[\text{collision} \leftarrow \textsc{Coll} \wedge \mathrm{g} \text{ has no collisions}\right] \tag{3.43}$$

$$\geq \mathbb{P}\left[\text{collision} \leftarrow \textsc{Coll} \mid \mathrm{g} \text{ has a collision}\right] \mathbb{P}\left[\mathrm{g} \text{ has a collision}\right] \tag{3.44}$$

$$\geq \frac{15}{16} \cdot \mathbb{P}\left[\mathrm{g} \text{ has a collision}\right]. \tag{3.45}$$

The main ingredient of bounding the probability of success of sponge-coll is bounding the probability that g has collisions.

The two regimes are distinguished by $\mathrm{b}(c) = \frac{r-n_{\text{PAD}}}{r-n_{\text{PAD}}+2}c - 2\log\lceil c/r \rceil - 2(r - n_{\text{PAD}}) - 5$. We choose this value so that in the case of short outputs ($\ell < \mathrm{b}(c)$) $2^{n'-c}$ is small.

Let us first deal with random functions, we start with the case $\ell < \mathrm{b}(c)$, where we look for output-collisions. To find the lower bound on $\mathbb{P}\left[\textsc{Sponge}_\mathbf{f} \text{ has a collision}\right]$ we focus on Equation (3.36). We use Equations (3.11) and (3.12) to get a bound on $|\mathcal{M}|$ in terms of n'. Given that

$n' := \frac{r-n_{\text{PAD}}+1}{r-n_{\text{PAD}}}\ell + 2(r - n_{\text{PAD}} + 1) + 6\frac{r-n_{\text{PAD}}+1}{r-n_{\text{PAD}}}$, we have an expression that crucially depends on ℓ and c. Our assumption $\ell < \mathbf{b}(c)$ guarantees that all terms are bounded by constants from below.

The same line of reasoning can be applied to the case $\ell < \mathbf{b}(c)$ with random permutations. Now instead of Equation (3.36) we focus on Equation (3.37).

In the case of $\ell \geq \mathbf{b}(c)$ the crucial bound is a lower bound on $\mathbb{P}\left[\text{SPONGE}_{\mathbf{f}} \text{ has an inner collision}\right]$. For the bounds we use Equation (3.34) in the case of functions and (3.35) in the case of permutations. We again appropriately bound $|\mathcal{M}|$. There is no need, though, to use $\ell \geq \mathbf{b}(c)$, as all the bounds on inner collisions do not depend on ℓ.

The number of queries made by SPONGE-COLL to SPONGE is bounded by $\frac{3}{2}2^{n'/3}$. A single evaluation of SPONGE requires at most

$$k + \lfloor \ell/r \rfloor = \left\lfloor \frac{n'}{2(r - n_{\text{PAD}} + 1)} \right\rfloor + \lfloor \ell/r \rfloor \tag{3.46}$$

$$\leq \begin{cases} \frac{1}{2(r-n_{\text{PAD}})}\ell + 1 + \frac{3}{r-n_{\text{PAD}}} + \frac{\ell}{r} & \text{if } \ell < \mathbf{b}(c) \\ \frac{1}{2(r-n_{\text{PAD}})}c + 1 + \frac{3}{r-n_{\text{PAD}}} & \text{if } \ell \geq \mathbf{b}(c) \end{cases} \tag{3.47}$$

queries to the internal function $\mathbf{f}$.

By bounding Equation (3.46) we get the following final parameters of the algorithm: In the case $\ell < \mathbf{b}(c)$ one query to SPONGE requires at most $\frac{3\ell+2(r-n_{\text{PAD}})+6}{2(r-n_{\text{PAD}})}$ queries to the internal function $\mathbf{f}$. Therefore, a collision in SPONGE can be found with at most $3k_{\text{coll}} \cdot \frac{3\ell+2(r-n_{\text{PAD}})+6}{4(r-n_{\text{PAD}})}2^{n'/3}$ queries to $\mathbf{f}$, where $n' = \frac{r-n_{\text{PAD}}+1}{r-n_{\text{PAD}}}\ell + 2(r - n_{\text{PAD}} + 1) + 6\frac{r-n_{\text{PAD}}+1}{r-n_{\text{PAD}}}$.

In the other case $\ell \geq \mathbf{b}(c)$, the algorithm makes at most $3k_{\text{coll}} \cdot \frac{c+2(r-n_{\text{PAD}})+6}{4(r-n_{\text{PAD}})}2^{n'/3}$ queries to $\mathbf{f}$, where $n' = \frac{r-n_{\text{PAD}}+1}{r-n_{\text{PAD}}}c + 2(r - n_{\text{PAD}} + 1) + 6\frac{r-n_{\text{PAD}}+1}{r-n_{\text{PAD}}}$. $\square$

3.3.2 Quantum Collision Finding With Random Oracle

In [Cza+18] the task of bounding the probability of collisions—that we solve in Lemmas 3.11 and 3.12—is solved with a use of an external independent random oracle. Instead of just relying on the randomness of $\text{SPONGE}_{\mathbf{f}}$ we can feed first the input to the hash function to a random function and with an algorithm similar to Algorithm 3.2 find collisions in any $\mathbf{h}$. Note that the algorithm that uses the random oracle works for any hash function, not only for $\text{SPONGE}_{\mathbf{f}}$. The drawback is though that it requires a lot of external randomness. One might consider using $\text{SPONGE}_{\mathbf{f}}$ for the external oracle as well, and use indistinguishability or indifferentiability results for the required randomness. Note however that for the number of queries made by the collision-finding algorithm the distinguishing advantage would no longer be negligible. This fact points to the statement that the algorithm does require an actual random oracle to work or a construction with much higher parameters—in case of sponges this crucial parameter would be the capacity.

We state the general result and point to Theorem 36 in [Cza+18] for the proof.

Theorem 3.13 (Theorem 36 in [Cza+18]). *For finite sets $\mathcal{X}, \mathcal{Y}$ with $|\mathcal{Y}| \geq 3$ and $|\mathcal{X}| \geq 40|\mathcal{Y}|$, there exists a quantum algorithm which requires access to a random oracle $\mathsf{H} : \mathcal{M} \to \mathcal{X}$, that outputs a collision in any any function $\mathbf{h} : \mathcal{X} \to \mathcal{Y}$ with probability at least $1/8$ after at most $k_{\text{coll}} \cdot |\mathcal{Y}|^{1/3}$ queries to $\mathbf{h}$ and at most $2k_{\text{coll}} \cdot |\mathcal{Y}|^{1/3} + 2$ queries to H where k_{coll} is the constant from Theorem 3.7.*

As noted after Theorem 3.7, there exist constants $k'_{\text{coll}}, k''_{\text{coll}}$ such that the running time and quantum memory of the collision-finding algorithm is at most $k'_{\text{coll}} \cdot |\mathcal{Y}|^{1/3} \cdot \log^{k''_{\text{coll}}}(|\mathcal{Y}|)$.

A corollary of the above theorem is a statement on the number of queries sufficient to find collisions in $\textsc{Sponge}_{\mathbf{f}}$ with high probability.

Corollary 3.14 (Theorem 37 in [Cza+18]). *There exists a quantum algorithm* $\textsc{coll-ro}$ *that finds a collision in* $\textsc{Sponge}_{\mathbf{f}}[\textsc{pad}, \{0,1\}^r, \{0,1\}^c, \ell]$, *a sponge construction, for an arbitrary internal function* $\mathbf{f}$. $\textsc{coll-ro}$ *makes at most $q_{\mathbf{f}}$ quantum queries to $\mathbf{f}$ and q_{H} quantum queries to a independent random oracle H.* $\textsc{coll-ro}$ *outputs colliding messages $m \neq m'$ such that $\textsc{Sponge}_{\mathbf{f}}(m) = \textsc{Sponge}_{\mathbf{f}}(m')$ with probability at least $1/8$, where*
$q_{\mathbf{f}} := 2k_{\text{coll}} \cdot \min\{\frac{c+6+2r}{r}2^{c/3}, \frac{2n+6+3r}{r}2^{n/3}\}$, *and* $q_{\mathcal{H}} := 2k_{coll} \cdot \min\{2^{c/3}, 2^{n/3}\} + 2$, *where k_{coll} is the constant from Theorem 3.7 and* $\textsc{pad}$ *is any padding function which appends at most $2r$ bits.*

Typical padding functions do not append more than $r + 1$ bits to the message, and are therefore covered by the theorem. Otherwise, the proof of Corollary 3.14 could be easily modified to take longer paddings into account, resulting in increased factors in the expression of $q_{\mathbf{f}}$ above.

3.4 Collapsingness of the Sponge Construction

The notion of collapsingness is explained in detail in Section 2.2.3. In [Cza+18] it is proven that the sponge construction is collapsing if the internal function is collapsing on the inner and outer part and the inner part is zero-preimage resistant. Here we state the result and give an idea of the proof, for proofs of the following statements we refer to [Cza+18]. In the end of this section we state a theorem connecting indifferentiability with collapsingness. We simplify the general t-valid adversaries from [Cza+18] and focus on 1-valid adversaries.

The proof of collapsingness of $\textsc{Sponge}_{\mathbf{f}}$ is done in three stages: first we prove collapsingness of the absorbing phase (shortened by one application of $\mathbf{f}$), then we prove collapsingness of the squeezing phase (modified by applying $\mathbf{f}$ before outputting the first block), and finally we conclude security of the whole sponge Sponge, consisting of padding, absorbing, and squeezing.

First, we analyze the absorbing phase. For the absorbing phase (without padding or squeezing) to be collapsing, we will need two properties of $\hat{\mathbf{f}}$:

- $\hat{\mathbf{f}}$ is collapsing. This is the main property required from the internal function $\mathbf{f}$. If we restricted the domain of Absorb to fixed length messages, then we could show the collapsing property of Absorb based on that property alone.

- $\hat{\mathbf{f}}$ is zero-preimage-resistant. To see why we need this property, consider an internal function $\mathbf{f}$ where the adversary can find, e.g., $x, y \in \{0,1\}^r$ with $f(x\|0^c) = y\|0^c$. Then we can see that $\textsc{Absorb}(x\|y) = 0^{c+r}$, and thus $\textsc{Absorb}(x\|y\|z) = z\|0^c = \textsc{Absorb}(z)$ for any $z \in \{0,1\}^r$. Thus Absorb would not be collision-resistant, and in particular not collapsing.

The reduction that proves collapsingness of $\textsc{Absorb}_{\mathbf{f}}$ is given below.

Lemma 3.15. *Let* (A, B) *be a* τ*-time adversary, valid against* Sponge *on* $(\{0,1\}^r)^*$*, with collapsing-advantage* ε*. Assume that* A *outputs messages of length at most* $l \cdot r$ *bits. Then there are:*

- *a* $(\tau + l\tau_{\mathbf{f}} + O(lc))$*-time adversary* A' *that finds a zero-preimage*[2] *of* $\hat{\mathbf{f}}$ *with probability* ε'*, and*

- *a* $(\tau + O(\tau_{\mathbf{f}}))$*-time adversary* $(\mathrm{A}'', \mathrm{B}'')$*, valid against* $\hat{\mathbf{f}}$*, with collapsing-advantage* ε''

such that $\varepsilon \leq \sqrt{\varepsilon'} + (l-1)\varepsilon''$*. Here* $\tau_{\mathbf{f}}$ *is the time required for a single classical invocation of* $\mathbf{f}$*. If time is measured in queries to* $\mathbf{f}$ *(instead of computation steps), the term* $O(lc)$ *in the runtime of* A' *can be omitted.*

The proof works by excluding zero-preimages via zero-preimage resistance and step by step (steps are the evaluations of $\mathbf{f}$) collapsing the states from the back of the evaluation of Sponge. Zero preimages have such a special role because they mark the beginning of the message, if one gets a zero inner state in the middle of an evaluation of Sponge one can get a collision (note that in Section 3.2 we have a similar situation) and hence win in the collapsingness game. We believe the necessity to assume zero-preimage resistance is just a consequence of the step by step nature of the proof. For random internal functions (or permutations) the probability that the inner-state equals to 0 is very small.

[2]By *zero-preimage* we mean any value x with $\hat{\mathbf{f}}(x) = 0^c$.

Next, we show that the squeezing phase is collapsing. Let $\bar{\mathbf{f}}_{\min}$ be defined for $\ell > 0$, as the first $\min(\ell, r)$ bits of the output of $\mathbf{f}$ (in particular, $\bar{\mathbf{f}}_{\min} = \bar{\mathbf{f}}$ for $\ell \geq r$). Then the collapsing property of the squeezing phase is a relatively trivial consequence of the fact that $\bar{\mathbf{f}}_{\min}$ is collapsing.

Lemma 3.16. *Let $\ell > 0$ be the output length of* Sponge. *Let (A, B) be a valid adversary against the squeezing phase of* $\textsc{Sponge}_{\mathbf{f}}$ *with runtime τ and collapsing-advantage ε. Then there is an adversary $(\mathrm{A}', \mathrm{B}')$ that is valid for $\bar{\mathbf{f}}_{\min}$, has runtime $\tau + O(\min(\ell, r))$, and collapsing-advantage ε. If time is measured in $\mathbf{f}$-queries (instead of computation steps), the runtime of $(\mathrm{A}', \mathrm{B}')$ is τ.*

Connecting the above statements via Lemma 2.11 we get

Theorem 3.17. *Let $\ell > 0$ be the output length of* Sponge. *Assume that* pad *is injective. Assume a τ-time adversary (A, B), valid for* Sponge, *with collapsing-advantage ε against* Sponge. *Then there are:*

- *a $(\tau + O(\tau_{\text{PAD}}) + O(\tau_{\text{Absorb}}))$-time adversary A_1 that finds a zero-preimage of $\hat{\mathbf{f}}$ with probability ε_1, and*
- *a $(\tau + O(\tau_{\text{PAD}}) + O(\tau_{\text{Absorb}}))$-time adversary $(\mathrm{A}_2, \mathrm{B}_2)$, valid against $\hat{\mathbf{f}}$, with collapsing-advantage ε_2, and*
- *a $(\tau + O(\tau_{\text{PAD}}) + O(\tau_{\text{Absorb}}))$-time adversary $(\mathrm{A}_3, \mathrm{B}_3)$ that is valid for $\bar{\mathbf{f}}_{\min}$, with collapsing-advantage ε_3,*

such that $\varepsilon \leq \sqrt{\varepsilon_1} + (\ell - 1)\varepsilon_2 + \varepsilon_3$. Here τ_{PAD} refers to the maximum runtime of pad. *And τ_{Absorb} refers to the maximum runtime of* Absorb *(on outputs of* pad*). If time is measured in $\mathbf{f}$-queries (instead of computation steps), the runtimes are $\tau + O(\ell\tau_{\mathbf{f}})$ where $\tau_{\mathbf{f}}$ is the number of H-queries performed by $\mathbf{f}$, and ℓ is an upper bound on the length (in r-bit blocks) of the messages m that A outputs on M_j.*

For a random function we have Lemma 2.12 and so for a random internal function we have:

Lemma 3.18. *If $\mathbf{f} : \{0,1\}^{r+c} \to \{0,1\}^{r+c}$ is a random function, and $(\mathrm{A}^{\mathbf{f}}, \mathrm{B}^{\mathbf{f}})$ is valid for $\bar{\mathbf{f}}_{\min}$ (or for $\hat{\mathbf{f}}$), makes $\leq q$ queries to $\mathbf{f}$, and has collapsing-advantage ε, then $\varepsilon \in O(q^{3/2} 2^{-\min(\ell, r)/2})$ (or $\varepsilon \in O(q^{3/2} 2^{-c/2})$).*

Using the above statement we can prove collapsingness of random sponges.

Theorem 3.19. *Let $\mathbf{f} : \{0,1\}^{r+c} \to \{0,1\}^{r+c}$ be a random function. Let (A, B) be an adversary that is valid against* Sponge *with output length n and that makes $\leq q$ queries to $\mathbf{f}$. Then (A, B) has collapsing-advantage*

$$O\left(\ell(q+\ell)^{3/2} 2^{-c/2} + (q+\ell)^{3/2} 2^{-\min(n,r)/2}\right). \tag{3.48}$$

In the rest of this section we state and prove that any function that is indifferentiable from a collapsing function is itself collapsing. In the context of sponges, together with Theorem. 7.11, we reprove the result of [Cza+18] in a modular way that might become useful when indifferentiability of sponges with permutations is established.

Theorem 3.20 (Quantum indifferentiability preserves collapsingness)**.** *Let* C *be a construction based on an internal function* $\mathbf{f}$*, and let* C *be* $(q, \varepsilon_I(q))$*-indifferentiable from an ideal function* $\mathsf{C}_{\text{ideal}}$ *with simulator* S*. Assume further that* $\mathsf{C}_{\text{ideal}}$ *allows for a collapsingness advantage at most* $\varepsilon_{\text{coll}}(q)$ *for a* q*-query adversary. Then* C *is collapsing with advantage* $\varepsilon_{\text{coll}}(q_{\mathsf{C}}, q_{\mathbf{f}}) = 2\,\varepsilon_I(q_{\mathsf{C}} + q_{\mathbf{f}}) + \varepsilon_{\text{coll}}(q_{\mathsf{C}} + \alpha q_{\mathbf{f}})$*, where* q_{C} *and* $q_{\mathbf{f}}$ *are the number of queries to* C *and* $\mathbf{f}$*, respectively, and* α *is the number of queries the simulator* S *makes (at most) to* $\mathsf{C}_{\text{ideal}}$ *each time it is queried.*

Proof. Given a collapsingness distinguisher $\tilde{\mathsf{D}}$ against C with advantage $\varepsilon \geq \varepsilon_{\text{coll}}(q_{\mathsf{C}} + \alpha q_{\mathbf{f}})$ that makes q_{C} queries to C and $q_{\mathbf{f}}$ queries to $\mathbf{f}$, we build an indifferentiability distinguisher D as follows. Chose $b \in \{0, 1\}$ at random. Running $\tilde{\mathsf{D}}$, if $b = 0$ simulate **Collapse 1**, if $b = 1$ simulate **Collapse 2**. Output 1 if $\tilde{\mathsf{D}}$ outputs b, and 0 otherwise.

In the real world, we have that

$$\mathbb{P}[1 \leftarrow \mathsf{D} : \mathbf{Real}] = \frac{1}{2}\left(\mathbb{P}[0 \leftarrow \tilde{\mathsf{D}}^{\mathsf{C},\mathbf{f}} : \mathbf{Collapse\ 1}] + \mathbb{P}[1 \leftarrow \tilde{\mathsf{D}}^{\mathsf{C},\mathbf{f}} : \mathbf{Collapse\ 2}]\right) \tag{3.49}$$

$$= \frac{1}{2} + \frac{1}{2}\left(\mathbb{P}[1 \leftarrow \tilde{\mathsf{D}}^{\mathsf{C},\mathbf{f}} : \mathbf{Collapse\ 2}] - \mathbb{P}[1 \leftarrow \tilde{\mathsf{D}}^{\mathsf{C},\mathbf{f}} : \mathbf{Collapse\ 1}]\right). \tag{3.50}$$

In the ideal world, the distinguisher together with the simulator S can be seen as a collapsingness distinguisher for $\mathsf{C}_{\text{ideal}}$. Therefore we get

$$\mathbb{P}[1 \leftarrow \mathsf{D} : \mathbf{Ideal}]$$
$$= \frac{1}{2} + \frac{1}{2}\left(\mathbb{P}[1 \leftarrow \tilde{\mathsf{D}}^{\mathsf{C}_{\text{ideal}},\mathsf{S}} : \mathbf{Collapse\ 2}] - \mathbb{P}[1 \leftarrow \tilde{\mathsf{D}}^{\mathsf{C}_{\text{ideal}},\mathsf{S}} : \mathbf{Collapse\ 1}]\right) \tag{3.51}$$

and hence

$$\Big|\mathbb{P}[1 \leftarrow \mathsf{D} : \mathbf{Real}] - \mathbb{P}[1 \leftarrow \mathsf{D} : \mathbf{Ideal}]\Big|$$

$$= \frac{1}{2}\Big|\mathbb{P}[1 \leftarrow \tilde{\mathsf{D}}^{\mathsf{C},\mathbf{f}} : \mathbf{Collapse\ 2}] - \mathbb{P}[1 \leftarrow \tilde{\mathsf{D}}^{\mathsf{C},\mathbf{f}} : \mathbf{Collapse\ 1}] \tag{3.52}$$

$$- \mathbb{P}[1 \leftarrow \tilde{\mathsf{D}}^{\mathsf{C}_{\text{ideal}},\mathsf{S}} : \mathbf{Collapse\ 2}] + \mathbb{P}[1 \leftarrow \tilde{\mathsf{D}}^{\mathsf{C}_{\text{ideal}},\mathsf{S}} : \mathbf{Collapse\ 1}]\Big| \tag{3.53}$$

$$\geq \frac{1}{2}\left(\varepsilon - \varepsilon_{\text{coll}}(q_{\mathsf{C}} + \alpha q_{\mathbf{f}})\right). \tag{3.54}$$

□

CHAPTER

4 QUANTUM INDISTINGUISHABILITY OF SPONGES

Chapter contents

In this chapter we show that the sponge construction can be used to construct quantum-secure pseudorandom functions. The main contribution is a proof that random sponges are quantum indistinguishable from random functions. Our proofs hold under the assumption that the internal function is a random function or permutation. We then use this result to obtain a quantum-security version of a result by Andreeva, Daemen, Mennink, and Van Assche [And+15] which shows that a sponge that uses a secure PRP or PRF as internal function is a secure PRF. This result also proves that the recent attacks against CBC-MAC in the quantum-access model by Kaplan, Leurent, Leverrier, and Naya-Plasencia [Kap+16] and Santoli and Schaffner [SS17] can be prevented by introducing a state with a non-trivial inner part.

The proof of our main contribution is derived by analyzing the joint distribution of any q input-output pairs. Our method analyzes the statistical behavior of the considered construction in great detail. The used techniques might prove useful in future analysis of different cryptographic primitives considering quantum adversaries. Using Zhandry's PRF/PRP switching lemma [Zha15a] we then obtain that quantum indistinguishability also holds if the internal block function is a random permutation.

4.1 Introduction

Originally introduced in the context of cryptographic hash functions, the sponge construction [Ber+07] became one of the most widely used constructions in symmetric cryptography. Consequently, sponges get used in keyed constructions, including message authentication codes (MAC), stream ciphers, and authenticated encryption (AE), see e.g. [Ber+10; Ber+11a; Ber+12b; MRV15; RS14; And+15; GPT15]. For all these applications it is either necessary or at least sufficient for security that a secretly keyed sponge is indistinguishable from a random function. That this is indeed the case was already shown in the original security proof for the sponge construction [Ber+08] where cryptographic sponges were shown to be indifferentiable from random functions. This result is widely applicable and consequently was followed up with several improved bounds for specific applications. Recent works [MRV15; And+15; GPT15] improved the bound for the setting of indistinguishability of secretly keyed sponges.

While these results show the applicability of the sponge construction in today's computing environment, they leave open the question of its applicability in a future post-quantum setting where adversaries have access to quantum computers. Such an attacker can for example run Shor's algorithm [Sho94] to break the security of constructions based on the RSA or discrete-logarithm problem. While such constructions are hardly ever considered for practical symmetric cryptography due to their slow operations, the impact of quantum adversaries goes beyond Shor's algorithm. Conventional security proofs, especially in idealized models, might break down in the light of quantum attackers who are allowed to ask queries in superposition [Bon+11]. Going even further, allowing adversaries superposition access to secretly keyed primitives, it was shown that several well known MACs and encryption schemes, including CBC-MAC and the Even-Mansour block cipher become insecure [KM12; Kap+16; SS17]. While these latter attacks are not applicable in the post-quantum setting, they are indications that secret-key cryptography does not trivially withstand quantum adversaries and that it is necessary to study the security of symmetric cryptography in the post-quantum setting. In this chapter we do exactly this: We study the security of secretly keyed sponges against quantum adversaries.

In this chapter we are concerned with proving that domain extension using the sponge construction leads to a somewhat random behaving function if the internal building block $\mathbf{f}$ behaves random in some way. Formally, we prove *indistinguishability* of the construction from a random oracle given that the internal building block $\mathbf{f}$ is a random function or permutation. When discussing the notion of indistinguishability of cryptographic constructions we assume that the distinguisher has oracle access either to the constructed function—e.g. $\textsc{Sponge}_{\mathbf{f}}$—or a random oracle, but not the internal function $\mathbf{f}$. Not allowing the distinguisher access to $\mathbf{f}$ is especially well justified for a secretly keyed building block such as a pseudorandom function or permutation.

Against non-quantum attackers, it was proven that $\textsc{Sponge}_{\mathbf{f}}$ with a random internal function is indistinguishable from a random function [Ber+07]. The proof presented by the inventors of the sponge construction relies on the fact that with polynomially many queries to $\textsc{Sponge}_{\mathbf{f}}$, the adversary is likely not to see collisions in the hidden part of the construction's state. That argument is no longer valid in the quantum world. As one can query $\textsc{Sponge}_{\mathbf{f}}$ on a superposition of exponentially many messages there will certainly occur collisions in different branches of the superposition. It is not at all obvious how to use this fact to break security but one certainly needs to use a different proof technique to establish security. Hence the classical argument is not useful if we assume that the adversary has access to a quantum computer.

Contributions. As the main contribution, we prove that the sponge construction using a random function or permutation is quantumly indistinguishable from a random function (see Theorems 4.2 and 4.9). This result can be used to obtain a quantum version of Theorem 1 from [And+15] (see Theorem 4.5) which states that the indistinguishability of keyed-internal-function sponges can be derived from the quantum-PRF-security (or quantum-PRP-security in case of a block-cipher) of the keyed internal function. Thereby we not only provide a proof for the security of keyed-internal-function sponges in the post-quantum setting, but even in the stronger quantum settings where the adversary gets full quantum-access to the keyed-internal-function sponge, i.e we prove that keyed-internal-function sponges are quantum PRFs.

Another implication of our result is that the quantum attacks against CBC-MAC mentioned above can be prevented using a state with a non-trivial inner part. The authors of the attack already noted[1] that their attack does not work in this case. More specifically, CBC-MAC can be viewed as full-width sponge (where the state has no inner part, i.e., the capacity is 0). On the other hand, a CBC-MAC where all message blocks are padded with $0 \in \mathcal{C}$ and the output is truncated to the first character can be viewed as an keyed-internal-function sponge. Hence, our result applies and shows that the quantum attacks by Kaplan, Leurent, Leverrier, and Naya-Plasencia [Kap+16] and Santoli and Schaffner [SS17] using Simon's algorithm are not applicable any longer. Even more, our result proves that this little tweak of CBC-MAC indeed results in a quantum secure MAC.

We also show a direct proof of indistinguishability for $\mathbf{f}$ being a random permutation. In this proof we state and prove a lemma that generalizes the average case polynomial method to allow for functions that are not necessarily polynomials but are close to one; this result is not necessary to achieve the main goal of the chapter but might be useful in other works using similar techniques.

In this chapter, we provide a quantum security guarantee more suitable for keyed primitives where an attacker does not have access to the internal building block. On the one hand, we increase the trust that hash functions based on the sponge construction are quantum safe and on the other hand, we formally prove that it is a quantum secure pseudorandom function when used with a keyed internal function—like it is used in the hash-based signatures scheme SPHINCS+ [Tea17] in the instantiation using the Haraka hash function [Köl+16].

A limitation. The authors of [And+15] use their Theorem 1 to show security of inner-keyed sponges using the PRP-security of single-key Even-Mansour. Their result does not carry over to the quantum setting as Even-Mansour is vulnerable in the quantum setting [KM12]. This does not lead to an actual attack on inner-keyed sponges in the quantum setting. The attack needs access to the full input to the Even-Mansour cipher, which is never the case for inner-keyed sponges as long as a non-trivial inner state is used. However, the attack on Even-Mansour does render the modular proof strategy not applicable for inner-keyed sponges. We also need to stress that our result does not cover the commonly used approaches to secretly key SHA-3 for this very reason.

Our approach. The main technical contribution of this chapter is a proof that the probability for any given input-output behavior of $\textsc{Sponge}_{\mathbf{f}}$ is a polynomial in the capacity of the sponge. This observation allows us then to apply the average-case polynomial method of [Zha12] (see Theorem 2.21 below).

[1]See slide 16 (page 26) of their Crypto 2016 presentation available at `https://who.rocq.inria.fr/Gaetan.Leurent/files/Simon_CR16_slides.pdf`.

In more detail, recall that the capacity of a Sponge$_\mathbf{f}$ is the size of the inner state (there are $|\mathcal{C}|$ possible inner states for a sponge as in Figure 2.4). If the capacity of a sponge increases, it becomes less and less likely that there are collisions in the inner state. Hence for infinite capacity, the inner states are unique and so the internal functions are called on unique inputs and therefore, the sponge behaves like a random function. Our proof formalizes this intuition by carefully analyzing the probabilities for q given input-output values of the sponge in terms of the capacity. We show that these probabilities are in fact polynomials in the inverse of the capacity of degree at most q times the length of the input-output values. We refer to Lemma 4.3 for the formal statement.

By establishing the capacity as this crucial parameter, we fit directly into the proof technique from [Zha12] that uses approximating polynomials of low degree to show closeness of distributions and in turn small quantum distinguishing advantage. By the PRF/PRP switching lemma from [Zha15a], quantum indistinguishability also holds for the case of $\mathbf{f}$ being a random permutation. In the appendix, we provide an alternative proof for this case by generalizing the proof technique of [Zha12] to the case of permutations.

To sum up, we deal directly with the statistical behavior of the sponge construction. Our proof technique might be generalized to analyze other cryptographic constructions or used for establishing other properties than indistinguishability. After all it provides insight into the workings of constructions based on repeatedly applying a fixed internal function $\mathbf{f}$. From a different perspective, our proof is based on the observations that the probability of any multiple input-output pairs occurring is a polynomial in $|\mathcal{C}|^{-1}$; the existence of such a parameter is a specific feature of the sponge construction. Hence, our method does not directly apply to other schemes like the Merkle-Damgård construction.

Let us also highlight the difference in our definition of the sponge construction compared to the standardized version. Instead of $\{0,1\}^r$ and $\{0,1\}^c$ we use $\mathcal{A}$ and $\mathcal{C}$ respectively. The fact that $\mathcal{C}$ can be any finite set allows us to use the polynomial method; If we want to use the statement about closeness of polynomials we have to show that $\mathbf{p}$ is a polynomial for any inverse integer and not only for 2^{-c}. Using the more general definition solves the problem of defining distributions for any integer, not only powers of 2. Note that of course all our results hold also for the standard definition.

Organization. In Section 4.2 we show that Sponge$_\mathbf{f}$ is indistinguishable from a random oracle in the conventional-access setting (in contrast to the quantum-access model). In Section 4.3 we state the main result of this chapter as well as several derived results. In Section 4.4 we provide an example proof valid for limited distinguishers but giving sufficient details to understand our approach and verify correctness without all the particulars of the full proof. Section 4.5 contains the proof of Lemma 4.3, the main technical result of this chapter. The case of random permutations is covered in Section 4.6.

4.2 Classical Indistinguishability of Random Sponges

In the following we state the indistinguishability result in the classical domain.

Theorem 4.1 (Classical indistinguishability of Sponge). *If* $\mathbf{f}$ *is a random transformation or a random permutation then* $\text{Sponge}_{\mathbf{f}}$ *defined in Algorithm 2.1 is classically indistinguishable from a random oracle. Namely for all quantum algorithms* A *making polynomially many classical queries there is a negligible function* ϵ *such that*

$$\left| \underset{\mathbf{f} \overset{\$}{\leftarrow} \mathcal{S}^{\mathcal{S}}}{\mathbb{P}} \left[b = 1 : b \leftarrow \mathrm{A}^{\text{Sponge}_{\mathbf{f}}} \right] - \underset{\mathbf{h} \leftarrow \mathfrak{R}}{\mathbb{P}} \left[b = 1 : b \leftarrow \mathrm{A}^{\mathbf{h}} \right] \right| \leq \epsilon, \tag{4.1}$$

where $\mathcal{S} = \mathcal{A} \times \mathcal{C}$, *and* $\mathfrak{R}$ *is defined according to Definition 2.13.*

Proof. The proof follows closely the proof of theorem 2 of [Ber+07]. Even though we give more power to the adversary by giving her access to a quantum computer, the queries are considered to be classical. All arguments in the proof of Bertoni and others depend only on the queries made by the adversary and not her computing power. For that reason we can use the discussion around Equation (7) from [MS17], which states that if a classical result holds for q-query distinguishers with unlimited computational power then it also holds for quantum distinguishers that make classical queries. The former class of distinguishers is exactly what we consider, as the classical result depends only on the number of queries done. □

4.3 Quantum Indistinguishability of Random Sponges

We want to show that the distribution corresponding to random sponges is quantumly indistinguishable from a random oracle. We can define a family of distributions indexed by the security parameter that intuitively gets closer to a random oracle with increasing parameter. For that reason Theorem 2.21 is a perfect theoretical tool to be used. The relevant tasks that remain are to identify the family of distributions that correspond to our figure of merit, to show that in fact the most secure member of the family with $t = \infty$ is a random oracle, and to prove that the assumptions of Theorem 2.21 are fulfilled.

The security parameter in Sponge is the capacity; we parametrize the family of random sponges by the size of the inner state space $t = |\mathcal{C}|$. Intuitively speaking, for $c \to \infty$ each evaluation of the internal function is done with a different inner state. In this case irrespective of the input, the output is a completely random string, which is the definition of a random oracle (R). Hence we conclude that we identified a family of distributions that is well suited to be used with Theorem 2.21. If we show that indeed for $t = \infty$ the member of the family is the random oracle we have that:

$$\begin{aligned}&\mathfrak{F}_{|\mathcal{C}|} \text{ is quantumly indistinguishable from } \mathfrak{F}_{\infty} \\ &\Rightarrow \text{ random sponge is quantumly indisitinguishable from } \mathsf{R}. \end{aligned} \tag{4.2}$$

We are left with the task to prove the left-hand side of the above statement. The assumption of Theorem 2.21 is that the probability of witnessing any input-output behavior on q queries is a polynomial in $1/|\mathcal{C}|$. Following the statement og Theorem 2.21 we define $\mathbf{p}(1/t) = \mathbb{P}_{\mathbf{h}\leftarrow\mathfrak{F}_t}[\forall i \in [2q] : \mathbf{h}(X^i) = Y^i]$. It remains to prove that $\mathbf{p}(|\mathcal{C}|^{-1})$ is in fact a polynomial in $|\mathcal{C}|^{-1}$, where by $|\mathcal{C}|$ we denote cardinality of the set. With that statement proven we fulfill the assumptions of Theorem 2.21 and show quantum indistinguishability of Sponge.

Let us now formally state the main claim of this chapter. We are going to focus on the internal function being modeled as a random function, in Section 4.6 though, we are going to cover the case of random permutations.

Theorem 4.2. Sponge$_{\mathbf{f}}$[pad, $\mathcal{A}, \mathcal{C}$] *for random* $\mathbf{f}$ *is quantumly indistinguishable from a random oracle. More concretely, for all quantum algorithms* A *making at most* q *quantum queries to* Sponge$_{\mathbf{f}}$*, such that the length of the padded input is at most* m *and the output length is at most* z *symbols of* $\mathcal{A}$*,*

$$\left| \underset{\mathbf{f}\xleftarrow{\$}\mathcal{S}^{\mathcal{S}}}{\mathbb{P}} \left[b = 1 : b \leftarrow \mathrm{A}^{|\textsc{Sponge}_{\mathbf{f}}\rangle} \right] - \underset{\mathbf{h}\leftarrow\mathfrak{R}}{\mathbb{P}} \left[b = 1 : b \leftarrow \mathrm{A}^{|\mathbf{h}\rangle} \right] \right| < \frac{\pi^2}{6}\eta^3|\mathcal{C}|^{-1}, \tag{4.3}$$

where $\eta := 2q(m + z - 2)$ *and* $\mathfrak{R}$ *is defined according to Definition 2.13. The domain is defined as* $\mathcal{S} = \mathcal{A} \times \mathcal{C}$ *for a finite* $\mathcal{A}$ *in an Abelian group* $(\mathcal{A}, \oplus)$ *and a non-empty finite set* $\mathcal{C}$*.*

Our result is stated for a general $\mathcal{A}$ but one can just consider $\mathcal{A} = \{0, 1\}^r$.

Before we prove the above theorem we state the main technical lemma. In the lemma we consider a set of messages that are already padded according to pad and the sponge itself is done without padding. This is done without loss of generality and simplifies the discussion.

Lemma 4.3. *Let us assume the padding function to be the identity,* pad = id*, and set* $\mathcal{M} = \mathcal{M}_{\textsc{pad}}$*, where* $\mathcal{M}_{\textsc{pad}}$ *is a set of messages padded according to* pad*. For a fixed* q *and for every* $(\mathbf{M}, \mathbf{Z}) := ((\mathbf{M}^i, \mathbf{Z}^i))_{i\in[2q]}$*, where* $\forall i \in [2q] : (\mathbf{M}^i, \mathbf{Z}^i) \in \mathcal{M} \times \mathcal{A}^*$*, such that* $\forall i \in [2q] : |\mathbf{M}^i| \le m, |\mathbf{Z}^i| \le z$*, it holds that*

(*i*) *the probability function is a polynomial in* $|\mathcal{C}|^{-1}$ *of degree* η

$$\mathbb{P}\left[\forall i \in [2q] : \textsc{Sponge}_{\mathbf{f}}(\mathbf{M}^i, \ell_i) = \mathbf{Z}^i\right] = \sum_{j=0}^{\eta} a_j |\mathcal{C}|^{-j} =: \mathbf{p}(|\mathcal{C}|^{-1}), \tag{4.4}$$

(ii) and the constant term is

$$a_0 = \prod_{i=1}^{2q} \boldsymbol{\delta}(\mathbf{M}, \mathbf{Z}, i)\, |\mathcal{A}|^{-|\mathbf{Z}^i|}. \tag{4.5}$$

All coefficients a_j are real and independent of $|\mathcal{C}|^{-1}$, the degree of the polynomial equals $\eta := 2q(m+z-2)$. In the equation describing a_0 we use $\boldsymbol{\delta}(\mathbf{M}, \mathbf{Z}, i)$ to denote a Boolean function that is 0 if $\mathbf{M}^i$ is input more than once and $\mathbf{Z}^i$ is inconsistent with other outputs (inputting the same message for the second time should yield the same output), the function is 1 otherwise.

The full proof is presented in Section 4.5.

Proof idea. Our goal is to explicitly evaluate $\mathbb{P}\left[\forall i \in [2q] : \text{SPONGE}_{\mathbf{f}}(\mathbf{M}^i, \ell_i) = \mathbf{Z}^i\right]$. We base all of our discussion on two facts: SPONGE has a structure that we know and it involves multiple evaluations of the internal function $\mathbf{f}$. $\mathbf{f}$ is a random function with well specified probability of yielding some output on a given input. The main idea of our approach is to extract terms like $\mathbb{P}[\mathbf{f}(S_1) = S_2]$ for some states S_1, S_2 from the overall probability expression and evaluate them.

Let us go through a more detailed plan of the proof. Fix $(\mathbf{M}, \mathbf{Z})$ and set $\ell_i := |\mathbf{Z}^i|$. In the first step we include all intermediate states in the probabilistic event $(\forall i \in [2q] : \text{SPONGE}_{\mathbf{f}}(\mathbf{M}^i, \ell_i) = \mathbf{Z}^i)$. We write explicitly all inner states and outer states not specified by the input-output pairs $(\mathbf{M}, \mathbf{Z})$. Next we rewrite the full probability expression in the form $\sum \prod \mathbb{P}[\mathbf{f}(S_1) = S_2 \mid \ldots]$. The sum comes from the fact that there are many possible intermediate states that yield the given input-output behavior. The product is the result of using Bayes' rule to isolate a single evaluation of $\mathbf{f}$ in the probability. To correctly evaluate the summands we need to analyze all states in $\mathbb{P}[\mathbf{f}(S_1) = S_2 \mid \ldots]$ from the perspective of *uniqueness*—we say a state is unique if it is input to $\mathbf{f}$ just a single time. Given a specific setup of unique states in all $2q$ evaluations of SPONGE we can easily evaluate the probabilities, as the only thing we need to know is that $\mathbf{f}$ is random. The final step of the proof is to calculate the number of states in the sum. We sum over all values of states that fulfill the constraints of $(\forall i \in [2q] : \text{SPONGE}_{\mathbf{f}}(\mathbf{M}^i, \ell_i) = \mathbf{Z}^i)$ and $\mathbf{f}$ being a function. The previous analysis of uniqueness of states makes it easier to include the latter constraint; non-unique states have predetermined outputs under $\mathbf{f}$ decreasing the number of possible states. After those steps we end up with an explicit expression for $\mathbb{P}\left[\forall i \in [2q] : \text{SPONGE}_{\mathbf{f}}(\mathbf{M}^i, \ell_i) = \mathbf{Z}^i\right]$, which allows us to show that $\mathbf{p}$ is a polynomial of the claimed degree and its limit in $t \to \infty$, i.e. the constant term a_0 is the probability of uniformly random outputs. $\square$

Proof of Theorem 4.2. Let us define a family $\mathfrak{F}_t$ indexed by $t \in \mathbb{N} \cup \{\infty\}, t > 0$. $\mathfrak{F}_t$ is a distribution on functions from $\mathcal{M} \times \mathbb{N}$ to $\mathcal{M}$, where $\mathcal{M}$ is an arbitrary set. The family is additionally parametrized by the choice of $r \in \mathbb{N}$ and a sponge-compliant padding function PAD. We define $\mathbf{h} \leftarrow \mathfrak{F}_t$ as follows:

- Choose $\mathbf{f}$ uniformly at random from $\mathcal{S}^{\mathcal{S}}$, where $\mathcal{S} := \mathcal{A} \times \mathcal{C}$ and $\mathcal{C}$ is any finite set of size $t > 0$.
- Use $\mathbf{f}, \mathcal{C}$, the fixed $\mathcal{A}$, and PAD to construct $\text{SPONGE}_{\mathbf{f}}[\text{PAD}, \mathcal{A}, \mathcal{C}]$.
- For each $(X, \ell) \in \mathcal{M} \times \mathbb{N}$ set $\mathbf{h}(X, \ell) := \text{SPONGE}_{\mathbf{f}}[\text{PAD}, \mathcal{A}, \mathcal{C}](X, \ell)$.

To show that we defined $\mathfrak{F}_t$ in the right way, let us analyze Equation (4.2) from the point of view of the newly defined distribution. On the one hand from our definition it follows that

$$\underset{\mathbf{h}\leftarrow\mathfrak{F}_t}{\mathbb{P}}\left[b = 1 : b \leftarrow \mathrm{A}^{|\mathbf{h}\rangle}\right] = \underset{\mathbf{f}\overset{\$}{\leftarrow}\mathcal{S}^{\mathcal{S}}}{\mathbb{P}}\left[b = 1 : b \leftarrow \mathrm{A}^{|\text{SPONGE}_{\mathbf{f}}\rangle}\right], \tag{4.6}$$

where the equality follows from our definition of $\mathbf{h}$. On the other hand if we take $t \to \infty$, then intuitively the internal function is going to be injective on its inner part. Namely $\hat{\mathbf{f}}$—the internal function with its output restricted to the inner part—is injective. That implies a different inner state in every evaluation of $\mathbf{f}$ in SPONGE what in turn implies a random and independent outer part in every step of generating the output, formally

$$\underset{\mathbf{h}\leftarrow\mathfrak{F}_\infty}{\mathbb{P}}\left[b = 1 : b \leftarrow \mathrm{A}^{|\mathbf{h}\rangle}\right] = \underset{\mathbf{h}\leftarrow\mathfrak{R}}{\mathbb{P}}\left[b = 1 : b \leftarrow \mathrm{A}^{|\mathbf{h}\rangle}\right]. \tag{4.7}$$

This intuition is formally captured by statement (ii) of Lemma 4.3, where we state that in the limit of $|\mathcal{C}| \to \infty$ the probability of getting particular outputs of SPONGE is the same as for a random oracle.

From the above discussion we get that

$$\begin{aligned}&\left|\underset{\mathbf{h}\leftarrow\mathfrak{F}_t}{\mathbb{P}}\left[b = 1 : b \leftarrow \mathrm{A}^{|\mathbf{h}\rangle}\right] - \underset{\mathbf{h}\leftarrow\mathfrak{F}_\infty}{\mathbb{P}}\left[b = 1 : b \leftarrow \mathrm{A}^{|\mathbf{h}\rangle}\right]\right| = \\ &\left|\underset{\mathbf{f}\overset{\$}{\leftarrow}\mathcal{S}^{\mathcal{S}}}{\mathbb{P}}\left[b = 1 : b \leftarrow \mathrm{A}^{|\text{SPONGE}_{\mathbf{f}}\rangle}\right] - \underset{\mathbf{h}\leftarrow\mathfrak{R}}{\mathbb{P}}\left[b = 1 : b \leftarrow \mathrm{A}^{|\mathbf{h}\rangle}\right]\right|,\end{aligned} \tag{4.8}$$

which is the crucial equality for using Theorem 2.21 to prove our statement. The last element of the proof is the assumption about $\mathbf{p}$ being a polynomial and that is exactly the statement of Lemma 4.3. Note that the statement of Theorem 2.21 does not include inputs like ℓ_i but is still general enough that it works in our setting. □

Quantum indistinguishability of commonly used sponges with binary state follows directly from the general result.

Corollary 4.4. *If* $\mathbf{f}$ *is a random function or a random permutation, then* $\text{SPONGE}_{\mathbf{f}}[\text{PAD}, \{0,1\}^r, \{0,1\}^c]$ *is quantumly indistinguishable from a random oracle.*

Proof. For a random function we use Theorem 4.2 and for a random permutation Theorem 4.9 and set $\mathcal{C} = \{0,1\}^c$. □

4.3.1 Application to keyed-internal-function sponges

We show that Theorem 4.2 implies that keyed-internal-function sponges are indistinguishable from a random oracle under quantum access if the used internal function is a quantum-secure PRF (or if the internal function is a permutation, a quantum-secure PRP). This means that in the case $\mathbf{f}$ is a quantum-secure pseudorandom function or permutation the sponge construction is a quantum-secure pseudorandom function. For keyed primitives, indistinguishability from a random oracle/permutation is exactly what we call pseudorandomness.

Now we state and prove a quantum version of theorem 1 of [And+15] which formalizes the above statement about quantum security of keyed-internal-function sponges. Note that we state the theorem for the general sponge construction but thanks to Corollary 4.4 it holds for the regular construction as well.

Theorem 4.5. *If the internal function* $\mathbf{f}$ *used in* $\textsc{Sponge}_{\mathbf{f}}$ *is a quantum-secure PRF/PRP, Definition 2.5 with advantage* $\boldsymbol{\epsilon}^{\mathrm{PR}}$*, then the resulting keyed-internal-function sponge is a quantum-secure PRF with advantage*

$$\left| \underset{K \xleftarrow{\$} \mathcal{K}}{\mathbb{P}} \left[b = 1 : b \leftarrow \mathrm{A}^{|\textsc{Sponge}_{\mathbf{f}_K}\rangle} \right] - \underset{\mathbf{g} \leftarrow \mathfrak{R}}{\mathbb{P}} \left[b = 1 : b \leftarrow \mathrm{A}^{|\mathbf{g}\rangle} \right] \right| \leq \boldsymbol{\epsilon}^{\mathrm{PR}} + \frac{\pi^2}{6} \eta^3 |\mathcal{C}|^{-1}, \tag{4.9}$$

where $\eta := 2q(m+z-2)$*,* q *is the number of queries* A *makes to its oracle,* m *and* z *are as defined in the statement of Theorem 4.2, and* $\mathfrak{R}$ *is defined according to definition 2.13.*

Proof. We give the proof for $\mathbf{f}$ being a keyed function. The proof when $\mathbf{f}$ is a keyed permutation is obtained by using Theorem 4.9 in place of Theorem 4.2 and restricting the sets from which $\mathbf{g}$ and $\mathbf{f}$ are drawn below to permutations.

We show that the advantage of any quantum adversary in distinguishing the keyed-internal-function sponge from a random oracle is bound by its ability to distinguish $\mathbf{f}$ from a random oracle (permutation, respectively) plus its ability to distinguish a random sponge from a random oracle. In the following calculation we use the triangle inequality and the result of Theorem 4.2.

$$\begin{aligned}
&\left| \underset{K \xleftarrow{\$} \mathcal{K}}{\mathbb{P}} \left[b = 1 : b \leftarrow \mathrm{A}^{|\textsc{Sponge}_{\mathbf{f}_K}\rangle} \right] - \underset{\mathbf{g} \leftarrow \mathfrak{R}}{\mathbb{P}} \left[b = 1 : b \leftarrow \mathrm{A}^{|\mathbf{g}\rangle} \right] \right| \\
=&\left| \underset{K \xleftarrow{\$} \mathcal{K}}{\mathbb{P}} \left[b = 1 : b \leftarrow \mathrm{A}^{|\textsc{Sponge}_{\mathbf{f}_K}\rangle} \right] - \underset{\mathbf{f} \xleftarrow{\$} \mathcal{S}^{\mathcal{S}}}{\mathbb{P}} \left[b = 1 : b \leftarrow \mathrm{A}^{|\textsc{Sponge}_{\mathbf{f}}\rangle} \right] + \right. \\
&\left. \underset{\mathbf{f} \xleftarrow{\$} \mathcal{S}^{\mathcal{S}}}{\mathbb{P}} \left[b = 1 : b \leftarrow \mathrm{A}^{|\textsc{Sponge}_{\mathbf{f}}\rangle} \right] - \underset{\mathbf{g} \leftarrow \mathfrak{R}}{\mathbb{P}} \left[b = 1 : b \leftarrow \mathrm{A}^{|\mathbf{g}\rangle} \right] \right|
\end{aligned} \tag{4.10}$$

$$\leq \underbrace{\left| \underset{K \overset{\$}{\leftarrow} \mathcal{K}}{\mathbb{P}} \left[b = 1 : b \leftarrow \mathrm{A}^{|\textsc{Sponge}_{\mathbf{f}_K}\rangle} \right] - \underset{\mathbf{f} \overset{\$}{\leftarrow} \mathcal{S}^{\mathcal{S}}}{\mathbb{P}} \left[b = 1 : b \leftarrow \mathrm{A}^{|\textsc{Sponge}_{\mathbf{f}}\rangle} \right] \right|}_{\leq \left| \underset{K \overset{\$}{\leftarrow} \mathcal{K}}{\mathbb{P}} \left[b=1 : b \leftarrow \mathrm{B}^{|\mathbf{f}_K\rangle} \right] - \underset{\mathbf{f} \overset{\$}{\leftarrow} \mathcal{S}^{\mathcal{S}}}{\mathbb{P}} \left[b=1 : b \leftarrow \mathrm{B}^{|\mathbf{f}\rangle} \right] \right|} +$$

$$\underbrace{\left| \underset{\mathbf{f} \overset{\$}{\leftarrow} \mathcal{S}^{\mathcal{S}}}{\mathbb{P}} \left[b = 1 : b \leftarrow \mathrm{A}^{|\textsc{Sponge}_{\mathbf{f}}\rangle} \right] - \underset{\mathbf{g} \leftarrow \mathfrak{R}}{\mathbb{P}} \left[b = 1 : b \leftarrow \mathrm{A}^{|\mathbf{g}\rangle} \right] \right|}_{\text{Quantum Indistinguishability, Theorem 4.2 or 4.9}} \leq \epsilon^{\mathrm{PR}} + \frac{\pi^2}{3} \eta^3 |\mathcal{C}|^{-1}, \tag{4.11}$$

where B is an adversary that uses A as a subroutine, simulating A's oracle using its own oracle and the sponge construction. B outputs the same output as A. □

4.4 Example proof of Lemma 4.3

In this section we prove Lemma 4.3 in a setting limited enough that every step can be done in all details. The main difficulty of our technique is of combinatorial nature, namely counting the possible values of intermediate states in multiple evaluations of Sponge. In the full proof we provide an algorithmic explanation of some steps but here we can execute these algorithms and explicitly write down their outputs.

We want to show that the probability function describing the input-output behavior of Sponge is a polynomial of bounded degree in $|\mathcal{C}|^{-1}$. By that we mean that the expression for $\mathbf{p}(|\mathcal{C}|^{-1})$ can be written as $\sum_i a_i |\mathcal{C}|^{-i}$. The proof goes as follows: Firstly we expand the event that on some inputs Sponge gives some outputs, this allows us to pinpoint the individual evaluations of $\mathbf{f}$. Secondly we impose an order on the evaluations of the internal function; which in turn allows us to exclude state values that would require f to output different values on the same input, calculate the probability of f having particular input-output behavior, and divide the set of state values in a way allowing to calculate its size. Finally we obtain a closed expression for $\mathbf{p}(|\mathcal{C}|^{-1})$.

The limitation we make in the example proof is to consider only single-query algorithms ($q = 1$). We also restrict ourselves to a *limited* Sponge that allows only 2-symbol inputs and always outputs a single element of $\mathcal{A}$. As $q = 1$ the number of input-output pairs we need to consider is 2. The array of inputs and outputs is $(\mathbf{M}, \mathbf{Z}) = ((\mathbf{M}^1, \mathbf{Z}^1), (\mathbf{M}^2, \mathbf{Z}^2))$ and for $i \in \{1, 2\} : \mathbf{M}^i = M_1^i \| M_2^i$, $\mathbf{Z}^i = Z_1^i$. In the following example $\mathbf{f} : \mathcal{S} \to \mathcal{S}$, where $\mathcal{S} := \mathcal{A} \times \mathcal{C}$.

The probabilistic event we analyze throughout this section is

$$\forall i \in [2] : \textsc{Sponge}(\mathbf{M}^i) = \mathbf{Z}^i \Leftrightarrow \forall i \in [2] : \bar{\mathbf{f}} \left(\bar{\mathbf{f}}(M_1^i, 0) \oplus M_2^i, \hat{\mathbf{f}}(M_1^i, 0) \right) = Z_1^i, \tag{4.12}$$

where by $\bar{\mathbf{f}} : \mathcal{S} \to \mathcal{A}$ and $\hat{\mathbf{f}} : \mathcal{S} \to \mathcal{C}$ we denote the outer part and the inner part of the output of $\mathbf{f}$ respectively.

In the following paragraph we are going to make explicit all inputs to $\mathbf{f}$. Particularly in Figure 4.1 we show the two evaluations of Sponge we analyze. The values of not-boxed-states are fixed by the requirement of inputs being $\mathbf{M}$ and outputs $\mathbf{Z}$. By M^i_j we denote the j-th block of $\mathbf{M}^i$, and similarly by Z^i_j the j-th block of $\mathbf{Z}^i$.

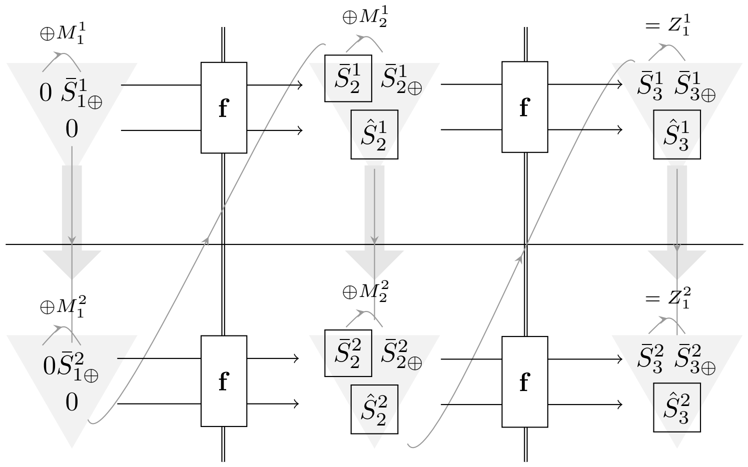


Figure 4.1: Table showing the intermediate states of the limited Sponge. Boxed states are the elements of ∇-$\mathbf{c}$ (pronounced as "nabla-c" and defined in Equation (4.16)) that do not have a fixed value across different ∇-$\mathbf{c} \in \nabla$-$\mathbf{C}(\mathbf{M}, \mathbf{Z})$, the arrows indicate the order in which Flag-Assign assigns flags.

Note that by including intermediate states we can further expand the above event, Equation (4.12). By *intermediate states* we mean the value of the state of Sponge during the calculation of Sponge$(\mathbf{M})$. Namely $\hat{S}^i_3$:

$$\begin{aligned} \forall i : & \bar{\mathbf{f}}\left(\bar{\mathbf{f}}(M^i_1, 0) \oplus M^i_2, \hat{\mathbf{f}}(M^i_1, 0)\right) = Z^i_1 \Leftrightarrow \\ \forall i : & \bigvee_{\hat{S}^i_3 \in \mathcal{C}} \mathbf{f}\left(\bar{\mathbf{f}}(M^i_1, 0) \oplus M^i_2, \hat{\mathbf{f}}(M^i_1, 0)\right) = (Z^i_1, \hat{S}^i_3). \end{aligned} \tag{4.13}$$

We are using the upper index to count the number of the evaluation of Sponge. Note that there is one inner state that we have not made explicit, the one being output by the first $\mathbf{f}$. Following the above reasoning we get

$$\forall i : \bigvee_{S^i_2 \in \mathcal{A} \times \mathcal{C}} \bigvee_{\hat{S}^i_3 \in \mathcal{C}} \mathbf{f}(M^i_1, 0) = S^i_2 \ \wedge\ \mathbf{f}(\bar{S}^i_2 \oplus M^i_2, \hat{S}^i_2) = (Z^i_1, \hat{S}^i_3), \tag{4.14}$$

where $S_2^i = (\bar{S}_2^i, \hat{S}_2^i)$, we denote the above as

$$\forall i : \bigvee_{S_2^i \in \mathcal{A}\times\mathcal{C}} \bigvee_{\hat{S}_3^i \in \mathcal{C}} \mathbf{f}(S_{1\oplus}^i) = S_2^i \ \wedge\ \mathbf{f}(S_{2\oplus}^i) = (\bar{S}_3^i, \hat{S}_3^i),$$
$$\text{where } \forall i : S_{1\oplus}^i := (M_1^i, 0) \ \wedge\ S_{2\oplus}^i := (\bar{S}_2^i \oplus M_2^i, \hat{S}_2^i) \ \wedge\ \bar{S}_3^i := Z_1^i. \tag{4.15}$$

By adding the subscript "$\oplus$" we highlight that the state output by $\mathbf{f}$ has been updated by adding the appropriate block of $\mathbf{M}$. Up to this point we have expanded the initial event from Equation (4.12) to a form with all inputs and outputs of $\mathbf{f}$ being explicit, namely $\mathbf{f}(S_1) = S_2$. From now on we are going to denote the set of the states by ∇-$\mathbf{c}$ (read as *"nabla configuration"*, where the ∇ is suggested by the three values that can be seen as vertices of a triangle), which we define as a matrix

$$\nabla\text{-}\mathbf{c} := \begin{bmatrix} \begin{pmatrix} \bar{S}_1^1 \ \bar{S}_{1\oplus}^1 \\ \hat{S}_1^1 \end{pmatrix} & \begin{pmatrix} \bar{S}_2^1 \ \bar{S}_{2\oplus}^1 \\ \hat{S}_2^1 \end{pmatrix} & \begin{pmatrix} \bar{S}_3^1 \ \bar{S}_{3\oplus}^1 \\ \hat{S}_3^1 \end{pmatrix} \\ \begin{pmatrix} \bar{S}_1^2 \ \bar{S}_{1\oplus}^2 \\ \hat{S}_1^2 \end{pmatrix} & \begin{pmatrix} \bar{S}_2^2 \ \bar{S}_{2\oplus}^2 \\ \hat{S}_2^2 \end{pmatrix} & \begin{pmatrix} \bar{S}_3^2 \ \bar{S}_{3\oplus}^2 \\ \hat{S}_3^2 \end{pmatrix} \end{bmatrix}, \tag{4.16}$$

where $\forall i : S_{1\oplus}^i = (M_1^i, 0) \ \wedge\ S_{2\oplus}^i = (\bar{S}_2^i \oplus M_2^i, \hat{S}_2^i) \ \wedge\ \bar{S}_3^i = Z_1^i = \bar{S}_{3\oplus}^i$; the constraints we impose fix the input-output behavior of the two evaluations of Sponge. Nabla-configurations ∇-$\mathbf{c}$ are matrices of triples but when we want to refer to a part of the triple in row i and column j of ∇-$\mathbf{c}$ we are going to write $S_j^i \in \nabla$-$\mathbf{c}$. More formally $S_j^i \in \nabla\text{-}\mathbf{c} \Leftrightarrow \nabla\text{-}\mathbf{c}_j^i = \begin{pmatrix} \bar{S} \ \bar{S}_\oplus \\ \hat{S} \end{pmatrix} \wedge S_j^i = (\bar{S}, \hat{S})$, where by ∇-$\mathbf{c}_j^i$ we denote the element of ∇-$\mathbf{c}$ in row i and column j, similarly for the second part of the state $S_{j\oplus}^i = (\bar{S}_{j\oplus}^i, \hat{S}_j^i)$, of the corresponding triple. We introduce this notation of ∇-$\mathbf{c}$ to capture possible values of the states in Sponge($\mathbf{M}^i$) that are consistent with $(\mathbf{M}, \mathbf{Z})$.

The set of all possible values of states in ∇-$\mathbf{c}$ is denoted by ∇-$\mathbf{C}(\mathbf{M}, \mathbf{Z})$ (*the set of nabla configurations*). The size of ∇-$\mathbf{C}(\mathbf{M}, \mathbf{Z})$ is the number of different ∇-$\mathbf{c}$ for a particular $(\mathbf{M}, \mathbf{Z})$,

$$|\,\nabla\text{-}\mathbf{C}(\mathbf{M}, \mathbf{Z})\,| = |\mathcal{A}^2 \times \mathcal{C}^4| = |\mathcal{A}|^2 \cdot |\mathcal{C}|^4. \tag{4.17}$$

In Figure 4.1, values of not-boxed-states are fixed by the requirement of inputs being $\mathbf{M}$ and outputs $\mathbf{Z}$. In what follows we analyze this set to find out how many possible values of states correspond to each value of probability of seeing ($\forall i \in [2]$: Sponge($\mathbf{M}^i$) $= \mathbf{Z}^i$). To better understand our approach we should clarify the implicit equivalence between ∇-$\mathbf{c}$—so values of the internal states of Sponge—and $\mathbf{f}$ i.e. the function taken at random from $\mathcal{S}^{\mathcal{S}}$. Note that for every $\mathbf{f} \in \mathcal{S}^{\mathcal{S}}$ we have at most a single ∇-$\mathbf{c}$, we say at most because some $\mathbf{f}$ are not consistent with the input-output pairs $(\mathbf{M}, \mathbf{Z})$. On the other hand, for a single ∇-$\mathbf{c}$ we have plenty of functions: all those that have input-output pairs consistent with values of states in ∇-$\mathbf{c}$ and any outputs on all inputs not present in ∇-$\mathbf{c}$. Also note that there are many ∇-$\mathbf{c}$ that will result in $(\forall i \in [2] : \textsc{Sponge}(\mathbf{M}^i) = \mathbf{Z}^i)$. What we do is basically counting the number of functions $\mathbf{f}$ that will result in Sponge evaluating to $\mathbf{Z}$ on $\mathbf{M}$ and dividing it by the number of all functions. The only difference is that we immediately simplify the result by not counting the functions with behavior outside of our scope—limited to few (in this section four) evaluations. This simplification is made easier by focusing on relevant values of inputs and outputs; on a few rows of the evaluation tables of $\mathbf{f}$.

The events we take the OR of are disjoint, so in terms of probabilities we get

$$\begin{aligned}&\underset{\mathbf{f}\xleftarrow{\$}\mathcal{S}^{\mathcal{S}}}{\mathbb{P}}\left[\forall i \in [2] : \textsc{Sponge}(\mathbf{M}^i) = \mathbf{Z}^i\right] \\ &= \sum_{S_2^1, S_2^2, \hat{S}_3^1, \hat{S}_3^2} \underset{\mathbf{f}\xleftarrow{\$}\mathcal{S}^{\mathcal{S}}}{\mathbb{P}}\left[\forall i \in [2] : \mathbf{f}(S^i_{1\oplus}) = S_2^i \wedge \mathbf{f}(S^i_{2\oplus}) = (\bar{S}_3^i, \hat{S}_j^i)\right], \qquad (4.18)\end{aligned}$$

where the sum is taken over $S_2^1, S_2^2 \in \mathcal{S}$ and $\hat{S}_3^1, \hat{S}_3^2 \in \mathcal{C}$ and $\bar{S}^i_{1\oplus}, \bar{S}^i_{2\oplus}, \bar{S}^i_{3\oplus}$ are constrained by $(\mathbf{M}, \mathbf{Z})$. Now that we have exposed the individual evaluations of $\mathbf{f}$ we can use the chain rule to specify the order in which we analyze the evaluations of the internal function. This order is only a tool for the analysis of the probability, not the actual time evolution. Note that the probability on the right hand side of Equation (4.18) is taken over a conjunction of events depending only on a single evaluation of $\mathbf{f}$. The next step is to extract events with a single evaluation of $\mathbf{f}$. We can do it simply by using Bayes' formula and the chain rule,

$$\underset{\mathbf{f}\xleftarrow{\$}\mathcal{S}^{\mathcal{S}}}{\mathbb{P}}\left[\textsc{Sponge}(\mathbf{M}) = \mathbf{Z}\right] = \sum_{S_2^1, S_2^2, \hat{S}_3^1, \hat{S}_3^2} \mathbb{P}[\forall i : \mathbf{f}(S^i_{1\oplus}) = S_2^i \ \wedge \ \mathbf{f}(S^i_{2\oplus}) = (\bar{S}_3^i, \hat{S}_3^i)] \qquad (4.19)$$

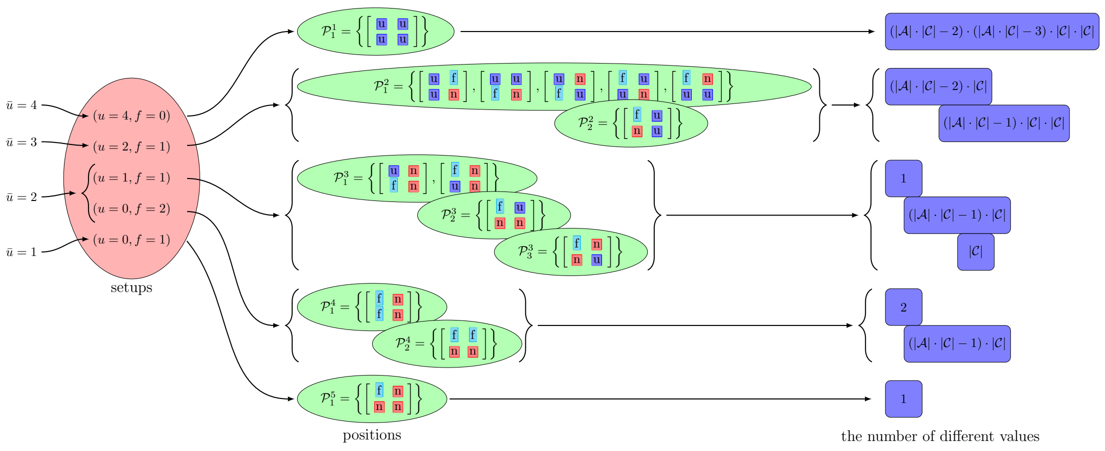
$$\begin{aligned}&= \sum_{\nabla\text{-}\mathbf{c}} \mathbb{P}[\forall i : \mathbf{f}(S^i_{1\oplus}) = S_2^i \ \wedge \ \mathbf{f}(S^i_{2\oplus}) = S_3^i; \forall i,j : S_j^i, S^i_{j\oplus} \in \nabla\text{-}\mathbf{c}] \\ &= \sum_{\nabla\text{-}\mathbf{c}} \mathbb{P}[\mathbf{f}(S^1_{1\oplus}) = S_2^1] \\ &\quad \cdot \mathbb{P}[\mathbf{f}(S^2_{1\oplus}) = S_2^2 \mid \mathbf{f}(S^1_{1\oplus}) = S_2^1] \\ &\quad \cdot \mathbb{P}[\mathbf{f}(S^1_{2\oplus}) = S_3^1 \mid \mathbf{f}(S^2_{1\oplus}) = S_2^2 \ \wedge \ \mathbf{f}(S^1_{1\oplus}) = S_2^1] \\ &\quad \cdot \mathbb{P}[\mathbf{f}(S^2_{2\oplus}) = S_3^2 \mid \mathbf{f}(S^1_{2\oplus}) = S_3^1 \ \wedge \ \mathbf{f}(S^2_{1\oplus}) = S_2^2 \ \wedge \ \mathbf{f}(S^1_{1\oplus}) = S_2^1], \qquad (4.20)\end{aligned}$$

where in the last equation we have omitted $\forall i,j : S_j^i, S^i_{j\oplus} \in \nabla$-$\mathbf{c}$ in each probability function. We denote the order specified above by "$\prec$".

We still cannot evaluate the above expression because we do not know if $\mathbf{f}$ is queried on a "fresh" input or not. First of all, note that thanks to conditioning on one event, we can treat $(\mathbf{f}(S^1_{1\oplus}) = S_2^1)$ from the second factor in Equation (4.20) as being prior to $(\mathbf{f}(S^1_{2\oplus}) = S^1_3)$. Prior in that case means that $\mathbf{f}$ is sampled on $S^1_{1\oplus}$ before it is sampled on $S^2_{1\oplus}$. That implies, e.g., that if $S^2_{1\oplus} = S^1_{1\oplus}$ then the outputs have to be the same, otherwise the probability is 0. This is what we mean when saying that an input is fresh or not. To separate a particular ∇-$\mathbf{c}$ with different numbers of fresh states, we perform a procedure on each ∇-$\mathbf{c}$ that assigns flags to the states. Flags mark whether the value of the state was previously input to $\mathbf{f}$ or not. By performing this procedure we want to divide ∇-$\mathbf{C}(\mathbf{M},\mathbf{Z})$ into subsets with the same probability—i.e. having the same probability of sampling $\mathbf{f}$ that yields a particular input-output behavior. Let us call this procedure FLAG-ASSIGN. Running it also identifies impossible values of internal states, calculates the probabilities of each transition, and divides ∇-$\mathbf{C}(\mathbf{M},\mathbf{Z})$ into sets of cardinalities we can compute.

Algorithm FLAG-ASSIGN, see Algorithm 4.2 in the next section, takes as input ∇-$\mathbf{c}$ and goes through each state starting from the first column going down, then down from the top of the second column and so on. The order in which FLAG-ASSIGN operates is depicted by arrows in Figure 4.1. If the value of the "$\oplus$" part of the state which is input to $\mathbf{f}$ appears in ∇-$\mathbf{c}$ just once, the algorithm assigns the flag "$\boxed{\mathbf{u}}$" to it, we call such states *unique*. If the value is not unique, i.e. it appears in ∇-$\mathbf{c}$ more than once, the state that is encountered first is assigned the flag "$\boxed{\mathbf{f}}$" and the rest of the states get the flag "$\boxed{\mathbf{n}}$". We call states with the flag "$\boxed{\mathbf{n}}$" *non-unique*. FLAG-ASSIGN also appends to each non-unique "$\oplus$"-state the output it should yield, i.e. the output of the corresponding "$\boxed{\mathbf{f}}$" state. If the state in ∇-$\mathbf{c}$ that follows the considered state is different than the claimed output we discard the whole configuration. We denote the set ∇-$\mathbf{C}(\mathbf{M},\mathbf{Z})$ without states that conflict with $\mathbf{f}$ being a proper map by $\mathbf{p}$-∇-$\mathbf{CF}(\mathbf{M},\mathbf{Z})$ (*set of p-nabla configurations with flags*, p emphasizes the fact that we have restricted $\mathbf{f}$ to proper transformations). By a proper map we mean that it does not output different states on the same input. Elements of $\mathbf{p}$-∇-$\mathbf{CF}(\mathbf{M},\mathbf{Z})$ are denoted by $\mathbf{p}$-∇-$\mathbf{cf}$ (*p-nabla configurations*).

After running FLAG-ASSIGN on every ∇-$\mathbf{c} \in \nabla$-$\mathbf{C}(\mathbf{M},\mathbf{Z})$ and discarding the configurations with bad output states we still need to add more details to our picture. The procedure we describe below is depicted in Figure 4.2. Firstly we discriminate between $\mathbf{p}$-∇-$\mathbf{cf}$ with different numbers of unique states. The total number of flags is 4, the final states are not inputs to $\mathbf{f}$ and are not assigned a flag. We denote the number of unique states by u, the number of states that are non-unique but appear for the first time is $\mathbf{f}$, and the number of non-unique states is n. Note that $u + f + n = 4$. In general there are 5 possible sets of those numbers in the case of $q = 1$ and lengths of the input and output strings we specified. These are as follows: $(u = 4, f = 0)$, $(u = 2, f = 1)$, $(u = 1, f = 1)$, $(u = 0, f = 2)$, and $(u = 0, f = 1)$. Secondly we discriminate between different placements of flags. For each setup there are several possible placements of flags. For $(u = 4, f = 0)$ and $(u = 0, f = 1)$, flags can be set in only one configuration. If we have 2 unique states and the setup is $(u = 2, f = 1)$ then there are 6 possible configurations of flags. For $(u = 1, f = 1)$ there are 4 and for $(u = 0, f = 2)$ only 2. While calculating the number of configurations it is important to remember that the flag of the first state $S^1_{1\oplus}$ is either **u** or **f**. There are some details of how to find the placements but they are made explicit only in the full proof of the lemma. All possible placements are depicted in Figure 4.2. The last step is calculating the number of distinct $\mathbf{p}\text{-}\nabla\text{-}\mathbf{cf}$. We consider



$\bar{u} = 4$
$\bar{u} = 3$
$\bar{u} = 2$
$\bar{u} = 1$
$(u = 4, f = 0)$
$(u = 2, f = 1)$
$(u = 1, f = 1)$
$(u = 0, f = 2)$
$(u = 0, f = 1)$
setups
$\mathcal{P}^1_1$
$\mathcal{P}^2_1$
$\mathcal{P}^2_2$
$\mathcal{P}^3_1$
$\mathcal{P}^3_2$
$\mathcal{P}^3_3$
$\mathcal{P}^4_1$
$\mathcal{P}^4_2$
$\mathcal{P}^5_1$
positions
$(|\mathcal{A}| \cdot |\mathcal{C}| - 2) \cdot (|\mathcal{A}| \cdot |\mathcal{C}| - 3) \cdot |\mathcal{C}| \cdot |\mathcal{C}|$
$(|\mathcal{A}| \cdot |\mathcal{C}| - 2) \cdot |\mathcal{C}|$
$(|\mathcal{A}| \cdot |\mathcal{C}| - 1) \cdot |\mathcal{C}| \cdot |\mathcal{C}|$
1
$(|\mathcal{A}| \cdot |\mathcal{C}| - 1) \cdot |\mathcal{C}|$
$|\mathcal{C}|$
2
$(|\mathcal{A}| \cdot |\mathcal{C}| - 1) \cdot |\mathcal{C}|$
1
the number of different values

Figure 4.2: Possible placements for the limited SPONGE. $\begin{bmatrix} \cdot & \cdot \\ \cdot & \cdot \end{bmatrix}$ denotes the flags assigned to the four states. $\bar{u}$ denotes the total number of unique states: $\bar{u} := u + f$.

the number of outputs of $\mathbf{f}$ consistent with the specified placement. In the first case in Figure 4.2 we have $(|\mathcal{A}| \cdot |\mathcal{C}| - 2)(|\mathcal{A}| \cdot |\mathcal{C}| - 3)$ values of states in the second column in Figure 4.1. This value comes from the fact that the first output is distinct from the two values in the first column and the second output is also distinct form the first output. The mentioned value is multiplied by $|\mathcal{C}|^2$ for the inner parts of the last states in $\mathbf{p}\text{-}\nabla\text{-}\mathbf{cf}$, this is true with the assumption that $(\mathbf{Z}^1_1, \mathbf{Z}^2_1)$ are distinct from each other and from $\mathbf{M}$. If $\mathbf{Z}$ does not fulfill this assumption the number of possible values is just instead multiplied by $|\mathcal{C}| - 1$. Whenever a state is not unique, then the number of values that can be output by $\mathbf{f}$ on it is 1. With such reasoning we calculate all the values in the last column of Figure 4.2.

In most steps we perform using FLAG-ASSIGN, the distinction between **u** and **f** seems unimportant. We will need it to properly identify different placements of flags in $\mathbf{p}\text{-}\nabla\text{-}\mathbf{CF}(\mathbf{M}, \mathbf{Z})$, but indeed in all other tasks one can treat them as a single "unique" flag.

The next step is calculating the number of values that can be assigned to states in a given setup and for a given placement of flags. We calculated those numbers assuming it is possible to have such placement. This assumption is not always fulfilled as particular messages and outputs exclude some options. For example, if both messages start with the same symbols then all positions where the two first states are unique are impossible. By CALC we denote the algorithm calculating the cardinality of subsets of $\mathbf{p}\text{-}\nabla\text{-}\mathbf{CF}(\mathbf{M}, \mathbf{Z})$, the details of CALC are specified below in Algorithm 4.3. It goes through a single placement of flags. The basic rules of its operation are: for every unique flag that maps to a unique flag multiply the result by $|\mathcal{A}| \cdot |\mathcal{C}|$, for every unique flag that maps to a non-unique flag multiply the result by 1, for every non-unique flag that maps to any state multiply the result by 1, and for every unique flag in the last column of flagged states multiply the result by $|\mathcal{C}|$. The first two rules are adjusted a bit to keep track of non-repeating unique states and allow for multiple values of non-unique states respectively. The last column in Figure 4.2 lists the results of CALC for placements in the respective rows. If the squeezing phases were longer we would have to account for the fact that the outer part of the state can be either unique or not which slightly changes the final outcome.

Now we want to show that the probability function $\mathbf{p}$ is a polynomial in $|\mathcal{C}|^{-1}$. Up to this point we have shown that

$$\sum_{\nabla\text{-}\mathbf{c}} \prod_{(i,j)=(1,1)}^{(2,2)} \mathbb{P}\Bigg[\mathbf{f}(S^i_{j\oplus}) = S^i_{j+1} \mid \bigwedge_{(i',j')\prec(i,j)} \mathbf{f}(S^{i'}_{j'\oplus}) = S^{i'}_{j'+1}; \forall i,j,i',j' : S^i_j, S^i_{j\oplus} \in \nabla\text{-}\mathbf{c}\Bigg]$$
$$= \sum_{\mathbf{p}\text{-}\nabla\text{-}\mathbf{cf}} \prod_{(i,j)=(1,1)}^{(2,2)} \mathbb{P}\Bigg[\mathbf{f}(S^i_{j\oplus}) = S^i_{j+1} \mid \bigwedge_{(i',j')\prec(i,j)} \mathbf{f}(S^{i'}_{j'\oplus}) = S^{i'}_{j'+1}; \forall i,j,i',j' : S^i_j, S^i_{j\oplus} \in \mathbf{p}\text{-}\nabla\text{-}\mathbf{cf}\Bigg], \tag{4.21}$$

where the order "$\prec$" is the same as in Equation (4.20). In the above equation we have discarded those $\nabla\text{-}\mathbf{c}$ that require $\mathbf{f}$ to output different states on the same input, because the probability is then 0. The sum in Equation (4.21) can be expanded to

$$\sum_{\mathbf{p}\text{-}\nabla\text{-}\mathbf{cf}} \cdots = \sum_{u=4} \cdots + \sum_{u=0}^{2} \sum_{f=1}^{\lfloor(4-u)/2\rfloor} \sum_{P\in\mathcal{P}(u,f)} \sum_{\mathbf{p}\text{-}\nabla\text{-}\mathbf{cf}(u,f,P)} \cdots, \tag{4.22}$$

where by $\mathbf{p}\text{-}\nabla\text{-}\mathbf{cf}(u, f, P)$ we denote the configuration with the number of unique and non-unique states and placement fixed. $\mathcal{P}(u, f)$ is the set of all placements in which the flags can be arranged given the number of unique and non-unique states. We omitted the input $(\mathbf{M}, \mathbf{Z})$ to $\mathcal{P}$ for brevity. Making use of information from Figure 4.2 we can now evaluate expression (4.21). Note that setting u and $\mathbf{f}$ to some particular values allows us to evaluate the probabilities. Denoting the total number of unique states by $\bar{u}$ we get that

$$\prod_{(i,j)=(1,1)}^{(2,2)} \mathbb{P}\left[\mathbf{f}(S^i_{j\oplus}) = S^i_{j+1} \mid \bigwedge_{(i',j')\prec(i,j)} \mathbf{f}(S^{i'}_{j'\oplus}) = S^{i'}_{j'+1}\right] = (|\mathcal{A}| \cdot |\mathcal{C}|)^{-\bar{u}} \tag{4.23}$$

for $\mathbf{p}$-∇-$\mathbf{cf}$ with $u + f = \bar{u}$. Finally we arrive at

$$\begin{aligned}\mathbf{p}(|\mathcal{C}|^{-1}) = &\sum_{P\in\mathcal{P}^1_1} (|\mathcal{A}|\cdot|\mathcal{C}|)^{-4} \cdot (|\mathcal{A}|\cdot|\mathcal{C}|-2)(|\mathcal{A}|\cdot|\mathcal{C}|-3)|\mathcal{C}|^2\boldsymbol{\delta}(P)\\ &+ \sum_{P\in\mathcal{P}^2_1} (|\mathcal{A}|\cdot|\mathcal{C}|)^{-3}\cdot(|\mathcal{A}|\cdot|\mathcal{C}|-2)|\mathcal{C}|\boldsymbol{\delta}(P)\\ &+ \sum_{P\in\mathcal{P}^2_2} (|\mathcal{A}|\cdot|\mathcal{C}|)^{-3}\cdot(|\mathcal{A}|\cdot|\mathcal{C}|-1)|\mathcal{C}|^2\boldsymbol{\delta}(P)\\ &+ \sum_{P\in\mathcal{P}^3_1} (|\mathcal{A}|\cdot|\mathcal{C}|)^{-2}\cdot\boldsymbol{\delta}(P) + \sum_{P\in\mathcal{P}^3_2} (|\mathcal{A}|\cdot|\mathcal{C}|)^{-2}\cdot(|\mathcal{A}|\cdot|\mathcal{C}|-1)|\mathcal{C}|\boldsymbol{\delta}(P)\\ &+ \sum_{P\in\mathcal{P}^3_3} (|\mathcal{A}|\cdot|\mathcal{C}|)^{-2}\cdot|\mathcal{C}|\boldsymbol{\delta}(P) + \sum_{P\in\mathcal{P}^4_1} (|\mathcal{A}|\cdot|\mathcal{C}|)^{-2}\cdot 2\boldsymbol{\delta}(P)\\ &+ \sum_{P\in\mathcal{P}^4_2} (|\mathcal{A}|\cdot|\mathcal{C}|)^{-2}\cdot(|\mathcal{A}|\cdot|\mathcal{C}|-1)|\mathcal{C}|\boldsymbol{\delta}(P) + \sum_{P\in\mathcal{P}^5_1} (|\mathcal{A}|\cdot|\mathcal{C}|)^{-1}\cdot\boldsymbol{\delta}(P),\end{aligned} \tag{4.24}$$

where the sets are denoted as in Figure 4.2. The function appearing in the above equation is defined as

$$\boldsymbol{\delta}(\mathbf{M},\mathbf{Z},P) := \begin{cases} 0 & \text{if } \mathbf{p}\text{-}\nabla\text{-}\mathbf{CF}(\mathbf{M},\mathbf{Z})(u,f,P) = \emptyset \\ 1 & \text{otherwise}\end{cases}, \tag{4.25}$$

where in Equation (4.24) we omitted the input of $(\mathbf{M},\mathbf{Z})$ for readability.

The degree of $\mathbf{p}(|\mathcal{C}|^{-1})$ is at most 2, as claimed for messages of length 2. The coefficient for $|\mathcal{C}|^0$ is

$$a_0 = \sum_{P\in\mathcal{P}^1_1} |\mathcal{A}|^{-2}\,\boldsymbol{\delta}(P) + \sum_{P\in\mathcal{P}^2_2} |\mathcal{A}|^{-2}\,\boldsymbol{\delta}(P) + \sum_{P\in\mathcal{P}^3_2} |\mathcal{A}|^{-1}\,\boldsymbol{\delta}(P) + \sum_{P\in\mathcal{P}^4_2} |\mathcal{A}|^{-1}\,\boldsymbol{\delta}(P). \tag{4.26}$$

Let us recapitulate the results of this section. First we characterized the possible internal functions by the outputs of their consecutive evaluations, Equation (4.19). Secondly we captured the features of the intermediate states that determine the probability of seeing a particular input-output behavior, Equation (4.23). Finally we calculated an explicit formula for the probability function, Equation (4.24), (4.26).

4.5 Proof of Lemma 4.3

In this section we give the complete proof of Lemma 4.3 for the general case of $q \geq 1$ queries the adversary makes and message lengths bounded by some m, not fixed to 2 like in the previous section. In subSection 4.5.1 we expand the probability expression to encompass all intermediate states of $(\forall i \in [2q] : \textsc{Sponge}_{\mathbf{f}}(\mathbf{M}^i, \ell_i) = \mathbf{Z}^i)$ and individual evaluations of $\mathbf{f}$. In subSection 4.5.2 we introduce the concept of unique states to evaluate the probabilities of $\mathbb{P}[\mathbf{f}(S_1) = S_2]$. In subSection 4.5.3 we define the algorithm that calculates the cardinality of the set of intermediate states—and equivalently inner functions—consistent with given characteristics. In subSection 4.5.4 we conclude the proof and provide the final expression for the probability of an input-output pair under a random $\textsc{Sponge}_{\mathbf{f}}$.

We omit the padding function of the sponge construction. This is done without loss of generality since we can just say that all the considered messages are in fact messages after padding and we do not use any properties of the padding in the proof. Also we focus on q evaluations of $\textsc{Sponge}$ instead of $2q$ to improve readability.

4.5.1 Expansion of the probability function

In this section we expand the probability function to the point that all intermediate states are accounted for. We consider the event

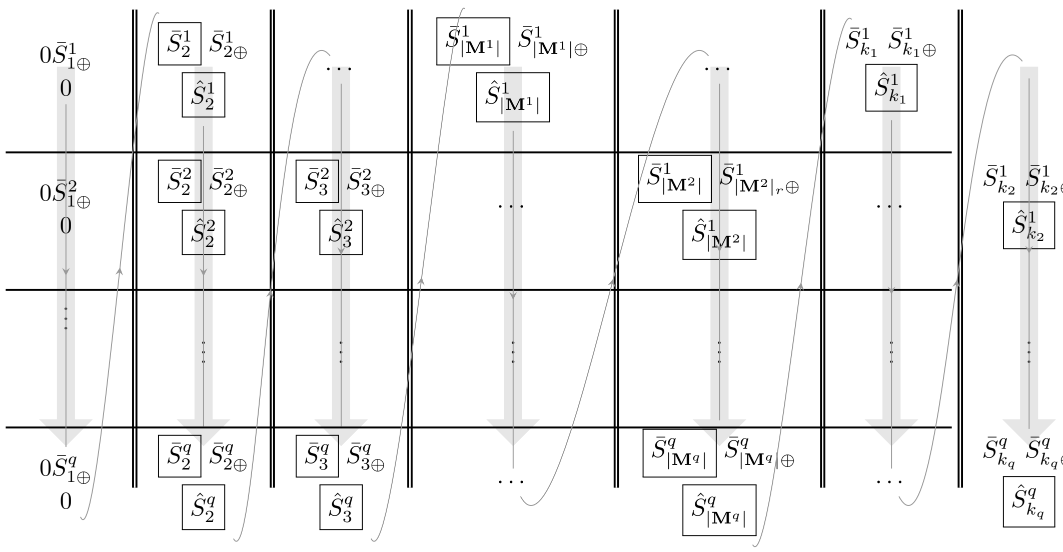
$$\left(\forall i \in [q] : \textsc{Sponge}_{\mathbf{f}}(\mathbf{M}^i, \ell_i) = \mathbf{Z}^i\right)$$

and then include the states that appear between consecutive evaluations of $\mathbf{f}$.

To keep track of the states we introduce the following notation. By the upper-index we denote the number of evaluations of $\textsc{Sponge}$, going from 1 to q. The lower index corresponds to the number of evaluations of $\mathbf{f}$ in the i-th calculation of $\textsc{Sponge}$. A state occurring during the calculation on $\mathbf{M}^i$ that is the input to the j-th evaluation of $\mathbf{f}$ is denoted by $S^i_{j\oplus}$. The output of that evaluation is S^i_{j+1}. States traversed in q evaluations of $\textsc{Sponge}$ can be represented by an array with q rows with $|\mathbf{M}^i| + |\mathbf{Z}^i|$ columns each. By *array* we mean a 2-dimensional matrix with unequal length of rows.

We call an array like that with values assigned to every state a *nabla configuration* ∇-$\mathbf{c}$. ∇ symbolizes the triangle shape in which we put states between evaluations of $\mathbf{f}$, each corner being an outer or inner part of the state. Now we define ∇-$\mathbf{c}$ relative to input-output pairs $(\mathbf{M}, \mathbf{Z})$. The size of the array is determined by the number of blocks in $\mathbf{M}^i$ and $\mathbf{Z}^i$.

Definition 4.6 (∇-$\mathbf{c}$)**.** *The nabla configuration* ∇-$\mathbf{c}$ *for* $(\mathbf{M}, \mathbf{Z})$ *is an array of triples* $\begin{pmatrix} \bar{S} \ \bar{S}_{\oplus} \\ \hat{S} \end{pmatrix} \in \mathcal{A}^2 \times \mathcal{C}$, *where* $\mathcal{C}$ *is an arbitrary non-empty finite set. The array* ∇-$\mathbf{c}$



$0\bar{S}^1_{1\oplus}$
0
$\bar{S}^1_2$ $\bar{S}^1_{2\oplus}$
$\hat{S}^1_2$
$\bar{S}^1_{|\mathbf{M}^1|}$ $\bar{S}^1_{|\mathbf{M}^1|\oplus}$
$\hat{S}^1_{|\mathbf{M}^1|}$
$\bar{S}^1_{k_1}$ $\bar{S}^1_{k_1\oplus}$
$\hat{S}^1_{k_1}$
$0\bar{S}^2_{1\oplus}$
0
$\bar{S}^2_2$ $\bar{S}^2_{2\oplus}$
$\hat{S}^2_2$
$\bar{S}^2_3$ $\bar{S}^2_{3\oplus}$
$\hat{S}^2_3$
$\bar{S}^1_{|\mathbf{M}^2|}$ $\bar{S}^1_{|\mathbf{M}^2|_r\oplus}$
$\hat{S}^1_{|\mathbf{M}^2|}$
$\bar{S}^1_{k_2}$ $\bar{S}^1_{k_2\oplus}$
$\hat{S}^1_{k_2}$
$0\bar{S}^q_{1\oplus}$
0
$\bar{S}^q_2$ $\bar{S}^q_{2\oplus}$
$\hat{S}^q_2$
$\bar{S}^q_3$ $\bar{S}^q_{3\oplus}$
$\hat{S}^q_3$
$\bar{S}^q_{|\mathbf{M}^q|}$ $\bar{S}^q_{|\mathbf{M}^q|\oplus}$
$\hat{S}^q_{|\mathbf{M}^q|}$
$\bar{S}^q_{k_q}$ $\bar{S}^q_{k_q\oplus}$
$\hat{S}^q_{k_q}$

Figure 4.3: Table depicting states traversed in q evaluations of Sponge. Vertical lines signify evaluations of the internal function.

consists of q rows. For every i, row i has k_i columns where $k_i := |\mathbf{M}^i| + |\mathbf{Z}^i|$ ($|\mathbf{M}^i|$ denotes the number of symbols in $\mathbf{M}^i$). Formally we have

$$\nabla\text{-}\mathbf{c} := \left[\begin{pmatrix} \bar{S}^i_j \; \bar{S}^i_{j\oplus} \\ \hat{S}^i_j \end{pmatrix}\right]_{\substack{i\in[q] \\ j\in[k_i]}} . \tag{4.27}$$

To refer to the element of ∇-$\mathbf{c}$ that lies in row i and column j we write ∇-$\mathbf{c}^i_j$. To refer to parts of the triple that lies in row i and column j we write

$$S^i_j = (a,b) \in \nabla\text{-}\mathbf{c} \Leftrightarrow \nabla\text{-}\mathbf{c}^i_j = \begin{pmatrix} a \; \bar{S}_\oplus \\ b \end{pmatrix} \text{ for some } \bar{S}_\oplus \tag{4.28}$$

$$S^i_{j\oplus} = (a,b) \in \nabla\text{-}\mathbf{c} \Leftrightarrow \nabla\text{-}\mathbf{c}^i_j = \begin{pmatrix} \bar{S} \; a \\ b \end{pmatrix} \text{ for some } \bar{S}. \tag{4.29}$$

Let us define the number of evaluations of $\mathbf{f}$ in ∇-$\mathbf{c}$ for $(\mathbf{M}, \mathbf{Z})$ as

$$\kappa := \sum_{i=1}^{q} (k_i - 1). \tag{4.30}$$

We denote the number of triples in ∇-$\mathbf{c}$ by $|\nabla\text{-}\mathbf{c}|$. With coefficient defined above we have $|\nabla\text{-}\mathbf{c}| = \kappa + q$.

To make good use of the newly introduced concept of nabla configurations $\nabla\text{-}\mathbf{c}$ we want to restrict the set of arrays we discuss. We want to put constraints on the set of $\nabla\text{-}\mathbf{c}$ to make explicit the requirement that states correspond to a correct input-output behavior of Sponge. The *set of* $\nabla\text{-}\mathbf{c}$ *for* $(\mathbf{M}, \mathbf{Z})$ is defined as follows.

Definition 4.7 ($\nabla\text{-}\mathbf{C}(\mathbf{M}, \mathbf{Z})$)**.** *The set of nabla configurations* $\nabla\text{-}\mathbf{c}$ *for* $(\mathbf{M}, \mathbf{Z})$ *is a set of arrays of size specified by* $(\mathbf{M}, \mathbf{Z})$*,* $\nabla\text{-}\mathbf{C}(\mathbf{M}, \mathbf{Z}) \subset (\mathcal{A}^2 \times \mathcal{C})^{\kappa+q}$*. We define* $\nabla\text{-}\mathbf{C}(\mathbf{M}, \mathbf{Z})$ *by the following constraints*

$$\forall i \in [q] : \hat{S}_1^i = 0, \tag{4.31}$$

$$\forall i \in [q] : \bar{S}_1^i = 0, \tag{4.32}$$

$$\forall i \in [q], 1 \leq j \leq |\mathbf{M}^i| : \bar{S}_{j\oplus}^i = \bar{S}_j^i \oplus M_j^i, \tag{4.33}$$

$$\forall i \in [q], |\mathbf{M}^i| < j \leq k_i : \bar{S}_{j\oplus}^i = \bar{S}_j^i = Z_{j-|\mathbf{M}^i|}^i. \tag{4.34}$$

The formal definition reads

$$\nabla\text{-}\mathbf{C}(\mathbf{M}, \mathbf{Z}) := \{\nabla\text{-}\mathbf{c} \text{ for } (\mathbf{M}, \mathbf{Z}) : \nabla\text{-}\mathbf{c} \text{ fulfills constraints (4.31)}\}. \tag{4.35}$$

In the following we assume that rows of all $\nabla\text{-}\mathbf{c} \in \nabla\text{-}\mathbf{C}(\mathbf{M}, \mathbf{Z})$ are initially sorted according to the following relation. We arrange $(\mathbf{M}^i, \mathbf{Z}^i)$ in non-decreasing order in terms of length, so $\forall i < j : k_i \leq k_j$, this also means that rows of $\nabla\text{-}\mathbf{c}$ are ordered in this way.

The set $\nabla\text{-}\mathbf{C}(\mathbf{M}, \mathbf{Z})$ can be seen as a collection of arrays (that can be viewed as sheets with states written down on them) filled with possible values of the internal states, like the one in Figure 4.4.

Having established the notation we move on to realizing the goal of this section: rewriting the probability function in a suitable way for further analysis. In the following when we consider $\left(\mathbf{f}(S_{j\oplus}^i) = S_{j+1}^i\right)$ for some $\nabla\text{-}\mathbf{c}$ we leave implicit that $S_{j\oplus}^i, S_{j+1}^i \in \nabla\text{-}\mathbf{c}$. We have that

$$\forall i \in [q] : \textsc{Sponge}(\mathbf{M}^i) = \mathbf{Z}^i$$

$$\Leftrightarrow \forall i \in [q] : \bigvee_{\nabla\text{-}\mathbf{c} \in \nabla\text{-}\mathbf{C}(\mathbf{M},\mathbf{Z})} \left(\mathbf{f}(S_{1\oplus}^i) = S_2^i\right) \wedge \cdots \wedge \left(\mathbf{f}(S_{(k_i-1)\oplus}^i) = S_{k_i}^i\right) \tag{4.36}$$

$$\Leftrightarrow \bigvee_{\nabla\text{-}\mathbf{c} \in \nabla\text{-}\mathbf{C}(\mathbf{M},\mathbf{Z})} \bigwedge_{i=1}^{q} \bigwedge_{j=1}^{k_i-1} \left(\mathbf{f}(S_{j\oplus}^i) = S_{j+1}^i\right). \tag{4.37}$$

In the above equations we first include the intermediate states and then combine all evaluations of $\mathbf{f}$. In the following we make use of the fact that the events we take the disjunction of are independent and the logical disjunction turns into a sum of the probability.

$$\mathop{\mathbb{P}}_{\mathbf{f} \xleftarrow{\$} \mathcal{S}^{\mathcal{S}}} \left[\forall i \in [q] : \textsc{Sponge}(\mathbf{M}^i) = \mathbf{Z}^i\right] = \mathbb{P}\left[\bigvee_{\nabla\text{-}\mathbf{c}} \bigwedge_{i=1}^{q} \bigwedge_{j=1}^{k_i-1} \left(\mathbf{f}(S_{j\oplus}^i) = S_{j+1}^i\right)\right]$$

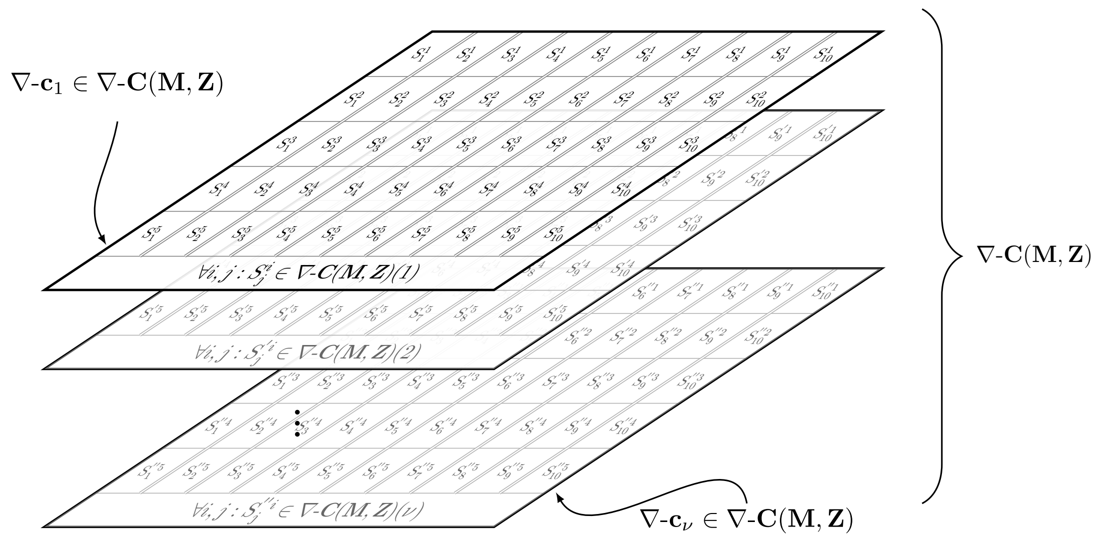

∇-c₁ ∈ ∇-C(M, Z)
∀i, j : S_j^i ∈ ∇-C(M, Z)(1)
∀i, j : S_j'^i ∈ ∇-C(M, Z)(2)
∀i, j : S_j''^i ∈ ∇-C(M, Z)(ν)
∇-C(M, Z)
∇-c_ν ∈ ∇-C(M, Z)

Figure 4.4: A pictorial description of the collection of arrays that is $\nabla\text{-}\mathbf{C}(\mathbf{M},\mathbf{Z}) = \{\nabla\text{-}\mathbf{c}_1, \nabla\text{-}\mathbf{c}_2, \dots, \nabla\text{-}\mathbf{c}_\nu\}$.

$$= \sum_{\nabla\text{-}\mathbf{c}\in\nabla\text{-}\mathbf{C}(\mathbf{M},\mathbf{Z})} \mathbb{P}\left[\bigwedge_{i=1}^{q}\bigwedge_{j=1}^{k_i-1}\left(\mathbf{f}(S^i_{j\oplus}) = S^i_{j+1}\right)\right]. \tag{4.38}$$

To further extract an expression involving the probability of a single $\left(\mathbf{f}(S^i_{j\oplus}) = S^i_{j+1}\right)$ we use Bayes' rule. By a chain of conditions we want to arrive at a function we can evaluate in the end. At this point we want to choose a particular order of the $\left(\mathbf{f}(S^i_{j\oplus}) = S^i_{j+1}\right)$ events. Let us define the order $\prec$ as

$$(i,j) \prec (i',j') \Leftrightarrow (j < j') \vee (j = j' \wedge i < i'). \tag{4.39}$$

The above rule imposes an order that begins with the top-left corner of a ∇-$\mathbf{c}$ and proceeds downwards to the end of the column to continue from the second column from the left etc.

$$\begin{aligned}
\mathbf{p}(|\mathcal{C}|^{-1}) &= \sum_{\nabla\text{-}\mathbf{c}\in\nabla\text{-}\mathbf{C}(\mathbf{M},\mathbf{Z})} \mathbb{P}\left[\bigwedge_{i=1}^{q}\bigwedge_{j=1}^{k_i-1}\left(\mathbf{f}(S^i_{j\oplus}) = S^i_{j+1}\right)\right] \\
&= \sum_{\nabla\text{-}\mathbf{c}} \mathbb{P}\left[\left(\mathbf{f}(S^q_{(k_q-1)\oplus}) = S^q_{k_q}\right) \mid \bigwedge_{(i,j)\prec(q,k_q-1)} \left(\mathbf{f}(S^i_{j\oplus}) = S^i_{j+1}\right)\right] \\
&\cdot \mathbb{P}\left[\bigwedge_{(i,j)\prec(q,k_q-1)} \left(\mathbf{f}(S^i_{j\oplus}) = S^i_{j+1}\right)\right]
\end{aligned}$$

$$= \sum_{\nabla\text{-}\mathbf{c}\in\nabla\text{-}\mathbf{C}(\mathbf{M},\mathbf{Z})} \prod_{(i,j)=(1,1)}^{(q,k_q-1)} \mathbb{P}\left[\left(\mathbf{f}(S^i_{j\oplus}) = S^i_{j+1}\right) \mid \bigwedge_{(i',j')\prec(i,j)} \left(\mathbf{f}(S^{i'}_{j'\oplus}) = S^{i'}_{j'+1}\right)\right]. \tag{4.40}$$

In the case that there is no state $(q-1, k_q-1)$ we just take the next state preceding (q, k_q-1) in the order given by Equation (4.39).

Up to this point we have performed some transformations of the event $(\forall i \in [q] : \text{Sponge}_\mathbf{f}(\mathbf{M}^i, \ell_i) = \mathbf{Z}^i)$, but we did not address the issue of the constraints put on elements of $\nabla\text{-}\mathbf{C}(\mathbf{M},\mathbf{Z})$. Is it correct to consider state values in evaluations of Sponge instead of different $\mathbf{f}$—are we in fact discussing the probability over the random choice of the internal function? The answer to this question is "yes", because of the equivalence of every $\nabla\text{-}\mathbf{c}$ with some set of $\mathbf{f}$. We can treat the input-output pairs for $\mathbf{f}$ assigned in $\nabla\text{-}\mathbf{c}$ as values in the function table of $\mathbf{f}$. By picking a single $\nabla\text{-}\mathbf{c}$ we fix at most κ rows of this table. As we sample $\mathbf{f}$ uniformly at random we are interested in the fraction of functions that are consistent with the input-output pairs $(\mathbf{M},\mathbf{Z})$ among all functions. Note however, that we only care about κ evaluations of $\mathbf{f}$ and all the details of those future evaluations are implicitly simplified in this fraction. This simplification allows us to focus only on the part of the function table corresponding to the few evaluations made in q queries and that is exactly $\nabla\text{-}\mathbf{c}$. The summing over nabla configurations $\nabla\text{-}\mathbf{c}$ corresponds to different values of the function table that are still consistent with $(\mathbf{M},\mathbf{Z})$.

The probability $\mathbb{P}\left[\left(\mathbf{f}(S^i_{j\oplus}) = S^i_{j+1}\right) \mid \bigwedge_{(i',j')\prec(i,j)} \left(\mathbf{f}(S^{i'}_{j'\oplus}) = S^{i'}_{j'+1}\right)\right]$ equals either $\frac{1}{|\mathcal{A}|\cdot|\mathcal{C}|}$ or 1 or 0: If the internal function is queried on a "fresh" input, it outputs any value with uniform probability. If on the other hand it is queried on the same input for the second time, it outputs the value it has output before with probability 1. One might think that the proof is finished, $\mathbf{p}(\lambda) = \sum_i \mathbf{w}_i(\lambda)$, where $\mathbf{w}_i$ are monomials in λ of degree up to $\kappa + q$. There is one problem with that reasoning, namely that the sum limits depend on the variable λ. Up until now we have shown that $\mathbf{p}(\lambda) = \sum_{i=1}^{\mathbf{v}(1/\lambda)} \mathbf{w}_i(\lambda)$, where $\mathbf{v}$ is another polynomial. Even for $\mathbf{v} = \mathbf{id}$ (the identity function) the degree of $\mathbf{p}$ is different from the maximal degree of $\mathbf{w}_i$. This means that we have to analyze the expression derived in Equation (4.40) in more detail. To this end, we add more structure to $\nabla\text{-}\mathbf{C}(\mathbf{M},\mathbf{Z})$ which will make it easier to count the number of values that the intermediate states can assume, i.e. the number of nabla configurations $\nabla\text{-}\mathbf{c}$ in $\nabla\text{-}\mathbf{C}(\mathbf{M},\mathbf{Z})$.

4.5.2 Unique and non-unique states

The goal of this section is to evaluate
$\mathbb{P}\left[\left(\mathbf{f}(S^i_{j\oplus}) = S^i_{j+1}\right) \mid \bigwedge_{(i',j')\prec(i,j)} \left(\mathbf{f}(S^{i'}_{j'\oplus}) = S^{i'}_{j'+1}\right)\right]$ for any $\nabla\text{-}\mathbf{c}$ and any $(\mathbf{M},\mathbf{Z})$. We approach this problem by recognizing which states in a particular $\nabla\text{-}\mathbf{c}$ are

fed to **f** once and which are repeated. We define an algorithm that includes the information about *uniqueness* of the intermediate states in ∇-**c**. The notion of uniqueness is derived relative to the events we condition on in Equation (4.40), that is why we took special care of the order in which we use the chain rule.

In this section we introduce two algorithms PREP and FLAG-ASSIGN. The former is an auxiliary algorithm that prepares the array ∇-**c** for further analysis. The latter algorithm assigns flags to states in ∇-**c**. Flags signify if a state appears once or more in the array. We use an algorithmic definition to explicitly show every step of the procedure.

Algorithm 4.1 takes as input an array ∇-**c** and groups its elements according to the value input to **f**. An important detail is the sorting rule among states with the same "$\oplus$"-state value; we use the order defined in Equation (4.39). The output of Algorithm 4.1 PREP(∇-**c**) is a vector (1-dimensional matrix), to access its l-th element we write ∇-$\mathbf{c}_l$.

Algorithm 4.1 PREP

```
    input: $\nabla$-c for $(\mathbf{M}, \mathbf{Z})$
    output: $\widetilde{\nabla\text{-}\mathbf{c}}$
 1: $\widetilde{\nabla\text{-}\mathbf{c}} := \nabla$-c, append three work spaces to each element of $\widetilde{\nabla\text{-}\mathbf{c}}$
 2: for all $1 \le i \le q, 1 \le j \le k_i - 1$ do
 3:     $\widetilde{\nabla\text{-}\mathbf{c}}^i_j = \left(\nabla\text{-}\mathbf{c}^i_j, \texttt{index}, \oplus\texttt{-state}, \texttt{image}\right) := \left(\nabla\text{-}\mathbf{c}^i_j, (i,j), S^i_{j\oplus}, S^i_{j+1}\right)$
 4: Sort $\widetilde{\nabla\text{-}\mathbf{c}}$ primarily according to the third entry and secondarily according
    to the second entry (using the order defined in Equation (4.39)).
 5: return $\widetilde{\nabla\text{-}\mathbf{c}}$
```

The main contribution of this subsection is Algorithm 4.2 which adds to each ∇-**c** information about the repetitions of the internal states. Running PREP groups the state values. The next step is to assign specific flags to states that are first (according to a specified rule) in each group. To each $S^i_{j\oplus}$ we will assign a flag, $\boxed{\mathrm{u}}$ for unique states, $\boxed{\mathrm{n}}$ for non-unique states, and $\boxed{\mathrm{f}}$ for states that appear twice or more in total but from our perspective it is their first appearance. The output of Algorithm 4.2 is FLAG-ASSIGN(∇-**c**) $=$ ∇-**cf** ("nabla configuration with flags") and $\forall i,j : \nabla\text{-}\mathbf{cf}^i_j = (\boxed{^{\mathrm{F}}}\, \nabla\text{-}\mathbf{c}^i_j, S)$, where the first register is the whole state between evaluations together with the assigned flag $\boxed{\mathrm{F}} \in \{\boxed{\mathrm{u}}, \boxed{\mathrm{f}}, \boxed{\mathrm{n}}\}$ of **f** and S is the corresponding image. To refer to the l-th register of ∇-$\mathbf{c}^i_j$ we write ∇-$\mathbf{c}^i_j(l)$. Flag $\boxed{\mathrm{f}}$ is important when discussing the relative position of unique flags ($\boxed{\mathrm{u}}$ or $\boxed{\mathrm{f}}$) in the array of ∇-**cf**. In the end of this section and in the beginning of the next section we are not going to need this distinction but it will become important when analyzing the final probability expression.

Let us define a simple function acting on elements of arrays ∇-**cf** output by

Algorithm 4.2 Flag-Assign

```
    input: $\nabla$-c for $(\mathbf{M}, \mathbf{Z})$
    output: $\nabla$-cf
 1: $\nabla\text{-}\mathbf{cf} = \emptyset$
 2: $\widetilde{\nabla\text{-}\mathbf{c}} := \textsc{Prep}(\nabla\text{-}\mathbf{c})$
 3: Set counter $l := 1$
 4: while $l \leq |\widetilde{\nabla\text{-}\mathbf{c}}| = \kappa + q$ do
 5:     Set counter $i := 1$                    ▷ the number of states with the same value
 6:     while $\widetilde{\nabla\text{-}\mathbf{c}}_{l+i}(3) = \widetilde{\nabla\text{-}\mathbf{c}}_l(3)$ do
 7:         $i := i + 1$
 8:     if $i = 1$ then
 9:         $\begin{pmatrix} \bar{S}\ \bar{S}_{\oplus} \\ \hat{S} \end{pmatrix} := \widetilde{\nabla\text{-}\mathbf{c}}_l(1)$, append $\left(\begin{pmatrix} \bar{S}\ \boxed{u}\bar{S}_{\oplus} \\ \boxed{u}\hat{S} \end{pmatrix}, \widetilde{\nabla\text{-}\mathbf{c}}_l(2), \widetilde{\nabla\text{-}\mathbf{c}}_l(4)\right)$ to $\nabla$-cf ▷
    state with the same value and a flag, indices, image
10:         $(i', j') := \widetilde{\nabla\text{-}\mathbf{c}}_l(2)$
11:     if $i > 1$ then
12:         $\begin{pmatrix} \bar{S}\ \bar{S}_{\oplus} \\ \hat{S} \end{pmatrix} := \widetilde{\nabla\text{-}\mathbf{c}}_l(1)$, append $\left(\begin{pmatrix} \bar{S}\ \boxed{f}\bar{S}_{\oplus} \\ \boxed{f}\hat{S} \end{pmatrix}, \widetilde{\nabla\text{-}\mathbf{c}}_l(2), \widetilde{\nabla\text{-}\mathbf{c}}_l(4)\right)$ to $\nabla$-cf
13:         for $j = 1, 2, \ldots, i - 1$ do
14:             $\begin{pmatrix} \bar{S}\ \bar{S}_{\oplus} \\ \hat{S} \end{pmatrix} := \widetilde{\nabla\text{-}\mathbf{c}}_l(1)$, append $\left(\begin{pmatrix} \bar{S}\ \boxed{n}\bar{S}_{\oplus} \\ \boxed{n}\hat{S} \end{pmatrix}, \widetilde{\nabla\text{-}\mathbf{c}}_l(2), \widetilde{\nabla\text{-}\mathbf{c}}_l(4)\right)$ to
    $\nabla$-cf
15:     $l := l + i$
16: Make a 2-dimensional array out of $\nabla$-cf according to the second entry in a
    standard left-to-right order $((i, j) \prec_{\text{l-r}} (i', j') \Leftrightarrow (i < i') \vee (i = i' \wedge j < j'))$,
    delete the second entry of $\nabla$-cf          ▷ $\nabla\text{-}\mathbf{cf}\,^i_j$ =(state with a flag, image)
17: return $\nabla$-cf
```

Flag-Assign. Flag : $\{\boxed{u}, \boxed{f}, \boxed{n}\} \times (\mathcal{A}^2 \times \mathcal{C}) \times (\mathcal{A} \times \mathcal{C}) \to \{\boxed{u}, \boxed{f}, \boxed{n}\}$,

$$\textsc{Flag}(\nabla\text{-}\mathbf{cf}^i_j) = \textsc{Flag}\left(\begin{pmatrix} \bar{S}^i_j\ \boxed{F}\bar{S}^i_{j\oplus} \\ \boxed{F}\hat{S}^i_j \end{pmatrix}, S\right) := \boxed{F}. \tag{4.41}$$

One can think about this function as a projection to the flag assigned to the triple. The transition probabilities in Equation (4.40) depend on the flags we assigned to states in ∇-c. We have that

$$\begin{aligned} &\textsc{Flag}(\nabla\text{-}\mathbf{cf}^i_j) \in \{\boxed{u}, \boxed{f}\} \\ &\Rightarrow \mathbb{P}\left[\mathbf{f}^{(\boxed{u} \vee \boxed{f})}(S^i_{j\oplus}) = S \mid \bigwedge_{(i',j') \prec (i,j)} \left(\mathbf{f}(S^{i'}_{j'\oplus}) = S^{i'}_{j'+1}\right)\right] = \frac{1}{|\mathcal{A}| \cdot |\mathcal{C}|}, \\ &\textsc{Flag}(\nabla\text{-}\mathbf{cf}^i_j) = \boxed{n} \end{aligned} \tag{4.42}$$

$$\Rightarrow \mathbb{P}\left[\mathbf{f}({}^{\boxed{\mathtt{n}}}S^{i}_{j\oplus}) = S \mid \bigwedge_{(i',j')\prec(i,j)} \left(\mathbf{f}(S^{i'}_{j'\oplus}) = S^{i'}_{j'+1}\right)\right] = \begin{cases} 1 & \text{if } S = \nabla\text{-}\mathbf{cf}^{i}_{j}(2) \\ 0 & \text{otherwise} \end{cases}. \tag{4.43}$$

4.5.3 Cardinality of ∇-$\mathrm{C}(\mathbf{M}, \mathbf{Z})$

In this section we evaluate the number of intermediate states that give $(\forall i \in [q] : \textsc{Sponge}_{\mathbf{f}}(\mathbf{M}^{i}, \ell_i) = \mathbf{Z}^{i})$. First we impose the constraint of $\mathbf{f}$ being a function. Then we want to calculate the product of probabilities in Equation (4.40). It depends on the number of unique states in ∇-$\mathbf{c}$ so we divide the set of possible states into subsets with the same number of states with the flag $\boxed{\mathtt{u}}$ or $\boxed{\mathtt{f}}$. The next steps involve further divisions of ∇-$\mathbf{C}(\mathbf{M}, \mathbf{Z})$.

In the process of calculating the conditional probabilities in Equation (4.40) we included in each state in ∇-$\mathbf{c}$ the image it should have under $\mathbf{f}$. The set ∇-$\mathbf{C}(\mathbf{M}, \mathbf{Z})$ does however contain states that would violate the constraint of $\mathbf{f}$ being a function—i.e. assign two or more distinct outputs to a single input. The first step to calculate the cardinality of ∇-$\mathbf{C}(\mathbf{M}, \mathbf{Z})$ is to exclude ∇-$\mathbf{c}$ that do not fulfill this requirement. The set of states that should be taken into consideration is defined below, we denote this set by p-∇-$\mathbf{CF}(\mathbf{M}, \mathbf{Z})$ (p emphasizes the fact that $\mathbf{f}$ is a proper function).

Definition 4.8 (p-∇-$\mathbf{CF}(\mathbf{M}, \mathbf{Z})$)**.** *The set of nabla configurations* ∇-$\mathbf{c}$ *for* $(\mathbf{M}, \mathbf{Z})$ *with flags and a proper function* $\mathbf{f}$ *is a set of arrays of size specified by* $(\mathbf{M}, \mathbf{Z})$*.* p-∇-$\mathbf{CF}(\mathbf{M}, \mathbf{Z}) \subset ((\{\boxed{\mathtt{u}}, \boxed{\mathtt{f}}, \boxed{\mathtt{n}}\} \times \mathcal{A}^2 \times \mathcal{C}) \times (\mathcal{A} \times \mathcal{C}))^{\kappa+q}$*, the set is defined in two steps, first we define the set of* ∇-$\mathbf{cf}$ *that are output by* $\textsc{Flag-Assign}$*,*

$$\nabla\text{-}\mathbf{CF}(\mathbf{M}, \mathbf{Z}) := \{\nabla\text{-}\mathbf{cf} : \exists\, \nabla\text{-}\mathbf{c}_0 \in \nabla\text{-}\mathbf{C}(\mathbf{M}, \mathbf{Z}), \nabla\text{-}\mathbf{cf} = \textsc{Flag-Assign}(\nabla\text{-}\mathbf{c}_0)\}\,. \tag{4.44}$$

We define p-∇-$\mathbf{CF}(\mathbf{M}, \mathbf{Z})$ *by the following constraints on* ∇-$\mathbf{CF}(\mathbf{M}, \mathbf{Z})$*:*

$$\forall S^{i}_{j} \in \nabla\text{-}\mathbf{cf}, \forall j > 1 : S^{i}_{j} = \nabla\text{-}\mathbf{cf}^{i}_{j-1}(2). \tag{4.45}$$

The formal definition reads

$$\text{p-}\nabla\text{-}\mathbf{CF}(\mathbf{M}, \mathbf{Z}) := \{\nabla\text{-}\mathbf{cf} \in \nabla\text{-}\mathbf{CF}(\mathbf{M}, \mathbf{Z}) : \nabla\text{-}\mathbf{cf} \text{ fulfills constraints (4.45)}\}\,. \tag{4.46}$$

One may think about p-∇-$\mathbf{CF}(\mathbf{M}, \mathbf{Z})$ as follows, first we consider ∇-$\mathbf{c}$: an array of states. The collection of all those arrays—with the exception of those that do not fulfill constraints (4.31)—is denoted by ∇-$\mathbf{C}(\mathbf{M}, \mathbf{Z})$. On each ∇-$\mathbf{c} \in \nabla$-$\mathbf{C}(\mathbf{M}, \mathbf{Z})$ we run the algorithm $\textsc{Flag-Assign}$, getting a collection of ∇-$\mathbf{cf}$—denoted by ∇-$\mathbf{CF}(\mathbf{M}, \mathbf{Z})$. Now we discard all those ∇-$\mathbf{cf}$ that do no fulfill constraints (4.45). The collection we are left with is denoted by p-∇-$\mathbf{CF}(\mathbf{M}, \mathbf{Z})$. We have the following relations between sets:

$$\nabla\text{-}\mathbf{CF}(\mathbf{M}, \mathbf{Z})(1) \overset{\text{omitting the flags}}{\simeq} \nabla\text{-}\mathbf{C}(\mathbf{M}, \mathbf{Z}), \tag{4.47}$$

$$\mathbf{p}\text{-}\nabla\text{-}\mathbf{CF}(\mathbf{M},\mathbf{Z}) \subset \nabla\text{-}\mathbf{CF}(\mathbf{M},\mathbf{Z})\,. \tag{4.48}$$

Each $\mathbf{p}\text{-}\nabla\text{-}\mathbf{cf} \in \mathbf{p}\text{-}\nabla\text{-}\mathbf{CF}(\mathbf{M},\mathbf{Z})$ has some number of unique states: with flag $\boxed{\mathrm{u}}$ or $\boxed{\mathrm{f}}$. Let us denote this number by $\bar{u}$. Equation (4.42) implies that no matter in what configurations the unique states are, the product of probabilities in Equation (4.40) is the same. Hence the first division of $\mathbf{p}\text{-}\nabla\text{-}\mathbf{CF}(\mathbf{M},\mathbf{Z})$ is in terms of the total number of unique states. We denote the state with a fixed number $\bar{u}$ by $\mathbf{p}\text{-}\nabla\text{-}\mathbf{CF}(\mathbf{M},\mathbf{Z},\bar{u})$, we have that

$$\mathbf{p}\text{-}\nabla\text{-}\mathbf{CF}(\mathbf{M},\mathbf{Z}) = \bigcup_{\bar{u}=1}^{\kappa} \mathbf{p}\text{-}\nabla\text{-}\mathbf{CF}(\mathbf{M},\mathbf{Z},\bar{u}). \tag{4.49}$$

The product in Equation (4.40) for $\mathbf{p}\text{-}\nabla\text{-}\mathbf{cf} \in \mathbf{p}\text{-}\nabla\text{-}\mathbf{CF}(\mathbf{M},\mathbf{Z},\bar{u})$ evaluates to

$$\prod_{(i,j)=(1,1)}^{(q,k_q-1)} \mathbb{P}\left[\left(\mathbf{f}(S^i_{j\oplus}) = S^i_{j+1}\right) \mid \bigwedge_{(i',j')\prec(i,j)} \left(\mathbf{f}(S^{i'}_{j'\oplus}) = S^{i'}_{j'+1}\right)\right] = \left(\frac{1}{|\mathcal{A}|\cdot|\mathcal{C}|}\right)^{\bar{u}}, \tag{4.50}$$

where all states $\mathbf{p}\text{-}\nabla\text{-}\mathbf{cf}$ are in $\mathbf{p}\text{-}\nabla\text{-}\mathbf{CF}(\mathbf{M},\mathbf{Z},\bar{u})$.

We have to work a bit more to calculate the total number of states. The number of possibilities in which a single transition event can be realized depends both on the input and on the output. For that reason we need to specify the configuration of flags in more detail, not just by the total number of unique states. Let us denote a transition event from a unique state to a unique state by $\left(\mathbf{f}({}^{(\boxed{\mathrm{u}}\vee\boxed{\mathrm{f}})}S_{\oplus}) = {}^{(\boxed{\mathrm{u}}\vee\boxed{\mathrm{f}})}S\right)$ and similarly for other flags. The flag of the output is defined by the XORed message symbol or the output symbol. Before we go into details of the analysis of the structure of $\mathbf{p}\text{-}\nabla\text{-}\mathbf{CF}(\mathbf{M},\mathbf{Z})$, we list the intuitive principles of counting the output states depending on the input and output states:

(a) $\left(\mathbf{f}({}^{(\boxed{\mathrm{u}}\vee\boxed{\mathrm{f}})}S_{\oplus}) = {}^{(\boxed{\mathrm{u}}\vee\boxed{\mathrm{f}})}S\right)$—the only constraint is that the output cannot be the same as any on the previous unique states, the number of possible output values is at most $|\mathcal{A}|\cdot|\mathcal{C}|$ (in the absorbing phase) or $|\mathcal{C}|$ (in the squeezing phase) and can be smaller by at most κ,

(b) $\left(\mathbf{f}({}^{(\boxed{\mathrm{u}}\vee\boxed{\mathrm{f}})}S_{\oplus}) = {}^{\boxed{\mathrm{n}}}S\right)$—the output has to be in the set of outputs of states with the flag $\boxed{\mathrm{f}}$, the number of possible output values is at most κ,

(c) $\left(\mathbf{f}({}^{\boxed{\mathrm{n}}}S_{\oplus}) = {}^{(\boxed{\mathrm{u}}\vee\boxed{\mathrm{f}}\vee\boxed{\mathrm{n}})}S\right)$—the output is defined by the image memorized in the second entry of the state, the number of possible output values $= 1$.

The actual numbers in the above guidelines can be calculated precisely but they depend on the actual case we deal with.

To properly treat the transition events we need to keep track of not only the total number of unique states but also the number of truly unique $\boxed{\mathtt{u}}$ states. We denote the latter by u and the set with those numbers fixed by $\mathbf{p}\text{-}\nabla\text{-}\mathbf{CF}(\mathbf{M}, \mathbf{Z}, \bar{u}, u)$. In the above paragraph we also noticed that we should include in our considerations the number of unique states in different phases of Sponge. The number of states with the flag $\boxed{\mathtt{u}}$ in the absorbing phase is denoted by u_{abs}. Note that we are addressing all q absorbing phases so we take into account flags of all states with indices (i, j) in $\{(i', j')\}_{i' \in \{1,\dots,q\}, j' \in \{1,\dots,|\mathbf{M}^{i'}|\}}$. The number of states with the flag $\boxed{\mathtt{u}}$ in the squeezing phase is denoted by u_{squ} and we take into account states with indices (i, j) in the set $\{(i', j')\}_{i' \in \{1,\dots,q\}, j' \in \{|\mathbf{M}^{i'}|+1,\dots,k_{i'}-1\}}$. Similarly the total number of unique states is denoted by $\bar{u}_{\text{abs}}$ and $\bar{u}_{\text{squ}}$.

Next we fix particular placements of flags in the arrays $\mathbf{p}\text{-}\nabla\text{-}\mathbf{cf} \in \mathbf{p}\text{-}\nabla\text{-}\mathbf{CF}(\mathbf{M}, \mathbf{Z}, \bar{u}_{\text{abs}}, u_{\text{abs}}, \bar{u}_{\text{squ}}, u_{\text{squ}})$. We no longer need to keep u and $\bar{u}$ explicit as $u = u_{\text{abs}} + u_{\text{squ}}$ and $\bar{u} = \bar{u}_{\text{abs}} + \bar{u}_{\text{squ}}$. Let us define a *placement* P for $(\mathbf{M}, \mathbf{Z})$ as an array of flags $\boxed{\mathtt{F}} \in \{\boxed{\mathtt{u}}, \boxed{\mathtt{f}}, \boxed{\mathtt{n}}\}$ with its dimensions determined by $(\mathbf{M}, \mathbf{Z})$ in the same way as for nabla configurations $\nabla\text{-}\mathbf{c}$. The set of placements $\mathcal{P}(\mathbf{M}, \mathbf{Z}, \bar{u}_{\text{abs}}, u_{\text{abs}}, \bar{u}_{\text{squ}}, u_{\text{squ}})$ is defined as the set of all placements P encountered in elements of $\mathbf{p}\text{-}\nabla\text{-}\mathbf{CF}(\mathbf{M}, \mathbf{Z}, \bar{u}_{\text{abs}}, u_{\text{abs}}, \bar{u}_{\text{squ}}, u_{\text{squ}})$. We are going to write $\textsc{Flag}(P_j^i)$ to determine the flag in the position (i, j) in placement P. For each P we are able to calculate the size of $\mathbf{p}\text{-}\nabla\text{-}\mathbf{CF}(\mathbf{M}, \mathbf{Z}, P)$, we no longer add $\bar{u}_{\text{abs}}$ and other parameters as they are already included in P. Before we define the algorithm performing this calculation we need to bound the number of different placements.

Let us assume for a moment that $(\mathbf{M}, \mathbf{Z})$ restrains only the size of $\mathbf{p}\text{-}\nabla\text{-}\mathbf{cf}$ and not the values of the states. If there were no constraints coming from the workings of Flag-Assign then unique states would be distributed in all combinations of picking $\bar{u}_{\text{abs}}$ elements among states in absorbing phases. Additionally, we also want to take into account combinations of u_{abs} elements among the $\bar{u}_{\text{abs}}$ flags. Let us recapitulate: first we distribute $\bar{u}_{\text{abs}}$ flags (without specifying whether they are $\boxed{\mathtt{u}}$ or $\boxed{\mathtt{f}}$) and then assign them concrete values ($\boxed{\mathtt{u}}$ or $\boxed{\mathtt{f}}$). The total number of state-triples in the absorbing phases of $\mathbf{p}\text{-}\nabla\text{-}\mathbf{cf}$ is $\mu := \sum_{i=1}^{q} |\mathbf{M}^i|$. The number of possibilities for the first step is $\binom{\mu}{\bar{u}_{\text{abs}}}$ and the second step is $\binom{\bar{u}_{\text{abs}}}{u_{\text{abs}}}$. The total number of possibilities of placing the unique flags in absorbing phases is $\binom{\mu}{\bar{u}_{\text{abs}}} \cdot \binom{\bar{u}_{\text{abs}}}{u_{\text{abs}}}$.

The problem of distributing unique states in squeezing phases is the same as in absorbing phases. The total number of state-triples with flags in the squeezing phases of $\mathbf{p}\text{-}\nabla\text{-}\mathbf{cf}$ is $\zeta := \sum_{i=1}^{q}(|\mathbf{Z}^i| - 1)$. The number of placements is $\binom{\zeta}{\bar{u}_{\text{squ}}}$. We also need to multiply this result by the number of placements of states with flag $\boxed{\mathtt{u}}$ among all unique states.

The two calculations above bring us to the conclusion that our analysis is sufficiently detailed; we have identified and taken into account all parts of $(\forall i \in [q] : \textsc{Sponge}_{\mathbf{f}}(\mathbf{M}^i, \ell_i) = \mathbf{Z}^i)$ that depend on $|\mathcal{C}|$. In summary we divided $\mathbf{p}\text{-}\nabla\text{-}\mathbf{CF}(\mathbf{M}, \mathbf{Z})$ into a small (relatively to $|\mathcal{C}|$) number of subsets whose size we can actually calculate. The last result assures that even though we do not formally describe the structure of the last level of division of $\mathbf{p}\text{-}\nabla\text{-}\mathbf{CF}(\mathbf{M}, \mathbf{Z})$, the number of possibilities of next divisions does not depend on $|\mathcal{C}|$. So we have that

$$\left|\mathcal{P}(\mathbf{M}, \mathbf{Z}, \bar{u}_{\text{abs}}, u_{\text{abs}}, \bar{u}_{\text{squ}}, u_{\text{squ}})\right| \leq \binom{\mu}{\bar{u}_{\text{abs}}}\binom{\bar{u}_{\text{abs}}}{u_{\text{abs}}} \cdot \binom{\zeta}{\bar{u}_{\text{squ}}}\binom{\bar{u}_{\text{squ}}}{u_{\text{squ}}} \tag{4.51}$$

$$\leq \binom{\mu}{\mu/2}^2 \binom{\zeta}{\zeta/2}^2 \leq \binom{\kappa}{\kappa/2}^4 \leq \kappa^{4\kappa}. \tag{4.52}$$

Our assumption is that κ is fixed so the number of placements is independent of $|\mathcal{C}|$. Note that we can compute $\left|\mathcal{P}(\mathbf{M}, \mathbf{Z}, \bar{u}_{\text{abs}}, u_{\text{abs}}, \bar{u}_{\text{squ}}, u_{\text{squ}})\right|$ for fixed parameters and the above inequality just shows that irrespective of the exact value of the calculation the number of placements does not depend on $|\mathcal{C}|$ and is relatively small.

Let us define a function that helps us accommodate for the fact that some subsets of $\mathbf{p}\text{-}\nabla\text{-}\mathbf{CF}(\mathbf{M}, \mathbf{Z})$ are empty for some specific $(\mathbf{M}, \mathbf{Z})$:

$$\boldsymbol{\delta}(\mathbf{M}, \mathbf{Z}, P) := \begin{cases} 1 \text{ if } \mathbf{p}\text{-}\nabla\text{-}\mathbf{CF}(\mathbf{M}, \mathbf{Z}, P) \neq \emptyset \\ 0 \text{ otherwise} \end{cases}. \tag{4.53}$$

In what follows we leave out the input to $\boldsymbol{\delta}$, as it can be inferred from context. For example $\boldsymbol{\delta}$ evaluates to 0 if the input includes $\bar{u}_{\text{abs}} = \mu$ and the first block of the input messages is not always different.

The last division we make is done be characterizing uniqueness of outer and inner parts of states. This step is done to get the precise and correct result, but the high-level explanation and an approximation of the output of Calc is already captured by principle (a). We have not captured this situation in detail in our example proof because it becomes important only if longer outputs are present. Here we explain the procedure of including the necessary details.

The main detail we add is assigning flags to outer and inner parts of states individually. We introduce those flags only now to keep the proof as clear as possible; technically to include the additional flags we modify the algorithm Flag-Assign in such a way that it runs over a configuration ∇-c two additional times but acting solely on outer states and inner states. Those two additional runs assign the same flags as the original one but corresponding to just one of the parts of $S_\oplus$ states. The rest of the discussion after applying Flag-Assign is unchanged and depends only on flags of the full states.

When discussing placements, note that a unique state ($\boxed{\text{u}}$ or $\boxed{\text{f}}$) can consist of a unique outer state and a unique inner state but also of a non-unique outer state and a unique inner state or vice versa. After we assign a particular placement $P \in \mathcal{P}(\mathbf{M}, \mathbf{Z}, \bar{u}_{\text{abs}}, u_{\text{abs}}, \bar{u}_{\text{squ}}, u_{\text{squ}})$ there are still many possibilities of arranging outer and inner states flags. There are exactly three possibilities every unique state can be arranged in: $\begin{pmatrix} \boxed{\mathrm{u}} \vee \boxed{\mathrm{f}} \\ \boxed{\mathrm{u}} \vee \boxed{\mathrm{f}} \end{pmatrix}$, $\begin{pmatrix} \boxed{\mathrm{u}} \vee \boxed{\mathrm{f}} \\ \boxed{\mathrm{n}} \end{pmatrix}$, and $\begin{pmatrix} \boxed{\mathrm{n}} \\ \boxed{\mathrm{u}} \vee \boxed{\mathrm{f}} \end{pmatrix}$, where we symbolize a state $S_\oplus$ by a column vector with flags assigned to its outer state in the first row and inner state in the second row. Hence, for every placement P we have $3^{\bar{u}_{\text{abs}}+\bar{u}_{\text{squ}}}$ placements of the outer and inner states flags. We are going to mark the fact that we have included those additional details into placements by adding a star to the set of placements $P \in \mathcal{P}^*(\mathbf{M}, \mathbf{Z}, \bar{u}_{\text{abs}}, u_{\text{abs}}, \bar{u}_{\text{squ}}, u_{\text{squ}})$. We have that

$$\left|\mathcal{P}^*(\mathbf{M}, \mathbf{Z}, \bar{u}_{\text{abs}}, u_{\text{abs}}, \bar{u}_{\text{squ}}, u_{\text{squ}})\right| \leq \kappa^{4\kappa} \cdot 3^{\bar{u}_{\text{abs}}+\bar{u}_{\text{squ}}}. \tag{4.54}$$

We also write $\textsc{Flag}(\bar{P}^i_j)$ and $\textsc{Flag}(\hat{P}^i_j)$ to access the flag of the outer and inner part of P^i_j respectively.

Algorithm 4.3 below shows the algorithm $\textsc{Calc}$ that outputs the number of different $\mathbf{p}\text{-}\nabla\text{-}\mathbf{cf} \in \mathbf{p}\text{-}\nabla\text{-}\mathbf{CF}(\mathbf{M}, \mathbf{Z}, \bar{u}_{\text{abs}}, u_{\text{abs}}, \bar{u}_{\text{squ}}, u_{\text{squ}})$ for some given placement $P \in \mathcal{P}^*(\mathbf{M}, \mathbf{Z}, \bar{u}_{\text{abs}}, u_{\text{abs}}, \bar{u}_{\text{squ}}, u_{\text{squ}})$. To capture the fact that the number of possible values a unique state can have depends on the number of unique states with already assigned values we define the following sets. For unique outer states we have

$$\bar{\mathbf{u}}_{\text{prev}}(P, i, j) := \left|\left\{P^{i'}_{j'} : (i', j') \prec (i, j) \wedge \textsc{Flag}(\bar{P}^{i'}_{j'}) \in \{\boxed{\mathrm{u}}, \boxed{\mathrm{f}}\}\right\}\right|, \tag{4.55}$$

$$\bar{\mathbf{u}}^{\boxed{\mathrm{f}}}_{\text{prev}}(P, i, j) := \left|\left\{P^{i'}_{j'} : (i', j') \prec (i, j) \wedge \textsc{Flag}(\bar{P}^{i'}_{j'}) = \boxed{\mathrm{f}}\right\}\right|. \tag{4.56}$$

For unique inner states we have

$$\hat{\mathbf{u}}_{\text{prev}}(P, i, j) := \left|\left\{P^{i'}_{j'} : (i', j') \prec (i, j) \wedge \textsc{Flag}(\hat{P}^{i'}_{j'}) \in \{\boxed{\mathrm{u}}, \boxed{\mathrm{f}}\}\right\}\right|, \tag{4.57}$$

$$\hat{\mathbf{u}}^{\boxed{\mathrm{f}}}_{\text{prev}}(P, i, j) := \left|\left\{P^{i'}_{j'} : (i', j') \prec (i, j) \wedge \textsc{Flag}(\hat{P}^{i'}_{j'}) = \boxed{\mathrm{f}}\right\}\right|. \tag{4.58}$$

Note that all of the above quantities (4.55, 4.56, 4.57, 4.58) are bounded by

$$1 \leq \bar{\mathbf{u}}_{\text{prev}}(P, i, j), \hat{\mathbf{u}}_{\text{prev}}(P, i, j), \bar{\mathbf{u}}^{\boxed{\mathrm{f}}}_{\text{prev}}(P, i, j), \hat{\mathbf{u}}^{\boxed{\mathrm{f}}}_{\text{prev}}(P, i, j) \leq \bar{u}_{\text{abs}} + \bar{u}_{\text{squ}} \leq \kappa. \tag{4.59}$$

In the algorithm we also use $\textsc{N-Possibilities}$ which is the number of possibilities in which one can assign values to non-unique states in a nabla configuration. $\textsc{N-Possibilities}$ is bounded by κ^κ, in the discussion proceeding Equation (4.64) we present its detailed derivation:

Let us consider the problem of assigning values to the non-unique states given some placement P. Let us denote the number of non-unique flags $\boxed{\mathrm{n}}$ by n and the number of "first" non-unique states with flags $\boxed{\mathrm{f}}$ by f. We are going to

Algorithm 4.3 CALC

```
    input: $P \in \mathcal{P}^*(\mathbf{M}, \mathbf{Z}, \bar{u}_{\text{abs}}, u_{\text{abs}}, \bar{u}_{\text{squ}}, u_{\text{squ}})$
    output: $\alpha \in \mathbb{N}$, cardinality of the set $\mathbf{p}\text{-}\nabla\text{-}\mathbf{CF}(\mathbf{M}, \mathbf{Z}, P)$
 1: $\alpha := 1$
 2: for $j = 1, \ldots, k_i - 2,\ i = 1, \ldots, q$ do
 3:     if $j < |\mathrm{M}^i|$ and $\textsc{Flag}(P^i_j) \in \{\boxed{u}, \boxed{f}\}$ then          ▷ Absorbing phases
 4:         if $\textsc{Flag}(P^i_{j+1}) \in \{\boxed{u}, \boxed{f}\}$ then
 5:             if $\textsc{Flag}(\bar{P}^i_{j+1}) \in \{\boxed{u}, \boxed{f}\}$ and $\textsc{Flag}(\hat{P}^i_{j+1}) \in \{\boxed{u}, \boxed{f}\}$ then
 6:                 $\alpha = \alpha \cdot \Big(|\mathcal{A}| - \bar{\mathbf{u}}_{\text{prev}}(P, i, j+1)\Big) \cdot \Big(|\mathcal{C}| - \hat{\mathbf{u}}_{\text{prev}}(P, i, j+1)\Big)$
 7:             if $\textsc{Flag}(\bar{P}^i_{j+1}) \in \{\boxed{u}, \boxed{f}\}$ and $\textsc{Flag}(\hat{P}^i_{j+1}) = \boxed{n}$ then
 8:                 $\alpha = \alpha \cdot \Big(|\mathcal{A}| - \bar{\mathbf{u}}_{\text{prev}}(P, i, j+1)\Big) \cdot \hat{\mathbf{u}}^{\boxed{f}}_{\text{prev}}(P, i, j+1)$
 9:             if $\textsc{Flag}(\bar{P}^i_{j+1}) = \boxed{n}$ and $\textsc{Flag}(\hat{P}^i_{j+1}) \in \{\boxed{u}, \boxed{f}\}$ then
10:                 $\alpha = \alpha \cdot \bar{\mathbf{u}}^{\boxed{f}}_{\text{prev}}(P, i, j+1) \cdot \Big(|\mathcal{C}| - \hat{\mathbf{u}}_{\text{prev}}(P, i, j+1)\Big)$
11:     if $j \geq |\mathrm{M}^i|$ and $\textsc{Flag}(P^i_j) \in \{\boxed{u}, \boxed{f}\}$ then          ▷ Squeezing phases
12:         if $\textsc{Flag}(P^i_{j+1}) \in \{\boxed{u}, \boxed{f}\}$ then
13:             if $\textsc{Flag}(\hat{P}^i_{j+1}) \in \{\boxed{u}, \boxed{f}\}$ then
14:                 $\alpha = \alpha \cdot \Big(|\mathcal{C}| - \hat{\mathbf{u}}_{\text{prev}}(P, i, j+1)\Big)$
15:             if $\textsc{Flag}(\hat{P}^i_{j+1}) = \boxed{n}$ then
16:                 $\alpha = \alpha \cdot \hat{\mathbf{u}}^{\boxed{f}}_{\text{prev}}(P, i, j+1)$
17: for $i = 1, \ldots, q,\ j = k_i - 1$ do
18:     if $\textsc{Flag}(P^i_j) \in \{\boxed{u}, \boxed{f}\}$ then
19:         $\alpha = \alpha \cdot |\mathcal{C}| \cdot |\mathcal{A}|^{|\mathbf{Z}^i|}$
20: $\alpha = \alpha \cdot \textsc{N-Possibilities}(\kappa - \bar{u}_{\text{abs}} - \bar{u}_{\text{squ}}, \bar{u}_{\text{abs}} + \bar{u}_{\text{squ}} - u_{\text{abs}} - u_{\text{squ}}, P)$
21: return $\alpha \cdot \boldsymbol{\delta}(\mathbf{M}, \mathbf{Z}, P)$
```

analyze the combinatorial problem of assigning n objects to $\mathbf{f}$ classes in a way that each class is assigned at least one object. Objects in a single class are indistinguishable but are distinguishable between different classes. For example three objects that we divide among two classes putting one object in the first class and two in the second can be assigned in three ways: we put into the first class the first object or the second object or the third. The objects are indistinguishable and we only count once the situation when the two objects that are in class two can be in one order or another. The number of permutations in such a problem in the general case of n objects and $\mathbf{f}$ classes is given by

$$\boldsymbol{\pi}\left(n; n_1, n_2, \ldots, n_f\right) = \left(\frac{n!}{n_1! n_2! \cdots n_f!}\right). \tag{4.60}$$

Note that the above formula requires that we specify the occupation of classes. These occupation numbers are not fixed by the placement P so we also need to analyze these occupation numbers. The occupations of the classes are defined by the possible distribution of objects among different classes. Let us define the set of possible distributions:

$$\mathcal{D}(n, f) := \Big\{(n_1, n_2, \ldots, n_f) \in \mathbb{N}^f : \forall i \in [f], n_i \geq 1, n_1 + n_2 + \cdots + n_f = n\Big\}. \tag{4.61}$$

There is one more detail we need to add to properly count all possible assignments of non-unique states; repeated values of each different $\mathbf{f}$ appear in P only after the initial unique state. Let us denote by $\Pi_{\text{class-ind}}(n; n_1, n_2, \ldots, n_f)$ the set of permutations with classes of indistinguishable objects, enumerated by $\boldsymbol{\pi}\left(n; n_1, n_2, \ldots, n_f\right)$. We need to implement the requirement coming from the nature of working of Flag-Assign. The set of permutations after including this constraint is

$$\begin{aligned}\Pi\left(P, n; n_1, n_2, \ldots, n_f\right) := \{&\pi \in \Pi_{\text{class-ind}}(n; n_1, n_2, \ldots, n_f) \mid \\ &\text{there are no objects in class } i \text{ prior to the } i\text{-th state according to } P\}.\end{aligned} \tag{4.62}$$

Eventually we count the number of possible assignments of values of non-unique states:

$$\text{N-Possibilities}(n, f, P) := \sum_{(n_1, n_2, \ldots, n_f) \in \mathcal{D}(n, f)} \left|\Pi\left(P, n; n_1, n_2, \ldots, n_f\right)\right|. \tag{4.63}$$

The most crucial observation of this subsection is that the number of possible assignments does not depend on $|\mathcal{C}|$ and

$$\text{N-Possibilities}(n, f, P) \leq f^n. \tag{4.64}$$

The above is a trivial bound found by ignoring all structure and only counting the total number of possibilities to put one of $\mathbf{f}$ values in every of the n places.

Thanks to the additional details we get the precise form of the expression $\mathbf{p}$.

4.5.4 Final expression

In the previous subsections we formalized algorithms that help us analyze the expression in Equation (4.40). First we introduced FLAG-ASSIGN that analyzes ∇-**c** from the perspective of having the same input to **f** multiple times. Then we defined CALC that counts the arrays of states that fulfill a given set of constraints, the number and arrangement of unique states. The final part of the proof of Lemma 4.3 is to use those algorithms to show that $\mathbf{p}(|\mathcal{C}|^{-1})$ is of the claimed form. We start by formally writing down the expression in terms of divisions of $\mathbf{p}\text{-}\nabla\text{-}\mathbf{CF}(\mathbf{M},\mathbf{Z})$ we introduced and the outputs of CALC. Next we identify crucial elements of the sum that lead to the claim of the lemma, showing the maximal degree of $|\mathcal{C}|^{-1}$ in the expression $\mathbf{p}(\lambda)$.

In the previous sections we showed that

$$\begin{aligned}\mathbf{p}(|\mathcal{C}|^{-1}) = &\sum_{\nabla\text{-}\mathbf{c}\in\nabla\text{-}\mathbf{C}(\mathbf{M},\mathbf{Z})} \\ &\prod_{(i,j)=(1,1)}^{(q,k_q-1)} \mathbb{P}\left[\left(\mathbf{f}(S^i_{j\oplus}) = S^i_{j+1}\right)\middle| \bigwedge_{(i',j')\prec(i,j)} \left(\mathbf{f}(S^{i'}_{j'\oplus}) = S^{i'}_{j'+1}\right)\right] \qquad (4.65)\\ = &\underbrace{\sum_{\mathbf{p}\text{-}\nabla\text{-}\mathbf{cf}\in\mathbf{p}\text{-}\nabla\text{-}\mathbf{CF}(\mathbf{M},\mathbf{Z})}}_{\text{Equation (4.67),(4.68)}} \\ &\underbrace{\prod_{(i,j)=(1,1)}^{(q,k_q-1)} \mathbb{P}\left[\left(\mathbf{f}(S^i_{j\oplus}) = S^i_{j+1}\right)\middle| \bigwedge_{(i',j')\prec(i,j)} \left(\mathbf{f}(S^{i'}_{j'\oplus}) = S^{i'}_{j'+1}\right)\right]}_{\text{Equation (4.50)}}, \qquad (4.66)\end{aligned}$$

where the second equality comes from the fact that constraints (4.45) exclude those ∇-**c** that have probability 0. Let us also make the division of $\mathbf{p}\text{-}\nabla\text{-}\mathbf{CF}(\mathbf{M},\mathbf{Z})$ explicit

$$\begin{aligned}&\mathbf{p}\text{-}\nabla\text{-}\mathbf{CF}(\mathbf{M},\mathbf{Z}) = \\ &\bigcup_{\bar{u}_{\text{abs}}=1}^{\mu} \bigcup_{u_{\text{abs}}=0}^{\mu} \bigcup_{\bar{u}_{\text{squ}}=0}^{\zeta} \bigcup_{u_{\text{squ}}=0}^{\zeta} \bigcup_{P\in\mathcal{P}^*(\mathbf{M},\mathbf{Z},\bar{u}_{\text{abs}},u_{\text{abs}},\bar{u}_{\text{squ}},u_{\text{squ}})} \mathbf{p}\text{-}\nabla\text{-}\mathbf{CF}(\mathbf{M},\mathbf{Z},P). \qquad (4.67)\end{aligned}$$

Next we use Equation (4.50) and the fact that for $P \in \mathcal{P}(\mathbf{M},\mathbf{Z},\bar{u}_{\text{abs}},u_{\text{abs}},\bar{u}_{\text{squ}},u_{\text{squ}})$ we have

$$|\mathbf{p}\text{-}\nabla\text{-}\mathbf{CF}(\mathbf{M},\mathbf{Z},P)| = \text{CALC}(P) \qquad (4.68)$$

to expand $\mathbf{p}(|\mathcal{C}|^{-1})$ to

$$\mathbf{p}(|\mathcal{C}|^{-1}) = \sum_{\bar{u}_{\text{abs}},u_{\text{abs}},\bar{u}_{\text{squ}},u_{\text{squ}},P} \text{CALC}(P)\left(\frac{1}{|\mathcal{A}|\cdot|\mathcal{C}|}\right)^{\bar{u}_{\text{abs}}+\bar{u}_{\text{squ}}}. \qquad (4.69)$$

To calculate a_0 and the maximal degree of $\mathbf{p}$ let us focus on $\mathbf{p}(|\mathcal{C}|^{-1})$ for all unique (with the flag u in both outer and inner part) sates:

$$\prod_{i=1}^{q} \prod_{j=1}^{|\mathbf{M}^i|-1} (|\mathcal{A}| - jq - i)\,(|\mathcal{C}| - jq - i)$$

$$\prod_{i=1}^{q} \prod_{j=|\mathbf{M}^i|}^{k_i-2} (|\mathcal{C}| - jq - i) \prod_{i=1}^{q} \left(|\mathcal{A}|^{|\mathbf{Z}^i|}\,|\mathcal{C}|\right) (|\mathcal{A}| \cdot |\mathcal{C}|)^{-\kappa}. \tag{4.70}$$

In the above expression if we take all messages of maximal length m and outputs of maximal length z we get a polynomial of degree $\kappa - q = q(m + z - 2)$. This is necessarily the maximal degree as every evaluation of $\mathbf{f}$ increases the degree by one, except for the last but this cannot be changed, the last column does not matter at all for the overall probability. Hence the maximal degree of $\mathbf{p}$ is as claimed

$$\eta := q(m + z - 2). \tag{4.71}$$

In the case all states are unique, i.e. $|\mathcal{C}| \to \infty$, $\mathbf{p}(|\mathcal{C}|^{-1})$ evaluates to $\sim |\mathcal{A}|^{-\sum_i \ell_i}$—where we used the fact that $|\mathbf{Z}_i| = \ell_i$. This expression corresponds to the output probability of a random oracle, exactly how expected of a sponge with all different inner states. If we only take the terms $|\mathcal{A}| \cdot |\mathcal{C}|$ and $|\mathcal{C}|$ and the probability we arrive at $|\mathcal{A}|^{-\sum_i \ell_i}$. This result is only one of the terms in a_0 but note that all other terms will correspond to different placements and will include $\boldsymbol{\delta}(\mathbf{M}, \mathbf{Z}, P)$ with different inputs, being non-zero for different $(\mathbf{M}, \mathbf{Z})$. Hence for any given input-output pairs $(\mathbf{M}, \mathbf{Z})$ for $|\mathcal{C}| \to \infty$ the probability function approaches the probability of a random oracle outputting $\mathbf{Z}$ on M.

In our proof we have focused on the case of $\mathbf{f}$ being a random transformation. In Section 4.6 we provide the details that should be considered to show that Theorem 4.2 holds also for random permutations.

4.6 Internal Permutations

In this section we prove the main result for the internal function $\mathbf{f}$ being a random permutation. We use Zhandry's PRF/PRP switching lemma from [Zha15a]. In subSection 4.6.1 we also give a direct proof, resulting in a slightly worse bound.

Theorem 4.9. $\textsc{Sponge}_{\mathbf{f}}[\textsc{pad}, \mathcal{A}, \mathcal{C}]$ *for a random permutation* $\mathbf{f}$ *is quantumly indistinguishable from a random oracle. More concretely, for all quantum algorithms* A *making at most* q *quantum queries to* Sponge, *such that the padded input length is at most* m *and the output length is at most* z,

$$\left| \underset{\mathbf{f} \xleftarrow{\$} \mathcal{I}_{\mathrm{per}}(\mathcal{S})}{\mathbb{P}} \left[b = 1 : b \leftarrow \mathrm{A}^{|\textsc{Sponge}_{\mathbf{f}}\rangle} \right] - \underset{\mathbf{h} \leftarrow \mathfrak{R}}{\mathbb{P}} \left[b = 1 : b \leftarrow \mathrm{A}^{|\mathbf{h}\rangle} \right] \right| < \frac{\pi^2}{3} \eta^3 |\mathcal{C}|^{-1}, \tag{4.72}$$

where the set of permutations is denoted by $\mathcal{I}_{\text{per}}(\mathcal{S}) := \{\mathbf{f} : \mathcal{S} \to \mathcal{S} \mid \mathbf{f} \text{ is a bijection}\}$. *The domain is defined as* $\mathcal{S} = \mathcal{A} \times \mathcal{C}$ *for a finite* $\mathcal{A}$ *in an Abelian group* $(\mathcal{A}, \oplus)$ *and a non-empty finite set* $\mathcal{C}$.

Proof. It was proven in [Zha15a] that a random permutation can be distinguished from a random function with probability at most $\pi^2 q^2/6|\mathcal{C}|$ for any adversary making at most q quantum queries. We can use this result in a reduction from distinguishing Sponge using a random permutation from Sponge using a random function to distinguishing of a random permutation from a random function. Using this result together with Theorem 4.2 gives us the resulting bound as follows

$$\begin{aligned}
&\left| \underset{\mathbf{f} \xleftarrow{\$} \mathcal{I}_{\text{per}}(\mathcal{S})}{\mathbb{P}} \left[b = 1 : b \leftarrow \mathrm{A}^{|\textsc{Sponge}_{\mathbf{f}}\rangle} \right] - \underset{\mathbf{h} \leftarrow \mathfrak{R}}{\mathbb{P}} \left[b = 1 : b \leftarrow \mathrm{A}^{|\mathbf{h}\rangle} \right] \right| \\
&\leq \left| \underset{\mathbf{f} \xleftarrow{\$} \mathcal{I}_{\text{per}}(\mathcal{S})}{\mathbb{P}} \left[b = 1 : b \leftarrow \mathrm{A}^{|\textsc{Sponge}_{\mathbf{f}}\rangle} \right] - \underset{\mathbf{f} \xleftarrow{\$} \mathcal{S}^{\mathcal{S}}}{\mathbb{P}} \left[b = 1 : b \leftarrow \mathrm{A}^{|\textsc{Sponge}_{\mathbf{f}}\rangle} \right] \right| \\
&+ \left| \underset{\mathbf{f} \xleftarrow{\$} \mathcal{S}^{\mathcal{S}}}{\mathbb{P}} \left[b = 1 : b \leftarrow \mathrm{A}^{|\textsc{Sponge}_{\mathbf{f}}\rangle} \right] - \underset{\mathbf{h} \leftarrow \mathfrak{R}}{\mathbb{P}} \left[b = 1 : b \leftarrow \mathrm{A}^{|\mathbf{h}\rangle} \right] \right| \qquad (4.73) \\
&\leq \left| \underset{\mathbf{f} \xleftarrow{\$} \mathcal{I}_{\text{per}}(\mathcal{S})}{\mathbb{P}} \left[b = 1 : b \leftarrow \mathrm{B}^{|\mathbf{f}\rangle} \right] - \underset{\mathbf{f} \xleftarrow{\$} \mathcal{S}^{\mathcal{S}}}{\mathbb{P}} \left[b = 1 : b \leftarrow \mathrm{B}^{|\mathbf{f}\rangle} \right] \right| + \frac{\pi^2}{6} \eta^3 |\mathcal{C}|^{-1} \qquad (4.74) \\
&\leq \frac{\pi^2}{3} \eta^3 |\mathcal{C}|^{-1}. \qquad (4.75)
\end{aligned}$$

□

4.6.1 Direct proof of indistinguishability with permutations

Here we prove Theorem 4.9 by direct application of Theorem 2.21 instead of relying on the PRF/PRP switching lemma. For this proof we need to generalize the average-case polynomial method. We show how to use it if the probability of a certain input-output behavior is not a polynomial but is close to a polynomial. This small generalization might prove useful in other applications of the polynomial method. The following is a restatement of Theorem 4.9, with a slightly worse bound.

Theorem 4.10. Sponge$_{\mathbf{f}}$ *for a random permutation* $\mathbf{f}$ *is quantumly indistinguishable from a random oracle. More concretely, for all quantum algorithms* A *making at most* q *quantum queries to* Sponge, *such that the input length is at most* $m \cdot r$ *bits long and the output length is at most* $z \cdot r$ *bits long,*

$$\left| \underset{\mathbf{f} \xleftarrow{\$} \mathcal{I}_{\text{per}}(\mathcal{S})}{\mathbb{P}} \left[b = 1 : b \leftarrow \mathrm{A}^{|\textsc{Sponge}_{\mathbf{f}}\rangle} \right] - \underset{\mathbf{h} \leftarrow \mathfrak{R}}{\mathbb{P}} \left[b = 1 : b \leftarrow \mathrm{A}^{|\mathbf{h}\rangle} \right] \right| < \frac{\pi^2}{6} (2\eta)^3 (|\mathcal{C}| - 1)^{-1}, \qquad (4.76)$$

where $\eta := 2q(m + z - 2)$*,* $\mathfrak{R}$ *is defined according to Definition 2.13, and the set of permutations is denoted by* $\mathcal{I}_{\text{per}}(\mathcal{S}) := \{\mathbf{f} : \mathcal{S} \to \mathcal{S} \mid \mathbf{f} \text{ is a bijection}\}$*. The domain is defined as* $\mathcal{S} = \mathcal{A} \times \mathcal{C}$ *for a finite* $\mathcal{A}$ *in an Abelian group* $(\mathcal{A}, \oplus)$ *and a non-empty finite set* $\mathcal{C}$*.*

Proof sketch. The proof follows the same reasoning as the proof of Theorem 4.2 with small differences explained in the following. We define the family of distributions $\mathfrak{F}_t$ with random permutations $\mathbf{f}$ from $\mathcal{I}_{\text{per}}(\mathcal{S})$. When we get to Equation (4.8) though, we need an argument different from Lemma 4.3 because it does not hold for permutations in SPONGE. We perform the same analysis of the probability function as in the proof of Lemma 4.3. Only the final argument is missing as we cannot use Theorem 2.21: $\mathbb{P}\left[\forall i \in [2q] : \text{SPONGE}_{\mathbf{f}}(\mathbf{M}^i, \ell_i) = \mathbf{Z}^i\right]$ is not a polynomial in $|\mathcal{C}|^{-1}$ if $\mathbf{f}$ is a permutation. Instead we formulate a generalization of Theorem 2.21 in Lemma 4.12 below that leads to the claimed bound. $\square$

Let us now highlight the differences we encounter when analyzing the case of permutations when following the reasoning of the proof of Lemma 4.3. The first and main difference is that the expression for the probability of a single evaluation of $\mathbf{f}$ (equations (4.42)) changes to:

$$\begin{aligned}
&\text{FLAG}(\nabla\text{-}\mathbf{cf}^i_j) \in \{\boxed{\mathrm{u}}, \boxed{\mathrm{f}}\} \\
&\Rightarrow \mathbb{P}\left[\mathbf{f}({}^{(\boxed{\mathrm{u}}\vee\boxed{\mathrm{f}})}S^i_{j\oplus}) = S \mid \bigwedge_{(i',j')\prec(i,j)} \left(\mathbf{f}(S^{i'}_{j'\oplus}) = S^{i'}_{j'+1}\right)\right] = \frac{1}{|\mathcal{A}| \cdot |\mathcal{C}| - \mathbf{u}_{\text{prev}}(i,j)},
\end{aligned} \tag{4.77}$$

$$\begin{aligned}
&\text{FLAG}(\nabla\text{-}\mathbf{cf}^i_j) = \boxed{\mathrm{n}} \\
&\Rightarrow \mathbb{P}\left[\mathbf{f}({}^{\boxed{\mathrm{n}}}S^i_{j\oplus}) = S \mid \bigwedge_{(i',j')\prec(i,j)} \left(\mathbf{f}(S^{i'}_{j'\oplus}) = S^{i'}_{j'+1}\right)\right] = \begin{cases} 1 & \text{if } S = \nabla\text{-}\mathbf{cf}^i_j(2) \\ 0 & \text{otherwise} \end{cases},
\end{aligned} \tag{4.78}$$

where

$$\mathbf{u}_{\text{prev}}(i,j) := \left|\left\{P^{i'}_{j'} : (i',j') \prec (i,j) \wedge \text{FLAG}(P^{i'}_{j'}) \in \{\boxed{\mathrm{u}}, \boxed{\mathrm{f}}\}\right\}\right| \tag{4.79}$$

is the number of unique states preceding the position (i, j). Note that we assume we have done all steps leading to Equation (4.42).

The product in Equation (4.40) for $\mathbf{p}\text{-}\nabla\text{-}\mathbf{cf} \in \mathbf{p}\text{-}\nabla\text{-}\mathbf{CF}(\mathbf{M}, \mathbf{Z}, \bar{u})$ and for $\mathbf{f}$ being a random permutation evaluates not to Equation (4.50), but instead to

$$\prod_{(i,j)=(1,1)}^{(q,k_q-1)} \mathbb{P}\left[\left(\mathbf{f}(S^i_{j\oplus}) = S^i_{j+1}\right) \mid \bigwedge_{(i',j')\prec(i,j)} \left(\mathbf{f}(S^{i'}_{j'\oplus}) = S^{i'}_{j'+1}\right)\right] = \prod_{i=0}^{\bar{u}-1} \frac{1}{|\mathcal{A}| \cdot |\mathcal{C}| - i}. \tag{4.80}$$

Algorithm 4.3 also changes when we consider random permutations. We need to shrink the set of possible states in the last column to $\left(|\mathcal{C}| - \mathbf{u}_{\text{prev}}(P, i, k_i)\right)$ (line **19** in CALC) and modify the number of possible assignments of values of non-unique states (line **20** in CALC). The rest is exactly the same, we will refer to the modified algorithm as CALCPER.

In the case of permutations we need to add one constraint to N-POSSIBILITIES

$$\begin{aligned}\Pi_{\text{per}}\left(P, n; n_1, n_2, \dots, n_f\right) := \{\pi \in \Pi_{\text{class-ind}}(n; n_1, n_2, \dots, n_f) \mid \\ \text{there are no objects in class } i \text{ prior to the } i\text{-th state according to } P \wedge \\ \text{non-unique n outputs of unique states (u} \vee \text{f) have different values}\}.\end{aligned} \tag{4.81}$$

Eventually we count the number of possible assignments of values of non-unique states:

$$\text{N-POSSIBILITIESPER}(n, f, P) := \sum_{(n_1, n_2, \dots, n_f) \in \mathcal{D}(n,f)} \left|\Pi_{\text{per}}\left(P, n; n_1, n_2, \dots, n_f\right)\right|. \tag{4.82}$$

Up to this point we have shown how to deal with combinatorial problems emerging from changing the internal function to a permutation. The main problem is different though; for random permutations the expression we derive in the last step of the proof in Section 4.5.4 is not an expression in $1/|\mathcal{C}|$. The expression we end up with is similar to Equation (4.69), but the probability comes from Equation (4.80):

$$\mathbf{p}(|\mathcal{C}|^{-1}) = \sum_{\bar{u}_{\text{abs}}, u_{\text{abs}}, \bar{u}_{\text{squ}}, u_{\text{squ}}, P} \text{CALCPER}(P) \prod_{i=0}^{\bar{u}-1} \frac{1}{|\mathcal{A}| \cdot |\mathcal{C}| - i}. \tag{4.83}$$

Unfortunately the above expression does not fit into the assumptions of Theorem 4.11 below which is the basis of the polynomial method allowing us to bound the adversary's advantage.

Theorem 4.11 (Theorem B.1. in [Zha12] for $\Delta = 1$)**.** *Let $\mathbf{p}(\lambda)$ be a polynomial in λ of degree d such that $0 \leq \mathbf{p}(0) \leq 1$, and $0 \leq \mathbf{p}(1/t) \leq 1$ for all $t \in \mathbb{Z}_+$ Then $|\mathbf{p}(1/t) - \mathbf{p}(0)| < \frac{\pi^2 d^3}{6t}$ for all $t \in \mathbb{Z}_+$.*

To deal with this problem we state a lemma relaxing a bit the requirements of Theorem 2.20.

Lemma 4.12. *For every $2q$ pairs $\forall i \in [2q] : (X^i, Y^i) \in \mathcal{X} \times \mathcal{Y}$ we define a function $\mathbf{g}_j(1/t) := \mathbb{P}_{\mathbf{h} \leftarrow \mathfrak{F}_t}[\forall i \in [2q] : \mathbf{h}(X^i) = Y^i]$, where j is the index enumerating different pairs. Assume that there exists $a \in \mathbb{N}$ and for every j there exist polynomials $\mathbf{p}'_j, \mathbf{p}''_j$,* *such that for every j and every $t \in \mathbb{N}$ with $t > a$, $\mathbf{g}_j(1/t)$ is bounded from below and above by polynomials $\mathbf{p}'_j, \mathbf{p}''_j$ respectively:*

$$\mathbf{p}'_j\left(\frac{1}{t-a}\right) \leq \mathbf{g}_j\left(\frac{1}{t}\right) \leq \mathbf{p}''_j\left(\frac{1}{t-a}\right), \tag{4.84}$$

such that $\mathbf{p}'_j(0) = \mathbf{p}''_j(0) = \mathbf{g}_j(0)$ and the degrees of the polynomials are d' and d'' respectively. Moreover $0 \leq \mathbf{p}'_j\left(\frac{1}{t-a}\right)$ and $\mathbf{p}''_j\left(\frac{1}{t-a}\right) \leq 1$ for all $t \in \mathbb{N}$ with $t > a$. Then

$$\left| \underset{\mathbf{h}\leftarrow\mathfrak{F}_t}{\mathbb{P}}\left[b = 1 : b \leftarrow \mathrm{A}^{|\mathbf{h}\rangle}\right] - \underset{\mathbf{h}\leftarrow\mathfrak{F}_\infty}{\mathbb{P}}\left[b = 1 : b \leftarrow \mathrm{A}^{|\mathbf{h}\rangle}\right]\right| < \frac{\pi^2(\max\{d', d''\})^3}{6(t-a)}. \tag{4.85}$$

Proof. Here we follow the proof of Theorem 7.3 in [Zha12] to make sure all assumptions are fulfilled to state that closeness of polynomials implies small adversarial advantage. Let us say that $\underset{\mathbf{h}\leftarrow\mathfrak{F}_t}{\mathbb{P}}\left[\mathrm{A}^{|\mathbf{h}\rangle}() = 1\right] - \underset{\mathbf{h}\leftarrow\mathfrak{F}_\infty}{\mathbb{P}}\left[\mathrm{A}^{|\mathbf{h}\rangle}() = 1\right] = \epsilon$, w.l.o.g. we can assume that $\epsilon > 0$. Also without loss of generality, using the fact that $\underset{\mathbf{h}\leftarrow\mathfrak{F}_t}{\mathbb{P}}\left[\mathrm{A}^{|\mathbf{h}\rangle}() = 1\right]$ is a linear combination of $\underset{\mathbf{h}\leftarrow\mathfrak{F}_t}{\mathbb{P}}[\forall i \in [2q] : \mathbf{h}(X^i) = Y^i] = \mathbf{g}_j(1/t)$ we can assume that all the coefficients in this combination are real. Therefore, we have

$$\epsilon = \sum_j \alpha_j(\mathbf{g}_j(1/t) - \mathbf{g}_j(0)) \leq \sum_j \alpha_j(\tilde{\mathbf{p}}_j\left(\frac{1}{t-a}\right) - \mathbf{g}_j(0)), \tag{4.86}$$

where $\tilde{\mathbf{p}}_j = \begin{cases}\mathbf{p}'_j & \text{if } \alpha_j < 0\\ \mathbf{p}''_j & \text{if } \alpha_j \geq 0\end{cases}$. Note that $\tilde{\mathbf{p}}(\frac{1}{t-a}) := \sum_j \alpha_j \tilde{\mathbf{p}}_j\left(\frac{1}{t-a}\right)$ is a polynomial of degree at most $\max\{d', d''\}$. If we set $t' := t - a$, it is straightforward to verify that $\tilde{\mathbf{p}}$ fulfills the assumptions of Theorem 4.11 above for all $t' \in \mathbb{Z}_+$. As $\mathbf{g}_j, \mathbf{p}'_j, \mathbf{p}''_j, \tilde{\mathbf{p}}$ all take on the same value for $t \to \infty$, we obtain that $\tilde{\mathbf{p}}(0) = \mathbf{g}_j(0)$, and hence the claim follows. $\square$

Now we just need to show that $\mathbb{P}[\forall i \in [2q] : \text{Sponge}_{\mathbf{f}}(\mathbf{M}^i, \ell_i) = \mathbf{Z}^i]$ is bounded by polynomials and find their degree. We are going to show that there are $\mathbf{p}', \mathbf{p}''$ (indexed with j in the statement of the lemma) that are polynomials and that fulfill the assumptions of Lemma 4.12 for $a = 1$. For each set of pairs of inputs and outputs we consider g to be a sum like in Equation (4.83). To do that we need to distinguish between placements P that involve at least two consecutive unique states and those that do not.

Let us deal with the case of P with at least two consecutive unique states first. We can bound the probability part of Equation (4.83) as follows:

$$\prod_{i=1}^{\bar{u}-1} \frac{1}{|\mathcal{A}| \cdot |\mathcal{C}| - i} \leq \prod_{i=1}^{\bar{u}-1} \frac{1}{|\mathcal{A}| \cdot |\mathcal{C}| - (\kappa - 1)} = \left(\frac{1}{|\mathcal{A}|} \cdot \frac{1}{|\mathcal{C}| - \frac{\kappa-1}{|\mathcal{A}|}}\right)^{\bar{u}-1} \tag{4.87}$$

$$\leq \frac{1}{|\mathcal{A}|^{\bar{u}-1}} \left(\frac{1}{|\mathcal{C}|-1}\right)^{\bar{u}-1}, \tag{4.88}$$

$$\prod_{i=1}^{\bar{u}-1} \frac{1}{|\mathcal{A}| \cdot |\mathcal{C}| - i} \geq \prod_{i=1}^{\bar{u}-1} \frac{1}{|\mathcal{A}| \cdot |\mathcal{C}|} \tag{4.89}$$

$$\geq \frac{1}{|\mathcal{A}|^{\bar{u}-1}} \left(1 - \frac{1}{|\mathcal{C}|-1}\right)^{\bar{u}-1} \left(\frac{1}{|\mathcal{C}|-1}\right)^{\bar{u}-1}. \tag{4.90}$$

Note that we have skipped the first term in the product that is supposed to range from 0 to $\bar{u}-1$. We have done it because in Equation (4.83) the fraction is multiplied by CalcPer and $1/|\mathcal{C}|$ simplifies. More concretely, the first term that is output by CalcPer necessarily involves $|\mathcal{C}|$ (not $|\mathcal{C}|-1$). Thanks to how we divide the product the final expression is a polynomial in $\frac{1}{|\mathcal{C}|-1}$.

In the latter case, no consecutive pairs of unique states, we bound every element (from $i=0$ to $\bar{u}-1$) of the product like in the above inequalities.

As for CalcPer we just treat $|\mathcal{C}|-1$ as the new variable. Note that now $\mathbf{p}'_j$ and $\mathbf{p}''_j$ are polynomials in $(|\mathcal{C}|-1)^{-1}$. Polynomials $\mathbf{p}''_j$ are defined as $\mathbf{g}_j$ (so Equation (4.83) for a specific input-output pair) with with $\prod_{i=1}^{\bar{u}-1} \frac{1}{|\mathcal{A}|\cdot|\mathcal{C}|-i}$ swapped to $\frac{1}{|\mathcal{A}|^{\bar{u}-1}} \left(\frac{1}{|\mathcal{C}|-1}\right)^{\bar{u}-1}$ from Equation (4.88)—with the exception of P with no consecutive pairs of unique states. Polynomials $\mathbf{p}'_j$ are defined similarly to $\mathbf{p}''_j$ but with Equation (4.90) being the exchange for the probability term in Equation (4.83).

Polynomial $\mathbf{p}''$ has the same degree as the the polynomial corresponding to $\mathbf{g}$ in the proof for functions, i.e. following the derivation of Equation (4.71) it equals $d'' = \eta = 2q(m+z-2)$. From the above lower bound in Equation (4.90) however, we get that $d' = 2d''$.

The last assumption we need to check is for $\mathbf{p}', \mathbf{p}''$ to be bounded by 0 and 1 for $|\mathcal{C}| > 1$. Note that it is enough to show that $\mathbf{p}' \geq 0$ and $\mathbf{p}'' \leq 1$. We already know that $\mathbf{p}'$ and $\mathbf{p}''$ bound $\mathbf{g}$. For the lower bound $\mathbf{p}' \geq 0$ comes from the fact that all coefficients are positive (they equal CalcPer(P)) and so is $\left(1 - \frac{1}{|\mathcal{C}|-1}\right) \frac{1}{|\mathcal{C}|-1}$ for $|\mathcal{C}| > 1$. For the upper bound of $\mathbf{g}$ we need to check that $\mathbf{p}'' \leq 1$. Following the algorithm Algorithm 4.3 we can see that CalcPer(P) is bounded by $|\mathcal{A}|^{\bar{u}-q} |\mathcal{C}|((|\mathcal{C}|-1)^{\bar{u}-1}$, so as long as the number of terms in Equation (4.83) is smaller than $|\mathcal{A}|^q$—which is our implicit assumption—then $\mathbf{p}'' \leq 1$.

The above discussion, together with Lemma 4.12 proves Theorem 4.10.

CHAPTER

5 DISPROVING THE CONJECTURED IMPOSSIBILITY OF QUANTUM INDIFFERENTIABILITY

Chapter contents

In this chapter, we analyze joint probability distributions that come from outcomes of sequences of quantum measurements performed on sets of quantum states. First, we identify some properties of these distributions that need to be fulfilled to get a classical behavior. Secondly, we prove that a joint distribution exists if and only if measurement operators "on-state" permute (permutability is the commutativity of more than two operators). By "on-state" we mean properties of operators that hold only on a subset of states in the Hilbert space. Then, we disprove a conjecture proposed by Carstens, Ebrahimi, Tabia, and Unruh [Car+18], which states that the property of partial on-state permutation implies full on-state permutation. We disprove this conjecture by finding a counterexample where pairwise "on-state" commutativity does not imply on state permutability, unlike in the case of commutativity for all states in the Hilbert space.

Finally, we explore the new concept of on-state commutativity by showing a simple proof that if two projections almost commute on state, then there is a commuting pair of operators that are (on state) close to the originals. This result was originally proven by Hastings [Has09] for general operators.

5.1 Introduction

In this chapter we propose a basic formalism for studying classical distributions that come from sequences of measurements on quantum states.

Our initial motivation comes from studying a conjecture proposed in a recent paper by Carstens, Ebrahimi, Tabia and Unruh [Car+18]. Their result on quantum indifferentiability relies on a conjecture proposed by them, which informally states that commutation of projectors with respect to a fixed quantum state implies a classical joint distribution of their measurement outcomes. More concretely, they conjecture the following for some t (a parameter of the statement).[1]

Conjecture 5.1 (Informal). *If we have a set of N binary projective measurements $\{\mathsf{P}_i, \mathbb{1} - \mathsf{P}_i\}_{i\in\{1,\dots,N\}}$ such that for any t of the projectors (or their complements $\mathbb{1} - \mathsf{P}$) we have $\mathsf{P}_{i_1}\cdots\mathsf{P}_{i_t}|\psi\rangle = \mathsf{P}_{\boldsymbol{\sigma}(i_1)}\cdots\mathsf{P}_{\boldsymbol{\sigma}(i_t)}|\psi\rangle$ for any permutation $\boldsymbol{\sigma}$ of the t-element set then there exist random variables $X_1,\dots,X_N$ with a joint distribution $\mathfrak{D}$ such that $\forall i_1,\dots,i_t$, the marginals of this distribution on $X_{i_1},\dots,X_{i_t}$ correspond to measuring $|\psi\rangle$ with measurements with indexes $i_1,\dots,i_t$.*

Motivated by this conjecture, our goal is to study the behavior of N random variables $X_1, X_2, \dots, X_N$ corresponding to the outcomes of a sequence of quantum measurements that commute (we say "commute" but technically mean

[1]See Conjecture 5.18 for the formal statement.

"permute" whenever we talk about more that two operators) on a set of quantum states $\mathcal{F} \subseteq \mathcal{D}(\mathcal{H})$. Surprisingly, such results have only been studied for $\mathcal{F} = \mathcal{D}(\mathcal{H})$, i.e. measurements commuting on *all* quantum states.

The focal point of this chapter is to study which are the necessary and sufficient properties of the quantum setup, so that such a joint probability distribution is well-defined. With this in hand, we then have two applications. First, we disprove Conjecture 5.1. Secondly, we show a simpler proof for a variant of the result by Hastings [Has09] on operators that almost-commute on specific states.

To be able to explain our contributions in more details, we will first start with a detour to very basic properties of probability distributions that arise from classical processes. Then, we discuss how these properties could be defined in the quantum setting (but, unfortunately, they do not hold for general quantum setups), and finally we state our results and discuss related works.

5.1.1 Distributions of Classical Experiments

We discuss here properties of probability distributions that may be obvious at first but are crucial and not trivial in the quantum world.

In the following, we let A, B, C be events that come from a classical experiment[2]. We denote the event corresponding to A not happening as $\overline{A}$, the probability that A and B both happen as $\mathbb{P}[A, B]$, and the probability that A happens conditioned on the fact that event B happens as $\mathbb{P}[A|B] = \frac{\mathbb{P}[A,B]}{\mathbb{P}[B]}$ (assuming $\mathbb{P}[B] \neq 0$).

The first property that we want to recall about classical distributions is that we can compute the *marginal* distribution when given the *joint* distribution:

Property 5.2 (Classical Marginals). $\mathbb{P}[A \mid C] = \mathbb{P}[A, B \mid C] + \mathbb{P}[A, \overline{B} \mid C]$.

A second property that we want to recall is that the probability that A and $\overline{A}$ occur is 0, even when considering other events:

Property 5.3 (Classical Disjointness). $\mathbb{P}[A, B \mid \bar{A}]\mathbb{P}[\overline{A}] = \mathbb{P}[A, B, \overline{A}] = 0$.

Another property that we have classically is *reducibility*, which says that the probability of events A and A both happening is the same as the probability of A.

Property 5.4 (Classical Reducibility). $\mathbb{P}[A, B \mid A]\mathbb{P}[A] = \mathbb{P}[A, B, A] = \mathbb{P}[A, B]$.

Finally, the last property we study is *sequential independence* of events. Roughly, this property just says that the probability that event A happens and that event B happens is the same as the probability that event B happens and that event A happens. Namely that $\mathbb{P}[A, B] = \mathbb{P}[B, A]$.

[2]Formally, a probability space $(\Omega, \mathbb{P})$

Property 5.5 (Classical Sequential Independence). $\mathbb{P}[A \mid B]\mathbb{P}[B] = \mathbb{P}[A, B] = \mathbb{P}[B \mid A]\mathbb{P}[A]$.

We stress that these properties hold trivially for *all* classical distributions and all events such that $\mathbb{P}[A] \neq 0$, $\mathbb{P}[\overline{A}] \neq 0$, $\mathbb{P}[B] \neq 0$, and $\mathbb{P}[C] \neq 0$.

5.1.2 Distributions of Quantum Experiments and their Properties

Our goal is to find necessary conditions for the existence of a classical description of the experiment where we perform a sequence of N general measurements, irrespective of the order. At this point we consider measurements and observables with an arbitrary number of outcomes. More concretely, we aim to find the properties of measurements $\mathcal{M}_1, \dots, \mathcal{M}_N$—defined as $\mathcal{M}_i = \{\mathsf{E}_i^x\}_{x \in \mathcal{X}_i}$ for every $i \in \{1, \dots, N\}$—on specific subsets of quantum states $\mathcal{F}$ so that there exists a joint distribution of random variables $X_1, X_2, \dots, X_N$ such that all marginals of this distribution on $X_{i_1}, \dots, X_{i_t}$ correspond to measuring a state $|\psi\rangle \in \mathcal{F}$ with measurements $\mathsf{E}_{i_1}, \dots, \mathsf{E}_{i_t}$.

The main obstacle in this task is the fact that quantum measurements do not necessarily commute, unlike in the classical world: the chosen order for performing the measurements influences the final probability distribution of the joint measurement outcomes. Because of that, we will consider the quantum analog of Properties 5.2 to 5.5, and study when such properties hold in the quantum case, and their implication for having such a joint distribution. Our connections closely follow [ME84], where they show that the existence of a joint distribution for *two* arbitrary quantum observables (Hermitian operators) on *every* quantum state is equivalent to their commutation. In this chapter, we show how to extend their analysis in two ways: we are interested in multiple general measurements[3] and we consider specific sets of quantum states. In order to carry out this analysis, we extend the properties described in Section 5.1.1 to quantum measurements and study their relations to each other. We leave the formal definitions of the quantum analogs of these classical properties to Section 5.2.1.

5.1.3 Our Results

Using the formalism described in the previous section, we prove the following connections between joint quantum distributions and the measurement operators.

[3]Observables can measured with a projective measurement, a subclass of the most general quantum measurements.

First, we show that in the on-state case, we have that there is a joint distribution if and only if all operators permute on the state. This result is a generalization of the classic results from [Nel67; Fin73; Fin82; ME84] to the on-state case.

Result 5.6 (Informal statement of Theorem 5.17)**.** *Fix a set of quantum states $\mathcal{F}$. A set of measurements yield a joint distribution on each state in $\mathcal{F}$ if and only if these operators permute on every state in $\mathcal{F}$.*

Then, we show that pairwise on-state commutation does not imply full on-state permutation, unlike in the case of permutation on all states. This fact—that we prove via a numerical counterexample—together with Result 5.6 implies that Conjecture 5.1 is false.

Result 5.7 (Informal statement of Theorem 5.19)**.** *Conjecture 5.1 is false.*

Finally, our last result is a simpler proof for a restricted version of Theorem 1 in [Has09], which states that if two operators A and B almost-commute, we can find commuting operators[4] A' and B' that are close to A and B, respectively. In our case, we consider on-state commutation instead of the regular one, and unlike in [Has09], our proof works only for projectors.

Result 5.8 (Making almost commuting projectors commute)**.** *Given any two projectors P_1 and P_2 and a state $|\psi\rangle$ we have that if $\|(\mathsf{P}_1\mathsf{P}_2 - \mathsf{P}_2\mathsf{P}_1)|\psi\rangle\| = \epsilon$ then there is a projector P_2' that is close to the original projector on the state $\|(\mathsf{P}_2' - \mathsf{P}_2)|\psi\rangle\| \leq \sqrt{2}\epsilon$ and $[\mathsf{P}_1, \mathsf{P}_2'] = 0$.*

5.1.4 Related Work

A prominent result in the literature is that a joint distribution for a set of measurements exists if and only if all the operators pairwise commute. Different versions of this result were previously proven: In [Nel67] the author considers the case of continuous variables and N observables. A similar result but without specifying the Hilbert space is achieved with different mathematical tools in [Fin82]. In the specific case where we have only two observables, we mention three works; In [Fin73] and [ME84] the authors prove the classic problem in a similar way, but using different mathematical tools. All but the first work mentioned here focus on the joint distribution as a functional from the space of states. An approach using $*$-algebras was presented by Hans Maassen in [Maa06; Maa10].

The authors of [GN02] analyze the case of general measurements but prove that the measurement operators pairwise commute if and only if the square-root operators permute (Corollaries 3 and 6 in [GN02]), in the sense of our Definition 5.15 (for all states in $\mathcal{H}$). In general the problem of conditional probabilities in Quantum Mechanics was discussed by Cassinelli and Zanghi in [CZ83].

[4]Operators A and B commute if $[\mathsf{A}, \mathsf{B}] := (\mathsf{AB} - \mathsf{BA}) = 0$.

The related problems of incompatible device measurement and joint measurability of quantum effects are covered in [HMZ16] and [BN18b] respectively.

In [Lin97; FR96] the authors prove that for any two Hermitian matrices if their commutator has small norm, then there are operators close to the originals that fully commute. In [Has09] Hastings proves how close the new operators are in terms of the norm of the commutator.

Organization

In Section 5.2, we discuss distributions of quantum experiments and their properties, in Section 5.3, we discuss the almost-commuting case.

5.2 Distributions of Quantum Experiments

In this section, we study the description of the statistics of outcomes derived from a sequence of measurements. Our approach is to consider the quantum version of the classical properties described in Section 5.1.1. Since quantum measurements do not commute in general, these quantum properties do not always hold. We then study the connection between properties of the measurements and the properties of their outcome distribution.

The structure of our proofs follows [ME84], where they show that for two Hermitian observables there is a joint distribution for the outcomes of their joint measurement if and only if they commute. We stress that the result in [ME84] only works for measurements that commute on every quantum state and our result extends it to the case of joint distributions defined for a limited set of states.

In the following, we denote POVM elements with Q and their square-roots with R. In Section 5.2.1, we define the quantum analogues of the classical properties of distributions defined in Section 5.1.1. We define a functional $\mathbf{W}$ (specified in the next section) as the main object of our discussion to justify the properties we impose on joint distributions and highlight our goal to define a classical distribution. Mathematically, the crucial objects of the next section are the operators Q. Then, in Section 5.2.2, we state and prove the main result of this section, where we show a connection between existence of a distribution and permutability—a generalization of commutativity—of the corresponding measurement operators.

5.2.1 Properties of Distributions of Quantum Experiments

Let N be a positive integer, let $\mathcal{X}_1, \ldots, \mathcal{X}_N$ be arbitrary finite non-empty sets, and let $\mathcal{H}$ be a Hilbert space. By $\mathcal{D}_{\succeq 0}(\mathcal{H})$ we denote the set of positive operators on $\mathcal{H}$. Then, for every POVM $\mathsf{Q}_{\mathcal{N}}$ on $\mathcal{H}$ with POVM elements $\mathsf{Q}_{\mathcal{N}}^{\vec{x}}$ indexed by $\vec{x} \in \mathcal{X}_1 \times \cdots \times \mathcal{X}_N$ we associate the functional $\mathbf{W}_{\mathcal{N}}^{\mathsf{Q}_{\mathcal{N}}} : \mathcal{D}_{\succeq 0}(\mathcal{H}) \times (\mathcal{X}_1 \times \cdots \times \mathcal{X}_N) \to [0,1]$, where we may omit the superscript $\mathsf{Q}_{\mathcal{N}}$ when the POVM is clear from the context. The subscript $\mathcal{N} := \{1, \ldots, N\}$ of $\mathbf{W}$ denotes the set of random variables that we talk about. Moreover, for every i $\mathcal{X}_i$ is the set of outcomes of a random variable X_i. We define this functional[5] as

$$\mathbf{W}_{\mathcal{N}}^{\mathsf{Q}_{\mathcal{N}},\rho}(\vec{x}) := \mathrm{Tr}\left(\mathsf{Q}_{\mathcal{N}}^{\vec{x}}\rho\right). \tag{5.1}$$

In the following we omit the superscript $\mathsf{Q}_{\mathcal{N}}$ whenever the particular POVM is clear from the context. This definition is similar to the one proposed by [ME84].

As we already mentioned, the aim of this section is to define the necessary properties of $\mathbf{W}_{\mathcal{N}}$ so that it is a joint distribution for a sequence of N measurements. This means that every random variable X_i, for $i \in \mathcal{N}$, corresponds to a general measurement (i.e. POVM) $\mathcal{M}_i := \{\mathsf{E}_i^x\}_{x\in\mathcal{X}_i}$, where all $\mathsf{E}_i^x \succeq 0$ and $\sum_{x\in\mathcal{X}_i} \mathsf{E}_i^x = \mathbb{1}$. The positive square-root operators of E_i^x are K_i^x. So we have to keep in mind that these operators are fixed throughout this chapter.

A functional $\mathbf{W}_{\mathcal{N}}$ defined as in Equation (5.1) is one of the three objects that we use to define a joint distribution describing outcomes of N general measurements. The other object is a family of POVM elements. For each $\mathcal{S} \subseteq \mathcal{N}$ (also for $\mathcal{N}$ itself) we let $\mathsf{Q}_{\mathcal{S}} := \{\mathsf{Q}_{\mathcal{S}}^{\vec{y}}\}_{\vec{y}}$ to be (for now arbitrary) POVMs where $\vec{y} \in \mathcal{X}_{\mathcal{S}(1)} \times \cdots \times \mathcal{X}_{\mathcal{S}(|\mathcal{S}|)}$. The fact that $\mathsf{Q}_{\mathcal{S}}$ is a POVM implies that for every set $\mathcal{S}$ we have $\forall \vec{y} \in \prod_{i\in\mathcal{S}} \mathcal{X}_i :\ \mathsf{Q}_{\mathcal{S}}^{\vec{y}} \succeq 0$ and $\sum_{\vec{y}} \mathsf{Q}_{\mathcal{S}}^{\vec{y}} = \mathbb{1}$.

In the following we define properties of the tuple $(\mathbf{W}_{\mathcal{N}}, \{\mathsf{Q}_{\mathcal{S}}\}_{\mathcal{S}\subseteq\mathcal{N}}, \{\mathcal{M}_i\}_{i\in\mathcal{N}})$ such that it is a joint distribution (to be defined in Definition 5.14). Note that in Equation (5.1) the operators from $\mathsf{Q}_{\mathcal{N}}$ are elements of the second part of the tuple $(\mathbf{W}_{\mathcal{N}}, \{\mathsf{Q}_{\mathcal{S}}\}_{\mathcal{S}\subseteq\mathcal{N}}, \{\mathcal{M}_i\}_{i\in\mathcal{N}})$. There are relations on the elements of the tuple we need to be aware of: Functional $\mathbf{W}_{\mathcal{N}}$ is defined using operators from the second element of the tuple, we keep it as a part of the tuple and a central object of this section to highlight our goal to define a *classical* joint distribution. The individual measurements $\{\mathcal{M}_i\}_i$ from the third element of the tuple are fixed as we strive to define a joint distribution for $\{\mathcal{M}_i\}_i$.

The first property is that $\mathbf{W}_{\mathcal{N}}$ is defined according to the Born rule, i.e. the rule that the probability of any event corresponds to a measurement operator and a quantum state. From Equation (5.1) we know that $\mathbf{W}_{\mathcal{N}}$ fulfills:

- Normalization: for all $\rho \in \mathcal{D}(\mathcal{H})$, we have that $\sum_{\vec{x}} \mathbf{W}_{\mathcal{N}}^{\rho}(\vec{x}) = 1$, this is implied by the fact that $\sum_{\vec{x}} \mathsf{Q}_{\mathcal{N}}^{\vec{x}} = \mathbb{1}$.

- Linearity: for every $\vec{x} \in \mathcal{X}_1 \times \cdots \times \mathcal{X}_N$, $\rho_1, \rho_2 \in \mathcal{D}(\mathcal{H})$ and $\lambda_1, \lambda_2 \in [0,1]$ such that $\lambda_1 + \lambda_2 = 1$, we have that

$$\mathbf{W}_{\mathcal{N}}^{\lambda_1\rho_1+\lambda_2\rho_2}(\vec{x}) = \lambda_1 \mathbf{W}_{\mathcal{N}}^{\rho_1}(\vec{x}) + \lambda_2 \mathbf{W}_{\mathcal{N}}^{\rho_2}(\vec{x}).$$

[5]Note that the second superscript of $\mathbf{W}_{\mathcal{N}}^{\mathsf{Q}_{\mathcal{N}},\rho}(\vec{x})$ denotes the first input to the functional, so we have $\mathbf{W}_{\mathcal{N}}^{\mathsf{Q}_{\mathcal{N}}}(\rho, \vec{x})$.

- Non-negativity: for every $\vec{x} \in \mathcal{X}_1 \times \cdots \times \mathcal{X}_N$ and $\rho \in \mathcal{D}(\mathcal{H})$, we have $\mathbf{W}^{\rho}_{\mathcal{N}}(\vec{x}) \geq 0$.

Below we describe the quantum analogues of the properties described in Section 5.1.1. We define a joint distribution through a measurement. By imposing properties on the distribution we impose constraints on the distribution but also on the measurement operators. Whenever possible we treat the functional $\mathbf{W}_{\mathcal{N}}$ as the central object of our definitions (and not the accompanying operators); This choice highlights our goal to define a classical description of a quantum event, hence we focus on the classical functional instead of on quantum operators. Nonetheless, from a mathematical point of view (and via the Born rule) all the constraints we put on $\mathbf{W}_{\mathcal{N}}$ are easily translated to constraints on $\mathsf{Q}_{\mathcal{N}}$. In the following sections we recast these constraints on the measurement operators as permutability of all the measurements.

Marginals. Given the operators from the second part of the tuple, we define marginal distributions in a similar way to the definition from Equation(5.1):

$$\forall \mathcal{S} \subseteq \mathcal{N} \quad \mathbf{W}^{\rho}_{\mathcal{S}}(\vec{y}) := \operatorname{Tr}\left(\mathsf{Q}^{\vec{y}}_{\mathcal{S}}\rho\right). \tag{5.2}$$

Given any $\mathsf{Q}^{\vec{y}}_{\mathcal{S}}$ and their corresponding positive square-root operators $\mathsf{R}^{\vec{y}}_{\mathcal{S}}$ and ρ, we have that if $\operatorname{Tr}\left(\mathsf{Q}^{\vec{y}}_{\mathcal{S}}\rho\right) \neq 0$, then we define the conditional distribution for any sequence $\vec{x}$ as

$$\mathbf{W}^{\rho}_{\mathcal{N}}(\vec{x} \mid \vec{y}) := \mathbf{W}^{\mathsf{R}^{\vec{y}}_{\mathcal{S}}\rho\mathsf{R}^{\vec{y}\dagger}_{\mathcal{S}}}_{\mathcal{N}}(\vec{x}) / \mathbf{W}^{\rho}_{\mathcal{S}}(\vec{y}). \tag{5.3}$$

For all the measurements from the second element of $(\mathbf{W}_{\mathcal{N}}, \{\mathsf{Q}_{\mathcal{S}}\}_{\mathcal{S}\subseteq\mathcal{N}}, \{\mathcal{M}_i\}_{i\in\mathcal{N}})$, for a set $\mathcal{F} \subseteq \mathcal{D}(\mathcal{H})$, and for $\mathcal{U} \subseteq \mathcal{N}$, we define the "orbit" of the post-measurement states. For any $\mathcal{T} \subseteq \mathcal{U}$ of size t, we take $s \leq t$ sets $\mathcal{S}_1, \ldots, \mathcal{S}_s$ that are a *partition* of $\mathcal{T}$. In details we mean $\mathcal{T} \subseteq \mathcal{U}, s \leq |\mathcal{T}|,\ \mathcal{S}_1, \ldots, \mathcal{S}_s \subseteq \mathcal{T}, \forall i \mathcal{S}_i \neq \emptyset, \bigcup_{i=1}^{s} \mathcal{S}_i = \mathcal{T}, \forall i \neq j\, \mathcal{S}_i \cap \mathcal{S}_j = \emptyset$. We denote the partition by $\bigsqcup_{i=1}^{s} \mathcal{S}_i = \mathcal{T}$, where $\sqcup$ denotes the disjoint union. As a shorthand for picking a $\mathcal{T} \subseteq \mathcal{U}$ and a partition of it, we just write $\bigsqcup_{i=1}^{s} \mathcal{S}_i \subseteq \mathcal{U}$. We consider the post-measurement states generated by sequences of measurements corresponding to $\mathcal{S}_i$:

$$\begin{aligned} \mathcal{G}_{\mathcal{U}}(\mathcal{F}) := \Big\{ & \mathsf{R}^{\vec{y}_s}_{\mathcal{S}_s} \cdots \mathsf{R}^{\vec{y}_1}_{\mathcal{S}_1} \psi \mathsf{R}^{\vec{y}_1\dagger}_{\mathcal{S}_1} \cdots \mathsf{R}^{\vec{y}_s\dagger}_{\mathcal{S}_s} : \\ & \psi \in \mathcal{F}, \bigsqcup_{i=1}^{s} \mathcal{S}_i \subseteq \mathcal{U}, \vec{y}_i \in \mathcal{X}_{\mathcal{S}_i(1)} \times \cdots \times \mathcal{X}_{\mathcal{S}_i(|\mathcal{S}_i|)} \Big\}, \end{aligned} \tag{5.4}$$

where $\mathsf{R}^{\vec{y}_i}_{\mathcal{S}_i}$ are the positive square-root operators of $\mathsf{Q}^{\vec{y}_i}_{\mathcal{S}_i} = \mathsf{R}^{\vec{y}_i\dagger}_{\mathcal{S}_i}\mathsf{R}^{\vec{y}_i}_{\mathcal{S}_i}$. The subscript of $\mathcal{G}$ denotes the set we take the subsets of, usually it is $\mathcal{N}$ but later we also consider $\mathcal{N} \setminus \mathcal{S}$ for some $\mathcal{S}$.

With our quantum marginals property, we require from the joint distribution that summing over a subset of variables yields a valid marginal distribution. Importantly, this translates to a constraint on the measurement operators $\mathsf{Q}_{\mathcal{S}}^{\vec{y}}$: a sum of $\operatorname{Tr}\left(\mathsf{Q}_{\mathcal{N}}^{\vec{x}}\rho\right)$ for different values x_i and variables i has to equal the appropriate $\operatorname{Tr}\left(\mathsf{Q}_{\mathcal{S}}^{\vec{y}}\rho\right)$. We would also like to note that we fixed the individual operators in the beginning so all the arbitrary measurements $\mathsf{Q}_{\mathcal{S}}^{\vec{y}}$ have to agree with E_i^x.

Property 5.9 (Quantum Marginals). *We say that* $(\mathbf{W}_{\mathcal{N}}, \{\mathsf{Q}_{\mathcal{S}}\}_{\mathcal{S}\subseteq\mathcal{N}}, \{\mathcal{M}_i\}_{i\in\mathcal{N}})$ *has the* quantum marginals property on set $\mathcal{F}$ *if for every* $\mathcal{S} \subseteq \mathcal{N}$ *and every value* $\vec{x} \in \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_N$*, denoting* $\vec{x} := (x_1, x_2, \ldots, x_N)$*, and for every operator* $\rho \in \mathcal{G}_{\mathcal{N}}(\mathcal{F})$ *defined with* $\{\mathsf{Q}_{\mathcal{S}}\}_{\mathcal{S}\subseteq\mathcal{N}}$ *as in Equation* (5.4) *we have that*

$$\sum_{i\in\mathcal{N}\setminus\mathcal{S}} \sum_{x_i\in\mathcal{X}_i} \mathbf{W}_{\mathcal{N}}^{\rho}(\vec{x}) = \mathbf{W}_{\mathcal{S}}^{\rho}(\vec{x}_{\mathcal{S}}), \tag{5.5}$$

where $\mathbf{W}_{\mathcal{S}}^{\rho}(\vec{x}_{\mathcal{S}}) = \operatorname{Tr}\left(\mathsf{Q}_{\mathcal{S}}^{\vec{x}_{\mathcal{S}}}\rho\right)$*. Moreover for all* $i \in \mathcal{N}$ *and all* $x_i \in \mathcal{X}_i$

$$\mathbf{W}_i^{\rho}(x_i) = \operatorname{Tr}\left(\mathsf{E}_i^{x_i}\rho\right). \tag{5.6}$$

On an intuitive level, Property 5.9 imposes constraints on all the measurement operators $\mathsf{Q}_{\mathcal{S}}^{\vec{y}}$. Crucially, for $|\mathcal{S}| = 1$ the constraints involve the sequence of measurements that we want to ultimately describe. Properties that we introduce in the remainder of this section can also be thought of as constraints on $\mathsf{Q}_{\mathcal{S}}^{\vec{y}}$. Note however that we always constrain only the trace, not the operator itself.

Disjointness. It follows from the definition of POVMs that the quantum measurement operators need not be orthogonal, and this implies that the disjointness property (Property 5.3) does not hold in generality quantumly.

Disjointness is a property that concerns a post-measurement state of a set $\mathcal{S}$ of variables.

Property 5.10 (Quantum Disjointness). *We say that* $(\mathbf{W}_{\mathcal{N}}, \{\mathsf{Q}_{\mathcal{S}}\}_{\mathcal{S}\subseteq\mathcal{N}}, \{\mathcal{M}_i\}_{i\in\mathcal{N}})$ *has the* quantum disjointness property on set $\mathcal{F}$ *if for every subset* $\mathcal{S} \subseteq \mathcal{N}$*, for every operator* $\rho \in \mathcal{G}_{\mathcal{N}\setminus\mathcal{S}}(\mathcal{F})$ *(defined with* $\{\mathsf{Q}_{\mathcal{S}}\}_{\mathcal{S}\subseteq\mathcal{N}}$ *as in Equation* (5.4)*), and for every value* $\vec{x} \in \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_N$ *and* $\vec{y} \in \prod_{i\in\mathcal{S}} \mathcal{X}_i$*, we have that if* $\vec{y} \neq \vec{x}_{\mathcal{S}}$*, then*

$$\mathbf{W}_{\mathcal{N}}^{\rho}(\vec{x} \mid \vec{y})\mathbf{W}_{\mathcal{S}}^{\rho}(\vec{y}) = 0, \tag{5.7}$$

where $\mathbf{W}_{\mathcal{N}}^{\rho}(\vec{x} \mid \vec{y})\mathbf{W}_{\mathcal{S}}^{\rho}(\vec{y}) = \mathbf{W}_{\mathcal{N}}^{\mathsf{R}_{\mathcal{S}}^{\vec{y}}\rho\mathsf{R}_{\mathcal{S}}^{\vec{y}\dagger}}(\vec{x}) = \operatorname{Tr}\left(\mathsf{Q}_{\mathcal{N}}^{\vec{x}}\mathsf{R}_{\mathcal{S}}^{\vec{y}}\rho\mathsf{R}_{\mathcal{S}}^{\vec{y}\dagger}\right)$.

Reducibility. Reducibility (Property 5.4) is a similar property to disjointness but with the key difference that we condition on the same event:

Property 5.11 (Quantum Reducibility)**.** *We say that* $(\mathbf{W}_{\mathcal{N}}, \{\mathsf{Q}_{\mathcal{S}}\}_{\mathcal{S}\subseteq\mathcal{N}}, \{\mathcal{M}_i\}_{i\in\mathcal{N}})$ *has the* quantum reducibility property on set $\mathcal{F}$ *if for every subset* $\mathcal{S} \subset \mathcal{N}$*, every operator* $\rho \in \mathcal{G}_{\mathcal{N}\setminus\mathcal{S}}(\mathcal{F})$ *(defined with* $\{\mathsf{Q}_{\mathcal{S}}\}_{\mathcal{S}\subseteq\mathcal{N}}$ *as in Equation* (5.4)*), and value* $\vec{x} \in \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_N$*, we have that*

$$\mathbf{W}^{\rho}_{\mathcal{N}}(\vec{x} \mid \vec{x}_{\mathcal{S}})\mathbf{W}^{\rho}_{\mathcal{S}}(\vec{x}_{\mathcal{S}}) = \mathbf{W}^{\rho}_{\mathcal{N}}(\vec{x}), \tag{5.8}$$

whenever $\mathbf{W}^{\rho}_{\mathcal{S}}(\vec{x}_{\mathcal{S}}) > 0$*. The above is equivalent to* $\mathbf{W}^{\mathsf{R}^{\vec{y}}_{\mathcal{S}}\rho\mathsf{R}^{\vec{y}\dagger}_{\mathcal{S}}}_{\mathcal{N}}(\vec{x}) = \mathbf{W}^{\rho}_{\mathcal{N}}(\vec{x})$ *or in other words* $\mathrm{Tr}\left(\mathsf{Q}^{\vec{x}}_{\mathcal{N}}\mathsf{R}^{\vec{x}_{\mathcal{S}}}_{\mathcal{S}}\rho\mathsf{R}^{\vec{x}_{\mathcal{S}}\dagger}_{\mathcal{S}}\right) = \mathrm{Tr}\left(\mathsf{Q}^{\vec{x}}_{\mathcal{N}}\rho\right)$*.*

Note that the last two properties together allow us to conclude that the operators are (intuitively) *on-state projections*: Property 5.10 plays the role of different projectors being orthogonal and Property 5.11 that projecting twice to the same space does not change the resulting state.

Sequential Independence As previously discussed, the notion of time order in the quantum setting is much more delicate as the probabilistic events no longer commute. Let us go back to the example of the simple sequence (A, B) from Section 5.1.1 but now consider A and B as quantum POVM elements measured on the state ρ. Let us assume for simplicity that A and B are projections. The probability of measuring a with A is $\mathrm{Tr}\,(\mathsf{A}\rho)$ and the state after this measurement is $\rho_a := \frac{\mathsf{A}\rho\mathsf{A}}{\mathrm{Tr}(\mathsf{A}\rho)}$ so the probability of measuring the sequence (a, b) equals

$$\mathbb{P}[b \leftarrow \mathsf{B}(\rho_a)]\mathbb{P}[a \leftarrow \mathsf{A}(\rho)] = \mathrm{Tr}\left(\mathsf{B}\frac{\mathsf{A}\rho\mathsf{A}}{\mathrm{Tr}\mathsf{A}\rho}\right)\mathrm{Tr}\,(\mathsf{A}\rho) = \mathrm{Tr}\,(\mathsf{ABA}\rho)\,. \tag{5.9}$$

On the other hand the probability of measuring the sequence (b, a) equals $\mathrm{Tr}\,(\mathsf{BAB}\rho)$ which is in general different[6] than Equation (5.9). This simple example shows that sequential independence from [GN02] is not attained by all sequences of quantum measurements.

The quantum counterpart of Property 5.5 that we define below is special in the sense that the constraint we pose is put on the measurement operators. The reason for that is to clarify presentation. At the end of this paragraph, however, we prove that the quantum sequential independence property is implied by the previous more natural Properties 5.9, 5.10, and 5.11.

[6] An easy example of this fact is when $\rho = |0\rangle\langle 0|$, $\mathsf{A} = |0\rangle\langle 0|$, and $\mathsf{B} = |+\rangle\langle +|$, where $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. Then we have $\mathrm{Tr}\,(\mathsf{ABA}\rho) = 1/2$ and $\mathrm{Tr}\,(\mathsf{BAB}\rho) = 1/4$.

Property 5.12 (Quantum Sequential Independence)**.** *We say that* $(\mathbf{W}_{\mathcal{N}}, \{\mathsf{Q}_{\mathcal{S}}\}_{\mathcal{S}\subseteq\mathcal{N}}, \{\mathcal{M}_i\}_{i\in\mathcal{N}})$ *has the* quantum sequential independence property on set $\mathcal{F}$ *if for every density operator* $\psi \in \mathcal{F}$*, for all partitions* $\bigsqcup_{i=1}^{s}\mathcal{S}_i \subseteq \mathcal{N}$*, for any permutation* $\boldsymbol{\sigma} \in \mathcal{I}_{\text{per}}(\{1,\dots,s\})$*, and* $\vec{x} \in \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_N$ *such that* $\vec{x} := (x_1, x_2, \dots, x_N)$ *and* $\vec{y}_i := \left(x_{\mathcal{S}_i(1)}, x_{\mathcal{S}_i(2)}, \dots, x_{\mathcal{S}_i(|\mathcal{S}_i|)}\right)$*, we have that*

$$\begin{aligned}&\mathrm{Tr}\left(\mathsf{Q}_{\mathcal{S}_s}^{\vec{y}_s}\mathsf{R}_{\mathcal{S}_{s-1}}^{\vec{y}_{s-1}}\cdots\mathsf{R}_{\mathcal{S}_1}^{\vec{y}_1}\psi\mathsf{R}_{\mathcal{S}_1}^{\vec{y}_1\dagger}\mathsf{R}_{\mathcal{S}_2}^{\vec{y}_2\dagger}\cdots\mathsf{R}_{\mathcal{S}_{s-1}}^{\vec{y}_{s-1}\dagger}\right)\\ &\quad=\mathrm{Tr}\left(\mathsf{Q}_{\mathcal{S}_{\sigma(s)}}^{\vec{y}_{\sigma(s)}}\mathsf{R}_{\mathcal{S}_{\sigma(s-1)}}^{\vec{y}_{\sigma(s-1)}}\cdots\mathsf{R}_{\mathcal{S}_{\sigma(1)}}^{\vec{y}_{\sigma(1)}}\psi\mathsf{R}_{\mathcal{S}_{\sigma(1)}}^{\vec{y}_{\sigma(1)}\dagger}\mathsf{R}_{\mathcal{S}_{\sigma(2)}}^{\vec{y}_{\sigma(2)}\dagger}\cdots\mathsf{R}_{\mathcal{S}_{\sigma(s-1)}}^{\vec{y}_{\sigma(s-1)}\dagger}\right).\end{aligned} \tag{5.10}$$

Theorem 5.13 (Marginals imply Sequential Independence)**.** *If* $(\mathbf{W}_{\mathcal{N}}, \{\mathsf{Q}_{\mathcal{S}}\}_{\mathcal{S}\subseteq\mathcal{N}}, \{\mathcal{M}_i\}_{i\in\mathcal{N}})$ *has Properties 5.9, 5.10, and 5.11, then it also has Property 5.12.*

Proof. First note that Properties 5.9, 5.10, and 5.11 directly imply:

$$\mathrm{Tr}\left(\mathsf{R}_{\mathcal{S}}^{\vec{y}\dagger}\mathsf{Q}_{\mathcal{N}}^{\vec{x}}\mathsf{R}_{\mathcal{S}}^{\vec{y}}\rho\right) = \delta_{\vec{x}_{\mathcal{S}},\vec{y}}\mathrm{Tr}\left(\mathsf{Q}_{\mathcal{N}}^{\vec{x}}\rho\right), \tag{5.11}$$

where $\rho \in \mathcal{G}_{\mathcal{N}\setminus\mathcal{S}}(\mathcal{F})$ and all operators are POVM elements (and their square-root operators) from the second element of the tuple in question.

With the above identity at hand, we can directly prove the statement. Let us take any $\mathcal{T} \subseteq \mathcal{N}$ and its partition into sets $\mathcal{S}_1, \dots, \mathcal{S}_s$, like in Property 5.12. Then by Property 5.9 we have

$$\mathrm{Tr}\left(\mathsf{Q}_{\mathcal{S}_s}^{\vec{y}_s}\underbrace{\mathsf{R}_{\mathcal{S}_{s-1}}^{\vec{y}_{s-1}}\cdots\mathsf{R}_{\mathcal{S}_1}^{\vec{y}_1}\psi\mathsf{R}_{\mathcal{S}_1}^{\vec{y}_1\dagger}\cdots\mathsf{R}_{\mathcal{S}_{s-1}}^{\vec{y}_{s-1}\dagger}}_{:=\rho}\right) = \sum_{\vec{y}'}\mathrm{Tr}\left(\mathsf{Q}_{\mathcal{N}}^{\vec{y}'}\rho\right), \tag{5.12}$$

where $\vec{y}'$ ranges over $\prod_{i\in\mathcal{N}}\mathcal{X}_i$ such that $\vec{y}'_{\mathcal{S}_s} = \vec{y}_s$. By $\vec{y}'_{\mathcal{S}_s}$ we denote the elements of the vector corresponding to values in $\prod_{i\in\mathcal{S}_s}\mathcal{X}_i$. Let $\rho' := \mathsf{R}_{\mathcal{S}_{s-2}}^{\vec{y}_{s-2}}\cdots\mathsf{R}_{\mathcal{S}_1}^{\vec{y}_1}\psi\mathsf{R}_{\mathcal{S}_1}^{\vec{y}_1\dagger}\cdots\mathsf{R}_{\mathcal{S}_{s-2}}^{\vec{y}_{s-2}\dagger}$, then we can continue Equation (5.12):

$$\sum_{\vec{y}'}\mathrm{Tr}\left(\mathsf{Q}_{\mathcal{N}}^{\vec{y}'}\mathsf{R}_{\mathcal{S}_{s-1}}^{\vec{y}_{s-1}}\rho'\mathsf{R}_{\mathcal{S}_{s-1}}^{\vec{y}_{s-1}\dagger}\right) = \sum_{\vec{y}'}\delta_{\vec{y}'_{\mathcal{S}_{s-1}},\vec{y}_{s-1}}\mathrm{Tr}\left(\mathsf{Q}_{\mathcal{N}}^{\vec{y}'}\rho'\right) \tag{5.13}$$

$$= \cdots = \sum_{\vec{y}'}\delta_{\vec{y}'_{\mathcal{S}_{s-1}},\vec{y}_{s-1}}\cdots\delta_{\vec{y}'_{\mathcal{S}_1},\vec{y}_1}\mathrm{Tr}\left(\mathsf{Q}_{\mathcal{N}}^{\vec{y}'}\psi\right) = \mathrm{Tr}\left(\mathsf{Q}_{\mathcal{T}}^{\vec{y}''}\psi\right), \tag{5.14}$$

where in the first equality above we use Equation (5.11). In the sequence of equalities that follow we repeatedly use Equation (5.11) with square-root operators that appear in the definition of ρ'. In the last sum we summed over the remaining $\mathcal{N} \setminus \mathcal{T}$ variables and get the marginal for $\vec{y}'' := \bigcup_{i=1}^{s}\vec{y}_i$.

We see that in Equation (5.14) we prove that the left hand side of Equation (5.12) is the same for any order of measurements. Our derivation does not depend on the order of the operators, which ends our proof. □

5.2.2 Joint Distributions and Permutability

In this section, we state the definition of a joint distribution that describes a sequence of quantum measurements performed on states from a restricted set. We also define a generalization of commutativity and prove that a joint distribution exists if and only if the measurement operators are permutable.

Definition 5.14 (Joint Distribution On State). *A tuple $(\mathbf{W}_{\mathcal{N}}, \{\mathsf{Q}_{\mathcal{S}}\}_{\mathcal{S}\subseteq\mathcal{N}}, \{\mathcal{M}_i\}_{i\in\mathcal{N}})$ of a linear functional defined as in Equation* (5.1) (*with an element of the family from the second entry*)*, a family of POVMs, and a set of measurements is a* joint distribution *of N random variables $X_1, \ldots, X_N$ that describe outcomes of general quantum measurements $\{\mathcal{M}_i\}_{i\in\mathcal{N}}$* (*from the third element of the tuple*) *of states $\psi \in \mathcal{F}$ if*

(1) *Quantum Marginals on states in $\mathcal{F}$, Property 5.9, holds,*

(2) *Quantum Disjointness on states in $\mathcal{F}$, Property 5.10, holds,*

(3) *Quantum Reducibility on states in $\mathcal{F}$, Property 5.11, holds.*

Next we show the connection between the existence of joint distributions and requirements on the measurement operators. But first let us define the on-state permutability notion.

Definition 5.15 (On-state permutator, (fully) permutable operators). *For any s operators R_i and a permutation of the s-element set $\boldsymbol{\sigma} \in \mathcal{I}_{\text{per}}(\{1,\ldots,s\})$ the permutator on ψ is defined as*

$$\begin{aligned}&[\mathsf{R}_1, \mathsf{R}_2, \ldots, \mathsf{R}_s]_\psi(\boldsymbol{\sigma}) := \\ &\operatorname{Tr}\left(\mathsf{R}_s\mathsf{R}_{s-1}\cdots\mathsf{R}_1\psi\mathsf{R}_1^\dagger\mathsf{R}_2^\dagger\cdots\mathsf{R}_s^\dagger\right) - \operatorname{Tr}\left(\mathsf{R}_{\boldsymbol{\sigma}(s)}\mathsf{R}_{\boldsymbol{\sigma}(s-1)}\cdots\mathsf{R}_{\boldsymbol{\sigma}(1)}\psi\mathsf{R}_{\boldsymbol{\sigma}(1)}^\dagger\mathsf{R}_{\boldsymbol{\sigma}(2)}^\dagger\cdots\mathsf{R}_{\boldsymbol{\sigma}(s)}^\dagger\right).\end{aligned} \tag{5.15}$$

We say that the operators $\mathsf{R}_1, \ldots, \mathsf{R}_s$ are permutable *on ψ if $[\mathsf{R}_1, \ldots, \mathsf{R}_s]_\psi(\boldsymbol{\sigma}) = 0$ for all $\boldsymbol{\sigma} \in \mathcal{I}_{\text{per}}(\{1,\ldots,s\})$.*

*Moreover, we call a set of measurements $\{\mathcal{M}_i\}_{i\in\mathcal{N}}$ (where for every $i \in \mathcal{N}$ we have $\mathcal{M}_i := \{\mathsf{Q}_i^{x_i}\}_{x_i\in\mathcal{X}_i}$) t-*permutable on $\mathcal{F}$ *if the square-root operators $\mathsf{R}_i^{x_i}$ of any t of these measurements are permutable on $\psi \in \mathcal{F}$. If $t = N$ we call the measurements "fully permutable".*

To (slightly) strengthen our result connecting existence of joint distributions and permutability we define the notion of on-state projectors relevant for sequences of measurements.

Definition 5.16 (On-State Projectors). *$\{\mathsf{R}_i^{x_i}\}_{i\in\mathcal{N},x_i\in\mathcal{X}_i}$ is a set of on-state projectors on $\psi \in \mathcal{F}$ if for all $\mathcal{S} \subseteq \mathcal{N}$, all $\vec{x} \in \mathcal{X}_1 \times \cdots \times \mathcal{X}_N$, all $\vec{y} \in \prod_{i\in\mathcal{S}}\mathcal{X}_i$, and for $\mathsf{R}_{\mathcal{S}}^{\vec{y}} := \mathsf{R}_{\mathcal{S}(1)}^{y_1}\mathsf{R}_{\mathcal{S}(2)}^{y_2}\cdots\mathsf{R}_{\mathcal{S}(|\mathcal{S}|)}^{y_{|\mathcal{S}|}}$ and $\mathsf{Q}_{\mathcal{S}}^{\vec{y}} := \mathsf{R}_{\mathcal{S}}^{\vec{y}\dagger}\mathsf{R}_{\mathcal{S}}^{\vec{y}}$, we have*

$$\operatorname{Tr}\left(\mathsf{R}_{\mathcal{S}}^{\vec{y}\dagger}\mathsf{Q}_{\mathcal{N}}^{\vec{x}}\mathsf{R}_{\mathcal{S}}^{\vec{y}}\psi\right) = \delta_{\vec{y},\vec{x}_{\mathcal{S}}}\operatorname{Tr}\left(\mathsf{Q}_{\mathcal{N}}^{\vec{x}}\psi\right). \tag{5.16}$$

We would like to note that if operators in $\{\mathsf{R}_i^{x_i}\}_{i\in\mathcal{N},x_i\in\mathcal{X}_i}$ are on-state permutable and are orthogonal projectors, then they are also on-state projectors.

Now we state the theorem connecting existence of joint distributions with permutability of the measurement operators. This statement extends Theorem 3.2 of [ME84], where they prove that if the joint distribution satisfies the marginals property (they use the name "nondisturbance"), then the operators pairwise commute.

Theorem 5.17 (Joint Distribution for Quantum Experiments and Permutability)**.** *There is a quantum joint distribution on states $\psi \in \mathcal{F}$ for measurements $\{\mathcal{M}_i\}_{i\in\mathcal{N}}$ if and only if all square roots K_i^x of the E_i^x operators of the measurements $\mathcal{M}_i$ permute on $\psi \in \mathcal{F}$ and $\{\mathsf{K}_i^{x_i}\}_{i\in\mathcal{S},x_i\in\mathcal{X}_i}$ is a set of on-state projectors according to Definition 5.16.*

Proof. ($\Rightarrow$) In Theorem 5.13 we have proven that marginals (Property 5.9) together with disjointness (Property 5.10) and reducibility (Property 5.11) imply sequential independence (Property 5.12). Permutability follows from Property 5.12 for N single-element sets $\mathcal{S}_i = \{i\}$. To show that $\{\mathsf{K}_i^{x_i}\}_{i\in\mathcal{S},x_i\in\mathcal{X}_i}$ is a set of on-state projectors we consider any $\mathsf{R}_{\mathcal{S}}^{\vec{y}} = \mathsf{R}_{\mathcal{S}(1)}^{y_1}\cdots\mathsf{R}_{\mathcal{S}(|\mathcal{S}|)}^{y_{|\mathcal{S}|}}$ from the statement of Definition 5.16 and $\rho = \mathsf{R}_{\mathcal{S}(2)}^{y_2}\cdots\mathsf{R}_{\mathcal{S}(|\mathcal{S}|)}^{y_{|\mathcal{S}|}}\psi\mathsf{R}_{\mathcal{S}(|\mathcal{S}|)}^{y_{|\mathcal{S}|}\dagger}\cdots\mathsf{R}_{\mathcal{S}(2)}^{y_2\dagger}$. First note that $\rho \in \mathcal{G}_{\mathcal{N}\setminus\mathcal{S}(1)}$. By applying Properties 5.10 and 5.11 we get

$$\mathrm{Tr}\left(\mathsf{Q}_{\mathcal{N}}^{\vec{x}}\mathsf{R}_{\mathcal{S}(1)}^{y_1}\rho\mathsf{R}_{\mathcal{S}(1)}^{y_1\dagger}\right) = \delta_{\vec{y}_1,\vec{x}_{\mathcal{S}(1)}}\mathrm{Tr}\left(\mathsf{Q}_{\mathcal{N}}^{\vec{x}}\rho\right). \tag{5.17}$$

We can repeat the above reasoning—stripping ρ of $\mathsf{R}_{\mathcal{S}(i)}^{y_i}$ operators one by one—to show the statement of Definition 5.16.

($\Leftarrow$) The other direction of the proof follows by setting the measurement operators to $\mathsf{Q}_{\mathcal{S}}^{\vec{y}} := \mathsf{K}_{\mathcal{S}(t)}^{y_t\dagger}\mathsf{K}_{\mathcal{S}(t-1)}^{y_{t-1}\dagger}\cdots\mathsf{K}_{\mathcal{S}(1)}^{y_1\dagger}\mathsf{K}_{\mathcal{S}(1)}^{y_1}\mathsf{K}_{\mathcal{S}(2)}^{y_2}\cdots\mathsf{K}_{\mathcal{S}(t)}^{y_t}$, for every set $\mathcal{S}\subseteq\mathcal{N}$ with $|\mathcal{S}| = t$. Similarly we define $\mathsf{R}_{\mathcal{S}}^{\vec{y}} := \mathsf{K}_{\mathcal{S}(1)}^{y_1}\mathsf{K}_{\mathcal{S}(2)}^{y_2}\cdots\mathsf{K}_{\mathcal{S}(t)}^{y_t}$.

The marginals property 5.9 reads as follows: $\sum_{i\in\mathcal{N}\setminus\mathcal{S}}\sum_{x_i\in\mathcal{X}_i}\mathrm{Tr}\left(\mathsf{Q}_{\mathcal{N}}^{\vec{x}}\rho\right) = \mathrm{Tr}\left(\mathsf{Q}_{\mathcal{S}}^{\vec{x}_{\mathcal{S}}}\rho\right)$. By definition of the square-root operators we have that $\mathrm{Tr}\left(\mathsf{Q}_{\mathcal{N}}^{\vec{x}}\rho\right) = \mathrm{Tr}\left(\mathsf{R}_{\mathcal{N}}^{\vec{x}}\rho\mathsf{R}_{\mathcal{N}}^{\vec{x}\dagger}\right)$. Thanks to how we defined the POVM elements and the assumed permutability we know that for any i

$$\mathrm{Tr}\left(\mathsf{R}_{\mathcal{N}}^{\vec{x}}\rho\mathsf{R}_{\mathcal{N}}^{\vec{x}\dagger}\right) = \mathrm{Tr}\left(\mathsf{R}_i^{x_i\dagger}\mathsf{R}_i^{x_i}\mathsf{R}_{\mathcal{N}\setminus\{i\}}^{\vec{x}_{\mathcal{N}\setminus\{i\}}}\rho\mathsf{R}_{\mathcal{N}\setminus\{i\}}^{\vec{x}_{\mathcal{N}\setminus\{i\}}\dagger}\right), \tag{5.18}$$

where x_i is the i-th element of $\vec{x}$. From the fact that $\sum_{x_i\in\mathcal{X}_i}\mathsf{E}_i^{x_i} = \mathbb{1}$ and the above transformation we get

$$\sum_{i\in\mathcal{N}\setminus\mathcal{S}}\sum_{x_i\in\mathcal{X}_i}\mathrm{Tr}\left(\mathsf{Q}_{\mathcal{N}}^{\vec{x}}\rho\right) = \sum_{i\in\mathcal{N}\setminus\mathcal{S}}\sum_{x_i\in\mathcal{X}_i}\mathrm{Tr}\left(\mathsf{E}_{i_1}^{x_{i_1}}\mathsf{R}_{\mathcal{N}\setminus\{i_1\}}^{\vec{x}_{\mathcal{N}\setminus\{i_1\}}}\rho\mathsf{R}_{\mathcal{N}\setminus\{i_1\}}^{\vec{x}_{\mathcal{N}\setminus\{i_1\}}\dagger}\right)$$

$$= \sum_{i \in \mathcal{N} \setminus (\mathcal{S} \cup \{i_1\})} \sum_{x_i \in \mathcal{X}_i} \mathrm{Tr}\left(\mathsf{E}_{i_2}^{x_{i_2}} \mathsf{R}_{\mathcal{N}\setminus\{i_1,i_2\}}^{\vec{x}_{\mathcal{N}\setminus\{i_1,i_2\}}} \rho \mathsf{R}_{\mathcal{N}\setminus\{i_1,i_2\}}^{\vec{x}_{\mathcal{N}\setminus\{i_1,i_2\}}\dagger} \right) = \cdots = \mathrm{Tr}\left(\mathsf{Q}_{\mathcal{S}}^{\vec{x}_{\mathcal{S}}} \rho \right). \quad (5.19)$$

Where $i_1, i_2, \ldots i_{|\mathcal{N}\setminus\mathcal{S}|}$ are indices of output sets we sum over, the order is arbitrary.

For the joint distribution to exist, Properties 5.10 and 5.11 also need to be satisfied. First we note that for square-root operators of the form $\mathsf{R}_{\mathcal{S}}^{\vec{y}} = \mathsf{K}_{\mathcal{S}(1)}^{y_1} \cdots \mathsf{K}_{\mathcal{S}(t)}^{y_t}$ we can simplify $\mathrm{Tr}\left(\mathsf{Q}_{\mathcal{N}}^{\vec{x}} \mathsf{R}_{\mathcal{S}}^{\vec{y}} \rho \mathsf{R}_{\mathcal{S}}^{\vec{y}\dagger} \right)$ appearing in both properties as follows: Let us consider any $\rho \in \mathcal{G}_{\mathcal{N}\setminus\mathcal{S}}(\mathcal{F})$, then by definition of set $\mathcal{G}_{\mathcal{N}\setminus\mathcal{S}}(\mathcal{F})$ it equals $\mathsf{R}_{\mathcal{S}'}^{\vec{y}'} \psi \mathsf{R}_{\mathcal{S}'}^{\vec{y}'\dagger}$. Additionally considering $\mathsf{R}_{\mathcal{S}}^{\vec{y}}$ and noting our assumption of permutability we get $\mathrm{Tr}\left(\mathsf{Q}_{\mathcal{N}}^{\vec{x}} \mathsf{R}_{\mathcal{S}}^{\vec{y}} \rho \mathsf{R}_{\mathcal{S}}^{\vec{y}\dagger} \right) = \mathrm{Tr}\left(\mathsf{Q}_{\mathcal{N}}^{\vec{x}} \mathsf{R}_{\mathcal{S}\cup\mathcal{S}'}^{\vec{y}\cup\vec{y}'} \psi \mathsf{R}_{\mathcal{S}\cup\mathcal{S}'}^{\vec{y}\cup\vec{y}'\dagger} \right)$. Similarly we have $\mathrm{Tr}\left(\mathsf{Q}_{\mathcal{N}}^{\vec{x}} \rho \right) = \mathrm{Tr}\left(\mathsf{Q}_{\mathcal{N}}^{\vec{x}} \mathsf{R}_{\mathcal{S}'}^{\vec{y}'} \psi \mathsf{R}_{\mathcal{S}'}^{\vec{y}'\dagger} \right)$. To prove that Properties 5.10 and 5.11 hold we apply Definition 5.16 to both traces:

$$\mathrm{Tr}\left(\mathsf{Q}_{\mathcal{N}}^{\vec{x}} \mathsf{R}_{\mathcal{S}}^{\vec{y}} \rho \mathsf{R}_{\mathcal{S}}^{\vec{y}\dagger} \right) = \mathrm{Tr}\left(\mathsf{Q}_{\mathcal{N}}^{\vec{x}} \mathsf{R}_{\mathcal{S}\cup\mathcal{S}'}^{\vec{y}\cup\vec{y}'} \psi \mathsf{R}_{\mathcal{S}\cup\mathcal{S}'}^{\vec{y}\cup\vec{y}'\dagger} \right) = \delta_{\vec{y}\cup\vec{y}', \vec{x}_{\mathcal{S}\cup\mathcal{S}'}} \mathrm{Tr}\left(\mathsf{Q}_{\mathcal{N}}^{\vec{x}} \psi \right) \quad (5.20)$$

and

$$\mathrm{Tr}\left(\mathsf{Q}_{\mathcal{N}}^{\vec{x}} \rho \right) = \mathrm{Tr}\left(\mathsf{Q}_{\mathcal{N}}^{\vec{x}} \mathsf{R}_{\mathcal{S}'}^{\vec{y}'} \psi \mathsf{R}_{\mathcal{S}'}^{\vec{y}'\dagger} \right) = \delta_{\vec{y}', \vec{x}_{\mathcal{S}'}} \mathrm{Tr}\left(\mathsf{Q}_{\mathcal{N}}^{\vec{x}} \psi \right). \quad (5.21)$$

Putting these observations together: if $\vec{y}$ is different from $\vec{x}_{\mathcal{S}}$, then Equation (5.20) yields 0 (as required by Property 5.10), else $\delta_{\vec{y}\cup\vec{y}', \vec{x}_{\mathcal{S}\cup\mathcal{S}'}} = \delta_{\vec{y}', \vec{x}_{\mathcal{S}'}}$ and Equation (5.20) equals to Equation (5.21) (as required by Property 5.11). □

5.2.3 Pairwise On-State Commutation Does Not Imply Full Commutation

We now investigate whether full permutability is the weakest assumption we can have for joint distributions to exist.

When we consider this question for the full Hilbert space $\mathcal{F} = \mathcal{D}(\mathcal{H})$, this problem has been considered by a number of works in the literature [Nel67; Fin73; Fin82; ME84] and it is well-known that it suffices for the measurement operators to pairwise commute, i.e., pairwise commutation on all possible quantum states implies permutability of the operators.

Our goal in this section is to consider the case where $\mathcal{F} \subsetneq \mathcal{D}(\mathcal{H})$. In particular, in [Car+18], in order to connect perfect quantum indifferentiability to classical indifferentiability with stateless simulators, they rely on the following conjecture. Note that the t in the conjecture is fixed, so we actually have a family of conjectures for different t and N.

Conjecture 5.18 (Conjecture 2 from [Car+18], rephrased). *Consider N binary measurements described by projectors $\mathsf{P}_1, \ldots, \mathsf{P}_N$, and a quantum state $|\Psi\rangle$.*

Assume that any t out of the N measurements permute on state $|\Psi\rangle$. That is, for any $\mathcal{I}$ with $|\mathcal{I}| = t$, if $\mathsf{P}'_1, \dots, \mathsf{P}'_t$ and $\mathsf{P}''_1, \dots, \mathsf{P}''_t$ are the projectors $\{\mathsf{P}_i\}_{i\in\mathcal{I}}$ (possibly in different order), then $\mathsf{P}'_t, \dots, \mathsf{P}'_1|\Psi\rangle = \mathsf{P}''_t, \dots, \mathsf{P}''_1|\Psi\rangle$.

Then there exist random variables $X_1, \dots, X_N$ with a joint distribution $\mathfrak{D}$ such that for any $\mathcal{I} = \{i_1, \dots, i_t\}$ the joint distribution of $X_{i_1}, \dots, X_{i_t}$ is the distribution of the outcomes when we measure $|\Psi\rangle$ with measurements $\mathsf{P}_{i_1}, \dots, \mathsf{P}_{i_t}$.

In the use of Conjecture 5.18 in [Car+18], they implicitly assume that we can replace P_i by its complement $\mathbb{1} - \mathsf{P}_i$ and the permutation still holds.

Conjecture 5.18 states that if any t measurement operators permute on state $|\Psi\rangle$ then there is a joint distribution. From Theorem 5.17, we know that if there is a joint distribution then the operators fully permute. Hence, the key point of the conjecture is that if any t operators permute on a state, then they fully permute on it. However, we show here that Conjecture 5.18 is not true, in general.

Theorem 5.19. *There exists a set of four projectors $\{\mathsf{P}_1, \mathsf{P}_2, \mathsf{P}_3, \mathsf{P}_4\}$ and a state $|\phi\rangle \in \mathbb{C}^8$ such that the projectors are 2-permutable (they pairwise commute) on state $|\phi\rangle$ and they are* not *4-permutable on $|\phi\rangle$.*

Proof. To prove the statement we found an example of such operators and a state numerically by random constrained search. We consider 4 projectors P_i and a state $|\phi\rangle$ of dimension 8. The constraints we impose are

$$\forall i, j \neq i \; [\mathsf{P}_i, \mathsf{P}_j]|\phi\rangle = 0, \tag{5.22}$$

moreover operators P_i are projectors: $\forall i \mathsf{P}_i^2 = \mathsf{P}_i$ and $|\phi\rangle$ is a unit-norm complex vector.

We look for an example to the statement that 2-permutability (commutativity) does not imply 4-permutability, so that

$$(\mathsf{P}_1\mathsf{P}_2\mathsf{P}_3\mathsf{P}_4 - \mathsf{P}_3\mathsf{P}_4\mathsf{P}_1\mathsf{P}_2)|\phi\rangle \neq 0. \tag{5.23}$$

To find such example we used software for symbolic computing to define the problem and maximize $\|(\mathsf{P}_1\mathsf{P}_2\mathsf{P}_3\mathsf{P}_4 - \mathsf{P}_3\mathsf{P}_4\mathsf{P}_1\mathsf{P}_2)|\phi\rangle\|$, where we maximize over operators P_i and the state $|\phi\rangle$.

The result of our optimization can be found in Section 5.2.4. Note that we consider vector equalities—instead of just traces like in Theorem 5.17—hence our example provides a slightly stronger argument for the necessity of full permutability. □

One can notice by looking at the optimization problem that it is not a semidefinite problem, nor are we aware of any other structure that is easy to exploit. For that reason finding larger instances is probably computationally very expensive.

We notice that Theorem 5.19 actually disproves a slightly stronger version of Conjecture 5.18; One that includes the complements of projectors. While this modification gives a slightly stronger assumption, our counterexample in Theorem 5.19 works just as well.

For the joint distribution in Conjecture 5.18 to exist, we know from Theorem 5.17 that all operators must on-state permute. An important observation is that Conjecture 5.18 considers vector equalities and Theorem 5.17 considers scalar equalities. The theorem is "easier" than the conjecture and our counterexample works with vector equalities, hence we disprove a stronger statement and indeed Theorem 5.19 disproves Conjecture 5.18.

In [Ebr18] the author uses a different conjecture and different reasoning to prove the existence of the joint distribution. Our counterexample does not disprove this other approach and we refer interested readers to [Ebr18].

5.2.4 Numerical values

Below we present the state and the projectors that are claimed in the proof of Theorem 5.19. The script used to generate these values can be found in [Cza21]. Before we write out the state and the projections that we found, let us state our violation of the permutator:

$$\|(\mathsf{P}_1\mathsf{P}_2\mathsf{P}_3\mathsf{P}_4 - \mathsf{P}_3\mathsf{P}_4\mathsf{P}_1\mathsf{P}_2)|\phi\rangle\| = 0.25 \pm 3\cdot 10^{-8}. \tag{5.24}$$

All the constraints listed in the proof of Theorem 5.19 are fulfilled up to the seventh decimal digit of precision, so up to 10^{-7}. Internal computations of the algorithm are performed with machine precision of 10^{-15}.

The state is

$$|\phi\rangle := \begin{pmatrix} -0.135381 - 0.0503468\mathrm{i} \\ 0.325588 - 0.222403\mathrm{i} \\ -0.209447 - 0.0404665\mathrm{i} \\ -0.418336 + 0.130098\mathrm{i} \\ -0.503693 - 0.299414\mathrm{i} \\ 0.379842 + 0.205081\mathrm{i} \\ -0.179291 - 0.0381456\mathrm{i} \\ 0.0840381 - 0.125995\mathrm{i} \end{pmatrix}. \tag{5.25}$$

We define the projectors by their eigenvectors:

$$\mathsf{P}_1 = |\pi^1\rangle\langle\pi^1|, \qquad \mathsf{P}_2 = |\pi^2_1\rangle\langle\pi^2_1| + |\pi^2_2\rangle\langle\pi^2_2|, \tag{5.26}$$

$$\mathsf{P}_3 = |\pi^3_1\rangle\langle\pi^3_1| + |\pi^3_2\rangle\langle\pi^3_2| + |\pi^3_3\rangle\langle\pi^3_3|, \qquad \mathsf{P}_4 = |\pi^4_1\rangle\langle\pi^4_1| + |\pi^4_2\rangle\langle\pi^4_2|. \tag{5.27}$$

The eigenvector of P_1 is:

$$|\pi^1\rangle := \begin{pmatrix} 0.440777 + 0.168408i \\ 0.208781 - 0.37351i \\ 0.247514 + 0.0276065i \\ -0.297971 + 0.0252308i \\ 0.118798 + 0.112225i \\ -0.293428 + 0.270889i \\ -0.193073 + 0.218869i \\ -0.41405 \end{pmatrix}. \tag{5.28}$$

The eigenvectors of P_2 are:

$$|\pi^2_1\rangle := \begin{pmatrix} -0.497016 - 0.094035i \\ 0.417527 - 0.0737062i \\ -0.000125303 + 0.35123i \\ 0.166569 - 0.187245i \\ -0.373202 + 0.205633i \\ 0.318452 - 0.251475i \\ -0.107473 - 0.123987i \\ -0.0711523 \end{pmatrix}, |\pi^2_2\rangle := \begin{pmatrix} 0.365906 + 0.0620997i \\ 0.418728 - 0.2059i \\ 0.229457 + 0.0557421i \\ -0.140393 + 0.0945029i \\ -0.199205 - 0.188139i \\ 0.103617 + 0.279644i \\ -0.546498 + 0.147197i \\ 0.275295 \end{pmatrix}. \tag{5.29}$$

The eigenvectors of P_3 are:

$$|\pi^3_1\rangle := \begin{pmatrix} -0.453059 + 0.181543i \\ -0.452841 + 0.0154095i \\ -0.17948 - 0.222827i \\ -0.230355 - 0.0526756i \\ -0.0918752 - 0.250754i \\ 0.242416 - 0.126917i \\ 0.300832 - 0.287566i \\ 0.315259 \end{pmatrix}, |\pi^3_2\rangle := \begin{pmatrix} -0.0586669 - 0.269559i \\ -0.280155 + 0.373271i \\ -0.150758 - 0.158539i \\ 0.158793 - 0.0454731i \\ 0.165888 + 0.362832i \\ -0.110453 - 0.310755i \\ 0.353894 - 0.00811586i \\ -0.487537 \end{pmatrix}, \tag{5.30}$$

$$|\pi^3_3\rangle := \begin{pmatrix} -0.182739 - 0.114718i \\ 0.246775 - 0.134678i \\ -0.513357 - 0.193655i \\ -0.10451 + 0.421294i \\ 0.111183 + 0.122625i \\ -0.200917 - 0.25897i \\ -0.0290851 + 0.398494i \\ 0.30081 \end{pmatrix}. \tag{5.31}$$

The eigenvectors of P_4 are:

$$|\pi_1^4\rangle := \begin{pmatrix} -0.464187 + 0.213035\mathrm{i} \\ -0.364421 + 0.119836\mathrm{i} \\ -0.324984 - 0.23097\mathrm{i} \\ -0.256841 + 0.0478513\mathrm{i} \\ -0.0700499 - 0.192822\mathrm{i} \\ 0.146148 - 0.225755\mathrm{i} \\ 0.243944 - 0.284786\mathrm{i} \\ 0.331272 \end{pmatrix}, |\pi_2^4\rangle := \begin{pmatrix} 0.111757 + 0.151275\mathrm{i} \\ 0.236223 - 0.323279\mathrm{i} \\ 0.157312 - 0.115385\mathrm{i} \\ -0.30864 + 0.0990552\mathrm{i} \\ -0.260931 - 0.236239\mathrm{i} \\ 0.240497 + 0.13559\mathrm{i} \\ -0.453404 + 0.12357\mathrm{i} \\ 0.490125 \end{pmatrix}. \tag{5.32}$$

5.3 Almost On-State Commutation

In the last part of the chapter we discuss almost commutativity in the on-state case. In particular, we show here that if we have two projectors that almost commute on a state then we can define a projector that fully commutes with one of the original operators and is close on state to the second one.

The main tool that we need to prove this result is Jordan's lemma.

Lemma 5.20 (Jordan's lemma [Jor75]). *Let P_1 and P_2 be two projectors with rank $r_i := \text{rank}(\mathsf{P}_i)$ for $i \in \{1,2\}$. Then both projectors can be decomposed simultaneously in the form $\mathsf{P}_i = \bigoplus_{k=1}^{r_i} \mathsf{P}_i^k$, where P_i^k denote rank-1 projectors acting on one- or two-dimensional subspaces. We denote the one- and two-dimensional subspaces by $S_1, \ldots, S_l$ and subspaces by $T_1, \ldots, T_{l'}$, respectively. The eigenvectors $|v_{k,1}\rangle$ and $|v_{k,2}\rangle$ of P_1^k and P_2^k respectively are related by:*

$$|v_{k,2}\rangle = \cos\theta_k |v_{k,1}\rangle + \sin\theta_k |v_{k,1}^\perp\rangle, |v_{k,1}\rangle = \cos\theta_k |v_{k,2}\rangle - \sin\theta_k |v_{k,2}^\perp\rangle. \tag{5.33}$$

We can now prove our result.

Theorem 5.21 (Making almost commuting projectors commute). *Given any two projectors P_1 and P_2 and a state $|\psi\rangle$ we have that if $\|(\mathsf{P}_1\mathsf{P}_2 - \mathsf{P}_2\mathsf{P}_1)|\psi\rangle\| = \epsilon$ then there is a projector P_2' that is close to the original projector on the state $\|(\mathsf{P}_2' - \mathsf{P}_2)|\psi\rangle\| \leq \sqrt{2}\epsilon$ and $[\mathsf{P}_1, \mathsf{P}_2'] = 0$.*

Proof. By Jordan's lemma (Lemma 5.20), there exist bits $\lambda_{i,1}, \lambda_{i,2} \in \{0,1\}$ and vectors $|u_1\rangle, ..., |u_m\rangle$ and $|v_{1,1}\rangle, |v_{1,2}\rangle, ..., |v_{\ell,1}\rangle, |v_{\ell,2}\rangle$, such that

1. $\mathsf{P}_1 = \sum_{i\in[m]} \lambda_{i,1} |u_i\rangle\langle u_i| + \sum_{i\in[\ell]} |v_{i,1}\rangle\langle v_{i,1}|$
   and $\mathsf{P}_2 = \sum_{i\in[m]} \lambda_{i,2} |u_i\rangle\langle u_i| + \sum_{i\in[\ell]} |v_{i,2}\rangle\langle v_{i,2}|$;

2. $\langle u_i|u_k\rangle = 0$ and $\langle u_i|v_{j,b}\rangle = 0$ for all b, i, j and $k \neq i$;

3. $\langle v_{j,b'}|v_{i,b}\rangle = 0$ for $i \neq j$ and any b, b';

4. $0 < \langle v_{i,1}|v_{i,2}\rangle < 1$.

Let θ_i be the angle between $|v_{i,1}\rangle$ and $|v_{i,2}\rangle$ (i.e. $\cos\theta_i = \langle v_{i,1}|v_{i,2}\rangle$), and $|v_{i,1}^\perp\rangle$ be the state orthogonal to $|v_{i,1}\rangle$ in the subspace spanned by these two vectors. Since the non-commuting part of P_1 and P_2 must come from the pairs $|v_{i,1}\rangle, |v_{i,2}\rangle$, we will define P_2' by removing the non-commuting part of P_2, shifting the vector $|v_{i,2}\rangle$, to either $|v_{i,1}\rangle$ or $|v_{i,1}^\perp\rangle$:

$$\mathsf{P}_2' = \sum_{i\in[m]} \lambda_{i,2}|u_i\rangle\langle u_i| + \sum_{i\in[\ell]:\theta_i\le\frac{\pi}{4}} |v_{i,1}\rangle\langle v_{i,1}| + \sum_{i\in[\ell]:\theta_i>\frac{\pi}{4}} |v_{i,1}^\perp\rangle\langle v_{i,1}^\perp|. \tag{5.34}$$

We have clearly that $[\mathsf{P}_1, \mathsf{P}_2'] = 0$ since the two projectors are simultaneously diagonalizable and we now want to prove that

$$\|(\mathsf{P}_2' - \mathsf{P}_2)|\psi\rangle\| \le \sqrt{2}\,\varepsilon\,. \tag{5.35}$$

Notice that

$$\begin{aligned}
&\|(\mathsf{P}_2' - \mathsf{P}_2)|\psi\rangle\|^2 \\
&= \Bigg\| \sum_{i\in[l']:\theta_i\le\frac{\pi}{4}} (|v_{i,1}\rangle\langle v_{i,1}|\psi\rangle - |v_{i,2}\rangle\langle v_{i,2}|\psi\rangle) \\
&+ \sum_{i\in[l']:\theta_i>\frac{\pi}{4}} \Big(|v_{i,1}^\perp\rangle\langle v_{i,1}^\perp|\psi\rangle - |v_{i,2}\rangle\langle v_{i,2}|\psi\rangle\Big)\Bigg\|^2 \qquad (5.36)\\
&= \sum_{i\in[l']:\theta_i\le\frac{\pi}{4}} \||v_{i,1}\rangle\langle v_{i,1}|\psi\rangle - |v_{i,2}\rangle\langle v_{i,2}|\psi\rangle\|^2 \\
&+ \sum_{i\in[l']:\theta_i>\frac{\pi}{4}} \Big\||v_{i,1}^\perp\rangle\langle v_{i,1}^\perp|\psi\rangle - |v_{i,2}\rangle\langle v_{i,2}|\psi\rangle\Big\|^2, \qquad (5.37)
\end{aligned}$$

where in the last step we used that $\langle v_{i,b'}|v_{j,b}\rangle = 0$ for $i \ne j$.

Using that $|v_{i,2}\rangle = \cos\theta_i|v_{i,1}\rangle + \sin\theta_i|v_{i,1}^\perp\rangle$, we have that if $\theta_i \le \frac{\pi}{4}$, then

$$\begin{aligned}
&\|\langle v_{i,1}|\psi\rangle|v_{i,1}\rangle - \langle v_{i,2}|\psi\rangle|v_{i,2}\rangle\|^2 \\
&= \sin^4\theta_i|\langle v_{i,1}|\psi\rangle|^2 - 2\sin^3\theta_i\cos\theta_i\mathfrak{Re}(\langle v_{i,1}^\perp|\psi\rangle\langle v_{i,1}|\psi\rangle) + \sin^2\theta_i\cos^2\theta_i|\langle v_{i,1}^\perp|\psi\rangle|^2 \\
&+ \sin^4\theta_i|\langle v_{i,1}^\perp|\psi\rangle|^2 + 2\sin^3\theta_i\cos\theta_i\mathfrak{Re}(\langle v_{i,1}|\psi\rangle\langle v_{i,1}^\perp|\psi\rangle) + \sin^2\theta_i\cos^2\theta_i|\langle v_{i,1}|\psi\rangle|^2 \\
&\le 2\sin^2\theta_i\cos^2\theta_i(|\langle v_{i,1}^\perp|\psi\rangle|^2 + |\langle v_{i,1}|\psi\rangle|^2), \qquad (5.38)
\end{aligned}$$

where in the inequality we used our assumption that $\theta_i \le \frac{\pi}{4}$ which implies that $\sin\theta_i \le \cos\theta_i$.

Using similar calculations, we have that if $\theta_i \ge \frac{\pi}{4}$

$$\Big\|\langle v_{i,1}^\perp|\psi\rangle|v_{i,1}^\perp\rangle - \langle v_{i,2}|\psi\rangle|v_{i,2}\rangle\Big\|^2 \le 2\sin^2\theta_i\cos^2\theta_i(|\langle v_{i,1}^\perp|\psi\rangle|^2 + |\langle v_{i,1}|\psi\rangle|^2). \tag{5.39}$$

We will show now that

$$\sum_i \sin^2\theta_i \cos^2\theta_i (|\langle v_{i,1}^{\perp}|\psi\rangle|^2 + |\langle v_{i,1}|\psi\rangle|^2) = \varepsilon^2,$$

which finishes the proof:

$$\varepsilon^2 = \|(\mathsf{P}_2\mathsf{P}_1 - \mathsf{P}_1\mathsf{P}_2)|\psi\rangle\|^2 \tag{5.40}$$

$$= \left\| \sum_{i\in[l']} |v_{i,1}\rangle\langle v_{i,1}|v_{i,2}\rangle\langle v_{i,2}|\psi\rangle - |v_{i,2}\rangle\langle v_{i,2}|v_{i,1}\rangle\langle v_{i,1}|\psi\rangle. \right\|^2 \tag{5.41}$$

$$= \left\| \sum_{i\in[l']} \cos\theta_i \left(\cos\theta_i\langle v_{i,1}|\psi\rangle + \sin\theta_i\langle v_{i,1}^{\perp}|\psi\rangle\right)|v_{i,1}\rangle \right.$$

$$\left. - \sum_{i\in[l']} \cos\theta_i\langle v_{i,1}|\psi\rangle \left(\cos\theta_i|v_{i,1}\rangle + \sin\theta_i|v_{i,1}^{\perp}\rangle\right) \right\|^2 \tag{5.42}$$

$$= \left\| \sum_{i\in[l']} \sin\theta_i\cos\theta_i \left(\langle v_{i,1}^{\perp}|\psi\rangle|v_{i,1}\rangle - \langle v_{i,1}|\psi\rangle|v_{i,1}^{\perp}\rangle\right) \right\|^2 \tag{5.43}$$

$$= \sum_{i\in[l']} \sin^2\theta_i\cos^2\theta_i \left(|\langle v_{i,1}^{\perp}|\psi\rangle|^2 + |\langle v_{i,1}|\psi\rangle|^2\right). \tag{5.44}$$

where in the second equality we again use that $|v_{i,2}\rangle = \cos\theta_i|v_{i,1}\rangle + \sin\theta_i|v_{i,1}^{\perp}\rangle$ and in the fourth equality we use the fact that $\langle v_{i,b'}|v_{j,b}\rangle = 0$ for $i \neq j$. □

Our proof relies solely on Jordan's Lemma. Note that Jordan's Lemma is sufficient only if we analyze commutation of projectors. Results that show how to make any Hermitian matrices commute [FR96; Has09] are much more complicated to prove and it is not clear how to translate them to the "on-state" case.

We stress that our proof only works for two projectors, since Jordan's lemma does not generalize for three or more projectors. Therefore, we leave as open problem (dis)proving a generalized version of Lemma 5.21 for more projectors.

In [Ebr18] the author proves Theorem 5.21 for $\varepsilon = 0$, but with a different proof. They use Halmo's two projections theorem instead of Jordan's lemma.

CHAPTER

6

QUANTUM GAME-PLAYING FRAMEWORK

Chapter contents

Game-playing proofs constitute a powerful framework for non-quantum cryptographic security arguments, notably applied in the context of indifferentiability. A common ingredient in such proofs is lazy sampling of random primitives. In [Cza+19] we developed a quantum game-playing proof framework by generalizing two recent proof techniques: Zhandry's compressed quantum oracles [Zha19a], that can be used to quantum lazy-sample a class of non-uniform function distributions. The second technique is Unruh's one-way-to-hiding lemma [Unr14], that can also be applied to compressed oracles, providing a quantum counterpart to the fundamental lemma of game-playing.

6.1 Introduction

The game-playing framework, described in detail in Section 2.4.2, provides proofs of security with a clear and easy-to-verify structure. Combined with the need of including fully quantum adversaries with quantum access to random oracles into the threat model (see Section 2.1.5), we encounter an obvious challenge: defining a *quantum* game-playing framework. In this chapter, we resolve that challenge, so that in later chapters we can apply the resulting framework to prove security of a number of cryptographic constructions. In the following paragraphs we describe our results and the main proof techniques we used to achieve them.

We devise a quantum game-playing framework for security proofs that involve fully-quantum[1] adversaries. Our framework is based on a combination of two recently developed proof techniques: compressed quantum random oracles by Zhandry [Zha19a] and the One-Way to Hiding (O2H) lemma by Unruh [Unr14; AHU19]. The former provides a way to lazy-sample a quantum-accessible random oracle, and the latter is a quantum counterpart of the Fundamental Game-Playing lemma—a key ingredient in the original game-playing framework. As our first main result we obtain a clean and powerful tool for proofs in post-quantum cryptography. The main advantage of the framework is the fact that it allows the translation of certain classical security proofs to the quantum setting, in a way that is arguably more straight-forward than for previously available proof techniques.

On the technical side, we begin by re-formalizing Zhandry's compressed-oracle technique, which, as a by-product, makes a modest generalization to some non-uniform distributions of oracles relatively straightforward. In particular, we generalize the compressed-oracle technique of [Zha19a] to a class of non-uniform distributions over functions, allowing a more general form of (quantum) lazy sampling. Our result allows to treat distributions with outputs that are independent for distinct inputs. Subsequently, we observe that the

[1]Meaning adversaries that have access to a quantum computer and make quantum queries to the primitives.

techniques of "puncturing oracles" proposed in [AHU19] can also be applied to compressed oracles, yielding a more general version of the O2H lemma which forms the quantum counterpart of the fundamental game-playing lemma.

There are already some examples in the literature where generalized compressed oracle for non-uniform distributions has been used, e.g. [Ala+18] (superposition oracle without compression that outputs 1 with probability λ, we define the sampling procedure for such distribution in Section 6.2.2.2) and [HM20] (a generalization similar to ours but presented after [Cza+19] was posted online). We believe that the generalized formalism developed here will continue to be useful.

Punctured oracles are quantum oracles measured after every adversarial query. An important lemma that we prove is a bound on the probability that any of these measurements succeeds. Our proof of this bound is general enough so that it can be applied with little changes to many different scenarios. The bound on the probability of any of the measurements in a punctured oracle succeeding, together with the O2H lemma for compressed oracles provides a bound on the distinguishing advantage between a regular compressed oracle and a punctured one. In Lemma 9 in [Zha19a], the indistinguishability of a compressed oracle and a punctured compressed oracle is also proven. The method, however, is different than ours and less details are provided. A crucial difference is that there are two technical claims left implicit. According to [Zha20] there exists a proof that maintains the claimed bound.

Related Work Recent work by Unruh and by Ambainis, Hamburg, and Unruh [Unr14; AHU19] form the second main ingredient of this result. Their O2H lemma provides a way to "reprogram" quantum accessible oracles on some set of inputs, formalized as "punctured" oracles in the latter paper.

In a recent article [Unr19] Unruh developed quantum Relational Hoare Logic for computer verification of proofs in (post-)quantum cryptography. There he also uses the approach of game-playing, but in general focuses on formal definitions of quantum programs and predicates. To investigate the relation between [Unr19] and this chapter in more detail one would have to express our results in the language of the new logic. We leave it as an interesting direction for the future.

The proof techniques of [Zha19a] and [AHU19] have been recently used to show quantum security of the 4-Round Feistel construction in [HI19] and of generic key-encapsulation mechanisms in [JZM19] respectively. In [CEV20] the authors use compressed oracles for randomness in an encryption scheme using a random tweakable permutation (that is given to the algorithm externally). In [Chu+20a] the authors prove quantum query complexity results using the compressed oracles technique and provide a framework that simplifies such tasks. In [Chu+20b], the authors use the compressed-oracle technique to prove tight quantum time-space tradeoffs. In the recent [Ros21] Rosmanis describes a new approach to the compressed-oracle technique, he also analyzes the case of random (invertible) permutations, although does not provide a way to efficiently compress them.

Organization. In Section 6.2 we generalize the compressed-oracle technique of [Zha19a] to non-uniform distributions over functions. In Section 6.3 we prove a generalization of the O2H lemma of [Unr14], adapted to the use with compressed oracles for non-uniform distributions. In Section 6.4 we provide a general bound on the probability of Find—the quantum counterpart of the Bad events in the classical game-playing proofs. We end in Section 6.5 with a discussion of how to use the techniques developed in this chapter to elevate proofs of classical security to proofs of quantum security, given that the classical proofs are written in the game-playing framework.

6.2 Quantum-Accessible Oracles

In the Quantum-Random-Oracle Model (QROM) [Bon+11], one assumes that the random oracle can be accessed in superposition. Quantum-accessible random oracles are motivated by the possibility of running an actual instantiation of the oracle as function on a quantum computer, which would allow for superposition access. In this section, oracles implement a function $\mathbf{f} : \mathcal{X} \to \mathcal{Y}$ distributed according to some probability distribution $\mathfrak{D}$ on the set $\mathcal{F}$ of functions from $\mathcal{X}$ to $\mathcal{Y}$. Without loss of generality we set $\mathcal{X} = \mathbb{Z}_M$ and $\mathcal{Y} = \mathbb{Z}_N$ for some integers $M, N > 0$.

In this section we give a formal treatment of quantum accessible oracles. We explain with special care the compressed-oracle technique of Zhandry [Zha19a]. A quantum oracle can be viewed as a purification (extension to a higher-dimensional Hilbert space) of the adversary's quantum state. The simplest purification extends the state to include a superposition of all full function tables from the set $\mathcal{F}$. Note that the oracle gives access to a random function from the set $\mathcal{F}$. The purification we talk about is called the oracle register. A quantum algorithm could simulate the access to the quantum oracle by preparing the oracle register and performing the correct update procedures every time the adversary makes a query. Such a simulator would not be efficient though, as the oracle register we just defined holds M entries (so one for each element of the domain $[M]$). The brilliant idea of Zhandry was to propose a procedure to lazy-sample a uniformly random function. By lazy-sampling we mean here to store just the queries asked by the adversary, not the whole function table. By doing that we limit the number of entries held by the simulator to q (the bound on the number of queries performed by the adversary). Our result in this section is generalizing Zhandry's technique to independent distributions on functions: Such that outputs are distributed independently for any distinct inputs.

Classically, an oracle for a function $\mathbf{f}$ is modeled via a tape with the queried input x written on it, the tape is then overwritten with $\mathbf{f}(x)$. The usual way of translating this functionality to the quantum circuit model is by introducing a special gate that implements the unitary $\mathsf{U}_\mathbf{f}|x,y\rangle = |x, y+\mathbf{f}(x)\rangle$. In the literature $+$ is usually the bitwise addition modulo 2, but in general it can be any group operation. We are going to use addition in $\mathbb{Z}_N$.[2]

In the case where the function $\mathbf{f}$ is a random variable, so is the unitary $\mathsf{U}_\mathbf{f}$. Sometimes this is not, however, the best way to think of a quantum random oracle, as the randomness of $\mathbf{f}$ is accounted for using classical probability theory, yielding a hybrid description. To capture the adversary's point of view more explicitly, it is necessary to switch to the *mixed-state formalism*. A mixed quantum state, or *density matrix*, is obtained by considering the projector onto the one-dimensional subspace spanned by a pure state, and then taking the expectation over any classical randomness. Say that the adversary sends the query state $|\Psi_0\rangle = \sum_{x,y} \alpha_{x,y}|x,y\rangle$ to the oracle, the output state is then

$$\begin{aligned}&\sum_{\mathbf{f}} \mathbb{P}[\mathbf{f} : \mathbf{f} \leftarrow \mathfrak{D}]\ \mathsf{U}_\mathbf{f}|\Psi_0\rangle\langle\Psi_0|\mathsf{U}_\mathbf{f}^\dagger \ \otimes\ |\mathbf{f}\rangle\langle\mathbf{f}|_F \\ &= \sum_{\mathbf{f}} \mathbb{P}[\mathbf{f} : \mathbf{f} \leftarrow \mathfrak{D}] \sum_{x,x',y,y'} \alpha_{x,y}\bar{\alpha}_{x',y'}|x, y+\mathbf{f}(x)\rangle\langle x', y'+\mathbf{f}(x')| \otimes |\mathbf{f}\rangle\langle\mathbf{f}|_F, \qquad (6.1)\end{aligned}$$

where by $\bar{\alpha}$ we denote the complex conjugate of α and we have recorded the random function choice in a classical register F holding the full function table of $\mathbf{f}$.

In quantum information science, a general recipe for simplifying the picture and to gain additional insight is to *purify* mixed states, i.e. to consider a pure quantum state on a system augmented by an additional register E, such that discarding E recovers the original mixed state. In [Zha19a] Zhandry applies this recipe to this quantum-random-oracle formalism.

In the resulting representation of a random oracle, the classical register F is replaced by a quantum register holding a superposition of functions from $\mathfrak{D}$. The joint state before an adversary makes the first query with a state $|\Psi_0\rangle_{XY}$ is $|\Psi_0\rangle_{XY} \sum_{\mathbf{f}\in\mathcal{F}} \sqrt{\mathbb{P}[\mathbf{f} : \mathbf{f} \leftarrow \mathfrak{D}]}\,|\mathbf{f}\rangle_F$. The unitary that corresponds to $\mathsf{U}_\mathbf{f}$ after purification will be called the *Standard Oracle* StO and works by reading the appropriate output of $\mathbf{f}$ from F and adding it to the algorithm's output register,

$$\mathsf{StO}|x,y\rangle_{XY}|\mathbf{f}\rangle_F := |x, y+\mathbf{f}(x)\rangle_{XY}|\mathbf{f}\rangle_F. \qquad (6.2)$$

[2]Note that introducing the formalism using the group $\mathbb{Z}_N$ for some $N \in \mathbb{N}$ is quite general in the following sense: Any finite Abelian group G is isomorphic to a product of cyclic groups, and the (quantum) Fourier transform with respect to such a group is the tensor product of the Fourier transforms on the cyclic groups, given the natural tensor product structure of $\mathbb{C}^G$.

Applied to a superposition of functions as intended, StO will entangle the adversary's registers XY with the oracle register F.

The main observation of [Zha19a] is that if we change the basis of the initial state of the oracle register F, the redundancy of this initial state becomes apparent. If we are interested in, e.g., an oracle for a uniformly random function, the Fourier transform changes the initial oracle state $\sum_{\mathbf{f}} \frac{1}{\sqrt{|\mathcal{F}|}} |\mathbf{f}\rangle$ to a state holding only zeros $|0^M\rangle$, where $0 \in \mathcal{Y}$.

Let us start by presenting the interaction of the adversary viewed in the same basis, called the Fourier basis. The unitary operation acting in the Fourier basis is called the *Fourier Oracle* FO. Another important insight from [Zha19a] is that the Fourier Oracle, instead of adding the output of the oracle to the adversary's output register, does the opposite: It adds the value of the adversary's *output* register to the (Fourier-)transformed truth table

$$\mathsf{FO}|x, \eta\rangle_{XY} |\phi\rangle_F := |x, \eta\rangle_{XY} |\phi - \chi_{x,\eta}\rangle_F, \tag{6.3}$$

where ϕ is the transformed truth table $\mathbf{f}$ and $\chi_{x,\eta} := (0, \ldots, 0, \eta, 0, \ldots, 0)$ is a transformed truth table equal to 0 in all rows except for row x, where it has the value η. Note that we subtract $\chi_{x,\eta}$ so that the reverse of QFT returns addition of $\mathbf{f}(x)$.

Classically, a (uniformly) random oracle can be "compressed" by lazy-sampling the responses, i.e. by answering with previous answers if there are any, and with a fresh random value otherwise. Is lazy-sampling possible for quantum accessible oracles? Surprisingly, the answer is yes. Thanks to the groundbreaking ideas presented in [Zha19a] we know that there exists a representation of a quantum random oracle that is efficiently implementable.

In the remainder of this section we present an efficient representation of oracles for functions $\mathbf{f}$ sampled from product distributions. In the first part we introduce a general structure of quantum-accessible oracles. In the second part we generalize the idea of compressed random oracles to deal with non-uniform distributions of functions. In the appendix of [Cza+19], we provide additional details on the implementation of the procedures introduced in this section and step-by-step calculations of important identities and facts concerning compressed oracles. In Section 6.2.2.1 we recall in detail the compressed oracle introduced in [Zha19a], where the distribution of functions is uniform and the functions map bitstrings to bitstrings. We show the oracle in different bases and present calculations that might be useful for developing intuition for working with the new view on quantum random oracles.

6.2.1 General Structure of the Oracles

In this subsection we describe the general structure of quantum-accessible oracles that will give us a high-level description of all the oracles we define in this thesis. A quantum-accessible random oracle consists of

1. Hilbert spaces for the input $\mathcal{H}_{\mathcal{X}}$, output $\mathcal{H}_{\mathcal{Y}}$, and state registers $\mathcal{H}_{\mathcal{F}}$,
2. a procedure $\mathsf{Samp}_{\mathfrak{D}}$ that, on input a subset of the input space of the functions in $\mathfrak{D}$, prepares a superposition of partial functions on that subset of inputs with weights according to the respective marginal of the distribution $\mathfrak{D}$,
3. an update unitary $\mathsf{FO}_{\mathfrak{D}}$ that might depend on $\mathfrak{D}$ (in the case of compressed oracles) or not (in the case of full oracles, Equation (6.3)).

First of all, let us note that we use the Fourier picture of the oracle as the basis for our discussion. This picture, even though less intuitive at first sight, is simpler to handle mathematically. The distribution of the functions we model by the quantum oracle are implicitly given by the procedure $\mathsf{Samp}_{\mathfrak{D}}$ that when acting on the $|0\rangle$ state generates a superposition of values consistent with outputs of a function $\mathbf{f}$ sampled from $\mathfrak{D}$.

In the above structure the way we implement the oracle—in a compressed way, or acting on full function tables—depends on the way we define $\mathsf{FO}_{\mathfrak{D}}$.

The definition of $\mathsf{Samp}_{\mathfrak{D}}$ is such that $\mathsf{Samp}_{\mathfrak{D}}(\mathcal{X})|0^M\rangle = \sum_{\mathbf{f}\in\mathcal{F}} \sqrt{\mathbb{P}[\mathbf{f} \leftarrow \mathfrak{D}]}|\mathbf{f}\rangle$ and is a unitary operator.

Quantum-accessible oracles work as follows. First the oracle state is prepared in an all-zero state. Then at every query by the adversary we run $\mathsf{FO}_{\mathfrak{D}}$ which updates the state of the database. Further details are provided in the following sections.

6.2.2 Non-uniform Oracles

One of the main results of this chapter is generalizing the idea of purification and compression of quantum random oracles to a class of non-uniform function distributions. We show that the compressed-oracle technique can be used to deal with distributions over functions with outputs independent of any prior interactions. Examples of such functions are random Boolean functions that output one with a given probability. In Section 6.2.2.1 we provide a number of examples of the compressed-oracle technique for the uniform distribution.

We want to compress the following oracle

$$\begin{aligned}
&\mathsf{StO}|x,y\rangle_{XY} \sum_{\mathbf{f}\in\mathcal{F}} \sqrt{\mathbb{P}[\mathbf{f} : \mathbf{f} \leftarrow \mathfrak{D}]}\,|\mathbf{f}\rangle_F \\
&= \sum_{\mathbf{f}\in\mathcal{F}} \sqrt{\mathbb{P}[\mathbf{f} : \mathbf{f} \leftarrow \mathfrak{D}]}\,|x, y + \mathbf{f}(x) \mod N\rangle_{XY}\,|\mathbf{f}\rangle_F, \qquad (6.4)
\end{aligned}$$

where $\mathfrak{D}$ is a distribution on the set of functions $\mathcal{F} = \{\mathbf{f} : \mathcal{X} \to \mathcal{Y}\}$. The first ingredient we need is an operation that prepares the superposition of function truth tables according to the given distribution. More formally, we know a unitary that for all $\mathcal{S} \subseteq \mathcal{X}$

$$\mathsf{Samp}_{\mathfrak{D}}(\mathcal{S})|0^{|\mathcal{S}|}\rangle_{F(\mathcal{S})} = \bigotimes_{x\in\mathcal{S}} \sum_{y_x\in\mathcal{Y}} \sqrt{\mathbb{P}\left[y_x = \mathbf{f}(x) : \mathbf{f} \leftarrow \mathfrak{D}\right]}|y_x\rangle_{F(x)}, \tag{6.5}$$

where by $F(x)$ we denote the register corresponding to x. In Section 6.2.2.2 we present an example of $\mathsf{Samp}_{\mathfrak{D}}$ for a non-uniform $\mathfrak{D}$. Applying QFT to the adversary's register[3] gives us the *Phase Oracle* PhO that changes the phase of the state according to the output value $\mathbf{f}(x)$. This picture is commonly used in the context of bitstrings but is not very useful in our context. Additionally transforming the oracle register brings us to the Fourier Oracle, that we will focus on. This series of transformations can be depicted as a chain of oracles:

$$\mathsf{StO} \xleftrightarrow{\mathsf{QFT}_N^Y} \mathsf{PhO} \xleftrightarrow{\mathsf{QFT}_N^F} \mathsf{FO}, \tag{6.6}$$

going "to the right" is done by applying QFT_N and "to the left" by applying the adjoint. Also note that register Y holds a single value in $\mathcal{Y}$ and register F holds values in $\mathcal{Y}^M$, the transform above is an appropriate tensor product of QFT_N. The non-uniform Fourier Oracle is defined as $\mathsf{FO} = \mathsf{QFT}_N^{YF} \circ \mathsf{StO} \circ \mathsf{QFT}_N^{\dagger\, YF}$, as a consequence of that definition we have

$$\mathsf{FO}|x,\eta\rangle_{XY} \sum_{\phi} \frac{1}{\sqrt{N^M}} \sum_{\mathbf{f}\in\mathcal{F}} \sqrt{\mathbb{P}[\mathbf{f} : \mathbf{f} \leftarrow \mathfrak{D}]}\, \omega_N^{\phi\cdot\mathbf{f}}\, |\phi\rangle_F \tag{6.7}$$

$$= |x,\eta\rangle_{XY} \sum_{\phi} \frac{1}{\sqrt{N^M}} \sum_{\mathbf{f}\in\mathcal{F}} \sqrt{\mathbb{P}[\mathbf{f} : \mathbf{f} \leftarrow \mathfrak{D}]}\, \omega_N^{\phi\cdot\mathbf{f}}\, |\phi - \chi_{x,\eta} \mod N\rangle_F. \tag{6.8}$$

The main difference between uniform oracles and non-uniform oracles is that in the latter, the initial state of the oracle in the Fourier basis is not necessarily an all-zero state. That is because the unitary $\mathsf{Samp}_{\mathfrak{D}}$—that is used to prepare the initial state—is not the adjoint of the transformation between oracle pictures, like it is the case for the uniform distribution.

Before we give all details of $\mathsf{Samp}_{\mathfrak{D}}$ let us discuss the two bases: the Fourier basis and the prepared basis. To deal with the difference between the initial 0 state and the initial Fourier basis truth tables we use yet another alphabet and define Д (pronounced as [dɛ]) which denotes the *unprepared* database. We call it like that because the initial state of Д is the $((\bot,0),\ldots,(\bot,0))$ state. Moreover only by applying $\mathsf{QFT}_N^D \circ \mathsf{Samp}_{\mathfrak{D}}^D$ we transform it to Δ, i.e the Fourier basis database. As we will see, operations on Д are more intuitive and easier to define. We denote an unprepared database by $|Д\rangle_D = |x_1, и_1\rangle_{D_1}|x_2, и_2\rangle_{D_2}\cdots|x_q, и_q\rangle_{D_q}$ (where the Cyrillic letter и is pronounced as [i]). By $\Delta^Y(x)$ we denote the η value corresponding

[3]We denote this by QFT_N^Y, where Y is the adversary's output register

to the pair in Δ containing x and by $Д^X$ we denote the x values in $Д$. The intuition behind the preparation procedure is to initialize the truth table of the correct distribution in the correct basis. This notion is not visible in the uniform-distribution case, because there the sampling procedure for the uniform distribution $\mathfrak{U}$ is the Fourier transform: $\mathsf{Samp}_{\mathfrak{U}} = \mathsf{QFT}_N^\dagger$, and the database pictures Δ and $Д$ are equivalent. The following chain of databases similar to Equation (6.6) represents different pictures, i.e. bases, in which the compressed database can be viewed

$$|Д\rangle \xleftrightarrow{\mathsf{Samp}_{\mathfrak{D}}} |D\rangle \xleftrightarrow{\mathsf{QFT}_N^{D^Y}} |\Delta\rangle. \tag{6.9}$$

Before defining compressed oracles for non-uniform function distributions, let us take a step back and think about classical lazy sampling for such a distribution. Let $\mathbf{f}$ be a random function from a distribution $\mathfrak{D}$. In principle, lazy sampling is always possible as follows. When the first input x_1 is queried, just sample from the marginal distribution for $\mathbf{f}(x_1)$. Say the outcome is y_1 for the next query with x_2, we sample from the *conditional distribution* of $\mathbf{f}(x_2)$ given that $\mathbf{f}(x_1) = y_1$, etc.

Whether actual lazy sampling is feasible depends on the complexity of sampling from the conditional distributions of function values given that a polynomial number of other function values are already fixed.

The method for quantum lazy sampling that we generalize in this chapter is applicable only to a certain class of distributions. The distributions that we analyze must be independent for every input. By $\mathbf{f}(\mathcal{S})$ we denote the part of the full truth table of $\mathbf{f}$ corresponding to inputs from $\mathcal{S}$. Below we provide a definition of *product* distributions:

Definition 6.1 (Product distribution). *A distribution $\mathfrak{D}$ on a set of functions $\mathcal{F} \subseteq \{\mathbf{f} : \mathcal{X} \to \mathcal{Y}\}$ is called* product *if for all disjoint $\mathcal{S}_1, \mathcal{S}_2 \subseteq \mathcal{X}$, $\mathbf{f}(\mathcal{S}_1)$ and $\mathbf{f}(\mathcal{S}_2)$ are independently distributed when $\mathbf{f} \leftarrow \mathfrak{D}$.*

The situation when constructing compressed superposition oracles for non-uniformly distributed random functions is very similar. In this case we need the operations $\mathsf{Samp}_{\mathfrak{D}}(\mathcal{S})$ to be efficiently implementable for the compressed oracle to be efficient. Here, $\mathcal{S} \subseteq \mathcal{X}$. By inputting a set to $\mathsf{Samp}_{\mathfrak{D}}$ we mean that the operation will prepare a superposition of outputs to elements of the set.

Let us now come back to Definition 6.1, we want to translate the constraint on distributions to constraints on the quantum sampling procedure. The definition requires that the distribution is independent for any $\mathcal{S}_1$ and $\mathcal{S}_2$, this leads to the following requirement on sampling procedures:

$$\forall \mathcal{S}_1, \mathcal{S}_2 \subseteq \mathcal{X} : \ \mathsf{Samp}_{\mathfrak{D}}(\mathcal{S}_1 \cup \mathcal{S}_2) = \mathsf{Samp}_{\mathfrak{D}}(\mathcal{S}_1) \circ \mathsf{Samp}_{\mathfrak{D}}(\mathcal{S}_2). \tag{6.10}$$

Let us present a detailed definition of sampling procedures for product distributions[4].

Definition 6.2 (Sampling procedure for a product $\mathfrak{D}$)**.** *A sampling procedure* $\mathsf{Samp}_{\mathfrak{D}}$ *for a product distribution* $\mathfrak{D}$ *(as defined in Definition 6.1) is a family of unitary operators*

$$\{\mathsf{Samp}_{\mathfrak{D}}(\mathcal{S}_1) : \mathcal{S}_1 \subseteq \mathcal{X}\}\,, \tag{6.11}$$

where each operator fulfills the following conditions:

(*i*) *It is efficiently implementable in the number of inputs* $|\mathcal{S}_1|$*.*

(*ii*) *It prepares the appropriate superposition on the zero state:*

$$\mathsf{Samp}_{\mathfrak{D}}(\mathcal{S}_1)|0^{|\mathcal{S}_1|}\rangle_{F_1} = \sum_{\vec{y}_1 \in \mathcal{Y}^{|\mathcal{S}_1|}} \sqrt{\underset{\mathbf{f} \leftarrow \mathfrak{D}}{\mathbb{P}}[\mathbf{f}(\mathcal{S}_1) = \vec{y}_1]}|\vec{y}_1\rangle_{F_1}. \tag{6.12}$$

(*iii*) *The operators are independent, so for* $\mathcal{S}_1, \mathcal{S}_2 \subseteq \mathcal{X}$ *such that* $\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset$ *we have:*

$$\mathsf{Samp}_{\mathfrak{D}}^{F_1F_2}(\mathcal{S}_1 \cup \mathcal{S}_2) = \mathsf{Samp}_{\mathfrak{D}}^{F_1}(\mathcal{S}_1) \circ \mathsf{Samp}_{\mathfrak{D}}^{F_2}(\mathcal{S}_2) \tag{6.13}$$

and F_1 *and* F_2 *are different quantum registers.*

Note that for $\mathsf{Samp}_{\mathfrak{D}}(\mathcal{S})$ to be efficient, it is not sufficient that the probability distributions $\mathfrak{D}$ are classically efficiently samplable. This is because running a reversible circuit obtained from a classical sampling algorithm on a superposition of random inputs will, in general, entangle the sample with the garbage output of the reversible circuit. The problem of efficiently creating a superposition with amplitudes $\sqrt{\mathbf{p}(x)}$ for some probability distribution $\mathbf{p}$ has appeared in other contexts, e.g. in classical-client quantum fully homomorphic encryption [Mah18].

Before we state the algorithm that realizes the general *Compressed Fourier Oracle* $\mathsf{CFO}_{\mathfrak{D}}$ we provide a high-level description of the procedure. The oracle $\mathsf{CFO}_{\mathfrak{D}}$ is a unitary algorithm that performs quantum lazy sampling, maintaining a compressed database of the adversary's queries. For the algorithm to be correct—indistinguishable for all adversaries from the full oracle—it has to respect the following invariants of the database: The full oracle is oblivious to the *order* in which a set of inputs is queried. Hence the same property has to hold for the compressed oracle, i.e. we cannot keep entries (x, η) in the order of queries. We ensure this property by keeping the database *sorted* according to x.

[4]An interesting example of a distribution that is not product but which we can quantumly lazy-sample is the following: It is uniform for inputs in $\{0,1\}^n \setminus \{x\}$ for any x and is fully determined on the "last" input: $\mathbf{f}(x) = \bigoplus_{x' \neq x} \mathbf{f}(x')$.

The second issue concerns the danger of storing too much information. If after the query we save (x, η) in the database but the resulting entry mapped to $(x, 0)$ in the *unprepared* basis, i.e. the basis before applying Samp, then the compressed database would entangle itself with the adversary, unlike in the case of the full oracle. Hence the database cannot contain 0 in the unprepared basis.

In the following we sketch the workings of the quantum algorithm $\mathsf{CFO}_{\mathfrak{D}}$ responsible for updating the oracle register. The set of inputs $\mathcal{X}$ is expanded by the symbol $\perp$, denoting an empty entry in the quantum database. If after q queries the database has a suffix of u pairs of the form $(\perp, 0)$, we say the database has $s = q - u$ non-padding entries.

$\mathsf{CFO}_{\mathfrak{D}}$: On input $|x, \eta\rangle_A$ and holding the database $|(x_1, \text{и}_1), \ldots, (x_s, \text{и}_s), (\perp, 0), \ldots, (\perp, 0)\rangle_D$ do the following:

1. Find the index $l \in [q]$ of the register holding the first x_l from the right that is $x_l \leq x$, we should insert (x, η) into this register.
2. If $x \neq x_l$: insert x in a register after the last element of the database and shift it to position l, moving the intermediate registers backwards.
3. Apply $\mathsf{QFT}_N^{D_l^Y} \circ \mathsf{Samp}_{\mathfrak{D}}^{D_l}(x)$ to change the basis to the Fourier basis (in which the adversary's η is encoded) and update register D_l (i.e. the l'th register in D) to contain $(x_l, \eta_l - \eta)$, change the basis back to original by applying $\mathsf{Samp}_{\mathfrak{D}}^{\dagger D_l}(x) \circ \mathsf{QFT}_N^{\dagger D_l^Y}$.
4. Check if register l contains a pair of the form $(x_l, 0)$, if yes subtract x from the first part to yield $(\perp, 0)$ and shift it back to the end of the database.
5. Uncompute[5] l.

Using this notation, Algorithm 6.1 defines the procedure of updates of the database of the compressed database. We refer to the appendix of [Cza+19] for the fully detailed description of $\mathsf{CFO}_{\mathfrak{D}}$.

Below in Algorithm 6.2 we explain how to uncompute l in Line 12 of Algorithm 6.1.

In Algorithm 6.1 we use the fact that $\mathsf{Samp}_{\mathfrak{D}}$ is a local sampling procedure, Definition 6.2; Note that we write $\mathsf{Samp}_{\mathfrak{D}}^{D(x)}(x)$, so the sampling is independent from all queries that are already in the database.

We would like to stress that to keep the compressed oracle $\mathsf{CFO}_{\mathfrak{D}}$ a unitary operation we always keep the database of size q. This can be easily changed by

[5]Uncomputing a function means in the context of quantum computing applying the conjugate of the unitary calculating this function.

Algorithm 6.1 General $\mathsf{CFO}_{\mathfrak{D}}$

```
    input: Unprepared database and adversary query: $|x,\eta\rangle_{XY}|Д\rangle_D$
    output: $|x,\eta\rangle_{XY}|Д'\rangle_D$
 1: Count in register $S$ the number of non-padding ($Д^X \neq \perp$) entries $s$ in $D$
 2: if $x \notin Д^X$ then                                              ▷ add
 3:     Insert $x$ in $Д^X$ in the right place and add 1 to $S$       ▷ keeping $Д^X$ sorted
 4: Apply $\mathsf{QFT}_N^{D^Y(x)}\mathsf{Samp}_{\mathfrak{D}}^{D^Y(x)}(x)$        ▷ prepare the database: $Д(x) \mapsto \Delta(x)$
 5: Subtract $\eta$ from $\Delta^Y(x)$                                   ▷ update entry with $x$
 6: Apply $\mathsf{Samp}_{\mathfrak{D}}^{\dagger D(x)}(x)\mathsf{QFT}_N^{\dagger D^Y(x)}$      ▷ unprepare the database: $\Delta(x) \mapsto Д(x)$
 7: In register $L$ save location $l$ of $x$ in $Д$
 8: if $Д_l^Y = 0$ then                                         ▷ remove or do nothing
 9:     Remove $x$ from $D_l^X$ and shift register $D_l^X$ to the back        ▷ $Д_l^X \mapsto \perp$
10: if $Д_l^X \neq x$ then                                           ▷ $x$ was removed
11:     Shift $D_l^Y$ to the back and subtract 1 from $S$
12: Uncompute $l$ from register $L$                                  ▷ Algorithm 6.2
13: Uncompute $s$ from register $S$
14: return $|x,\eta\rangle_{XY}|Д'\rangle_D$                          ▷ $Д'$ is the modified database
```

Algorithm 6.2 Uncompute L in Line 12 of Algorithm 6.1

```
1: Control on registers $X$ and $D^X$
2: for $i = 1\ldots, s-1$ do
3:     if $Д_i^X = x$ then
4:         Subtract $i$ from $L$
5:     else if $Д_i^X < x$ and $x < Д_{i+1}^X$ then
6:         Subtract $i+1$ from $L$
```

always appending an empty register at the beginning of each query of adversary A. The current formulation of $\mathsf{CFO}_{\mathfrak{D}}$ assumes that there is an upper bound on the number of queries made by the adversary, this is not a fundamental requirement.

The interface corresponding to the compressed Fourier oracle $\mathsf{CFO}_{\mathfrak{D}}$ interprets the adversary's output register in the Fourier basis. When we want to change the basis to the standard one, we apply $\mathsf{QFT}_N^{D^Y}$ to the database register and QFT_N^{Y} to the adversary's output register. These basis changes give rise to the versions of oracle analogous to the full-oracle case:

$$\mathsf{CStO} \xleftrightarrow{\mathsf{QFT}_N^Y} \mathsf{CPhO} \xleftrightarrow{\mathsf{QFT}_N^{D^Y}} \mathsf{CFO}. \tag{6.14}$$

The intermediate oracle is the compressed phase oracle.

The decompression procedure for the general Compressed Fourier Oracle is given by Algorithm 6.3. The output of the decompression procedure $\phi(Д)$ is

Algorithm 6.3 General Decompression Procedure $\mathsf{Dec}_{\mathfrak{D}}$

```
    input: Unprepared database: $|Д\rangle_D$
    output: Prepared, Fourier-basis truth table: $|\phi(Д)\rangle$
 1: Count in register $S$ the number of non-padding ($Д^X \neq \perp$) entries $s$ in $D$
 2: Initialize register $F$ in the state $\bigotimes_{x\in\mathcal{X}}|0\rangle$
 3: for $i = 1, 2, \ldots, s$ do                                  ▷ Controlled on $S$
 4:     Swap register $D_i^Y$ with register $F(x_i)$
 5: for $x \in \mathcal{X}$ in descending order do
 6:     if $F(x)$ holds a value $\neq 0$ then
 7:         Subtract $x$ from register $D_s^X$                    ▷ $Д_s^X \mapsto \perp$
 8:         Subtract $1$ from register $S$
 9: Discard $D$ and $S$
10: Apply $\mathsf{QFT}_N^F \mathsf{Samp}_{\mathfrak{D}}^F(\mathcal{X})$          ▷ Prepare the database
```

the state holding the prepared Fourier-basis truth table of the functions from $\mathfrak{D}$, which by construction is consistent with the adversary's interaction with the compressed oracle.

The decompression can be informally described as follows. The first operation coherently counts the number of $Д^X \neq \perp$ and stores the result in a register S. Next we prepare a fresh all-zero initial state of a function from $\mathcal{X}$ to $\mathcal{Y}$, i.e. $|\mathcal{X}|$ registers of dimension N, all in the zero state. These registers will hold the final FO superposition oracle state. The next step is swapping each Y-type register of the CFO-database with the prepared zero state in the FO at the position indicated by the corresponding X-type register in the CFO database. This FOR loop is controlled on register S. Note that after preparing S we do not modify S anymore in this step. The task left to do is deleting x's from D. It is made possible by the fact that the non-padding entries of the CFO database are nonzero and ordered. That is why we can iterate over the entries of the truth table F and, conditioned on the entry not being 0, delete the last entry of D^X and reducing S by one to update the number of remaining non-padding entries in the CFO-database. Here the loop range does not depend on the size of the database, just the size of the domain. Finally, we switch to the correct basis to end up with a full oracle of Fourier type, i.e. a FO.

Theorem 6.3 (Correctness of $\mathsf{CFO}_{\mathfrak{D}}$)**.** *Say $\mathfrak{D}$ is a product distribution* (*Definition 6.1*) *over functions, let $\mathsf{CFO}_{\mathfrak{D}}$ be defined as in Algorithm 6.1 and* FO *as in Equation*(6.8)*. Let z be a random arbitrarily distributed string. Then for any quantum adversary* A *making q quantum queries we have*

$$\left|\mathbb{P}\left[b = 1 : b \leftarrow \mathrm{A}^{\mathsf{FO}}(z)\right] - \mathbb{P}\left[b = 1 : b \leftarrow \mathrm{A}^{\mathsf{CFO}}(z)\right]\right| = 0. \tag{6.15}$$

Proof. We will show that

$$|\Psi_{\mathsf{FO}}\rangle_{AF} = \mathsf{Dec}_{\mathfrak{D}}^D |\Psi_{\mathsf{CFO}}\rangle_{AD}, \tag{6.16}$$

where $|\Psi_{\mathsf{FO}}\rangle_{AF}$ is the joint state of the adversary and the oracle resulting from the interaction of A with FO and $|\Psi_{\mathsf{CFO}}\rangle_{AD}$ is the state resulting from the interaction of A with $\mathsf{CFO}_{\mathfrak{D}}$. The state $|\Psi_{\mathsf{FO}}\rangle_{AF}$ is generated by applying $\prod_{i=1}^{q} \mathsf{U}_i \circ \mathsf{FO}$ to $|\psi_0\rangle_A|0^M\rangle_F$, where the $|\psi_0\rangle_A$ is the initial state of the adversary. In the case of the compressed oracle, the state $|\Psi_{\mathsf{CFO}}\rangle_{AD}$ is generated by applying $\prod_{i=1}^{q} \mathsf{U}_i \circ \mathsf{CFO}$ to $|\Psi_0\rangle_A|(\perp,0)^q\rangle_D$, where $(\perp,0)^q$ denotes q pairs $(\perp,0)$.

We can focus on the state equality from Equation (6.16) because if they are indeed equal, then any adversary measurement on $|\Psi_{\mathsf{FO}}\rangle_{AF}$ will yield the output $b=1$ with the same probability as on $\mathsf{Dec}_{\mathfrak{D}}^{D}|\Psi_{\mathsf{CFO}}\rangle_{AD}$.

Let us call a database state

$$|Д(\vec{x},\vec{и})\rangle := |x,\eta\rangle_{XY}|x_1,и_1\rangle_{D_1}\cdots|x_s,и_s\rangle_{D_s}\cdots|\perp,0\rangle_{D_q}, \tag{6.17}$$

where $\vec{x}:=(x_1,x_2,\ldots,x_s)$ and $\vec{и}:=(и_1,и_2,\ldots,и_s)$ well-formed, if no x_i in $\vec{x}$ is $\perp$ and no $и_i$ in $\vec{и}$ is zero.

To prove Equation (6.16) we show that

$$\mathsf{FO}\circ\mathsf{Dec}_{\mathfrak{D}}|Д(\vec{x},\vec{и})\rangle = \mathsf{Dec}_{\mathfrak{D}}\circ\mathsf{CFO}_{\mathfrak{D}}|Д(\vec{x},\vec{и})\rangle. \tag{6.18}$$

This is sufficient for the proof of the theorem as $|\Psi_{\mathsf{FO}}\rangle$ is generated by a series of the adversary's unitaries intertwined with oracle calls. If we show that $\mathsf{FO} = \mathsf{Dec}_{\mathfrak{D}}\circ\mathsf{CFO}_{\mathfrak{D}}\circ\mathsf{Dec}_{\mathfrak{D}}^{\dagger}$, when acting on well-formed databases, then everything that happens on the oracle's register side can be compressed. Note that as we start from the empty oracle state and only apply the oracle to the oracle register, the database will always be well-formed.

We study the action of $\mathsf{Dec}_{\mathfrak{D}}$ on the state in Equation (6.17). To write the output state we need to name the matrix elements of the sampling unitary: $(\mathsf{Samp}_{\mathfrak{D}}(\mathcal{X}))_{\mathbf{f}\vec{И}} = a_{\mathbf{f}\vec{И}}(\mathcal{X})$, the column index consists of a vector of size M with exactly s non-zero entries: $\vec{и} = (0,\ldots,0,и_1,0\ldots,0,и_2,0,\ldots)$. The decompressed state is

$$\begin{aligned}|\Upsilon(\vec{x},\vec{и})\rangle_F &:= \mathsf{Dec}_{\mathfrak{D}}|Д(\vec{x},\vec{и})\rangle\\ &= \sum_{\phi\in\mathcal{F}}\frac{1}{\sqrt{N^M}}\sum_{\mathbf{f}\in\mathcal{F}}\omega_N^{\phi\cdot\mathbf{f}}\, a_{\mathbf{f}\vec{И}}(\mathcal{X})\;|\phi_0\rangle_{F(0)}\cdots|\phi_{M-1}\rangle_{F(M-1)},\end{aligned} \tag{6.19}$$

where $\phi\cdot\mathbf{f} = \sum_{x\in\mathcal{X}}\phi_x\mathbf{f}(x) \mod N$ and by $\mathbf{f}(x)$ we denote row number x of the function truth table $\mathbf{f}$.

Using the fact that $\mathsf{Samp}_{\mathfrak{D}}$ is defined for a product distribution, as in Definition 6.2, we have that $\mathsf{Samp}_{\mathfrak{D}}(\mathcal{X})$
$= \mathsf{Samp}_{\mathfrak{D}}(\mathcal{X}\setminus\{x\})\circ\mathsf{Samp}_{\mathfrak{D}}(x)$ and we can focus our attention on some fixed x: isolate register $F(x)$ with amplitudes depending only on x. Let us compute

this state after application of FO, note that FO only subtracts η from $\mathsf{F}(x)$:

$$\mathsf{FO}|x,\eta\rangle_{XY}|\Upsilon(\vec{x},\vec{\text{и}})\rangle_F = |x,\eta\rangle_{XY} \sum_{\phi',\mathbf{f}'\in\mathcal{F}(\mathcal{X}\setminus\{x\})} \frac{1}{\sqrt{N^{M-1}}}\,\omega_N^{\phi'\cdot\mathbf{f}'}\, a_{\mathbf{f}'\vec{\text{и}}'}(\mathcal{X}\setminus\{x\})$$
$$\cdot\,|\phi_0\rangle_{F(0)}\cdots\left(\sum_{\zeta,z\in[N]}\frac{1}{\sqrt{N}}\,\omega_N^{\zeta\cdot z}\, a_{z\text{и}_x}(x)\,|\zeta-\eta\rangle_{F(x)}\right)\cdots|\phi_{M-1}\rangle_{F(M-1)}, \tag{6.20}$$

where $\vec{\text{и}}' \in \mathcal{Y}^{M-1}$ denotes the vector of и_i without the row with index x. Note that $\text{и}_x = 0$ if x was not in $\vec{x}$ before decompression and $\text{и}_x \neq 0$ otherwise.

The harder part of the proof is showing that the right hand side of Equation (6.18) actually equals the left hand side that we just analyzed. Let us inspect $|\text{Д}(\vec{x},\vec{\text{и}})\rangle$ after application of the compressed oracle

$$\mathsf{CFO}_{\mathfrak{D}}|x,\eta\rangle_{XY}|\text{Д}(\vec{x},\vec{\text{и}})\rangle_D = |x,\eta\rangle_{XY}$$
$$\cdot\left(\sum_{\tilde{\text{и}}_x\neq 0}\alpha(x,\eta,\text{и}_x,\tilde{\text{и}}_x)\,|\text{Д}'_{\text{ADD/UPD}}\rangle_D + \alpha(x,\eta,\text{и}_x,0)\,|\text{Д}'_{\text{REM/NOT}}\rangle_D\right), \tag{6.21}$$

where $\tilde{\text{и}}_x$ is the new value of $\text{Д}^Y(x)$ and и_x is the old content of the database. By $\text{Д}'_{\text{ADD/UPD}}$ we denote the database $\text{Д}(\vec{x},\vec{\text{и}})$ with entry $\tilde{\text{и}}_x \neq 0$, it corresponds to x being added or updated. By $\text{Д}'_{\text{REM/NOT}}$ we denote the database where $\tilde{\text{и}}_x = 0$, meaning x was removed from Д or nothing happened. The function $\alpha(\cdot)$ denotes the corresponding amplitudes.

Before we proceed with decompression of the above state let us calculate the amplitudes α. Again using the definition of $\mathsf{Samp}_{\mathfrak{D}}$ we describe the action of the compressed oracle on a single x step by step. Below we denote by Rem removing $\text{и} = 0$ from Д and by Sub subtraction of η from database register D^Y. We start with a database containing $(x,\text{и}_x)$, which we can always assume due to line 3 in Algorithm 6.1. In the case x was not already in Д, then $\text{и}_x = 0$, otherwise it is the value defined in previous queries. The simplification we make is to describe $\mathsf{CFO}_{\mathfrak{D}}$ acting on a single-entry database. We do not lose generality by that as the only thing that changes for q larger than one is maintaining proper sorting and padding, which can be easily done (see appendix of [Cza+19] for details). The calculation of $\mathsf{CFO}_{\mathfrak{D}}$ on a basis state follows:

$$|x,\eta\rangle_{XY}|x,\text{и}_x\rangle_D \overset{\mathsf{Samp}_{\mathfrak{D}}}{\mapsto} |x,\eta\rangle_{XY}\sum_{z\in[N]} a_{z\text{и}_x}(x)\,|x,z\rangle_D \tag{6.22}$$

$$\overset{\mathsf{QFT}_N^{D^Y}}{\mapsto} |x,\eta\rangle_{XY}\sum_{z\in[N]} a_{z\text{и}_x}(x)\sum_{\zeta\in[N]}\frac{1}{\sqrt{N}}\,\omega_N^{\zeta\cdot z}\,|x,\zeta\rangle_D \tag{6.23}$$

$$\overset{\mathsf{Sub}}{\mapsto}|x,\eta\rangle_{XY}\sum_{z,\zeta\in[N]} a_{z\text{и}_x}(x)\frac{1}{\sqrt{N}}\,\omega_N^{\zeta\cdot z}\,|x,\zeta-\eta\rangle_D \tag{6.24}$$

$$\overset{\mathsf{QFT}_N^{\dagger D^Y}}{\mapsto} |x,\eta\rangle_{XY} \sum_{z,\zeta\in[N]} a_{z\text{и}_x}(x)\frac{1}{\sqrt{N}}\,\omega_N^{\zeta\cdot z} \sum_{z'\in[N]} \frac{1}{\sqrt{N}}\,\bar{\omega}_N^{z'\cdot(\zeta-\eta)}\,|x,z'\rangle_D \tag{6.25}$$

$$=|x,\eta\rangle_{XY} \sum_{z\in[N]} a_{z\text{и}_x}(x) \underbrace{\sum_{z',\zeta\in[N]} \frac{1}{N}\,\omega_N^{\zeta\cdot z}\,\bar{\omega}_N^{z'\cdot(\zeta-\eta)}}_{=\bar{\omega}_N^{-z\cdot\eta}\,\delta(z',z)} |x,z'\rangle_D \tag{6.26}$$

$$\overset{\mathsf{Samp}_{\mathfrak{D}}^{\dagger D}(x)}{\mapsto} |x,\eta\rangle_{XY} \sum_{z\in[N]} a_{z\text{и}_x}(x)\,\omega_N^{z\cdot\eta} \sum_{\tilde{\text{и}}_x\in[N]} \bar{a}_{z\tilde{\text{и}}_x}(x)\,|x,\tilde{\text{и}}_x\rangle_D \tag{6.27}$$

$$=|x,\eta\rangle_{XY} \sum_{\tilde{\text{и}}_x\in[N]} \underbrace{\sum_{z\in[N]} a_{z\text{и}_x}(x)\,\omega_N^{z\cdot\eta}\,\bar{a}_{z\tilde{\text{и}}_x}(x)}_{:=\alpha(x,\eta,\text{и}_x,\tilde{\text{и}}_x)} |x,\tilde{\text{и}}_x\rangle_D \tag{6.28}$$

$$\overset{\mathsf{Rem}^D}{\mapsto} |x,\eta\rangle_{XY}\left(\sum_{\text{и}\in[N]\setminus\{0\}} \alpha(x,\eta,\text{и}_x,\tilde{\text{и}}_x)\,|x,\tilde{\text{и}}_x\rangle_D + \alpha(x,\eta,\text{и}_x,0)\,|\bot,0\rangle_D\right). \tag{6.29}$$

In the above equations we have defined α as

$$\alpha(x,\eta,\text{и}_x,\tilde{\text{и}}_x) := \sum_{z\in[N]} a_{z\text{и}_x}(x)\,\bar{a}_{z\tilde{\text{и}}_x}(x)\,\omega_N^{z\cdot\eta}. \tag{6.30}$$

After decompressing the state from Equation (6.21), the resulting database state will be $\sum_{\tilde{\text{и}}_x\neq 0}\alpha(x,\eta,\text{и}_x,\tilde{\text{и}}_x)\,|\Upsilon(\text{Д}'_{\text{ADD/UPD}})\rangle+\alpha(x,\eta,\text{и}_x,0)\,|\Upsilon(\text{Д}'_{\text{REM/NOT}})\rangle_D$, where we overload notation of $|\Upsilon(\vec{x},\vec{\text{и}})\rangle$ to denote that $(\vec{x},\vec{\text{и}})$ consists of values in the respective databases. We can write down this state in more detail using Equation (6.20):

$$\begin{aligned}
&\mathsf{Dec}_{\mathfrak{D}}\circ\mathsf{CFO}_{\mathfrak{D}}|x,\eta\rangle_{XY}|\text{Д}(\vec{x},\vec{\text{и}})\rangle_D\\
&=\sum_{\phi',\mathbf{f}'\in\mathcal{F}(\mathcal{X}\setminus\{x\})} \frac{1}{\sqrt{N^{M-1}}}\,\omega_N^{\phi'\cdot\mathbf{f}'}\,a_{\mathbf{f}'\vec{\text{и}}'}(\mathcal{X}\setminus\{x\})\,|\phi_0\rangle_{F(0)}\cdots\\
&\cdot\left(\sum_{\tilde{\text{и}}_x\neq 0}\alpha(x,\eta,\text{и}_x,\tilde{\text{и}}_x)\sum_{\zeta,z\in[N]}\frac{1}{\sqrt{N}}\omega_N^{\zeta\cdot z}a_{z\tilde{\text{и}}_x}(x)|\zeta\rangle_{F(x)}\right.\\
&\left.+\alpha(x,\eta,\text{и}_x,0)\sum_{\zeta,z\in[N]}\frac{1}{\sqrt{N}}\omega_N^{\zeta\cdot z}a_{z0}(x)|\zeta\rangle_{F(x)}\right)\cdots|\phi_{M-1}\rangle_{F(M-1)}.
\end{aligned} \tag{6.31}$$

In the above equation we notice that

$$\begin{aligned}
&\sum_{\tilde{\text{и}}_x\neq 0}\alpha(x,\eta,\text{и}_x,\tilde{\text{и}}_x)\sum_{\zeta,z\in[N]}\frac{1}{\sqrt{N}}\omega_N^{\zeta\cdot z}a_{z\tilde{\text{и}}_x}(x)|\zeta\rangle_{F(x)}\\
&+\alpha(x,\eta,\text{и}_x,0)\sum_{\zeta,z\in[N]}\frac{1}{\sqrt{N}}\omega_N^{\zeta\cdot z}a_{z0}(x)|\zeta\rangle_{F(x)}
\end{aligned}$$

$$= \sum_{\zeta, z \in [N]} \frac{1}{\sqrt{N}} \, \omega_N^{\zeta \cdot z} \sum_{\tilde{\text{и}}_x \in [N]} \alpha(x, \eta, \text{и}_x, \tilde{\text{и}}_x) \; a_{z\tilde{\text{И}}_x}(x) \; |\zeta\rangle_{F(x)} \tag{6.32}$$

which comes from the fact that $\mathsf{Samp}_{\mathfrak{D}}$ is a unitary and $\sum_{j \in [N]} a_{ij} \bar{a}_{kj} = \delta_{ik}$ and therefore we have

$$\begin{aligned} &\sum_{\tilde{\text{и}}_x \in [N]} \alpha(x, \eta, \text{и}_x, \tilde{\text{и}}_x) \; a_{z\tilde{\text{И}}_x}(x) \\ &= \sum_{z' \in [N]} \underbrace{\sum_{\tilde{\text{и}}_x \in [N]} \bar{a}_{z'\tilde{\text{И}}_x}(x) \; a_{z\tilde{\text{И}}_x}(x)}_{=\delta_{z',z}} a_{z'\text{И}_x}(x) \; \omega_N^{z' \cdot \eta} = a_{z\text{И}_x}(x) \; \omega_N^{z \cdot \eta}. \end{aligned} \tag{6.33}$$

Together with changing the variable $\zeta \mapsto \zeta - \eta$ and observing Equation (6.20) we derive the claimed identity:

$$\begin{aligned} &\mathsf{Dec}_{\mathfrak{D}} \circ \mathsf{CFO}_{\mathfrak{D}} |x, \eta\rangle_{XY} |\text{Д}(\vec{x}, \vec{\text{и}})\rangle_D \\ &= \mathsf{FO} \; |x, \eta\rangle_{XY} |\Upsilon(\vec{x}, \vec{\text{и}})\rangle = \mathsf{FO} \circ \mathsf{Dec}_{\mathfrak{D}} \; |x, \eta\rangle_{XY} |\text{Д}(\vec{x}, \vec{\text{и}})\rangle_D. \end{aligned} \tag{6.34}$$

□

6.2.2.1 Uniform Oracles

The uniform distribution is an especially important distribution we can lazy sample. In this case the unprepared basis simplifies to the Fourier basis. The sampling procedure $\mathsf{Samp}_{\mathfrak{U}}$ equals $\mathsf{QFT}_N^\dagger$. We denote the compressed oracle for $\mathfrak{U}$ over the set $\mathcal{X}^{\mathcal{Y}}$ by $\mathsf{CStO}_{\mathcal{Y}}$.

For ease of exposition, and to highlight the connection to the formalism in [Zha19a], we present a discussion of compressed oracles with *uniform oracles* that model functions sampled uniformly at random from $\mathcal{F} := \{f : \{0,1\}^m \to \{0,1\}^n\}$.

We denote the uniform distribution over $\mathcal{F}$ by $\mathfrak{U}$. The cardinality of the set of functions is $|\mathcal{F}| = 2^{n2^m}$ and the truth table of any $f \in \mathcal{F}$ can be represented by 2^m rows of n bits each. Uniform oracles are the most studied in the random-oracle model and are also analyzed in [Zha19a].

The transformation we use in the case of uniformly sampled functions is the Hadamard transform. The unitary operation to change between types of oracles is defined as

$$\mathsf{HT}_n |x\rangle := \frac{1}{\sqrt{2^n}} \sum_{\xi \in \{0,1\}^n} (-1)^{\xi \cdot x} |\xi\rangle, \tag{6.35}$$

where $\xi \cdot x$ is the inner product modulo two between the n-bit strings ξ and x viewed as vectors. In this section the registers X, Y are vectors in the n-qubit Hilbert space $(\mathbb{C}^2)^{\otimes n}$.

In what follows we first focus on *full* oracles, i.e. not compressed ones. We analyze in detail the relations between different pictures of the oracles: the Standard Oracle, the Fourier Oracle, and the intermediate Phase Oracle. Next we provide an explicit algorithmic description of the compressed oracle and discuss the behavior of the compressed oracle in different pictures.

For the QROM, usually the Standard Oracle is the oracle used. The initial state of the oracle is the uniform superposition of truth tables $\mathbf{f}$ representing functions $\mathbf{f} : \{0,1\}^m \to \{0,1\}^n$. The Standard Oracle acts as follows

$$\mathsf{StO}_{\mathfrak{U}}|x,y\rangle_{XY}\frac{1}{\sqrt{|\mathcal{F}|}}\sum_{\mathbf{f}\in\mathcal{F}}|\mathbf{f}\rangle_F = \frac{1}{\sqrt{|\mathcal{F}|}}\sum_{\mathbf{f}\in\mathcal{F}}|x,y\oplus\mathbf{f}(x)\rangle_{XY}\otimes|\mathbf{f}\rangle_F, \tag{6.36}$$

where instead of modular addition we use bitwise XOR denoted by $\oplus$. Note that in the above formulation $\mathsf{StO}_{\mathfrak{U}}$ is just a controlled XOR operation from the x-th row of the truth table to the output register Y. We add the subscript $\mathfrak{U}$ to denote that in the case of uniform distribution we also fix the input and output sets to bit-strings and the operation the oracle performs is not addition modulo N. The register F contains vectors in $(\mathbb{C}^2)^{\otimes n2^m}$.

The Fourier Oracle that stores the queries of the adversary is defined as

$$\mathsf{FO}_{\mathfrak{U}}|x,\eta\rangle_{XY}|\phi\rangle_F := |x,\eta\rangle_{XY}|\phi\oplus\chi_{x,\eta}\rangle_F, \tag{6.37}$$

where $\chi_{x,\eta} := (0^n,\dots,0^n,\eta,0^n,\dots,0^n)$ is a table with 2^m rows, among which only the x-th row equals η and the rest are filled with zeros. Note that initially the Y register is in the Hadamard basis, for that reason we use Greek letters to denote its value.

To model the random oracle we initialize the oracle register F in the Hadamard basis in the all 0 state $|\phi\rangle = |0^{n2^m}\rangle$.

If we take the Standard Oracle again and transform the adversary's Y register instead, again using HT, we recover the commonly used Phase Oracle. More formally, the phase oracle is defined as

$$\mathsf{PhO}_{\mathfrak{U}} := (\mathbb{1}_m^X\otimes\mathsf{HT}_n^Y)\otimes\mathbb{1}_{n2^m}^F\circ\mathsf{StO}_{\mathfrak{U}}\circ(\mathbb{1}_m^X\otimes\mathsf{HT}_n^Y)\otimes\mathbb{1}_{n2^m}^F, \tag{6.38}$$

where $\mathbb{1}_n$ is the identity operator acting on n qubits.

Applying the Hadamard transform also to register F will give us the Fourier Oracle

$$\mathsf{FO}_{\mathfrak{U}} = (\mathbb{1}^{XY})\otimes\mathsf{HT}_{n2^m}^F\circ\mathsf{PhO}_{\mathfrak{U}}\circ(\mathbb{1}^{XY})\otimes\mathsf{HT}_{n2^m}^F\,. \tag{6.39}$$

The above relations show that we have a chain of oracles, similar to Equation (6.6):

$$\mathsf{StO}_{\mathfrak{U}}\xleftrightarrow{\mathsf{HT}_n^Y}\mathsf{PhO}_{\mathfrak{U}}\xleftrightarrow{\mathsf{HT}_{n2^m}^F}\mathsf{FO}_{\mathfrak{U}}. \tag{6.40}$$

In the following paragraphs we present some calculations explicitly showing how to use the technique and helping understanding why it is correct.

Full Oracles, Additional Details In this section we show detailed calculations of identities claimed in Section 6.2.2.1. First we analyze the Phase Oracle, introduced in Equation (6.38). We can check by direct calculation that this yields the standard Phase Oracle,

$$\mathsf{PhO}_{\mathfrak{U}}|x,\eta\rangle_{XY}|\mathbf{f}\rangle_F = (-1)^{\eta\cdot\mathbf{f}(x)}|x,\eta\rangle_{XY}|\mathbf{f}\rangle_F. \tag{6.41}$$

Including the full initial state of the oracle register, we calculate

$$\begin{aligned}
&\mathsf{PhO}_{\mathfrak{U}}|x,\eta\rangle_{XY}\frac{1}{\sqrt{|\mathcal{F}|}}\sum_{\mathbf{f}\in\mathcal{F}}|\mathbf{f}\rangle_F\\
&= (\mathbb{1}_m^X\otimes\mathsf{HT}_n^Y)\otimes\mathbb{1}_{n2^m}^F\mathsf{StO}_{\mathfrak{U}}|x\rangle_X\frac{1}{\sqrt{2^n}}\sum_y(-1)^{\eta\cdot y}|y\rangle_Y\frac{1}{\sqrt{|\mathcal{F}|}}\sum_{\mathbf{f}\in\mathcal{F}}|\mathbf{f}\rangle_F && (6.42)\\
&= (\mathbb{1}_m^X\otimes\mathsf{HT}_n^Y)\otimes\mathbb{1}_{n2^m}^F|x\rangle_X\frac{1}{\sqrt{2^n}}\sum_y\sum_{\mathbf{f}\in\mathcal{F}}(-1)^{\eta\cdot y}|y\oplus\mathbf{f}(x)\rangle_Y\frac{1}{\sqrt{|\mathcal{F}|}}|\mathbf{f}\rangle_F && (6.43)\\
&= \frac{1}{\sqrt{|\mathcal{F}|}}\sum_{\mathbf{f}\in\mathcal{F}}|x\rangle_X\sum_\zeta\underbrace{\frac{1}{2^n}\sum_y(-1)^{\eta\cdot y}(-1)^{(y\oplus\mathbf{f}(x))\cdot\zeta}}_{=\delta(\eta,\zeta)(-1)^{\zeta\cdot\mathbf{f}(x)}}|\zeta\rangle_Y|\mathbf{f}\rangle_F && (6.44)\\
&= \frac{1}{\sqrt{|\mathcal{F}|}}\sum_{\mathbf{f}\in\mathcal{F}}(-1)^{\eta\cdot f(x)}|x\rangle_X|\eta\rangle_Y|\mathbf{f}\rangle_F. && (6.45)
\end{aligned}$$

Applying the Hadamard transform also to register F will give us the Fourier Oracle. In the following calculation we denote acting on register F with $\mathsf{HT}_{n2^m}^{\otimes 2^m}$ by $\mathsf{HT}_{n2^m}^F$.

$$\begin{aligned}
&\mathsf{HT}_{n2^m}^F\circ\mathsf{PhO}_{\mathfrak{U}}\circ\mathsf{HT}_{n2^m}^F|x,\eta\rangle_{XY}|0^{2^m n}\rangle_F = \mathsf{HT}_{n2^m}^F\frac{1}{\sqrt{|\mathcal{F}|}}\sum_{\mathbf{f}\in\mathcal{F}}(-1)^{\eta\cdot\mathbf{f}(x)}|x,\eta\rangle|\mathbf{f}\rangle_F\\
&= \frac{1}{|\mathcal{F}|}\sum_{\phi,\mathbf{f}}(-1)^{\phi\cdot\mathbf{f}}(-1)^{\eta\cdot\mathbf{f}(x)}|x,\eta\rangle|\phi\rangle_F\\
&= \sum_\phi\underbrace{\frac{1}{2^{n(2^m-1)}}\sum_{\mathbf{f}(x'\neq x)}(-1)^{\phi_{x'}\cdot\mathbf{f}(x')}}_{=\delta(\phi_{x'},0^n)}\underbrace{\frac{1}{2^n}\sum_{\mathbf{f}(x)}(-1)^{\phi_x\cdot\mathbf{f}(x)}(-1)^{\eta\cdot\mathbf{f}(x)}}_{=\delta(\phi_x,\eta)}|x,\eta\rangle|\phi\rangle_F\\
&= |x,\eta\rangle|0^{2^m n}\oplus\chi_{x,\eta}\rangle && (6.46)
\end{aligned}$$

where we write $\mathbf{f}(x)$ and ϕ_x to denote the x-th row of the truth table $\mathbf{f}$ and ϕ respectively.

Compressed Oracles, Additional Details Let us state the input-output behavior of the compressed oracle $\mathsf{CFO}_{\mathfrak{U}}$ for uniform distributions. The input-output behavior of $\mathsf{CFO}_{\mathfrak{U}}$ on basis states is given by the following equation, x_r is the smallest $x_i \in D^X$ such that $x_r \geq x$ and $|\psi_{r-1}\rangle := |x_1, \eta_1\rangle_{D_1} \cdots |x_{r-1}, \eta_{r-1}\rangle_{D_{r-1}}$:

$$\mathsf{CFO}_{\mathfrak{U}}|x,\eta\rangle_{XY}|x_1,\eta_1\rangle_{D_1}\cdots|x_{q-1},\eta_{q-1}\rangle_{D_{q-1}}|\bot,0^n\rangle_{D_q} = |x,\eta\rangle_{XY}|\psi_{r-1}\rangle$$
$$\otimes\begin{cases} |x_r,\eta_r\rangle_{D_r}\cdots|x_{q-1},\eta_{q-1}\rangle_{D_{q-1}}|\bot,0^n\rangle_{D_q} & \text{if } \eta = 0^n,\\ |x,\eta\rangle_{D_r}|x_r,\eta_r\rangle_{D_{r+1}}\cdots|x_{q-1},\eta_{q-1}\rangle_{D_q} & \text{if } \eta \neq 0^n, x \neq x_r,\\ |x_r,\eta_r\oplus\eta\rangle_{D_r}\cdots|x_{q-1},\eta_{q-1}\rangle_{D_{q-1}}|\bot,0^n\rangle_{D_q} & \text{if } \eta \neq 0^n, x = x_r,\\ & \quad \eta \neq \eta_r,\\ |x_{r+1},\eta_{r+1}\rangle_{D_r}\cdots|x_{q-1},\eta_{q-1}\rangle_{D_{q-2}}|\bot,0^n\rangle_{D_{q-1}}|\bot,0^n\rangle_{D_q} & \text{if } \eta \neq 0^n, x = x_r,\\ & \quad \eta = \eta_r. \end{cases} \tag{6.47}$$

In the following let us change the picture of the compressed oracle to see how the Compressed Standard Oracle and Compressed Phase Oracle act on basis states. Let us begin with the Phase Oracle, given by the Hadamard transform of the oracle database

$$\mathsf{CPhO}_{\mathfrak{U}} := \left(\mathbb{1}_{n+m} \otimes \mathsf{HT}_n^{D^Y}\right) \circ \mathsf{CFO}_{\mathfrak{U}} \circ \left(\mathbb{1}_{n+m} \otimes \mathsf{HT}_n^{D^Y}\right), \tag{6.48}$$

where by $\mathsf{HT}_n^{D^Y}$ we denote transforming just the Y registers of the database: $\mathsf{HT}_n^{D^Y} := (\mathbb{1}_m \otimes \mathsf{HT}_n)^{\otimes q}$. Let us calculate the outcome of applying CPhO to a state for the first time, for simplicity we omit all but the first register of D

$$\mathsf{CPhO}_{\mathfrak{U}}|x,\eta\rangle_{XY}\frac{1}{\sqrt{2^n}}\sum_{z\in\{0,1\}^n}|\bot,z\rangle_D = \left(\mathbb{1}_{n+m}\otimes\mathsf{HT}_n^{D^Y}\right)\circ\mathsf{CFO}_{\mathfrak{U}}|x,\eta\rangle_{XY}|\bot,0^n\rangle_D \tag{6.49}$$

$$= \left(\mathbb{1}_{n+m}\otimes\mathsf{HT}_n^{D^Y}\right)\left((1-\delta(\eta,0^n))|x,\eta\rangle_{XY}|x,\eta\rangle_D + \delta(\eta,0^n)|x,\eta\rangle_{XY}|\bot,0^n\rangle_D\right) \tag{6.50}$$

$$= \frac{1}{\sqrt{2^n}}\sum_{z\in\{0,1\}^n}\left((1-\delta(\eta,0^n))(-1)^{\eta\cdot z}|x,\eta\rangle_{XY}|x,z\rangle_D + \delta(\eta,0^n)|x,0^n\rangle_{XY}|\bot,z\rangle_D\right). \tag{6.51}$$

If we defined the Compressed Phase Oracle from scratch we might be tempted to omit the coherent deletion of $\eta = 0^n$. The following attack shows that this would brake the correctness of the compressed oracles: The adversary inputs the equal superposition in the X register $\frac{1}{\sqrt{2^m}}\sum_x|x,0^n\rangle_{XY}$, after interacting with the regular $\mathsf{CPhO}_{\mathfrak{U}}$ the state after a single query is

$$\frac{1}{\sqrt{2^m}}\sum_x|x,0^n\rangle_{XY} \overset{\mathsf{CPhO}_{\mathfrak{U}}}{\mapsto} \frac{1}{\sqrt{2^m}}\sum_x|x,0^n\rangle_{XY}\frac{1}{\sqrt{2^n}}\sum_z|\bot,z\rangle_D, \tag{6.52}$$

but with a modified oracle that does not take care of this deleting, simply omits the term with $\delta(\eta,0^n)$, let us call it $\mathsf{CPhO}'_{\mathfrak{U}}$, the resulting state is

$$\frac{1}{\sqrt{2^m}}\sum_x|x,0^n\rangle_{XY} \overset{\mathsf{CPhO}'_{\mathfrak{U}}}{\mapsto} \frac{1}{\sqrt{2^m}}\sum_x|x,0^n\rangle_{XY}\frac{1}{\sqrt{2^n}}\sum_z|x,z\rangle_D. \tag{6.53}$$

Performing a measurement of the X register in the Hadamard basis distinguishes the two states with probability $1 - \frac{1}{2^m}$.

Let us inspect the state after making two queries to the Compressed Phase Oracle

$$\begin{aligned}
&\mathsf{CPhO}_{\mathfrak{U}}|x_2, \eta_2\rangle_{X_2Y_2}\mathsf{CPhO}_{\mathfrak{U}}|x_1, \eta_1\rangle_{X_1Y_1}\frac{1}{2^n}\sum_{z_1,z_2\in\{0,1\}^n}|\bot, z_1\rangle_{D_1}|\bot, z_2\rangle_{D_2} \\
&= |x_2, \eta_2\rangle|x_1, \eta_1\rangle\frac{1}{2^n}\sum_{z_1,z_2}\Bigg((-1)^{\eta_1\cdot z_1}\delta(\eta_2, 0^n)(1-\delta(\eta_1, 0^n))\underbrace{|x_1, z_1\rangle_{F_1}|\bot, z_2\rangle_{F_2}}_{=|\psi^{\mathrm{NOT}})} \\
&+ \delta(\eta_2, 0^n)\delta(\eta_1, 0^n)\underbrace{|\bot, z_1\rangle_{F_1}|\bot, z_2\rangle_{F_2}}_{=|\psi^{\mathrm{NOT}})} \\
&+ (-1)^{\eta_2\cdot z_1}(1-\delta(\eta_2, 0^n))\delta(\eta_1, 0^n)\underbrace{|x_2, z_1\rangle_{F_1}|\bot, z_2\rangle_{F_2}}_{=|\psi^{\mathrm{ADD}})} \\
&+ (-1)^{\eta_1\cdot z_1}(-1)^{\eta_2\cdot z_2}(1-\delta(\eta_2, 0^n))(1-\delta(x_1, x_2))(1-\delta(\eta_1, 0^n))\underbrace{|x_1, z_1\rangle_{F_1}|x_2, z_2\rangle_{F_2}}_{=|\psi^{\mathrm{ADD}})} \\
&+ (1-\delta(\eta_2, 0^n))\delta(x_1, x_2)\delta(\eta_1, \eta_2)(1-\delta(\eta_1, 0^n))\underbrace{|\bot, z_1\rangle_{F_1}|\bot, z_2\rangle_{F_2}}_{=|\psi^{\mathrm{REM}})} \\
&+ (1-\delta(\eta_2, 0^n))\delta(x_1, x_2)(1-\delta(\eta_1, \eta_2))(1-\delta(\eta_1, 0^n)) \\
&\cdot(-1)^{(\eta_1\oplus\eta_2)\cdot z_1}\underbrace{|x_1, z_1\rangle_{F_1}|\bot, z_2\rangle_{F_2}}_{=|\psi^{\mathrm{UPD}})}\Bigg),
\end{aligned} \tag{6.54}$$

where by the superscripts we denote the operation performed by $\mathsf{CPhO}_{\mathfrak{U}}$ on the compressed database. By ADD we denote adding a new pair (x, η), by UPD changing the Y register of an already stored database entry, REM signifies removal of a database entry, and NOT stands for doing nothing, that happens if the queried $\eta = 0^n$.

Let us discuss the Compressed Standard Oracle. We know that it is the Hadamard transform of the adversary's register followed by $\mathsf{CPhO}_{\mathfrak{U}}$

$$\mathsf{CStO}_{\mathfrak{U}} = \mathbb{1}_m \otimes \mathsf{HT}_n^Y \circ \mathsf{CPhO}_{\mathfrak{U}} \circ \mathbb{1}_m \otimes \mathsf{HT}_n^Y. \tag{6.55}$$

Let us present the action of CStO in the first query of the adversary

$$\begin{aligned}
&\mathsf{CStO}_{\mathfrak{U}}|x, y\rangle_{XY}\frac{1}{\sqrt{2^n}}\sum_{z\in\{0,1\}^n}|\bot, z\rangle_D \\
&= \mathbb{1}_m \otimes \mathsf{HT}_n^Y \circ \mathsf{CPhO}_{\mathfrak{U}}\frac{1}{\sqrt{2^n}}\sum_{\eta\in\{0,1\}^n}(-1)^{\eta\cdot y}|x, \eta\rangle_{XY}\frac{1}{\sqrt{2^n}}\sum_{z\in\{0,1\}^n}|\bot, z\rangle_D \qquad (6.56) \\
&= \mathbb{1}_m \otimes \mathsf{HT}_n^Y\frac{1}{\sqrt{2^n}}\sum_{\eta\in\{0,1\}^n}\frac{1}{\sqrt{2^n}}\sum_{z\in\{0,1\}^n}(-1)^{\eta\cdot y}
\end{aligned}$$

$$\left((1-\delta(\eta,0^n))(-1)^{\eta\cdot z}|x,\eta\rangle_{XY}|x,z\rangle_D+\delta(\eta,0^n)|x,0^n\rangle_{XY}|\perp,z\rangle_D\right) \tag{6.57}$$

$$=\frac{1}{2^n}\sum_{y',\eta}\frac{1}{\sqrt{2^n}}\sum_z(-1)^{\eta\cdot y}(-1)^{y'\cdot\eta}\Bigg((1-\delta(\eta,0^n))(-1)^{\eta\cdot z}|x,y'\rangle_{XY}|x,z\rangle_D$$

$$+\delta(\eta,0^n)|x,y'\rangle_{XY}|\perp,z\rangle_D\Bigg) \tag{6.58}$$

$$=\sum_{y'}\frac{1}{\sqrt{2^n}}\sum_z\underbrace{\frac{1}{2^n}\sum_{\eta\neq 0}(-1)^{\eta\cdot y}(-1)^{y'\cdot\eta}(-1)^{\eta\cdot z}}_{=\delta(y',y\oplus z)-\frac{1}{2^n}}|x,y'\rangle_{XY}|x,z\rangle_D$$

$$+\sum_{y'}\frac{1}{\sqrt{2^n}}\sum_z\frac{1}{2^n}|x,y'\rangle_{XY}|\perp,z\rangle_D \tag{6.59}$$

$$=\frac{1}{\sqrt{2^n}}\sum_z\Bigg(|x,y\oplus z\rangle_{XY}|x,z\rangle_D$$

$$-\frac{1}{2^n}\sum_{y'}|x,y'\rangle_{XY}|x,z\rangle_D+\frac{1}{2^n}\sum_{y'}|x,y'\rangle_{XY}|\perp,z\rangle_D\Bigg). \tag{6.60}$$

We would like to note that a similar calculation and resulting state is presented in [HI19].

6.2.2.2 Example Non-Uniform Distributions

Let us say we want to efficiently simulate a quantum oracle for a random function $\mathbf{h}:\{0,1\}^m\to\{0,1\}$, such that $\mathbf{h}(x)=1$ with probability λ. Then the adding function of the corresponding compressed oracle is $\forall x\in\{0,1\}^m$:

$$\mathsf{Samp}_\lambda(x):=\begin{pmatrix}\sqrt{1-\lambda} & \sqrt{\lambda}\\ \sqrt{\lambda} & -\sqrt{1-\lambda}\end{pmatrix}, \tag{6.61}$$

independent from any previous queries. This observation comes in useful in tasks like search in a sparse database.

6.2.2.3 Multiple Quantum Interfaces

We model access to multiple quantum oracles at once by specifying a special quantum register I. This register holds information about the particular interface that is queried. Note that this setup allows the adversary to make a superposition of queries to different oracles. If however one would want to make the interface register classical, we can just perform a standard basis measurement of I.

In the context of compressed oracles the multiple interfaces hold different databases. Of course if oracles are somehow related then so are the databases.

6.3 One-way to Hiding Lemma for Compressed Oracles

The fundamental game-playing lemma, Lemma 2.22, is a very powerful tool in proofs that include a random oracle. A common use of the framework is to reprogram the random oracle in a useful way. The fundamental lemma gives us a simple way of calculating how much the reprogramming costs in terms of the adversary's advantage—the difference between probabilities of A outputting 1 when interacting with one game or the other. The lemma that provides a counterpart to Lemma 2.22 in this context valid for quantum accessible oracles is the *One-Way to Hiding* (O2H) Lemma first introduced by Unruh in [Unr14].

In this section we generalize the O2H lemma to work with the compressed-oracle technique. The oracle register in this technique is a superposition over databases of input-output pairs. A relation on a database is a specific set of databases that fulfill some requirement, e.g., contains a collision (two entries with distinct inputs and the same output). The O2H lemma, as stated in [AHU19], works with punctured oracles, these are quantum oracles that include a binary measurement after every query. After introducing the notion of relations on databases we bring the concept of punctured oracles to the compressed-oracles technique. Punctured compressed oracles involve measurements on superpositions of databases. These measurements allow to analyze adversaries that had access to oracles that e.g. never output colliding outputs. This is a very useful situation, considering how often we lazy-sample functions in cryptographic proofs and then want to focus on some transcripts of input-output pairs. Our version of the O2H lemma provides a bound on the distinguishing advantage between an oracle that is not punctured and an oracle that is. The bound in the O2H lemma is stated in terms of the probability of any measurement in the punctured oracle succeeding, i.e., finding a database in the oracle register that fulfills the relation we discuss. The strength of our result lies in how versatile the new O2H lemma is, moreover the proof of the lemma is almost the same as the one in [AHU19].

In the original statement of the O2H lemma, the main idea is that there is a marked subset of inputs to the random oracle H, and an adversary tries to distinguish the situation in which she interacts with the normal oracle from an interaction with an oracle G that differs only on this set. The lemma states a bound for the distinguishing advantage which depends on the probability of an external algorithm measuring the input register of the adversary and seeing an element of the marked set. This probability is usually small, for random marked sets.

Recently this technique was generalized by Ambainis, Hamburg, and Unruh in [AHU19]. The main technical idea introduced by the generalized O2H lemma is to exchange the oracle G with a so-called *punctured oracle* that measures the input of the adversary after every query. The bound on the adversary's advantage is given by the probability of this measurement succeeding. This technique forms the link with the classical identical-until-bad games: we perform a binary measurement on the "bad" event and bound the advantage by the probability of observing this bad event.

In this chapter we present a generalization of this lemma that involves the use of compressed oracles. Our idea is to measure the database of the compressed oracle, which makes the lemma more versatile and easier to use for more general quantum oracles.

Below we state our generalized O2H lemmas. Most proofs of [AHU19] apply almost word by word so we just describe the differences and refer the reader to the original work.

6.3.1 Relations on Databases

The key concept we use are relations on the database of the compressed oracle.

Definition 6.4 (Relation R on D)**.** *Let D be a database of size at most q pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$. We call a subset*[6] $R \subseteq \mathcal{X} \times \bigcup_{t\in[q+1]} (\mathcal{X} \times \mathcal{Y})^t$ *a* relation R on D.

The set $\mathcal{X}$ that precedes the set of databases of different sizes corresponds to the adversary's input register. Including this set in Definition 6.4 allow us to consider relations being subsets of inputs, that is how we recover punctured oracles from [AHU19]. Usually though, we focus on relations limited to the actual database stored in the oracle registers. We provide two important examples for relations, the zero-preimage and the collision relation. We only consider preimages of zero so we call the former relation just the preimage relation. It is satisfied when the output of the oracle is 0:

$$R_{\text{preim}} := \{((x_1, y_1), \cdots, (x_s, y_s)) \in \bigcup_{t\in[q+1]} (\mathcal{X} \times \mathcal{Y})^t : \exists i : y_i = 0\}. \tag{6.62}$$

Note that we omit the set of adversary's inputs $\mathcal{X}$, technically the definition should be $\mathcal{X} \times R_{\text{preim}}$, we leave out $\mathcal{X}$, though, for the sake of notation. The collision relation is satisfied when there are two outputs in D that are the same:

$$R_{\text{coll}} := \Bigg\{((x_1, y_1), \ldots, (x_s, y_s)) \in \bigcup_{t\in[q+1]} (\mathcal{X} \times \mathcal{Y})^t : \\ \exists_{i,j}\ i \neq j, x_i \neq x_j, y_i = y_j \Bigg\}. \tag{6.63}$$

[6]Note that $[q + 1] := \{0, 1, \ldots, q\}$

Note however, that it is only reasonable to check if the *non-padding entries* are in R, omitting the $(\bot, 0)$ pairs at the end of D. If D is held in a quantum register, the relation R has a corresponding projective measurement J_R such that $\|\mathsf{J}_R|(x_1,y_1),\cdots,(x_q,y_q)\rangle_D\| = 1$ if and only if for some s it holds that $\big((x_1,y_1),\cdots,(x_s,y_s)\big) \in R$ and for the remaining $i > s$, the (x_i,y_i) are padding entries.

We also state an explicit algorithm to implement the measurement of a relation R, given that membership in R is efficiently decidable. To denote the single-bit membership decision by $(D \in R)$, the bit is 1 if and only if database D is in R. For the sake of notation we include the adversary's input in D, this is for when the relation depends on register X. To measure the relation we define a unitary V_R^{XDSJ} that XORs a bit $(D \in R)$ to register J; This unitary is controlled on registers S and D, the former holds the information about the size of the database and the latter the database itself. Algorithm 6.4 defines the measurement procedure of measuring R on quantum databases in the standard basis.

Algorithm 6.4 Measurement of a relation R

```
   input: Adversary's input and a database in the standard basis $|x\rangle_X|D\rangle_D$
   output: Outcome $j$ and post-measurement state $|D'\rangle_D$
1: Count in register $S$ the number of non-padding $(D^X \neq \bot)$ entries $s$ in $D$
2: Initialize a new qubit register $|0\rangle_J$
3: Apply $\mathsf{V}_R^{XDSJ}$ that XORs a bit $j := (D \in R)$ to register $J$
4: Uncompute register $S$, measure register $J$, output the outcome $j$
```

An important issue concerning measuring relations is the basis in which we store the quantum database. For the measurement to be meaningful it has to be done in the standard basis, so it is easiest to analyze $\mathsf{CStO}_{\mathfrak{D}}$ or $\mathsf{CPhO}_{\mathfrak{D}}$, defined by Equation (6.14).

While not directly relevant to our applications, we keep the generality of [AHU19] by introducing the notion of *query depth* as the number of sets of parallel queries an algorithm makes. We usually assume quantum algorithms make q quantum queries in total and d (as in "query depth") sequentially, but those queries in sequence may involve a number of parallel queries. A parallel query of width p to an oracle H involves p applications of H to p query registers. Note that if H is considered to be a compressed oracle, p-parallel queries are processed by sequentially applying the compressed oracle unitary p times.

First we define a compressed oracle H punctured on relation R, denoted by $\mathsf{H} \setminus R$.

Definition 6.5 (Punctured compressed oracle $\mathsf{H} \setminus R$)**.** *Let* H *be a compressed oracle and* R *a relation on its database. The punctured compressed oracle* $\mathsf{H} \setminus R$ *is equal to* H*,* *except that R is measured after every query as described in Algorithm 6.4. By* Find *we denote the event that R outputs* 1 *at least once among all queries.*

Full oracles can be punctured as well, the relation is then checked only on the queried entries of the function table—those queried entries need to be identified (like in $\mathsf{Dec}_{\mathfrak{D}}$ from Algorithm 6.3) prior to the measurement of R.

In many applications of punctured oracles we might want to apply $\mathsf{H} \setminus R$ only if some condition is fulfilled. Moreover, this condition might be quantum—in other words we control $\mathsf{H} \setminus R$ on some quantum register. To avoid the situation of a measurement being performed or not depending on a state of a quantum register—which is not permitted by quantum mechanics—we propose the following solution: We postpone the measurement to the end of the quantum query. Namely, we omit the measurement of register J in Algorithm 6.4 and perform it at the end of the compressed-oracle algorithm. After the measurement we can uncompute the outcome register J. We are not changing notation and implicitly assume the postponement of puncturing.

Similarly we define the above notions for a pair of databases.

Definition 6.6 (Relation R on (D_1, D_2))**.** *Let (D_1, D_2) be two databases each of size at most q: Database D_1 of pairs $(x, y) \in \mathcal{X}_1 \times \mathcal{Y}_1$ and database D_2 of pairs $(x, y) \in \mathcal{X}_2 \times \mathcal{Y}_2$. A* relation R on (D_1, D_2) *is a subset*

$$R \subseteq (\mathcal{X}_1 \times \mathcal{X}_2) \times \bigcup_{t_1 \in [q+1]} (\mathcal{X}_1 \times \mathcal{Y}_1)^{t_1} \times \bigcup_{t_2 \in [q+1]} (\mathcal{X}_2 \times \mathcal{Y}_2)^{t_2} . \qquad (6.64)$$

The measurement of relations defined on pairs of databases is done in the same way as in Algorithm 6.4 with the difference that registers D and S consist of two registers: D_1 and D_2 and S_1 and S_2 respectively. The result of the measurement is still a single bit stating whether $(D_1, D_2) \in R$.

The above discussion about relations defined on pairs of databases can be easily extended to multiple databases.

6.3.2 One-way-to-Hiding Lemma

Using the definitions from the previous sections we can prove a generalization of Theorem 1 from [AHU19].

Let us also comment on the differences of the O2H lemma in [AHU19] and in this thesis. The main difference is that in our generalization we no longer focus solely (we can recover the original O2H lemma though) on the adversary's inputs but also treat the outputs of the oracle. Function outputs are also important in [AHU19], but the oracle is not lazy sampled, there they pick a subset of the domain such that e.g. the output is 0 and then puncture on inputs in this random set. We use lazy sampled functions and puncture on databases, so functions defined only on the queried inputs. In addition, defining the puncturing operation on the compressed oracle-database is more expressive, as it allows puncturing conditions depending on more than one input-output pair, hence allowing us to treat a larger class of relations.

Theorem 6.7 (Compressed oracle O2H)**.** *Let R_1 and R_2 be relations on the database (or multiple databases) of a quantum oracle* H *(possibly combining multiple interfaces). Let z be a random string. R_1, R_2, and z may have arbitrary joint distribution. Let* A *be an oracle algorithm of query depth d, then*

$$\begin{aligned}&\left|\mathbb{P}\left[b=1 : b \leftarrow \mathrm{A}^{\mathsf{H}\setminus R_1}(z)\right] - \mathbb{P}\left[b=1 : b \leftarrow \mathrm{A}^{\mathsf{H}\setminus R_1\cup R_2}(z)\right]\right| \\ &\leq \sqrt{(d+1)\mathbb{P}[\mathsf{Find} : \mathrm{A}^{\mathsf{H}\setminus R_1\cup R_2}(z)]}, \text{ and} \qquad (6.65)\\ &\left|\sqrt{\mathbb{P}[b=1 : b \leftarrow \mathrm{A}^{\mathsf{H}\setminus R_1}(z)]} - \sqrt{\mathbb{P}[b=1 : b \leftarrow \mathrm{A}^{\mathsf{H}\setminus R_1\cup R_2}(z)}\right| \\ &\leq \sqrt{(d+1)\mathbb{P}[\mathsf{Find} : \mathrm{A}^{\mathsf{H}\setminus R_1\cup R_2}(z)]}, \qquad (6.66)\end{aligned}$$

where Find *is the event that measuring $R_1 \cup R_2$ succeeds.*

Proof. The proof works almost the same as the proof of Theorem 1 of [AHU19]. Let us state the analog of Lemma 5 from [AHU19]. In the following we write $R := R_1 \cup R_2$.

For the following lemma let us first define two algorithms. Let $\mathrm{A}^{\mathsf{H}}(z)$ be a unitary quantum algorithm with oracle access to H with query depth d. Let Q denote the quantum register of A and D the database of the compressed oracle H. We also need a "query log" register L consisting of d qubits.

Let $\mathrm{B}^{\mathsf{H},R}(z)$ be a unitary quantum algorithm acting on registers Q and L and having oracle access to H. First we define the following unitary

$$\mathsf{V}_{R,i}|D\rangle_D|l_1,l_2,\dots,l_d\rangle_L := \begin{cases}|D\rangle_D|l_1,l_2,\dots,l_d\rangle_L & \text{if } D \notin R\\ |D\rangle_D|l_1,\dots,l_i\oplus 1,\dots,l_d\rangle_L & \text{if } D \in R\end{cases}, \qquad (6.67)$$

where $R(|D\rangle_D)$ denotes the outcome of the projective binary measurement on D. The unitary exists for all relations. One can just coherently compute $R(D)$ into an auxiliary register, apply CNOT from that register to L_i and then uncompute $R(D)$. If the relation is efficiently computable, then so is the unitary. We define $\mathrm{B}^{\mathsf{H},R}(z)$ as:

- Initialize the register L with $|0^d\rangle$.
- Perform all operations that $\mathrm{A}^{\mathsf{H}}(z)$ does.
- For all i, after the i-th query of A apply the unitary $\mathsf{V}_{R,i}$ to registers D, L.

Let $|\Psi_{\mathrm{A}}\rangle$ denote the final state of $\mathrm{A}^{\mathsf{H}}(z)$, and $|\Psi_{\mathrm{B}}\rangle$ the final state of $\mathrm{B}^{\mathsf{H},R}(z)$. Let $\tilde{P}_{\mathsf{find}}$ be the probability that a measurement of L in the computational basis in the state $|\Psi_{\mathrm{B}}\rangle$ returns $l \neq 0^d$, i.e. $\tilde{P}_{\mathsf{find}} := \left\|\mathbb{1}^{Q,D} \otimes (\mathbb{1}^L - |0^d\rangle_L\langle 0^d|)|\Psi_{\mathrm{B}}\rangle\right\|^2$.

To deal with relation R_1 we consider algorithms with all measurements postponed to the end of their operation; Instead of performing the actual measurement we save the outcome into a fresh quantum register—with V_R as in Algorithm 6.4, note that prior to the measurement this fresh register can hold a superposition. Moreover we postpone the measurement of the auxiliary register until the very end of the run of the quantum algorithm. The coherent evaluation of R_1 happens in both algorithms. In addition, the proof below does not make use of the particular form of the unitaries that are applied between the measurements of R_2, so the evaluation of R_1 can be absorbed into the compressed oracle unitary.

Lemma 6.8 (Compressed-oracle O2H for pure states)**.** *Fix a joint distribution for R_1, R_2, z. Consider the definitions of algorithms* A *and* B *and their quantum states, then*

$$\left\| |\Psi_{\mathrm{A}}\rangle \otimes |0^d\rangle_L - |\Psi_{\mathrm{B}}\rangle \right\|^2 \leq (d+1)\tilde{P}_{\mathrm{find}}. \tag{6.68}$$

Proof. This lemma can be proved in the same way as Lemma 5 of [AHU19]. Here we omit some details and highlight the most important observation of the proof.

First define $\mathrm{B}_{\mathrm{count}}$ that works in the same way as B but instead of storing L, the log of queries with D in relation, it keeps *count*—in register C—of how many times a query resulted in $R(|D\rangle_D) = 1$. The state that results from running $\mathrm{B}_{\mathrm{count}}$ is $|\Psi_{\mathrm{B}_{\mathrm{count}}}\rangle = \sum_{i=0}^{d} |\Psi^i_{\mathrm{B}_{\mathrm{count}}})|i\rangle_C$ and similarly $|\Psi_{\mathrm{B}}\rangle = \sum_{l\in\{0,1\}^d} |\Psi^l_{\mathrm{B}})|l\rangle_L$, where $|\Psi)$ denotes a sub-normalized state. We can observe that $|\Psi_{\mathrm{A}}\rangle = \sum_{i=0}^{d} |\Psi^i_{\mathrm{B}_{\mathrm{count}}})$. As $\tilde{P}_{\mathrm{find}}$ is the probability of measuring at least one bit in the register L of B, or counting at least one fulfilling of R in C, we have that $|\Psi^{0^d}_{\mathrm{B}}) = |\Psi^0_{\mathrm{B}_{\mathrm{count}}})$. From the definition we also have $\tilde{P}_{\mathrm{find}} = 1 - \left\| |\Psi^0_{\mathrm{B}_{\mathrm{count}}}) \right\|^2$. Using the above identities we can calculate the bound

$$\begin{aligned} \left\| |\Psi_{\mathrm{B}}\rangle - |\Psi_{\mathrm{A}}\rangle \otimes |0^d\rangle_L \right\|^2 &= \left\| \sum_{i=1}^{d} |\Psi^i_{\mathrm{B}_{\mathrm{count}}}) \right\|^2 + \tilde{P}_{\mathrm{find}} \overset{\triangle}{\leq} \left(\sum_{i=1}^{d} \left\| |\Psi^i_{\mathrm{B}_{\mathrm{count}}}) \right\| \right)^2 + \tilde{P}_{\mathrm{find}} \\ &\overset{\text{J-I}}{\leq} d \underbrace{\sum_{i=1}^{d} \left\| |\Psi^i_{\mathrm{B}_{\mathrm{count}}}) \right\|^2}_{=\tilde{P}_{\mathrm{find}}} + \tilde{P}_{\mathrm{find}} = (d+1)\tilde{P}_{\mathrm{find}}, \end{aligned} \tag{6.69}$$

where $\triangle$ denotes the triangle inequality and J-I denotes the Jensen's inequality. It is apparent that introducing $\mathrm{B}_{\mathrm{count}}$ gave us a more coarse-grained look at the initial algorithm B, resulting in a tighter bound. □

The rest of the proof of Theorem 6.7 follows the same reasoning as the proof of Lemma 6 in [AHU19] with the modifications shown in the above lemma. Using bounds on fidelity (Lemma 3 and Lemma 4 of [AHU19]) and monotonicity and joint concavity of fidelity (from Theorem 9.6 and Equation 9.95 of [NC10]) one can generalize the results to the case of arbitrary mixed states. □

The original Theorem 1 from [AHU19] can be recovered from Theorem 6.7 when we puncture on relations defined solely on $\mathcal{X}$ corresponding to the adversary's A^X register.

We continue by deriving an explicit formula for $\mathbb{P}[\text{Find}]$. Let A be a quantum algorithm with oracle access to H, making at most q quantum queries with depth d. Let R be a relation on the database of H and z an input to A. R and z can have any joint distribution. J_R is the projector from the measurement of R on D, $\mathsf{U}_i^{\mathsf{H}}$ is the i-th unitary performed by $\mathrm{A}^{\mathsf{H}\setminus R}$ together with a—possibly parallel—query to H, and $|\Psi_0\rangle$ is the initial state of A. Then we have the formula

$$\mathbb{P}[\text{Find} : \mathrm{A}^{\mathsf{H}\setminus R}(z)] = 1 - \left\| \left(\prod_{i=1}^{d} (\mathbb{1} - \mathsf{J}_R) \mathsf{U}_i^{\mathsf{H}} \right) |\Psi_0\rangle \right\|^2 . \tag{6.70}$$

Let us now discuss the notion of "identical-until-bad" games in the case of compressed oracles. For random oracles, the notion was introduced in [AHU19]. The definition is rather straightforward as H and G are considered identical until bad if they had the same outputs except for some marked set. When using compressed oracles, the outputs of H and G are quantum lazy-sampled, making the definition of what it means for two oracles to be identical until bad require more care. Here we state a definition that captures useful notions of identical-until-bad punctured oracles.

Definition 6.9 (Almost identical oracles)**.** *Let* H *and* G *be compressed oracles and* R_i*,* $i = 1, 2$ *relations on their databases. We call the oracles* $\mathsf{H} \setminus R_1$ *and* $\mathsf{G} \setminus R_2$ almost identical *if they are equal conditioned on the events* $\neg\text{Find}_1$ *and* $\neg\text{Find}_2$ *respectively, i.e. for any string* z *and any quantum algorithm* A

$$\mathbb{P}[b = 1 : b \leftarrow A^{\mathsf{H}\setminus R_1}(z) \mid \neg\text{Find}_1] = \mathbb{P}[b = 1 : b \leftarrow A^{\mathsf{G}\setminus R_2}(z) \mid \neg\text{Find}_2]. \tag{6.71}$$

Note that not punctured compressed oracles are a special case of punctured ones (for $R_1 = \emptyset$ or $R_2 = \emptyset$), so the above definition can be applied to a pair of oracles where one is punctured and one is not. We can prove the following bound on the adversary's advantage in distinguishing almost identical punctured oracles.

Lemma 6.10 (Distinguishing almost identical punctured oracles)**.** *If* $\mathsf{H} \setminus R_1$ *and* $\mathsf{G} \setminus R_2$ *are almost identical according to Def.6.9 then for any* $b \in \{0, 1\}$

$$\begin{aligned} &\left| \mathbb{P}[b = 1 : b \leftarrow A^{\mathsf{H}\setminus R_1}(z)] - \mathbb{P}[b = 1 : b \leftarrow A^{\mathsf{G}\setminus R_2}(z)] \right| \\ &\leq 2\mathbb{P}[\text{Find}_1 : A^{\mathsf{H}\setminus R_1}(z)] + 2\mathbb{P}[\text{Find}_2 : A^{\mathsf{G}\setminus R_2}(z)]. \end{aligned} \tag{6.72}$$

Proof. For the sake fo readability, in the following we omit the "$b = 1 :$" in the events we analyze the probability of. We bound

$$\left| \mathbb{P}[b \leftarrow A^{\mathsf{H}\setminus R_1}(z)] - \mathbb{P}[b \leftarrow A^{\mathsf{G}\setminus R_2}(z)] \right|$$

$$\overset{\text{Def. 6.9}}{=} \Big|\mathbb{P}[b \leftarrow A^{\mathsf{H}\setminus R_1}(z) \mid \neg\mathrm{Find}_1]\Big(\mathbb{P}[\neg\mathrm{Find}_1 : A^{\mathsf{H}\setminus R_1}(z)] - \mathbb{P}[\neg\mathrm{Find}_2 : A^{\mathsf{G}\setminus R_2}(z)]\Big)$$

$$+ \mathbb{P}[b \leftarrow A^{\mathsf{H}\setminus R_1}(z) \mid \mathrm{Find}_1]\mathbb{P}[\mathrm{Find}_1 : A^{\mathsf{H}\setminus R_1}(z)]$$

$$-\mathbb{P}[b \leftarrow A^{\mathsf{G}\setminus R_2}(z) \mid \mathrm{Find}_2]\mathbb{P}[\mathrm{Find}_2 : A^{\mathsf{G}\setminus R_2}(z)]\Big| \tag{6.73}$$

$$\overset{\triangle}{\leq} \left|\underbrace{\mathbb{P}[b \leftarrow A^{\mathsf{H}\setminus R_1}(z) \mid \neg\mathrm{Find}_1]}_{\leq 1}\underbrace{\Big(\mathbb{P}[\neg\mathrm{Find}_1 : A^{\mathsf{H}\setminus R_1}(z)] - \mathbb{P}[\neg\mathrm{Find}_2 : A^{\mathsf{G}\setminus R_2}(z)]\Big)}_{=\mathbb{P}[\mathrm{Find}_2 : A^{\mathsf{G}\setminus R_2}(z)] - \mathbb{P}[\mathrm{Find}_1 : A^{\mathsf{H}\setminus R_1}(z)]}\right|$$

$$+ \left|\underbrace{\mathbb{P}[b \leftarrow A^{\mathsf{H}\setminus R_1}(z) \mid \mathrm{Find}_1]}_{\leq 1}\mathbb{P}[\mathrm{Find}_1 : A^{\mathsf{H}\setminus R_1}(z)]\right|$$

$$+ \left|\underbrace{\mathbb{P}[b \leftarrow A^{\mathsf{G}\setminus R_2}(z) \mid \mathrm{Find}_2]}_{\leq 1}\mathbb{P}[\mathrm{Find}_2 : A^{\mathsf{G}\setminus R_2}(z)]\right| \tag{6.74}$$

$$\overset{\triangle}{\leq} 2\mathbb{P}[\mathrm{Find}_1 : A^{\mathsf{H}\setminus R_1}(z)] + 2\mathbb{P}[\mathrm{Find}_2 : A^{\mathsf{G}\setminus R_2}(z)], \tag{6.75}$$

where by $\triangle$ we denote the triangle inequality. □

Note that for $R_2 = \emptyset$, the above lemma is essentially a special case of the well known Gentle-Measurement Lemma of [Win99].

It is a fact of quantum mechanics that measurements disturb the state. Considering that, one might be curious if measuring the database does not disturb it too much. As an example, note that after a measurement of the collision relation, Equation (6.63), the database does not necessarily consist of only non-Fourier-0 entries. Even though this is true, if the disturbance of the oracle is low enough, then the adversary will not notice it. This is exactly the case of the O2H lemma, the disturbance is low enough so the adversary does not notice any difference in the content of the oracle's output.

6.4 Bound on $\mathbb{P}$[Find]

We state a lemma giving a bound on the probability of Find for the uniform distribution over the sets $\{\mathbf{f}_1 : \mathcal{X}_1 \to \mathcal{Y}_1\}$ and $\{\mathbf{f}_2 : \mathcal{X}_2 \to \mathcal{Y}_2\}$ and for a general relation, possibly defined on multiple databases. We also allow for R to depend on an external random oracle R. In this proof we explicitly analyze adversaries with two interfaces H_1 and H_2 (for a discussion of multiple interfaces we refer to Section 6.2.2.3). We allow them to make queries to different interfaces in superposition. The register encoding the interface is I and holds $a \in \{1, 2\}$. In what follows we assume $\mathcal{Y}_1 = [N_1]$ and $\mathcal{Y}_2 = [N_2]$. By $\bar{a}$ we denote the index other that a, namely $\bar{a} = 3 - a$. Whenever we refer to a we mean by it the interface that is queried.

For $a \in \{1,2\}$ we write $\vec{x}_a \in (\mathcal{X}_a \times \{1,2\})^q$ to denote all the previous inputs asked by the adversary to H_a, we always consider queries x to be pairs of the query value and the interface. We mostly leave the interface part implicit. A vector with a fixed a has a fixed interface a in all tuples. (x, η, a) is the last query. Whenever $\vec{x}$ denotes queries, the vector is sorted in a rising fashion. We denote the outputs given to A by $\vec{y}_a := (y_1^a, \dots, y_{s_a}^a)$, where $y_i^a \in \mathcal{Y}_a \times \{1,2\}$ are pairs of values with interface, similarly to inputs. Vector of outputs is sorted according to the corresponding inputs. The set of all queries is $\vec{x} = \vec{x}_1 \cup \vec{x}_2$, similarly for $\vec{y}$. Databases are denoted as $D_a = \left((x_1^a, y_1^a), \dots, (x_{s_a}^a, y_{s_a}^a)\right)$. When we use set operations[7] on vectors we mean a set consisting of entries of $\vec{x}$, note that if there are no repetitions in $\vec{x}$, then there is no ambiguity.

In this section our primary subject are databases and their membership in the relation. To this end we define sets of good and bad outputs. For a relation R, the database $D = (D_1, D_2)$ that contains $\vec{x}_1$ and $\vec{x}_2$ of sizes s_1 and s_2 respectively, and $x \notin D_a^X$ we have

$$\mathcal{G}^R(\vec{x}_1, \vec{x}_2) := \left\{(D_1^Y(\vec{x}_1), D_2^Y(\vec{x}_2)) \in \mathcal{Y}_1^{s_1} \times \mathcal{Y}_2^{s_2} : (D_1, D_2) \notin R\right\}, \tag{6.76}$$

$$\mathcal{G}_{\bar{a}}^R(\vec{x}_1, \vec{x}_2 \mid D_a) := \left\{D_{\bar{a}}^Y(\vec{x}_{\bar{a}}) \in \mathcal{Y}_{\bar{a}}^{s_{\bar{a}}} : (D_1, D_2) \notin R\right\}, \tag{6.77}$$

$$\mathcal{B}_a^R(x \mid D) := \{y \in \mathcal{Y}_a : (D_a \cup \{(x,y)\}, D_{\bar{a}}) \in R\}. \tag{6.78}$$

The bad set defined above is the subset of the codomain of the sampled function corresponding to the new value bringing D to be in R. By $\mathcal{G}_a^R(\vec{x}_1, \vec{x}_2)$ we denote the part of $\mathcal{G}^R(\vec{x}_1, \vec{x}_2)$ corresponding to $D_a^Y(\vec{x}_a)$.

Our assumptions on R are the following: The relation does not depend on the adversary's input. The size of $\mathcal{G}^R(\vec{x}_1, \vec{x}_2)$ depends only on s_1 and s_2. When addressing the size of $\mathcal{G}$ we often write $\left|\mathcal{G}^R(s_1, s_2)\right|$. Moreover $\left|\mathcal{B}_a^R(x \mid D)\right|$ is the same for all $x \notin D_a^X$.

We also define a coefficient that gives the number of outputs that bring the database to R, defined as:

$$b_a^R(s_1, s_2) := \left|\mathcal{B}_a^R(x \mid D)\right|, \text{ where } x \notin D, D \notin R, \text{ and } |D_a| = s_a - 1, |D_{\bar{a}}| = s_{\bar{a}}, \tag{6.79}$$

as one can see from the definition (the argument of b_a^R does not include particular values in $\vec{x}_1$ and $\vec{x}_2$) above we use the assumption that $\left|\mathcal{B}_a^R(x \mid D)\right|$ is the same for all $x \notin D_a^X$. When making a query to database a we use the notation $\left|\mathcal{G}^R(s_a - 1, s_{\bar{a}})\right|$ for $\left|\mathcal{G}^R(s_1 - 1, s_2)\right|$ if $a = 1$ and $\left|\mathcal{G}^R(s_1, s_2 - 1)\right|$ if $a = 2$. Similarly we use $b_a^R(s_a + 1, s_{\bar{a}})$. An important identity that we will use later in this section is:

$$\left|\mathcal{G}^R(s_1, s_2)\right| = \left|\mathcal{G}^R(s_a - 1, s_{\bar{a}})\right| (|\mathcal{Y}_a| - b_a^R(s_1, s_2)). \tag{6.80}$$

[7]Like the union $\cup$, intersection $\cap$, or subtraction $\setminus$.

To get some intuition for the above equality, let us consider a database D of size $s_a - 1 + s_{\bar{a}}$ that is not in R. According to the definition from Equation (6.78), there are $b_a^R(s_1, s_2)$ outputs $y \in \mathcal{Y}_a$, such that $D \cup \{(x, y)\}$ for any $x \in \mathcal{X}_a$ that is not in D^X, is in R. As this holds for any value x, for every good database we have $|\mathcal{Y}_a| - b_a^R(s_1, s_2)$ good database with a single query added to D_a.

In general as in the good and bad sets, as well as the coefficient b, we omit the superscript R whenever the relation is clear from the context. As examples of b, consider relations on a single database, if the relation is R_{preim}, then $b(s) = 1$, there is just one value $y = 0$ that causes a fresh query to be in relation; For R_{coll} we have $b(s) = s - 1$, the new y can be any of the previously queried values to make D fulfill the relation.

Two sets important in our treatment of multiple databases are $\mathcal{H}_a^{\text{ADD}}(\vec{x}_1, \vec{x}_2, \vec{y}_a)$ and $\mathcal{H}_a^{\text{REM}}(\vec{x}_1, \vec{x}_2, \vec{y}_a)$. To properly define them we generalize the definition of the good set conditioned on a database:

$$\begin{aligned}&\mathcal{G}_{\bar{a}}(\vec{x}_a, \vec{x}_{\bar{a}} \mid \vec{y}_a)\\ &:= \left\{ D_{\bar{a}}^Y(\vec{x}_{\bar{a}}) \in \mathcal{Y}_{\bar{a}}^{s_{\bar{a}}} : \exists \vec{y}_a^* \in \mathcal{Y}_a^{s_a - |\vec{y}_a|}, D_a^Y(\vec{x}_a) := \vec{y}_a \cup \vec{y}_a^*, (D_1, D_2) \notin R \right\}. \end{aligned} \tag{6.81}$$

Intuitively speaking the above set is the set $\mathcal{G}_{\bar{a}}(\vec{x}_a, \vec{x}_{\bar{a}} \mid D_a)$ defined in Equation (6.77) with the difference that we do not specify all values in D_a^Y. Moreover the more entries are in $\vec{x}_a$ the more "restrictions" are on good $D_{\bar{a}}$, meaning the size of the good in principle gets smaller with $\vec{x}_a$ getting bigger. Then the two sets are defined as

$$\mathcal{H}_a^{\text{ADD}}(\vec{x}_1, \vec{x}_2, \vec{y}_a) := \mathcal{G}_{\bar{a}}(\vec{x}_1, \vec{x}_2 \mid \vec{y}_a) \setminus \mathcal{G}_{\bar{a}}(\vec{x}_a \cup \{x\}, \vec{x}_{\bar{a}} \mid \vec{y}_a), \tag{6.82}$$

$$\left|\mathcal{H}_a^{\text{ADD}}(s_1, s_2)\right| := \left|\mathcal{H}_a^{\text{ADD}}(\vec{x}_1, \vec{x}_2, \vec{y}_a)\right|, \tag{6.83}$$

and

$$\begin{aligned}&\mathcal{H}_a^{\text{REM}}(\vec{x}_1, \vec{x}_2, \vec{y}_a) := \mathcal{G}_{\bar{a}}(\vec{x}_a \setminus \{x\}, \vec{x}_{\bar{a}} \mid \vec{y}_a) \setminus \mathcal{G}_{\bar{a}}(\vec{x}_1, \vec{x}_2 \mid \vec{y}_a),\\ &\left|\mathcal{H}_a^{\text{REM}}(s_1, s_2)\right| := \left|\mathcal{H}_a^{\text{REM}}(\vec{x}_1, \vec{x}_2, \vec{y}_a)\right|. \end{aligned} \tag{6.84}$$

The intuition one should have for $\mathcal{H}_a^{\text{ADD}}(\vec{x}_1, \vec{x}_2, \vec{y}_a)$ and $\mathcal{H}_a^{\text{REM}}(\vec{x}_1, \vec{x}_2, \vec{y}_a)$ is that for the relations we discuss in this thesis, they are very small sets.

The assumption that is important for when R is defined on two databases is that if good outputs of H_1 depend on inputs to H_2, we never make a query to H_2 that automatically brings D to be in R. An example of such relation is $y_1 = x_2$ (outputs of H_1 equal to any input to H_2). For these relations it is trivial to fulfill them—by just querying one of the outputs of H_1 to H_2—so the oracles have to be constructed in a way that avoids this attack. By constructing we mean adding a quantum algorithm managing queries to different interfaces. We say that such *trivial attacks* are of concern when D_a^X interacts with $D_{\bar{a}}^Y$.

Below we state a lemma bounding the probability of Find, which gives great utility to the quantum game-playing framework. The result depends only on measurements performed on the database. The basis of the database matters, as we define the relation in a particular (standard) basis. Hence, this result works exactly the same for CStO.

Lemma 6.11. *Let* A *be a quantum adversary interacting with a compressed punctured oracle* $\mathsf{H} \setminus R$*, with* $\mathsf{H} = \mathsf{S}(\mathsf{H}_1, \mathsf{H}_2)$*, where* S *is any quantum algorithm that ensures that the trivial attacks (important when* D_a^X *interacts with* $D_{\bar{a}}^Y$*) are avoided,* $\mathsf{H}_1 = \mathsf{CPhO}_{\mathcal{Y}_1}$ *and* $\mathsf{H}_2 = \mathsf{CPhO}_{\mathcal{Y}_2}$*. Moreover* R *is a relation following Definition 6.6, such that*

1. $\left|\mathcal{G}^R(\vec{x}_1, \vec{x}_2)\right|$ *from Equation* (6.94) *depends only on* s_1 *and* s_2*,*
2. $\left|\mathcal{B}_a^R(x \mid D)\right|$ *from Equation* (6.78) *is the same for all* $x \notin D_a^X$*.*

Then the probability of Find *is bounded by:*

$$\begin{aligned}
\mathbb{P}\left[\text{Find} : \mathsf{A}[\mathsf{H} \setminus R]\right] \leq \sum_{i=1}^{q} \Bigg(\sum_{j=1}^{i-1} \max_{a\in\{1,2\}, s_1, s_2 \leq j-1} \Bigg(2\frac{b_a(s_1,s_2)}{N_a} \\
+ \frac{b_a(s_1,s_2)}{\sqrt{N_a(N_a - b_a(s_1,s_2))}} \sqrt{\frac{|\mathcal{G}_{\bar{a}}(s_a-1,s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}} \\
+ \frac{b_a(s_1,s_2)}{N_a} \sqrt{\frac{|\mathcal{G}_{\bar{a}}(s_a-1,s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|} - \left(\sqrt{\frac{|\mathcal{G}_{\bar{a}}(s_a-1,s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}} - 1\right)} \\
+ \frac{b_a(s_a+1,s_{\bar{a}})}{N_a} \sqrt{\frac{|\mathcal{G}_{\bar{a}}(s_a+1,s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|} - \left(\sqrt{\frac{|\mathcal{G}_{\bar{a}}(s_a+1,s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}} - 1\right)} \Bigg) \Bigg) \\
+ \max_{a\in\{1,2\}, s_1, s_2 \leq i-1} \Bigg(\sqrt{\frac{N_a - b_a(s_a+1,s_{\bar{a}})}{N_a}} \sqrt{\frac{|\mathcal{H}_a^{\mathrm{ADD}}(s_1,s_2)|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}} \\
+ \sqrt{\frac{b_a(s_a+1,s_{\bar{a}})}{N_a}} + \frac{b_a(s_1,s_2)^{3/2}}{N_a\sqrt{N_a - b_a(s_1,s_2)}} \\
+ \frac{b_a(s_1,s_2)}{\sqrt{N_a(N_a - b_a(s_1,s_2))}} \mathrm{sgn}\left(\left|\mathcal{H}_a^{\mathrm{REM}}(s_1,s_2)\right|\right) \sqrt{\frac{|\mathcal{G}_{\bar{a}}(s_a-1,s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}} \\
+ \sqrt{\frac{N_a - b_a(s_1,s_2)}{N_a}} \sqrt{\frac{|\mathcal{H}_a^{\mathrm{REM}}(s_1,s_2)|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}} + \frac{\sqrt{b_a(s_1,s_2)(N_a - b_a(s_1,s_2))}}{N_a} \Bigg) \Bigg)^2, \quad (6.85)
\end{aligned}$$

where $a \in \{1,2\}$*,* q *is the maximal number of queries made by* A*, and* sgn *is the sign function equal* 0 *whenever the argument is* 0*.*

6.4.1 Proof of Lemma 6.11

Proof.

6.4.1.1 Overview

We first provide a high level overview of the proof.

In Equation (6.92) we divide the probability of Find into a sum of probabilities of q sub-events. Each sub-event is Find after i queries conditioned on $\neg$Find. The next task is to find a bound for the probability of every sub-event.

To achieve this goal we define the *good* state $|\Psi_i^{\mathrm{Good}}\rangle$. This is an auxiliary state of the adversary and the oracle register that is easier to handle from the true state $|\Phi_i\rangle$ resulting from the interaction of A with the punctured $\mathsf{H} \setminus R$. We use this state by swapping the real state with it in the i-th step we introduced in the previous point. This swapping is presented in Equation (6.99).

As we see from Equation (6.100), to evaluate the error introduced by swapping the original state with the good state we need to inspect the good state after a single query. Calculating $\mathsf{H} \setminus R|\Psi_i^{\mathrm{Good}}\rangle$ and comparing it to $|\Psi_{i+1}^{\mathrm{Good}}\rangle$ is a rather complicated task that involves taking care of many details. We dedicate Section 6.4.1.5 to defining $\mathsf{H} \setminus R|\Psi_i^{\mathrm{Good}}\rangle$ and identifying the parts of the queried state that differ from $|\Psi_{i+1}^{\mathrm{Good}}\rangle$.

The final step of the proof consists of plugging in bounds on $\left\||\Psi_{i+1}^{\mathrm{Good}}\rangle - \mathsf{H} \setminus R|\Psi_i^{\mathrm{Good}}\rangle\right\|$ and on the probability that Find happens when the joint state is $|\Psi_i^{\mathrm{Good}}\rangle$. These bounds are calculated in sections 6.4.1.6 and 6.4.1.7 and stated in Lemmas 6.12 and 6.14. On a technical level we bound the norms of the parts of the state $\mathsf{H} \setminus R|\Psi_i^{\mathrm{Good}}\rangle$ that do not appear in $|\Psi_{i+1}^{\mathrm{Good}}\rangle$, we call these parts *errors*. The proof of Lemma 6.12 consists of two main parts, as the bound is a sum of two types of errors.

6.4.1.2 Introduction

We start the proof with a few definitions concerning compressed oracles. Some notions from Section 6.2 require to be stated a bit more concretely for our proof. The measurement that we apply after $\mathsf{CPhO}_{\mathcal{Y}}$, in line 4 of Algorithm 6.4 is

$$\mathsf{J}_R := \mathbb{1} \otimes |1\rangle_J\langle 1| \text{ and} \tag{6.86}$$

$$\bar{\mathsf{J}}_R := \mathbb{1} \otimes |0\rangle_J\langle 0| \tag{6.87}$$

is the projection to D being not in relation.

In the following we focus on the punctured oracle just prior to measurement J_R. A unitary that omits the last step of Algorithm 6.4 in $\mathsf{H} \setminus R$ acts on registers ADJ, we define it as

$$\mathsf{H} \setminus \mathsf{V}_R := \mathsf{Queries}^\dagger \circ \mathsf{V}_R \circ \mathsf{Queries} \circ \mathsf{H}, \tag{6.88}$$

where the unitary $\mathsf{Queries}$ counts the number of $x \neq \perp$ in D (line 1 and 4 in Algorithm 6.4) and V_R checks whether the queried values in registers D fulfill the relation R and saves the single bit answer to register J.

We proceed by rephrasing the definition of $\mathbb{P}[\text{Find} : \mathsf{A}[\mathsf{H} \setminus R]]$, after that we treat the part specific to our relation. We follow Equation (6.70) to analyze the probability of Find:

$$\mathbb{P}[\text{Find} : \mathsf{A}[\mathsf{H} \setminus R_{\text{preim}} \cup R_{\text{coll}}]] = 1 - \left\| \left(\prod_{i=q}^{1} \bar{\mathsf{J}}_R \mathsf{U}_i \mathsf{H} \setminus \mathsf{V}_R \right) |\Psi_0\rangle |0\rangle_J \right\|^2 \tag{6.89}$$

$$= 1 - \left\| \left(\prod_{i=q-1}^{1} \bar{\mathsf{J}}_R \mathsf{U}_i \mathsf{H} \setminus \mathsf{V}_R \right) |\Psi_0\rangle |0\rangle_J \right\|^2$$

$$+ \left\| \mathsf{J}_R \mathsf{U}_q \mathsf{H} \setminus \mathsf{V}_R \left(\prod_{i=q-1}^{1} \bar{\mathsf{J}}_R \mathsf{U}_i \mathsf{H} \setminus \mathsf{V}_R \right) |\Psi_0\rangle |0\rangle_J \right\|^2 = \cdots = \tag{6.90}$$

$$= \sum_{i=1}^{q} \left\| \mathsf{J}_R \mathsf{U}_i \mathsf{H} \setminus \mathsf{V}_R \underbrace{\left(\prod_{j=i-1}^{1} \bar{\mathsf{J}}_R \mathsf{U}_j \mathsf{H} \setminus \mathsf{V}_R \right) |\Psi_0\rangle |0\rangle_J}_{:= \mathsf{U}_{i-1} |\Phi_{i-1}\rangle} \right\|^2 \tag{6.91}$$

$$= \sum_{i=1}^{q} \left\| \mathsf{J}_R \mathsf{U}_i \mathsf{H} \setminus \mathsf{V}_R \mathsf{U}_{i-1} |\Phi_{i-1}\rangle \right\|^2, \tag{6.92}$$

where $|\Psi_0\rangle$ is the initial state of the adversary. Note that in the definition

$$|\Phi_{i-1}\rangle := \mathsf{U}_{i-1}^{\dagger} \left(\prod_{j=i-1}^{1} \bar{\mathsf{J}}_R \mathsf{U}_j \mathsf{CPhO}_{\mathcal{Y}} \setminus \mathsf{V}_R \right) |\Psi_0\rangle |0\rangle_J \tag{6.93}$$

we use[8] $[\mathsf{U}_{i-1}, \bar{\mathsf{J}}_R] = 0$. Here, the second and third equations follow from the fact that $\| |v\rangle \|^2 = \| \mathsf{P} |v\rangle \|^2 + \| (\mathbb{1} - \mathsf{P}) |v\rangle \|^2$ for all $|v\rangle$ and projectors P.

In what follows we analyze $\| \mathsf{J}_R \mathsf{U}_i \mathsf{H} \setminus \mathsf{V}_R \mathsf{U}_{i-1} |\Phi_{i-1}\rangle \|^2$. Our approach is to propose a state $|\Psi_{i-1,R}^{\text{Good}}\rangle |0\rangle_J$, close to the original $|\Phi_{i-1}\rangle$, for which bounding $\left\| \mathsf{J}_R \mathsf{U}_i \mathsf{H} \setminus \mathsf{V}_R \mathsf{U}_{i-1} |\Psi_{i-1,R}^{\text{Good}}\rangle |0\rangle_J \right\|^2$ is easy. The intuition behind $|\Psi_{i-1,R}^{\text{Good}}\rangle$ is to have a superposition over databases that are not in relation.

6.4.1.3 The good state

The state $|\Psi_{i,R}^{\text{Good}}\rangle_{AD}$ corresponds to the adversary's state just after the i-th query and before the application of U_i. The size of the database s_a depends on whether the new query x was added to, updated, or removed from the database, it

[8]The commutator of two operators (matrices) is defined as $[\mathsf{A}, \mathsf{B}] := \mathsf{AB} - \mathsf{BA}$.

equals $|\vec{x}_a \cup \{x\}|$, $|\vec{x}_a|$, or $|\vec{x}_a \setminus \{x\}|$ respectively. After i queries s_a can range from 0 to i and the joint state of A and the oracle can be a superposition over different database sizes. By $D(\perp)$ we denote the part of the database containing empty entries. The adversary's work register is denoted by A^W and its contents by $\psi(x, \eta, \vec{x}, \vec{\eta}, w)$, where w can be any value of finite size. We define the good state as:

$$\begin{aligned}
|\Psi^{\text{Good}}_{i,R}\rangle_{AD} := &\sum_{x,\eta,a,\vec{x},\vec{\eta},w} \alpha_{x,\eta,a,\vec{x},\vec{\eta},w}|x,\eta,a\rangle_{A^{XYI}}|\psi(x,\eta,a,\vec{x},\vec{\eta},w)\rangle_{A^W}\\
&\sum_{\vec{y}\in\mathcal{G}^R(\vec{x}_1,\vec{x}_2)} \frac{1}{\sqrt{|\mathcal{G}^R(s_1,s_2)|}}\omega_N^{\vec{\eta}_1\cdot\vec{y}_1}|(x^1_1,y^1_1),\dots,(x^1_{s_1},y^1_{s_1})\rangle_{D_1(\vec{x}_1)}\\
&\sum_{y_{s_1+1},\dots,y_q\in[N]} \frac{1}{\sqrt{N^{q-s_1}}}|(\perp,y_{s_1+1}),\dots,(\perp,y_q)\rangle_{D_1(\perp)}\\
&\omega_N^{\vec{\eta}_2\cdot\vec{y}_2}|(x^2_1,y^2_1),\dots,(x^2_{s_2},y^2_{s_2})\rangle_{D_2(\vec{x}_2)}\\
&\sum_{y_{s_2+1},\dots,y_q\in[N]} \frac{1}{\sqrt{N^{q-s_2}}}|(\perp,y_{s_2+1}),\dots,(\perp,y_q)\rangle_{D_2(\perp)}. \qquad (6.94)
\end{aligned}$$

In case we have added x to D_a, the full database D above contains (x, y^a_j). In the rest of the proof we omit the subscript R, however note that $|\Psi^{\text{Good}}_i\rangle$ does indeed depend on R.

Another way to define the good state is to consider the joint state of the adversary and the non-punctured oracle H just after the i-th query. The good state is then this state after a projection of register D with $\bar{\mathsf{J}}_R$. Normalization of the projected state comes from multiplying each branch corresponding to a given size of the database by an appropriate $\sqrt{\frac{N_1^{s_1}N_2^{s_2}}{|\mathcal{G}^R(s_1,s_2)|}}$ factor. The reason why the good state is normalized is that for a fixed set of queries we can think of definining it as A interacting with the normalized database register using PhO instead of CPhO. This intuition works for every branch of the superposition separately. Now combining all branches together also gives a normalized state, because they origin from a valid interaction of a unitary adversary with CPhO (as mentioned in the beginning of this section).

6.4.1.4 Final Bound

To calculate the probability of measuring R, Equation (6.92) implies

$$\mathbb{P}[\text{Find}] \le \sum_{i=1}^{q} \|\mathsf{J}_R\mathsf{U}_i\mathsf{H}\setminus\mathsf{V}_R\mathsf{U}_{i-1}|\Phi_{i-1}\rangle\|^2. \qquad (6.95)$$

We use the good state to bound the elements of the sum in the following way:

$$\|\mathsf{J}_R\mathsf{U}_i\mathsf{H}\setminus\mathsf{V}_R\mathsf{U}_{i-1}|\Phi_{i-1}\rangle\| \le \left\||\Phi_{i-1}\rangle - |\Psi^{\text{Good}}_{i-1}\rangle\right\| + \left\|\mathsf{J}_R\mathsf{U}_i\mathsf{H}\setminus\mathsf{V}_R\mathsf{U}_{i-1}|\Psi^{\text{Good}}_{i-1}\rangle\right\|. \qquad (6.96)$$

Next we bound the two norms in Equation (6.96). First we bound the distance of the good state from the state resulting from the interaction with the non-punctured oracle $|\Phi_i\rangle_{ADJ}$. We simplify this task with the following derivation:

$$\begin{aligned}
&\left\| |\Psi_i^{\text{Good}}\rangle_{AD}|0\rangle_J - |\Phi_i\rangle_{ADJ} \right\| \\
&= \left\| |\Psi_i^{\text{Good}}\rangle_{AD}|0\rangle_J - \bar{\mathsf{J}}_R\mathsf{H}\setminus \mathsf{V}_R\mathsf{U}_{i-1}|\Phi_{i-1}\rangle_{ADJ} \right\| && (6.97)\\
&\leq \left\| |\Psi_i^{\text{Good}}\rangle_{AD}|0\rangle_J - \bar{\mathsf{J}}_R\mathsf{H}\setminus \mathsf{V}_R\mathsf{U}_{i-1}|\Psi_{i-1}^{\text{Good}}\rangle_{AD}|0\rangle_J \right\| \\
&+ \left\| \bar{\mathsf{J}}_R\mathsf{H}\setminus \mathsf{V}_R\mathsf{U}_{i-1}|\Psi_{i-1}^{\text{Good}}\rangle_{AD}|0\rangle_J - \bar{\mathsf{J}}_R\mathsf{H}\setminus \mathsf{V}_R\mathsf{U}_{i-1}|\Phi_{i-1}\rangle_{ADJ} \right\| && (6.98)\\
&\leq \varepsilon_{\text{step}}(i) + \left\| |\Psi_{i-1}^{\text{Good}}\rangle_{AD}|0\rangle_J - |\Phi_{i-1}\rangle_{ADJ} \right\| \leq \sum_{j=1}^{i} \varepsilon_{\text{step}}(j), && (6.99)
\end{aligned}$$

where we use the triangle inequality and recursively get rid of all queries made by A. The definition of a single step is

$$\varepsilon_{\text{step}}(j) := \left\| |\Psi_j^{\text{Good}}\rangle_{AD}|0\rangle_J - \bar{\mathsf{J}}_R\mathsf{H}\setminus \mathsf{V}_R\mathsf{U}_{j-1}|\Psi_{j-1}^{\text{Good}}\rangle_{AD}|0\rangle_J \right\|_2 . \quad (6.100)$$

To calculate the bound on $\varepsilon_{\text{step}}(j)$ we first calculate how a query affects the good state. The full calculations are presented in Section 6.4.1.5. Using these findings we prove Lemma 6.12 that states a bound on the norm of the difference of the good and original states.

We define the second part in Equation (6.96) as

$$\varepsilon_{\text{Find}}(i) := \left\| \mathsf{J}_R\mathsf{U}_i\mathsf{H}\setminus \mathsf{V}_R\mathsf{U}_{i-1}|\Psi_{i-1}^{\text{Good}}\rangle \right\| . \quad (6.101)$$

Using the techniques developed to bound $\varepsilon_{\text{step}}(j)$, we bound $\varepsilon_{\text{Find}}(i)$ in Section 6.4.1.7 and state the bounds in Lemma 6.14.

The final bound is

$$\mathbb{P}\left[\text{Find} : \mathsf{A}[\mathsf{H}\setminus R]\right] \leq \sum_{i=1}^{q} \left(\sum_{j=1}^{i-1} \varepsilon_{\text{step}}(j) + \varepsilon_{\text{Find}}(i) \right)^2 , \quad (6.102)$$

with Lemma 6.12 and Lemma 6.14 we get the final bound. □

6.4.1.5 $|\Psi_{i-1}^{\text{Good}}\rangle$ after a query

To prove the main technical lemmas of this section we need to analyze how a single query to the oracle affects the good state.

To prove Lemma 6.12 we analyze how far apart the state $|\Psi_{i-1}^{\text{Good}}\rangle$ is after a query from $|\Psi_i^{\text{Good}}\rangle$. To achieve this goal we inspect in detail the state $\mathsf{H}\setminus \mathsf{V}_R\mathsf{U}_{i-1}|\Psi_{i-1}^{\text{Good}}\rangle_{AD}|0\rangle_J$. We distinguish different modes of operation: ADD when the queried x is added to D, UPD when x was already in D and is not removed from the database, REM when we remove x from D, and NOT where register A^Y is in state $|0\rangle$. These modes correspond to different branches of superposition in $\mathsf{H} \setminus \mathsf{V}_R \mathsf{U}_{i-1}|\Psi_{i-1}^{\text{Good}}\rangle_{AD}|0\rangle_J$. We write

$$\mathsf{U}_{i-1}|\Psi_{i-1}^{\text{Good}}\rangle_{AD} = |\xi_{i-1}(\text{ADD})\rangle + |\xi_{i-1}(\text{UPD})\rangle + |\xi_{i-1}(\text{REM})\rangle + |\xi_{i-1}(\text{NOT})\rangle \tag{6.103}$$

and analyze the action of $\mathsf{H} \setminus \mathsf{V}_R$ on the above states separately.

For $|\xi_{i-1}(\text{NOT})\rangle$ there is no change to the state. For $|\xi_{i-1}(\text{UPD})\rangle$ and $|\xi_{i-1}(\text{REM})\rangle$, we treat the updated x as the last one in D_a, this does not have to be true but it simplifies notation. Note that we want the corresponding y_{s_a} to depend on previous queries to H_a. This assumption is without loss of generality as there is no fixed order for $\sum_{\vec{y}_a}$ in Equation (6.94). The empty register is moved to the back of D, we do not write it out for simplicity but still consider it done.

After querying $|\Psi_{i-1}^{\text{Good}}\rangle|0\rangle_J$ we encounter states multiplied by $|0\rangle_J$ that do not appear in the definition of the good state and those multiplied by $|1\rangle_J$. We call these vectors *errors*. We mark the former errors by a superscript Bad and the latter with Find, note that indeed all branches of superposition that have $|1\rangle_J$ increase $\mathbb{P}\left[\text{Find}\right]$.

In general, a query to H_a can cause errors in D_a and $D_{\bar{a}}$. The former results from, e.g., adding a new entry to D_a; We sample a uniform entry and some values bring D_a to be in relation. The latter errors occur when the set of good outputs in $D_{\bar{a}}$ changes after we, e.g., add a new entry to D_a. The rule we follow is that $y_{s_a}^a$ is the last value sampled. The second rule is that all values can be sampled one by one, query by query. These rules imply that we can sample $\vec{y}_a$ first, then $\vec{y}_{\bar{a}}$, then $y_{s_a}^a$. This reasoning, however does not apply to relations that depend on inputs, so whenever contents of D^X (a or $\bar{a}$) changes we need to make up for it by changing the set we sample $\vec{y}_{\bar{a}}$ from.

Adding a new entry to a database results in setting the register corresponding to (x, a) to $\sum_{y_{s_a+1}^a \in [N_a]} \frac{1}{\sqrt{N_a}} \omega_{N_a}^{\eta y_{s_a+1}^a} |x, y_{s_a+1}^a\rangle$, just as expected from a phase oracle for the uniform distribution. As we mentioned earlier, there are errors in two databases, D_a and $D_{\bar{a}}$. First we go over the errors in D_a and leave $D_{\bar{a}}$ unchanged. In the equality that follows we single out all the branches of superposition that are not parts of $|\xi_i(\text{ADD})\rangle$:

$$\mathsf{H}|\xi_{i-1}(\text{ADD})\rangle = \sum_{x,\eta,a,\vec{x},\vec{\eta},w} \alpha_{x,\eta,a,\vec{x},\vec{\eta},w} |x, \eta, a\rangle_{A^{XYI}} |\psi(x, \eta, a, \vec{x}, \vec{\eta}, w)\rangle_{A^W}$$
$$\sum_{\vec{y}_a \in \mathcal{G}_a(\vec{x}_1, \vec{x}_2)} \frac{1}{\sqrt{|\mathcal{G}_a(s_1, s_2)|}} \omega_{N_a}^{\vec{\eta}_a \cdot \vec{y}_a} |(x_1^a, y_1^a), \ldots, (x_{s_a}^a, y_{s_a}^a)\rangle_{D_a(\vec{x}_a)}$$
$$\sum_{\vec{y}_{\bar{a}} \in \mathcal{G}_{\bar{a}}(\vec{x}_1, \vec{x}_2 | \vec{y}_a)} \frac{1}{\sqrt{|\mathcal{G}_{\bar{a}}(s_1, s_2)|}} \left(\frac{1}{\sqrt{N_a}} \sum_{y_{s_a+1}^a \notin \mathcal{B}_a(x|D(\vec{x}))} \omega_{N_a}^{\eta y_{s_a+1}^a} |x, y_{s_a+1}^a\rangle_{D_a(x)} \right.$$

$$
\begin{aligned}
&\left.+\frac{1}{\sqrt{N_a}}\sum_{y^a_{s_a+1}\in\mathcal{B}_a(x|D(\vec{x}))}\omega_{N_a}^{\eta y^a_{s_a+1}}|x,y^a_{s_a+1}\rangle_{D_a(x)}\right)\\
&\omega_{N_{\bar{a}}}^{\vec{\eta}_{\bar{a}}\cdot\vec{y}_{\bar{a}}}|(x_1^{\bar{a}},y_1^{\bar{a}}),\dots,(x^{\bar{a}}_{s_{\bar{a}}},y^{\bar{a}}_{s_{\bar{a}}})\rangle_{D_{\bar{a}}(\vec{x}_{\bar{a}})}\\
&\sum_{y^a_{s_a+2},\dots,y^a_q\in[N_a]}\frac{1}{\sqrt{N_a^{q-s_a-1}}}|(\perp,y^a_{s_a+2}),\dots,(\perp,y^a_q)\rangle_{D_a(\perp)}\\
&\sum_{y^{\bar{a}}_{s_{\bar{a}}+1},\dots,y^{\bar{a}}_q\in[N_{\bar{a}}]}\frac{1}{\sqrt{N_{\bar{a}}^{q-s_{\bar{a}}}}}|(\perp,y^{\bar{a}}_{s_{\bar{a}}+1}),\dots,(\perp,y^{\bar{a}}_q)\rangle_{D_{\bar{a}}(\perp)}.
\end{aligned}\tag{6.104}
$$

Errors that are left to be analyzed come from $D_{\bar{a}}$, let us present the split in the sum over $\vec{y}_a$ that we will use in the ADD case:

$$
\begin{aligned}
&\sum_{\vec{y}_a\in\mathcal{G}_a(\vec{x}_1,\vec{x}_2)}\ \sum_{\vec{y}_{\bar{a}}\in\mathcal{G}_{\bar{a}}(\vec{x}_1,\vec{x}_2|\vec{y}_a)}\ \sum_{y^a_{s_a+1}\notin\mathcal{B}_a(x|D(\vec{x}))} = \sum_{\vec{y}_a\in\mathcal{G}_a(\vec{x}_1,\vec{x}_2)}\ \sum_{\vec{y}_{\bar{a}}\in\mathcal{G}_{\bar{a}}(\vec{x}_a\cup\{x\},\vec{x}_{\bar{a}}|\vec{y}_a)}\ \sum_{y^a_{s_a+1}\notin\mathcal{B}_a(x|D(\vec{x}))}\\
&+\sum_{\vec{y}_a\in\mathcal{G}_a(\vec{x}_1,\vec{x}_2)}\ \sum_{\vec{y}_{\bar{a}}\in\mathcal{G}_{\bar{a}}(\vec{x}_1,\vec{x}_2|\vec{y}_a)\setminus\mathcal{G}_{\bar{a}}(\vec{x}_a\cup\{x\},\vec{x}_{\bar{a}}|\vec{y}_a)}\ \sum_{y^a_{s_a+1}\notin\mathcal{B}_a(x|D(\vec{x}))}.
\end{aligned}\tag{6.105}
$$

Next we include the full impact of V_R. Two things happen in the expression below. First we split the sum over $\vec{y}_{\bar{a}}$ in the first element in the parentheses, secondly we rewrite the normalization factors to simplify the analysis later on. We underline parts of the state that are important later on. With red color we denote the errors. After applying $\mathsf{Queries}^\dagger\circ\mathsf{V}_R\circ\mathsf{Queries}$ the state is:

$$
\begin{aligned}
&\mathsf{ADD}:\mathsf{H}\setminus\mathsf{V}_R|\xi_{i-1}(\mathsf{ADD})\rangle|0\rangle_J\\
&=\sum_{x,\eta,a,\vec{x},\vec{\eta},w}\alpha_{x,\eta,a,\vec{x},\vec{\eta},w}|x,\eta,a\rangle_{A^{XYI}}|\psi(x,\eta,a,\vec{x},\vec{\eta},w)\rangle_{A^W}\\
&\sum_{\vec{y}_a\in\mathcal{G}_a(\vec{x}_1,\vec{x}_2)}\frac{1}{\sqrt{|\mathcal{G}_a(s_1,s_2)|}}\omega_{N_a}^{\vec{\eta}_a\cdot\vec{y}_a}|(x_1^a,y_1^a),\dots,(x^a_{s_a},y^a_{s_a})\rangle_{D_a(\vec{x}_a)}\\
&\Bigg(\sqrt{\frac{N_a-b_a(s_a+1,s_{\bar{a}})}{N_a}}\sqrt{\frac{|\mathcal{G}_{\bar{a}}(s_a+1,s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}}\\
&\underbrace{\sum_{\vec{y}_{\bar{a}}\in\mathcal{G}_{\bar{a}}(\vec{x}_a\cup\{x\},\vec{x}_{\bar{a}}|\vec{y}_a)}\frac{1}{\sqrt{|\mathcal{G}_{\bar{a}}(s_a+1,s_{\bar{a}})|}}}_{\text{(i) }|\Psi_i^{\mathrm{Good}}(\mathrm{ADD},a,s_1,s_2)\rangle}\\
&\underbrace{\sum_{y^a_{s_a+1}\notin\mathcal{B}_a(x|D(\vec{x}))}\frac{1}{\sqrt{N_a-b_a(s_a+1,s_{\bar{a}})}}\omega_{N_a}^{\eta y^a_{s_a+1}}|x,y^a_{s_a+1}\rangle_{D_a(x)}|0\rangle_J}_{\text{(ii) }|\Psi_i^{\mathrm{Good}}(\mathrm{ADD},a,s_1,s_2)\rangle}\\
&\underbrace{+\sqrt{\frac{N_a-b_a(s_a+1,s_{\bar{a}})}{N_a}}\sqrt{\frac{|\mathcal{H}_a^{\mathrm{ADD}}(s_1,s_2)|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}}\sum_{\vec{y}_{\bar{a}}\in\mathcal{H}_a^{\mathrm{ADD}}(\vec{x}_1,\vec{x}_2,\vec{y}_a)}\frac{1}{\sqrt{|\mathcal{H}_a^{\mathrm{ADD}}(a,s_1,s_2)|}}}_{{\color{red}\text{(i) }|\Psi_{i,1}^{\mathrm{Find}}(\mathrm{ADD},a,s_1,s_2)\rangle}}
\end{aligned}
$$

$$\underbrace{\sum_{y^a_{s_a+1}\notin\mathcal{B}_a(x|D(\vec{x}))}\frac{1}{\sqrt{N_a-b_a(s_a+1,s_{\bar{a}})}}\omega_{N_a}^{\eta y^a_{s_a+1}}|x,y^a_{s_a+1}\rangle_{D_a(x)}|1\rangle_J}_{\text{(ii) }|\Psi^{\text{Find}}_{i,1}(\text{ADD},a,s_1,s_2)\rangle}$$

$$\underbrace{+\sqrt{\frac{b_a(s_a+1,s_{\bar{a}})}{N_a}}\sum_{\vec{y}_{\bar{a}}\in\mathcal{G}_{\bar{a}}(\vec{x}_1,\vec{x}_2|\vec{y}_a)}\frac{1}{\sqrt{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}}}_{\text{(i) }|\Psi^{\text{Find}}_{i,2}(\text{ADD},a,s_1,s_2)\rangle}$$

$$\left.\underbrace{\sum_{y^a_{s_a+1}\in\mathcal{B}_a(x|D(\vec{x}))}\frac{1}{\sqrt{b_a(s_a+1,s_{\bar{a}})}}\omega_{N_a}^{\eta y^a_{s_a+1}}|x,y^a_{s_a+1}\rangle_{D_a(x)}|1\rangle_J}_{\text{(ii) }|\Psi^{\text{Find}}_{i,2}(\text{ADD},a,s_1,s_2)\rangle}\right)$$

$$\omega_{N_{\bar{a}}}^{\vec{\eta}_{\bar{a}}\cdot\vec{y}_{\bar{a}}}|(x^{\bar{a}}_1,y^{\bar{a}}_1),\ldots,(x^{\bar{a}}_{s_{\bar{a}}},y^{\bar{a}}_{s_{\bar{a}}})\rangle_{D_{\bar{a}}(\vec{x}_{\bar{a}})}$$
$$\sum_{y^a_{s_a+2},\ldots,y^a_q\in[N_a]}\frac{1}{\sqrt{N_a^{q-s_a-1}}}|(\perp,y^a_{s_a+2}),\ldots,(\perp,y^a_q)\rangle_{D_a(\perp)}$$
$$\sum_{y^{\bar{a}}_{s_{\bar{a}}+1},\ldots,y^{\bar{a}}_q\in[N_{\bar{a}}]}\frac{1}{\sqrt{N_{\bar{a}}^{q-s_{\bar{a}}}}}|(\perp,y^{\bar{a}}_{s_{\bar{a}}+1}),\ldots,(\perp,y^{\bar{a}}_q)\rangle_{D_{\bar{a}}(\perp)},\tag{6.106}$$

where the appropriate position of register J is after D. The size of the domain of $\vec{y}_{\bar{a}}$ is denoted by

$$|\mathcal{G}_{\bar{a}}(s_a+1,s_{\bar{a}})|:=|\mathcal{G}_{\bar{a}}(\vec{x}_a\cup\{x\},\vec{x}_{\bar{a}}\mid\vec{y}_a)|\,,\tag{6.107}$$

which uses our assumption that the size of the good set does not depend on the actual values stored in D. By $|\Psi^{\text{Good}}_i(\text{ADD};a,s_1,s_2)\rangle$, $|\Psi^{\text{Find}}_{i,1}(\text{ADD};a,s_1,s_2)\rangle$, and $|\Psi^{\text{Find}}_{i,2}(\text{ADD};a,s_1,s_2)\rangle$ we mean states equal to the above state but with just the underlined part in the parentheses. We used color in Equation (6.106) to indicate the parts that we consider errors. By adding arguments to states we mean that these values are fixed. We add a as the argument to specify the queried interface and s_1 and s_2 to specify the sizes of the databases. The formal definition of states with a, s_1, or s_2 specified is the underlined branch of the superposition projected to register A^I containing a and databases with s_1 and s_2 inputs not equal $\perp$. Above we use $\left|\mathcal{H}^{\text{ADD}}_a(s_1,s_2)\right|$ defined in Equation (6.82). In Equation (6.82) we use the fact that the cardinality of $\mathcal{G}$ depends only on s_1 and s_2, conditioning on $\vec{y}_a$ does not influence the cardinality either, so we omit it in the arguments of $\left|\mathcal{H}^{\text{ADD}}_a(s_1,s_2)\right|$.

For the state $|\Psi^{\text{Find}}_{i,2}(\text{ADD};a,s_1,s_2)\rangle$ we omit entirely the analysis of the other database. That is because the register $D(x)$ is the one responsible for D being in relation and there is no need to analyze $D_{\bar{a}}$.

When we update or remove from the database we start by presenting the non-punctured oracle to make clear the source of errors when discussing the punctured oracle. The counting procedure Queries acts by just analyzing D^X. The only point where we operate in the Fourier basis is when in Algorithm 6.1 we update the count in line 8. Below we are a bit sloppy with notation of $\vec{\eta}$, and $\vec{\eta}_a$ does not contain $\eta^a_{s_a}$:

$$\begin{aligned}
&\mathsf{H}\left(|\xi_{i-1}(\text{UPD})\rangle + |\xi_{i-1}(\text{REM})\rangle\right)\\
&= \sum_{x,\eta,a,\vec{x},\vec{\eta},w} \alpha_{x,\eta,a,\vec{x},\vec{\eta},w} |x,\eta,a\rangle_{A^{XYI}} |\psi(x,\eta,a,\vec{x},\vec{\eta},w)\rangle_{A^W}\\
&\sum_{\vec{y}_a\in\mathcal{G}_a(\vec{x}_a\setminus\{x\},\vec{x}_{\bar{a}})} \frac{1}{\sqrt{|\mathcal{G}_a(s_a-1,s_{\bar{a}})|}} \omega_{N_a}^{\vec{\eta}_a\cdot\vec{y}_a} |(x^a_1,y^a_1),\ldots,(x^a_{s_a-1},y^a_{s_a-1})\rangle_{D_a(\vec{x}_a\setminus\{x\})}\\
&\sum_{\vec{y}_{\bar{a}}\in\mathcal{G}_{\bar{a}}(\vec{x}_1,\vec{x}_2|\vec{y}_a)} \frac{1}{\sqrt{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}}\\
&\Bigg(\sum_{y^a_{s_a}\notin\mathcal{B}_a(x|D(\vec{x}\setminus\{x\}))} \frac{1}{\sqrt{N_a-b_a(s_1,s_2)}} \omega_{N_a}^{(\eta^a_{s_a}+\eta)y^a_{s_a}} |x,y^a_{s_a}\rangle_{D_a(x)}\\
&-\frac{1}{\sqrt{N_a(N_a-b_a(s_1,s_2))}} \sum_{y^a_{s_a}\notin\mathcal{B}_a(x|D(\vec{x}\setminus\{x\}))} \omega_{N_a}^{(\eta^a_{s_a}+\eta)y^a_{s_a}} \sum_{y^{a\prime}_{s_a}\in[N_a]} \frac{1}{\sqrt{N_a}} |x,y^{a\prime}_{s_a}\rangle_{D(x)}\\
&+\frac{1}{\sqrt{N_a(N_a-b_a(s_1,s_2))}} \sum_{y^a_{s_a}\notin\mathcal{B}_a(x|D(\vec{x}\setminus\{x\}))} \omega_{N_a}^{(\eta^a_{s_a}+\eta)y^a_{s_a}} \sum_{y^{a\prime}_{s_a}\in[N_a]} \frac{1}{\sqrt{N_a}} |\perp,y^{a\prime}_{s_a}\rangle_{D(x)}\Bigg)\\
&\omega_{N_{\bar{a}}}^{\vec{\eta}_{\bar{a}}\cdot\vec{y}_{\bar{a}}} |(x^{\bar{a}}_1,y^{\bar{a}}_1),\ldots,(x^{\bar{a}}_{s_{\bar{a}}},y^{\bar{a}}_{s_{\bar{a}}})\rangle_{D_{\bar{a}}(\vec{x}_{\bar{a}})}\\
&\sum_{y^a_{s_a+1},\ldots,y^a_q\in[N_a]} \frac{1}{\sqrt{N_a^{q-s_a}}} |(\perp,y^a_{s_a+1}),\ldots,(\perp,y^a_q)\rangle_{D_a(\perp)}\\
&\sum_{y^{\bar{a}}_{s_{\bar{a}}+1},\ldots,y^{\bar{a}}_q\in[N_{\bar{a}}]} \frac{1}{\sqrt{N_{\bar{a}}^{q-s_{\bar{a}}}}} |(\perp,y^{\bar{a}}_{s_{\bar{a}}+1}),\ldots,(\perp,y^{\bar{a}}_q)\rangle_{D_{\bar{a}}(\perp)}, \qquad (6.108)
\end{aligned}$$

The states that we add to the first element in the parentheses come from performing the Fourier transform on a state that is not of the form $\mathsf{QFT}_{N_a}|\eta\rangle$. Note that this discrepancy is the result of considering the good state. Whether we are in the branch UPD or REM depends on whether $\eta = -\eta_s$ or not.

Similarly to the case of ADD, we first present the state with the error parts of D_a exposed.

$$\begin{aligned}
\mathsf{H}|\xi_{i-1}(\text{UPD})\rangle = &\sum_{x,\eta,a,\vec{x},\vec{\eta},w} \alpha_{x,\eta,a,\vec{x},\vec{\eta},w} |x,\eta,a\rangle_{A^{XYI}} |\psi(x,\eta,a,\vec{x},\vec{\eta},w)\rangle_{A^W}\\
&\sum_{\vec{y}_a\in\mathcal{G}_a(\vec{x}_a\setminus\{x\},\vec{x}_{\bar{a}})} \frac{1}{\sqrt{|\mathcal{G}_a(s_a-1,s_{\bar{a}})|}} \omega_{N_a}^{\vec{\eta}_a\cdot\vec{y}_a} |(x^a_1,y^a_1),\ldots,(x^a_{s_a-1},y^a_{s_a-1})\rangle_{D_a(\vec{x}_a\setminus\{x\})}\\
&\sum_{\vec{y}_{\bar{a}}\in\mathcal{G}_{\bar{a}}(\vec{x}_1,\vec{x}_2|\vec{y}_a)} \frac{1}{\sqrt{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}}
\end{aligned}$$

$$\Bigg(\sum_{y^a_{s_a}\notin\mathcal{B}_a(x|D(\vec{x}\backslash\{x\}))}\frac{1}{\sqrt{N_a-b_a(s_1,s_2)}}\omega_N^{(\eta^a_{s_a}+\eta)y^a_{s_a}}|x,y^a_{s_a}\rangle_{D_a(x)}$$
$$+\frac{1}{N_a\sqrt{N_a-b_a(s_1,s_2)}}\sum_{y^a_{s_a}\in\mathcal{B}_a(x|D(\vec{x}\backslash\{x\}))}\omega_{N_a}^{(\eta^a_{s_a}+\eta)y^a_{s_a}}\sum_{y^{a\prime}_{s_a}\notin\mathcal{B}_a(x|D(\vec{x}\backslash\{x\}))}|x,y^{a\prime}_s\rangle_{D(x)}$$
$$+\frac{1}{N_a\sqrt{N_a-b_a(s_1,s_2)}}\sum_{y^a_{s_a}\in\mathcal{B}_a(x|D(\vec{x}\backslash\{x\}))}\omega_{N_a}^{(\eta^a_{s_a}+\eta)y^a_{s_a}}\sum_{y^{a\prime}_{s_a}\in\mathcal{B}_a(x|D(\vec{x}\backslash\{x\}))}|x,y^{a\prime}_{s_a}\rangle_{D(x)}$$
$$-\frac{1}{N_a\sqrt{N_a-b_a(s_1,s_2)}}\sum_{y^a_{s_a}\in\mathcal{B}_a(x|D(\vec{x}\backslash\{x\}))}\omega_{N_a}^{(\eta^a_{s_a}+\eta)y^a_{s_a}}\sum_{y^{a\prime}_{s_a}\in[N_a]}|\bot,y^{a\prime}_{s_a}\rangle_{D_a(x)}\Bigg)$$
$$\omega_{N_{\bar{a}}}^{\vec{\eta}_{\bar{a}}\cdot\vec{y}_{\bar{a}}}|(x^{\bar{a}}_1,y^{\bar{a}}_1),\ldots,(x^{\bar{a}}_{s_{\bar{a}}},y^{\bar{a}}_{s_{\bar{a}}})\rangle_{D_{\bar{a}}(\vec{x}_{\bar{a}})}$$
$$\sum_{y^a_{s_a+1},\ldots,y^a_q\in[N_a]}\frac{1}{\sqrt{N_a^{q-s_a}}}|(\bot,y^a_{s_a+1}),\ldots,(\bot,y^a_q)\rangle_{D_a(\bot)}$$
$$\sum_{y^{\bar{a}}_{s_{\bar{a}}+1},\ldots,y^{\bar{a}}_q\in[N_{\bar{a}}]}\frac{1}{\sqrt{N_{\bar{a}}^{q-s_{\bar{a}}}}}|(\bot,y^{\bar{a}}_{s_{\bar{a}}+1}),\ldots,(\bot,y^{\bar{a}}_q)\rangle_{D_{\bar{a}}(\bot)}, \tag{6.109}$$

where we have used the fact that $\eta^a_{s_a}+\eta\neq 0$ which implies $\sum_{y^a_{s_a}\notin\mathcal{B}_a(x|D(\vec{x}\backslash\{x\}))}$ $\omega_{N_a}^{(\eta^a_{s_a}+\eta)y^a_{s_a}}=-\sum_{y^a_{s_a}\in\mathcal{B}_a(x|D(\vec{x}\backslash\{x\}))}\omega_{N_a}^{(\eta^a_{s_a}+\eta)y^a_{s_a}}$.

Splitting the sums in the case of removing an entry works as follows:

$$\sum_{\vec{y}_a\in\mathcal{G}_a(\vec{x}\backslash\{x\})}\sum_{\vec{y}_{\bar{a}}\in\mathcal{G}_{\bar{a}}(\vec{x}|\vec{y}_a)}=\sum_{\vec{y}_a\in\mathcal{G}_a(\vec{x}\backslash\{x\})}\sum_{\vec{y}_{\bar{a}}\in\mathcal{G}_{\bar{a}}(\vec{x}\backslash\{x\}|\vec{y}_a)}$$
$$-\sum_{\vec{y}_a\in\mathcal{G}_a(\vec{x}\backslash\{x\})}\sum_{\vec{y}_{\bar{a}}\in\mathcal{G}_{\bar{a}}(\vec{x}\backslash\{x\}|\vec{y}_a)\backslash\mathcal{G}_{\bar{a}}(\vec{x}|\vec{y}_a)}. \tag{6.110}$$

Let us present in detail the effect of Queries$^\dagger\circ\mathsf{V}_R\circ$ Queries on the branch of updated databases:

$$\mathsf{UPD}:\mathsf{H}\setminus\mathsf{V}_R|\xi_{i-1}(\mathsf{UPD})\rangle|0\rangle_J$$
$$=\sum_{x,\eta,a,\vec{x},\vec{\eta},w}\alpha_{x,\eta,a,\vec{x},\vec{\eta},w}|x,\eta,a\rangle_{A^{XYI}}|\psi(x,\eta,a,\vec{x},\vec{\eta},w)\rangle_{A^W}$$
$$\sum_{\vec{y}_a\in\mathcal{G}_a(\vec{x}_a\backslash\{x\},\vec{x}_{\bar{a}})}\frac{1}{\sqrt{|\mathcal{G}_a(s_a-1,s_{\bar{a}})|}}\omega_{N_a}^{\vec{\eta}_a\cdot\vec{y}_a}|(x^a_1,y^a_1),\ldots,(x^a_{s_a-1},y^a_{s_a-1})\rangle_{D_a(\vec{x}_a\backslash\{x\})}$$
$$\Big(|\Psi_i^{\mathsf{Good}}(\mathsf{UPD};a,s_1,s_2)\rangle|0\rangle_J$$
$$+|\Psi_{i,1}^{\mathsf{Bad}}(\mathsf{UPD};a,s_1,s_2)\rangle|0\rangle_J$$
$$+|\Psi_{i,1}^{\mathsf{Find}}(\mathsf{UPD};a,s_1,s_2)\rangle|1\rangle_J$$
$$-|\Psi_{i,2}^{\mathsf{Bad}}(\mathsf{UPD};a,s_1,s_2)\rangle|0\rangle_J+|\Psi_{i,2}^{\mathsf{Find}}(\mathsf{UPD};a,s_1,s_2)\rangle|1\rangle_J\Big)$$
$$\omega_{N_{\bar{a}}}^{\vec{\eta}_{\bar{a}}\cdot\vec{y}_{\bar{a}}}|(x^{\bar{a}}_1,y^{\bar{a}}_1),\ldots,(x^{\bar{a}}_{s_{\bar{a}}},y^{\bar{a}}_{s_{\bar{a}}})\rangle_{D_{\bar{a}}(\vec{x}_{\bar{a}})}$$
$$\sum_{y^a_{s_a+1},\ldots,y^a_q\in[N_a]}\frac{1}{\sqrt{N_a^{q-s_a}}}|(\bot,y^a_{s_a+1}),\ldots,(\bot,y^a_q)\rangle_{D_a(\bot)}$$

$$\sum_{y^{\bar{a}}_{s_{\bar{a}}+1},\dots,y^{\bar{a}}_q\in[N_{\bar{a}}]} \frac{1}{\sqrt{N_{\bar{a}}^{q-s_{\bar{a}}}}}|(\perp,y^{\bar{a}}_{s_{\bar{a}}+1}),\dots,(\perp,y^{\bar{a}}_q)\rangle_{D_{\bar{a}}(\perp)}, \tag{6.111}$$

where the states with superscripts Good, Bad, and Find denote the states that are defined as the state from Equation (6.111) with expressions from Equations (6.112), (6.113), (6.114), (6.115), or (6.116) below put in the correct spot, without any other element from the parentheses. Note that the states in the equation above come from splitting the sum over $\vec{y}_{\bar{a}}$ into the parts of Equation (6.109) from the corresponding lines. Below we define in detail all the states from Equation (6.111).

The good state gives the following expression in the parentheses in Equation (6.111):

$$|\Psi_i^{\text{Good}}(\text{UPD};a,s_1,s_2)\rangle : \sum_{\vec{y}_{\bar{a}}\in\mathcal{G}_{\bar{a}}(\vec{x}_1,\vec{x}_2|\vec{y}_a)} \frac{1}{\sqrt{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}}$$
$$\sum_{y^a_{s_a}\notin\mathcal{B}_a(x|D(\vec{x}\setminus\{x\}))} \frac{1}{\sqrt{N_a-b_a(s_1,s_2)}}\omega_N^{(\eta^a_{s_a}+\eta)y^a_{s_a}}|x,y^a_{s_a}\rangle_{D_a(x)}. \tag{6.112}$$

Bad states are those with $|0\rangle_J$ that are not good:

$$|\Psi_{i,1}^{\text{Bad}}(\text{UPD};a,s_1,s_2)\rangle : \frac{1}{N_a}\sum_{\vec{y}_{\bar{a}}\in\mathcal{G}_{\bar{a}}(\vec{x}_1,\vec{x}_2|\vec{y}_a)} \frac{1}{\sqrt{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}}\sum_{y^a_{s_a}\in\mathcal{B}_a(x|D(\vec{x}\setminus\{x\}))}\omega_{N_a}^{(\eta^a_{s_a}+\eta)y^a_{s_a}}$$
$$\sum_{y^{a\prime}_{s_a}\notin\mathcal{B}_a(x|D(\vec{x}\setminus\{x\}))} \frac{1}{\sqrt{N_a-b_a(s_1,s_2)}}|x,y^{a\prime}_{s_a}\rangle_{D(x)}, \tag{6.113}$$

$$|\Psi_{i,2}^{\text{Bad}}(\text{UPD};a,s_1,s_2)\rangle : \frac{1}{\sqrt{N_a(N_a-b_a(s_1,s_2))}}\sqrt{\frac{|\mathcal{G}_{\bar{a}}(s_a-1,s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}}$$
$$\sum_{\vec{y}_{\bar{a}}\in\mathcal{G}_{\bar{a}}(\vec{x}_a\setminus\{x\},\vec{x}_{\bar{a}}|\vec{y}_a)} \frac{1}{\sqrt{|\mathcal{G}_{\bar{a}}(s_a-1,s_{\bar{a}})|}}$$
$$\sum_{y^a_{s_a}\in\mathcal{B}_a(x|D(\vec{x}\setminus\{x\}))}\omega_{N_a}^{(\eta^a_{s_a}+\eta)y^a_{s_a}}\sum_{y^{a\prime}_{s_a}\in[N_a]}\frac{1}{\sqrt{N_a}}|\perp,y^{a\prime}_{s_a}\rangle_{D_a(x)}. \tag{6.114}$$

The states with the superscript Find are states for which $D\in R$:

$$|\Psi_{i,1}^{\text{Find}}(\text{UPD};a,s_1,s_2)\rangle : \sqrt{\frac{b_a(s_1,s_2)}{N_a^2(N_a-b_a(s_1,s_2))}}\sum_{\vec{y}_{\bar{a}}\in\mathcal{G}_{\bar{a}}(\vec{x}_1,\vec{x}_2|\vec{y}_a)}\frac{1}{\sqrt{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}}$$
$$\sum_{y^a_{s_a}\in\mathcal{B}_a(x|D(\vec{x}\setminus\{x\}))}\omega_{N_a}^{(\eta^a_{s_a}+\eta)y^a_{s_a}}\sum_{y^{a\prime}_{s_a}\in\mathcal{B}_a(x|D(\vec{x}\setminus\{x\}))}\frac{1}{\sqrt{b_a(s_1,s_2)}}|x,y^{a\prime}_{s_a}\rangle_{D(x)}, \tag{6.115}$$

$$|\Psi_{i,2}^{\text{Find}}(\text{UPD};a,s_1,s_2)\rangle : \frac{1}{\sqrt{N_a(N_a-b_a(s_1,s_2))}}\sqrt{\frac{|\mathcal{H}_a^{\text{REM}}(s_1,s_2)|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}}\sum_{\vec{y}_{\bar{a}}\in\mathcal{H}_1^{\text{REM}}(\vec{x}_1,\vec{x}_2,\vec{y}_a)}$$

$$\frac{1}{\sqrt{\left|\mathcal{H}_a^{\mathrm{REM}}(s_1,s_2)\right|}}\sum_{y^a_{s_a}\in\mathcal{B}_a(x|D(\vec{x}\setminus\{x\}))}\omega_{N_a}^{(\eta^a_{s_a}+\eta)y^a_{s_a}}\sum_{y^{a\prime}_{s_a}\in[N_a]}\frac{1}{\sqrt{N_a}}|\perp,y^{a\prime}_{s_a}\rangle_{D_a(x)}, \tag{6.116}$$

where $\left|\mathcal{H}_a^{\mathrm{REM}}(s_1,s_2)\right|$ is defined in Equation (6.84).

The branch of superposition corresponding to removing x from D with just the errors in D_a exposed is

$$\begin{aligned}
&\mathrm{REM}:\mathsf{H}|\xi_{i-1}(\mathrm{REM})\rangle\\
&=\sum_{x,\eta,a,\vec{x},\vec{\eta},w}\alpha_{x,\eta,a,\vec{x},\vec{\eta},w}|x,\eta,a\rangle_{A^{XYI}}|\psi(x,\eta,a,\vec{x},\vec{\eta},w)\rangle_{A^W}\\
&\sum_{\vec{y}_a\in\mathcal{G}_a(\vec{x}_a\setminus\{x\},\vec{x}_{\bar{a}})}\frac{1}{\sqrt{|\mathcal{G}_a(s_a-1,s_{\bar{a}})|}}\omega_{N_a}^{\vec{\eta}_a\cdot\vec{y}_a}|(x^a_1,y^a_1),\ldots,(x^a_{s_a-1},y^a_{s_a-1})\rangle_{D_a(\vec{x}_a\setminus\{x\})}\\
&\sum_{\vec{y}_{\bar{a}}\in\mathcal{G}_{\bar{a}}(\vec{x}_1,\vec{x}_2|\vec{y}_a)}\frac{1}{\sqrt{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}}\left(\sqrt{\frac{N_a-b_a(s_1,s_2)}{N_a}}\sum_{y^a_{s_a}\in[N_a]}\frac{1}{\sqrt{N_a}}|\perp,y^a_{s_a}\rangle_{D(x)}\right.\\
&+\frac{b_a(s_1,s_2)}{N_a}\sum_{y^a_{s_a}\notin\mathcal{B}_a(x|D(\vec{x}\setminus\{x\}))}\frac{1}{\sqrt{N_a-b_a(s_1,s_2)}}|x,y^a_{s_a}\rangle_{D_a(x)}\\
&\left.+\frac{\sqrt{b_a(s_1,s_2)(N_a-b_a(s_1,s_2))}}{N_a}\sum_{y^a_{s_a}\in\mathcal{B}_a(x|D(\vec{x}\setminus\{x\}))}\frac{1}{\sqrt{b_a(s_1,s_2)}}|x,y^a_{s_a}\rangle_{D_a(x)}\right)\\
&\omega_{N_{\bar{a}}}^{\vec{\eta}_{\bar{a}}\cdot\vec{y}_{\bar{a}}}|(x^{\bar{a}}_1,y^{\bar{a}}_1),\ldots,(x^{\bar{a}}_{s_{\bar{a}}},y^{\bar{a}}_{s_{\bar{a}}})\rangle_{D_{\bar{a}}(\vec{x}_{\bar{a}})}\\
&\sum_{y^a_{s_a+1},\ldots,y^a_q\in[N_a]}\frac{1}{\sqrt{N_a^{q-s_a}}}|(\perp,y^a_{s_a+1}),\ldots,(\perp,y^a_q)\rangle_{D_a(\perp)}\\
&\sum_{y^{\bar{a}}_{s_{\bar{a}}+1},\ldots,y^{\bar{a}}_q\in[N_{\bar{a}}]}\frac{1}{\sqrt{N_{\bar{a}}^{q-s_{\bar{a}}}}}|(\perp,y^{\bar{a}}_{s_{\bar{a}}+1}),\ldots,(\perp,y^{\bar{a}}_q)\rangle_{D_{\bar{a}}(\perp)},
\end{aligned} \tag{6.117}$$

where we simplified the formula from Equation (6.108) using the fact $\eta=-\eta^a_{s_a}$. Splitting the sums in the case of removing an entry works as shown in Equation (6.110).

Now we present the full state, after checking for $D\in R$:

$$\begin{aligned}
&\mathrm{REM}:\mathsf{H}\setminus\mathsf{V}_R|\xi_{i-1}(\mathrm{REM})\rangle|0\rangle_J\\
&=\sum_{x,\eta,a,\vec{x},\vec{\eta},w}\alpha_{x,\eta,a,\vec{x},\vec{\eta},w}|x,\eta,a\rangle_{A^{XYI}}|\psi(x,\eta,a,\vec{x},\vec{\eta},w)\rangle_{A^W}\\
&\sum_{\vec{y}_a\in\mathcal{G}_a(\vec{x}_a\setminus\{x\},\vec{x}_{\bar{a}})}\frac{1}{\sqrt{|\mathcal{G}_a(s_a-1,s_{\bar{a}})|}}\omega_{N_a}^{\vec{\eta}_a\cdot\vec{y}_a}|(x^a_1,y^a_1),\ldots,(x^a_{s_a-1},y^a_{s_a-1})\rangle_{D_a(\vec{x}_a\setminus\{x\})}\\
&\left(\sqrt{\frac{N_a-b_a(s_1,s_2)}{N_a}}\sqrt{\frac{|\mathcal{G}_{\bar{a}}(s_a-1,s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}}\right.
\end{aligned}$$

$$
\begin{gathered}
\underbrace{\sum_{\vec{y}_{\bar{a}}\in\mathcal{G}_{\bar{a}}(\vec{x}_a\setminus\{x\},\vec{x}_{\bar{a}}|\vec{y}_a)} \frac{1}{\sqrt{|\mathcal{G}_{\bar{a}}(s_a-1,s_{\bar{a}})|}} \sum_{y^a_{s_a}\in[N_a]} \frac{1}{\sqrt{N_a}} |\perp, y^a_{s_a}\rangle_{D(x)} |0\rangle_J}_{|\Psi_i^{\text{Good}}(\text{REM},a,s_1,s_2)\rangle} \\
-\underbrace{\sqrt{\frac{N_a-b_a(s_1,s_2)}{N_a}}\sqrt{\frac{|\mathcal{H}_a^{\text{REM}}(s_1,s_2)|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}} \sum_{\vec{y}_{\bar{a}}\in\mathcal{H}_a^{\text{REM}}(\vec{x}_1,\vec{x}_2,\vec{y}_a)} \frac{1}{\sqrt{|\mathcal{H}_a^{\text{REM}}(s_1,s_2)|}}}_{\text{(i) } |\Psi_{i,1}^{\text{Find}}(\text{REM},a,s_1,s_2)\rangle} \\
\underbrace{\sum_{y^a_{s_a}\in[N_a]} \frac{1}{\sqrt{N_a}} |\perp, y^a_{s_a}\rangle_{D(x)} |1\rangle_J}_{\text{(ii) } |\Psi_{i,1}^{\text{Find}}(\text{REM},a,s_1,s_2)\rangle} \\
+\underbrace{\frac{b_a(s_1,s_2)}{N_a} \sum_{\vec{y}_{\bar{a}}\in\mathcal{G}_{\bar{a}}(\vec{x}_1,\vec{x}_2|\vec{y}_a)} \frac{1}{\sqrt{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}}}_{\text{(i) } |\Psi_i^{\text{Bad}}(\text{REM};a,s_1,s_2)\rangle} \\
\underbrace{\sum_{y^a_{s_a}\notin\mathcal{B}_a(x|D(\vec{x}\setminus\{x\}))} \frac{1}{\sqrt{N_a-b_a(s_1,s_2)}} |x, y^a_{s_a}\rangle_{D_a(x)} |0\rangle_J}_{\text{(ii) } |\Psi_i^{\text{Bad}}(\text{REM};a,s_1,s_2)\rangle} \\
+\underbrace{\frac{\sqrt{b_a(s_1,s_2)(N_a-b_a(s_1,s_2))}}{N_a} \sum_{\vec{y}_{\bar{a}}\in\mathcal{G}_{\bar{a}}(\vec{x}_1,\vec{x}_2|\vec{y}_a)} \frac{1}{\sqrt{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}}}_{\text{(i) } |\Psi_{i,2}^{\text{Find}}(\text{REM},a,s_1,s_2)\rangle} \\
\left.\underbrace{\sum_{y^a_{s_a}\in\mathcal{B}_a(x|D(\vec{x}\setminus\{x\}))} \frac{1}{\sqrt{b_a(s_1,s_2)}} |x, y^a_{s_a}\rangle_{D_a(x)} |1\rangle_J}_{\text{(ii) } |\Psi_{i,2}^{\text{Find}}(\text{REM},a,s_1,s_2)\rangle}\right) \\
\omega_{N_{\bar{a}}}^{\vec{\eta}_{\bar{a}}\cdot\vec{y}_{\bar{a}}} |(x_1^{\bar{a}}, y_1^{\bar{a}}),\dots,(x_{s_{\bar{a}}}^{\bar{a}}, y_{s_{\bar{a}}}^{\bar{a}})\rangle_{D_{\bar{a}}(\vec{x}_{\bar{a}})} \\
\sum_{y^a_{s_a+1},\dots,y^a_q\in[N_a]} \frac{1}{\sqrt{N_a^{q-s_a}}} |(\perp, y^a_{s_a+1}),\dots,(\perp, y^a_q)\rangle_{D_a(\perp)} \\
\sum_{y^{\bar{a}}_{s_{\bar{a}}+1},\dots,y^{\bar{a}}_q\in[N_{\bar{a}}]} \frac{1}{\sqrt{N_{\bar{a}}^{q-s_{\bar{a}}}}} |(\perp, y^{\bar{a}}_{s_{\bar{a}}+1}),\dots,(\perp, y^{\bar{a}}_q)\rangle_{D_{\bar{a}}(\perp)}, \tag{6.118}
\end{gathered}
$$

where $\mathcal{H}_a^{\text{REM}}(\vec{x}_1,\vec{x}_2,\vec{y}_a) = \mathcal{G}_{\bar{a}}(\vec{x}_a\setminus\{x\},\vec{x}_{\bar{a}} \mid \vec{y}_a) \setminus \mathcal{G}_{\bar{a}}(\vec{x}_1,\vec{x}_2 \mid \vec{y}_a)$.

6.4.1.6 Bound on $\varepsilon_{\text{step}}(j)$

We want to show that after any query, $|\Phi_i\rangle_{ADJ}$ is close to $|\Psi_i^{\text{Good}}\rangle_{AD}|0\rangle_J$. One way of looking at the lemma below is from the perspective of an adversary searching for inputs that provide outputs of a random function that are in R. Normally this task does not involve a punctured oracle but a regular one. We show here the error introduced by puncturing the oracle; The two states that we consider come from projecting with $\overline{\mathsf{J}}_R$ either the state after interacting with a non-punctured oracle or the state after interacting with a punctured oracle (given $\neg$Find). This intuition, however, is not crucial for our proof, as we focus solely on punctured oracles.

Lemma 6.12. *For states defined in the preceding sections we have*

$$\begin{aligned}
&\left\| |\Psi_i^{\text{Good}}\rangle_{AD}|0\rangle_J - |\Phi_i\rangle_{ADJ} \right\| \leq \sum_{j=1}^{i} \varepsilon_{\text{step}}(j) \\
&\leq \sum_{j=1}^{i} \max_{a\in\{1,2\}, s_1, s_2 \leq j-1} \left(2\frac{b_a(s_1,s_2)}{N_a} + \frac{b_a(s_1,s_2)}{\sqrt{N_a(N_a - b_a(s_1,s_2))}} \sqrt{\frac{|\mathcal{G}_{\bar{a}}(s_a-1,s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}} \right. \\
&+ \frac{b_a(s_1,s_2)}{N_a} \sqrt{\frac{|\mathcal{G}_{\bar{a}}(s_a-1,s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}} - \left(\sqrt{\frac{|\mathcal{G}_{\bar{a}}(s_a-1,s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}} - 1 \right) \\
&\left. + \frac{b_a(s_a+1,s_{\bar{a}})}{N_a} \sqrt{\frac{|\mathcal{G}_{\bar{a}}(s_a+1,s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}} - \left(\sqrt{\frac{|\mathcal{G}_{\bar{a}}(s_a+1,s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}} - 1 \right) \right). \qquad (6.119)
\end{aligned}$$

Proof. We are going to prove the statement by recursion over the number of queries made by the adversary. The exact derivation is shown in Equation (6.99).

In the following we calculate $\varepsilon_{\text{step}}(j)$ defined in Equation (6.100). For $i = 0$ the statement is true, as $|\Psi_0^{\text{Good}}\rangle|0\rangle_J = |\Phi_0\rangle = |\Psi_0\rangle|0\rangle_J$.

From Equations (6.106), (6.111), and (6.118) we know how querying works for $|\Psi_{j-1}^{\text{Good}}\rangle$, now we distinguish two types of errors compared to $|\Psi_j^{\text{Good}}\rangle|0\rangle_J$: an additive error of adding a small-weight state to the original one and a multiplicative error where one branch of the superposition is multiplied by some factor.

The additive error includes all states of small-weight states multiplied by $|0\rangle_J$ with the superscript Bad. In the branches of the superposition where we add or remove an entry from the database we see that we recover $|\Psi_j^{\text{Good}}\rangle|0\rangle_J$ after multiplying a branch of $\mathsf{H} \setminus \mathsf{V}_R \mathsf{U}_{j-1}|\Psi_{j-1}^{\text{Good}}\rangle|0\rangle_J$ by some factor.

Our approach to the rest of the proof consists of first dealing with the additive and later with the multiplicative error. To this end let us define $|\psi_j^\times\rangle_{ADJ}$ as the state $\overline{\mathsf{J}}_R \mathsf{H} \setminus \mathsf{V}_R \mathsf{U}_{j-1}|\Psi_{j-1}^{\text{Good}}\rangle|0\rangle_J$ with all branches classified as the additive error excluded. By "classified as the additive error" we mean states with superscript Bad and highlighted in red in Equations (6.106, 6.111, 6.118). The new state is defined as

$$|\psi_j^\times\rangle_{ADJ} := \left(\sum_{a,s_1,s_2} |\Psi_j^{\text{Good}}(\text{NOT}; a, s_1, s_2)\rangle \right.$$

$$
\begin{aligned}
&+\sqrt{\frac{N_a - b_a(s_a+1, s_{\bar{a}})}{N_a}}\sqrt{\frac{|\mathcal{G}_{\bar{a}}(s_a+1, s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1, s_2)|}}|\Psi_j^{\text{Good}}(\text{ADD}; a, s_1, s_2)\rangle \\
&+|\Psi_j^{\text{Good}}(\text{UPD}; a, s_1, s_2)\rangle \\
&+\sqrt{\frac{N_a - b_a(s_1, s_2)}{N_a}}\sqrt{\frac{|\mathcal{G}_{\bar{a}}(s_a-1, s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1, s_2)|}}|\Psi_j^{\text{Good}}(\text{REM}; a, s_1, s_2)\rangle\Bigg)|0\rangle_J, \qquad (6.120)
\end{aligned}
$$

where the states above correspond to branches of superposition where we do nothing (NOT, for $\eta = 0$), add an entry, update the database, and remove an entry from D. Bounding the difference of the states is done as follows

$$
\begin{aligned}
&\left\||\Psi_j^{\text{Good}}\rangle|0\rangle_J - \mathsf{J}_R\mathsf{H}\setminus\mathsf{V}_R\mathsf{U}_{j-1}|\Psi_{j-1}^{\text{Good}}\rangle|0\rangle_J\right\| \\
&\leq \left\||\Psi_j^{\text{Good}}\rangle|0\rangle_J - |\psi_j^{\times}\rangle_{ADJ}\right\| + \left\||\psi_j^{\times}\rangle_{ADJ} - \mathsf{J}_R\mathsf{H}\setminus\mathsf{V}_R\mathsf{U}_{j-1}|\Psi_{j-1}^{\text{Good}}\rangle|0\rangle_J\right\|. \qquad (6.121)
\end{aligned}
$$

The second term above is just the norm of all states amplifying the additive error—we call them the bad states.

We bound the additive error $\||\psi_j^{\times}\rangle_{ADJ} - \mathsf{J}_R\mathsf{H}\setminus\mathsf{V}_R\mathsf{U}_{j-1}|\Psi_{j-1}^{\text{Good}}\rangle|0\rangle_J\|$ by first splitting the three cases underlined above:

$$
\left\||\Psi_j^{\text{Bad}}\rangle\right\| \leq \left\||\Psi_{j,1}^{\text{Bad}}(\text{UPD})\rangle\right\| + \left\||\Psi_{j,2}^{\text{Bad}}(\text{UPD})\rangle\right\| + \left\||\Psi_j^{\text{Bad}}(\text{REM})\rangle\right\|, \qquad (6.122)
$$

where $|\Psi_j^{\text{Bad}}\rangle$ is the sum of all three bad states, the bound follows from the triangle inequality.

Calculating all of the three norms above is done by first focusing on a particular interface that is queried and by focusing on particular sizes of databases:

$$
\left\||\Psi_j^{\text{Bad}}\rangle\right\| = \sqrt{\sum_a \sum_{s_1,s_2=0}^{j} |\beta(a, s_1, s_2)|^2 \left\||\Psi_j^{\text{Bad}}(a, s_1, s_2)\rangle\right\|^2}, \qquad (6.123)
$$

where $\beta(a, s_1, s_2)$ is the amplitude of the good state projected to states with the specified parameters: For a projector P_{a,s_1,s_2} to adversaries that query interface a and databases of sizes s_1 and s_2 we have $\beta(a, s_1, s_2) := \mathsf{P}_{a,s_1,s_2}|\Psi_j^{\text{Good}}\rangle$ and $|\Psi_j^{\text{Bad}}(a, s_1, s_2)\rangle := \mathsf{P}_{a,s_1,s_2}|\Psi_j^{\text{Bad}}\rangle$.

Additive errors Dealing with additive errors, we begin with the UPD branch. In the bad states in the UPD case, Equation (6.111), we need to take special care of $\sum_{y_{s_a}^a \in \mathcal{B}(x|D(\vec{x}\setminus\{x\}))} \omega_N^{(\eta_{s_a}^a+\eta)y_{s_a}^a}$; This is a a complex number that depends on $\eta_{s_a}^a$, so it enters the norm in a non-trivial way. The first step is a change of variables: Instead of summing over elements of the bad state we sum over $y_{s_a}^a \in [b_a(s_1, s_2)]$ and change $y_{s_a}^a$ in the expression to $\mathcal{B}_a(x \mid D(\vec{x}\setminus\{x\}))(y_{s_a}^a)$, by which we denote the $y_{s_a}^a$-th element of $\mathcal{B}_a(x \mid D(\vec{x}\setminus\{x\}))$. Note that there is a natural order in the bad set, as $\mathcal{Y}_a = [N_a]$.

Given the change of variables we can use the triangle inequality to focus on the norm of a state with a single phase factor $\omega_N^{(\eta^a_{s_a}+\eta)\mathcal{B}_a(x|D(\vec{x}\setminus\{x\}))(y^a_{s_a})}$, instead of the whole sum:

$$\begin{aligned}&\left\| |\Psi_j^{\text{Bad}}(\text{UPD}; a, s_1, s_2)\rangle \right\| \\ &\leq \sum_{y^a_{s_a}\in[b_a(s_1,s_2)]} \left\| |\Psi_j^{\text{Bad}}(\text{UPD}; a, s_1, s_2, \mathcal{B}_a(x \mid D(\vec{x}\setminus\{x\}))(y^a_{s_a}))\rangle \right\|, \qquad (6.124)\end{aligned}$$

where we omit the index of the UPD errors because the techniques here work in almost the same way for both states. The input $D(\vec{x}\setminus\{x\})$ should not be treated as an actual argument of the state, we still consider the superposition over different inputs, we just mean that in the state $|\Psi_j^{\text{Bad}}(\text{UPD}; a, s_1, s_2)\rangle$ we change the variable $y^a_{s_a}$. In what follows we denote the state on the right hand side of the above equation by $|\Psi_j^{\text{Bad}}(\text{UPD}; a, s_1, s_2, \mathcal{B}'(y^a_{s_a}))\rangle$.

Now we focus on the state with a fixed $\mathcal{B}'(y^a_{s_a})$, we bound the norm of this state.

Claim 6.13. *For all* $y^a_{s_a} \in [b_a(s_1,s_2)]$

$$\left\| |\Psi_{j,1}^{\text{Bad}}(\text{UPD}; a, s_1, s_2, \mathcal{B}_a(x \mid D(\vec{x}\setminus\{x\}))(y^a_{s_a}))\rangle \right\| \leq \frac{1}{N_a} \text{ and} \qquad (6.125)$$

$$\begin{aligned}&\left\| |\Psi_{j,2}^{\text{Bad}}(\text{UPD}; a, s_1, s_2, \mathcal{B}_a(x \mid D(\vec{x}\setminus\{x\}))(y^a_{s_a}))\rangle \right\| \\ &\leq \sqrt{\frac{1}{N_a(N_a - b_a(s_1,s_2))}}\sqrt{\frac{|\mathcal{G}_{\bar{a}}(s_a-1, s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}}. \qquad (6.126)\end{aligned}$$

Proof. Our idea for the proof is to first show that the norm of a good state in the UPD branch with a modified sum over $y^a_{s_a}$ is not greater than 1. Then to prove that the norm of $|\Psi_j^{\text{Bad}}(\text{UPD}; a, s_1, s_2, \mathcal{B}_a(x \mid D(\vec{x}\setminus\{x\}))(y^a_{s_a}))\rangle$ multiplied by the corresponding right hand side of Equation (6.125) and (6.126) equals the norm of the good state we mentioned earlier.

We start by defining two states:

$$\begin{aligned}&\sum_{x,\eta,a,\vec{x},\vec{\eta},w} \alpha_{x,\eta,a,\vec{x},\vec{\eta},w} |x,\eta,a\rangle_{A^{XYI}} |\psi(x,\eta,a,\vec{x},\vec{\eta},w)\rangle_{A^W} \\ &\sum_{\vec{y}_a\in\mathcal{G}_a(\vec{x}_a\setminus\{x\},\vec{x}_{\bar{a}})} \frac{1}{\sqrt{|\mathcal{G}_a(s_a-1,s_{\bar{a}})|}} \omega_{N_a}^{\vec{\eta}_a\cdot\vec{y}_a} |(x^a_1,y^a_1),\ldots,(x^a_{s_a-1},y^a_{s_a-1})\rangle_{D_a(\vec{x}_a\setminus\{x\})} \\ &\sum_{\vec{y}_{\bar{a}}\in\mathcal{G}_{\bar{a}}(\vec{x}_1,\vec{x}_2|\vec{y}_a)} \frac{1}{\sqrt{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}} \omega_{N_{\bar{a}}}^{\vec{\eta}_{\bar{a}}\cdot\vec{y}_{\bar{a}}} |(x^{\bar{a}}_1,y^{\bar{a}}_1),\ldots,(x^{\bar{a}}_{s_{\bar{a}}},y^{\bar{a}}_{s_{\bar{a}}})\rangle_{D_{\bar{a}}(\vec{x}_{\bar{a}})} \\ &\sum_{y^a_{s_a+1},\ldots,y^a_q\in[N_a]} \frac{1}{\sqrt{N_a^{q-s_a}}} |(\perp,y^a_{s_a+1}),\ldots,(\perp,y^a_q)\rangle_{D_a(\perp)} \\ &\sum_{y^{\bar{a}}_{s_{\bar{a}}+1},\ldots,y^{\bar{a}}_q\in[N_{\bar{a}}]} \frac{1}{\sqrt{N_{\bar{a}}^{q-s_{\bar{a}}}}} |(\perp,y^{\bar{a}}_{s_{\bar{a}}+1}),\ldots,(\perp,y^{\bar{a}}_q)\rangle_{D_{\bar{a}}(\perp)}\end{aligned}$$

$$\otimes\begin{cases}\sum_{y^a_{s_a}\in\mathcal{B}_a(x|D(\vec{x}\setminus\{x\}))}\frac{1}{\sqrt{b_a(s_1,s_2)}}\omega_N^{(\eta^a_{s_a}+\eta)y^a_{s_a}}|x,y^a_{s_a}\rangle_{D_a(x)} =: |\overline{\Psi}^{\text{Good}}_j(\text{UPD};a,s_1,s_2)\rangle,\\ \sum_{y^a_{s_a}\in[N_a]}\frac{1}{\sqrt{N_a}}\omega_{N_a}^{(\eta^a_{s_a}+\eta)y^a_{s_a}}|x,y^a_{s_a}\rangle_{D_a(x)} =: |\widetilde{\Psi}^{\text{Good}}_j(\text{UPD};a,s_1,s_2)\rangle.\end{cases} \tag{6.127}$$

The first one, $|\overline{\Psi}^{\text{Good}}_j(\text{UPD};a,s_1,s_2)\rangle$ is the one that we use in the last step of the proof, as described in the previous paragraph. The second one will be used to show that the norm of $|\overline{\Psi}^{\text{Good}}_j(\text{UPD};a,s_1,s_2)\rangle$ is bounded by 1.

One more statement that we need to prove is that $\left\||\widetilde{\Psi}^{\text{Good}}_j(\text{UPD};a,s_1,s_2)\rangle\right\| = \left\||\Psi^{\text{Good}}_j(\text{UPD};a,s_1,s_2)\rangle\right\|$. To prove our statement we need to note two things. First being that the projection is done on a single branch of superposition, so its weight remains the same for all states considered in this paragraph. The second being that the database register is normalized, by that we mean that just dividing by the square-root of the number of distinct databases yields a normalized database register. The first fact is obvious when noting that to define the two good states we can first take the corresponding $(\text{UPD};a,s_1,s_2)$ branch of the non-punctured state and then project to databases that are not in R. By the up-punctured state we mean the state generated by the same adversary interacting with the un-punctured oracle. The second observation comes from the fact that the good state is normalized. Hence, just taking care of proper weighting of the database entries (normalization factors for every y^a_i) ensures the database register is normalized.

The fact that the state with $\sum_{y^a_{s_a}\in[N_a]}$ is sub-normalized is important because now we can bound the norm of $|\overline{\Psi}^{\text{Good}}_j(\text{UPD};a,s_1,s_2)\rangle$. Having in mind that $\sum_{y^a_{s_a}\in\mathcal{B}_a(x|D(\vec{x}\setminus\{x\}))} = \sum_{y^a_{s_a}\in[N_a]} - \sum_{y^a_{s_a}\notin\mathcal{B}_a(x|D(\vec{x}\setminus\{x\}))}$ we see that

$$\begin{aligned}&b_a(s_1,s_2)\left\||\overline{\Psi}^{\text{Good}}_j(\text{UPD};a,s_1,s_2)\rangle\right\|^2\\ &= N_a\left\||\widetilde{\Psi}^{\text{Good}}_j(\text{UPD};a,s_1,s_2)\rangle\right\|^2\\ &-(N_a-b_a(s_1,s_2))\left\||\Psi^{\text{Good}}_j(\text{UPD};a,s_1,s_2)\rangle\right\|^2 \le b_a(s_1,s_2),\end{aligned} \tag{6.128}$$

hence $\left\||\overline{\Psi}^{\text{Good}}_j(\text{UPD};a,s_1,s_2)\rangle\right\|^2 \le 1$.

Now that we know that $|\overline{\Psi}^{\text{Good}}_j(\text{UPD};a,s_1,s_2)\rangle$ is sub-normalized we show that

$$\left\||\overline{\Psi}^{\text{Good}}_j(\text{UPD};a,s_1,s_2,\mathcal{B}'(y^a_{s_a}))\rangle\right\| \le \frac{1}{\sqrt{b_a(s_1,s_2)}}. \tag{6.129}$$

To prove this bound, consider measuring register $D_a(x)$ of $|\overline{\Psi}^{\text{Good}}_j(\text{UPD};a,s_1,s_2)\rangle$ in the computational basis. The probability of getting any outcome $y^a_{s_a}$ is necessarily $\frac{1}{b_a(s_1,s_2)}$, as the outputs of the oracle are uniformly random. The post-measurement state, for an outcome $y^a_{s_a}$, is $\sqrt{b_a(s_1,s_2)}\cdot$ $|\overline{\Psi}_j^{\text{Good}}(\text{UPD}; a, s_1, s_2, \mathcal{B}'(y^a_{s_a}))\rangle$. Naturally, norm of this post-measurement state is at most 1.

Now we can use the state $|\overline{\Psi}_j^{\text{Good}}(\text{UPD}; a, s_1, s_2, \mathcal{B}'(y^a_{s_a}))\rangle$ to analyze the norm of $|\Psi_j^{\text{Bad}}(\text{UPD}; a, s_1, s_2, \mathcal{B}'(y^a_{s_a}))\rangle$. First let us inspect the norm squared of the bad state:

$$\left\| |\Psi_j^{\text{Bad}}(\text{UPD}; a, s_1, s_2, \mathcal{B}'(y^a_{s_a}))\rangle \right\|^2 = \sum_{x,\eta,a,\vec{x},\vec{\eta}',\vec{\eta},w',w} \sum_{\eta^{a\prime}_{s_a},\eta^a_{s_a}} \bar{\alpha}'_{x,\eta,a,\vec{x},\vec{\eta}',\eta^{a\prime}_{s_a},w'} \alpha'_{x,\eta,a,\vec{x},\vec{\eta},\eta^a_{s_a},w}$$
$$\langle \psi(x,\eta,a,\vec{x},\vec{\eta}',\eta^{a\prime}_{s_a},w') | \psi(x,\eta,a,\vec{x},\vec{\eta},\eta^a_{s_a},w)\rangle$$
$$\sum_{\vec{y}_a \in \mathcal{G}_a(\vec{x}_a \setminus \{x\}, \vec{x}_{\bar{a}})} \frac{1}{|\mathcal{G}_a(s_a-1,s_{\bar{a}})|} \omega_{N_a}^{\vec{\eta}_a\cdot\vec{y}_a} \sum_{\vec{y}_{\bar{a}} \in \mathcal{G}_{\bar{a}}(\vec{x}_1,\vec{x}_2|\vec{y}_a)} \frac{1}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|} \omega_{N_{\bar{a}}}^{\vec{\eta}_{\bar{a}}\cdot\vec{y}_{\bar{a}}}$$
$$\frac{1}{N_a^2(N_a - b_a(s_1,s_2))} \bar{\omega}_{N_a}^{(\eta^{a\prime}_{s_a}+\eta)\mathcal{B}'(y^a_{s_a})} \omega_{N_a}^{(\eta^a_{s_a}+\eta)\mathcal{B}'(y^a_{s_a})} \gamma^2 \underbrace{\sum_{y^{a\prime}_{s_a}\in[\nu]}}_{=\nu}, \tag{6.130}$$

where $\nu = N_a - b_a(s_1,s_2)$ and $\gamma = 1$ for $|\Psi_{j,1}^{\text{Bad}}(\text{UPD}; a, s_1, s_2, \mathcal{B}'(y^a_{s_a}))\rangle$ and $\nu = N_a$ and $\gamma = \sqrt{\frac{|\mathcal{G}_{\bar{a}}(s_a-1,s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}}$ for $|\Psi_{j,2}^{\text{Bad}}(\text{UPD}; a, s_1, s_2, \mathcal{B}'(y^a_{s_a}))\rangle$ (in the second case the sum goes over $y^{a\prime}_{s_a} \notin \mathcal{B}_a(x \mid D(\vec{x} \setminus \{x\}))$ instead of $y^{a\prime}_{s_a} \in [\nu]$). It is easy to notice, that the only difference between Equation (6.130) and norm squared of $|\overline{\Psi}_j^{\text{Good}}(\text{UPD}; a, s_1, s_2, \mathcal{B}'(y^a_{s_a}))\rangle$ lies in the factor $\frac{\nu\gamma^2}{N_a^2(N_a - b_a(s_1,s_2))}$. This factor in the modified good state equals $\frac{1}{b_a(s_1,s_2)}$. This observation implies that

$$\begin{aligned}&\left\| |\Psi_j^{\text{Bad}}(\text{UPD}; a, s_1, s_2, \mathcal{B}'(y^a_{s_a}))\rangle \right\| \\ &= \sqrt{\frac{b(s)\cdot\nu\gamma^2}{N_a^2(N_a - b_a(s_1,s_2))}} \left\| |\overline{\Psi}_j^{\text{Good}}(\text{UPD}; s, \mathcal{B}'(y^a_{s_a}))\rangle \right\|.\end{aligned} \tag{6.131}$$

Together with the bound on the norm in the left hand side this proves the claimed bounds. □

Claim 6.13, together with the bound from Equation (6.124) gives us:

$$\left\| |\Psi_{j,1}^{\text{Bad}}(\text{UPD}; a, s_1, s_2)\rangle \right\| \le \frac{b_a(s_1,s_2)}{N_a}, \tag{6.132}$$
$$\left\| |\Psi_{j,2}^{\text{Bad}}(\text{UPD}; a, s_1, s_2)\rangle \right\| \le \frac{b_a(s_1,s_2)}{\sqrt{N_a(N_a - b_a(s_1,s_2))}} \sqrt{\frac{|\mathcal{G}_{\bar{a}}(s_a-1,s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}}. \tag{6.133}$$

The bounds from Equation (6.132) in Equation (6.123) give us the bound on the additive error in the UPD branch. The additive error for the REM branch ($|\Psi_j^{\text{Bad}}(\text{REM})\rangle$ in Equation (6.118)) is much easier to calculate: As register $D(x)$

is normalized and all the rest of the state is the same as $|\Psi_j^{\mathrm{Good}}(\mathrm{REM})\rangle$, the only error comes from the factor $\frac{b_a(s_1,s_2)}{N_a}$. To calculate the norm of the state we can follow the analysis of Equation (6.130). Finally we get:

$$\left\| |\Psi_{j,1}^{\mathrm{Bad}}(\mathrm{UPD})\rangle \right\| \leq \max_{a,s_1,s_2} \left(\frac{b_a(s_1,s_2)}{N_a} \right), \tag{6.134}$$

$$\left\| |\Psi_{j,2}^{\mathrm{Bad}}(\mathrm{UPD})\rangle \right\| \leq \max_{a,s_1,s_2} \left(\frac{b_a(s_1,s_2)}{\sqrt{N_a(N_a - b_a(s_1,s_2))}} \sqrt{\frac{|\mathcal{G}_{\bar{a}}(s_a - 1, s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}} \right), \tag{6.135}$$

$$\left\| |\Psi_{j}^{\mathrm{Bad}}(\mathrm{REM})\rangle \right\| \leq \max_{a,s_1,s_2} \left(\frac{b_a(s_1,s_2)}{N_a} \right), \tag{6.136}$$

where $a \in \{1,2\}$ and $s_1, s_2 \leq j-1$.

Multiplicative errors The multiplicative error is a factor that multiplies a part of the state $|\psi_j^\times\rangle_{ADJ}$. Similarly as before we need to take care of the fact that the joint state of the adversary and the oracle is a sum over databases of different sizes and queries to different interfaces:

$$|\psi_j^\times\rangle = \sum_{a,s_1,s_2} |\psi_j^\times(a,s_1,s_2)\rangle, \tag{6.137}$$

where the states $|\psi_j^\times(a,s_1,s_2)\rangle$ are orthogonal. The terms are also orthogonal in $|\Psi_j^{\mathrm{Good}}\rangle = \sum_{a,s_1,s_2} |\Psi_j^{\mathrm{Good}}(a,s_1,s_2)\rangle$.

There are two sources of multiplicative errors, ADD from Equation (6.106) and REM from Equation (6.118), we split the two sources with the triangle inequality. We deal with both in the same way, just the final bound is different.

Let us write down the two parts, one affected by the error and the second not:

$$|\Psi_j^{\mathrm{Good}}\rangle_{AD}|0\rangle_J = \sum_{a,s_1,s_2} \alpha(a,s_1,s_2)|\varphi_1(a,s_1,s_2)\rangle + \beta(a,s_1,s_2)|\varphi_2(a,s_1,s_2)\rangle, \tag{6.138}$$

$$\begin{aligned} |\psi_j^\times\rangle_{ADJ} = \sum_{a,s_1,s_2} & \alpha(a,s_1,s_2)|\varphi_1(a,s_1,s_2)\rangle \\ & + (1+g)\sqrt{1-e}\,\beta(a,s_1,s_2)|\varphi_2(a,s_1,s_2)\rangle, \end{aligned} \tag{6.139}$$

where $(1+g)\sqrt{1-e}$ is the multiplicative error, in the case ADD the error is $g = \sqrt{\frac{|\mathcal{G}_{\bar{a}}(s_a+1,s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}} - 1$ and $e = \frac{b_a(s_a+1,s_{\bar{a}})}{N_a}$. In the case REM the error is $g = \sqrt{\frac{|\mathcal{G}_{\bar{a}}(s_a-1,s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}} - 1$ and $e = \frac{b_a(s_1,s_2)}{N_a}$. We know that
$\sum_{a,s_1,s_2} |\alpha(a,s_1,s_2)|^2 + |\beta(a,s_1,s_2)|^2 \leq 1$, because we excluded a single branch

of the superposition, for ADD and REM. This inequality implies $\sum_{a,s_1,s_2} |\beta(a,s_1,s_2)|^2 \leq 1$. We continue with the bound

$$\begin{aligned}
&\left\| |\psi_j^{\times}\rangle_{ADJ} - |\Psi_j^{\text{Good}}\rangle_{AD}|0\rangle_J \right\| \\
&= \left\| \sum_{a,s_1,s_2} \left(1-(1+g)\sqrt{1-e}\right) \beta(a,s_1,s_2)|\varphi_2(a,s_1,s_2)\rangle \right\| && (6.140)\\
&= \sqrt{\sum_{a,s_1,s_2} \left(1-(1+g)\sqrt{1-e}\right)^2 |\beta(a,s_1,s_2)|^2} \leq \max_{a,s_1,s_2} \left(1-(1+g)\sqrt{1-e}\right) \\
&\leq \max_{a,s_1,s_2} \left((1+g)e-g\right). && (6.141)
\end{aligned}$$

Maximization is done over $a \in \{1,2\}$ and $s_1, s_2 \leq j-1$.

Bound on one step From Equations (6.121), (6.134), and (6.141) (for the two sources of error) the bound on the single step is

$$\begin{aligned}
\varepsilon_{\text{step}}(j) \leq &\max_{a\in\{1,2\},s_1,s_2\leq j-1} \left(2\frac{b_a(s_1,s_2)}{N_a} + \frac{b_a(s_1,s_2)}{\sqrt{N_a(N_a-b_a(s_1,s_2))}} \sqrt{\frac{|\mathcal{G}_{\bar{a}}(s_a-1,s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}} \right.\\
&+ \frac{b_a(s_1,s_2)}{N_a}\sqrt{\frac{|\mathcal{G}_{\bar{a}}(s_a-1,s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}} - \left(\sqrt{\frac{|\mathcal{G}_{\bar{a}}(s_a-1,s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}} - 1\right)\\
&\left.+ \frac{b_a(s_a+1,s_{\bar{a}})}{N_a}\sqrt{\frac{|\mathcal{G}_{\bar{a}}(s_a+1,s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}} - \left(\sqrt{\frac{|\mathcal{G}_{\bar{a}}(s_a+1,s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}} - 1\right)\right) \qquad (6.142)
\end{aligned}$$

and the final bound is

$$\begin{aligned}
&\left\| |\Psi_i^{\text{Good}}\rangle_{AD}|0\rangle_J - |\Phi_i\rangle_{ADJ} \right\| \\
&\leq \sum_{j=1}^{i} \max_{a\in\{1,2\},s_1,s_2\leq j-1} \left(2\frac{b_a(s_1,s_2)}{N_a} + \frac{b_a(s_1,s_2)}{\sqrt{N_a(N_a-b_a(s_1,s_2))}} \sqrt{\frac{|\mathcal{G}_{\bar{a}}(s_a-1,s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}} \right.\\
&+ \frac{b_a(s_1,s_2)}{N_a}\sqrt{\frac{|\mathcal{G}_{\bar{a}}(s_a-1,s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}} - \left(\sqrt{\frac{|\mathcal{G}_{\bar{a}}(s_a-1,s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}} - 1\right)\\
&\left.+ \frac{b_a(s_a+1,s_{\bar{a}})}{N_a}\sqrt{\frac{|\mathcal{G}_{\bar{a}}(s_a+1,s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}} - \left(\sqrt{\frac{|\mathcal{G}_{\bar{a}}(s_a+1,s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}} - 1\right)\right). \qquad (6.143)
\end{aligned}$$

□

6.4.1.7 Bound on $\varepsilon_{\text{Find}}(i)$

Our task here is bounding the norm of $\left\| \mathsf{J}_R\mathsf{U}_i\mathsf{H} \setminus \mathsf{V}_R\mathsf{U}_{i-1}|\Psi_{i-1}^{\text{Good}}\rangle \right\|$. All states (among the states defined in Section 6.4.1.5) that give non-zero contributions to this norm are the ones that we give the superscript Find, they contain $|1\rangle_J$.

Lemma 6.14. *For states defined in preceding sections we have*

$$\begin{aligned}
&\left\| \mathsf{J}_R \mathsf{U}_i \mathsf{H} \setminus \mathsf{V}_R \mathsf{U}_{i-1} |\Psi^{\text{Good}}_{i-1}\rangle \right\| = \varepsilon_{\text{Find}}(i) \\
&\leq \max_{a\in\{1,2\}, s_1, s_2 \leq i-1} \left(\sqrt{\frac{N_a - b_a(s_a+1, s_{\bar{a}})}{N_a}} \sqrt{\frac{|\mathcal{H}^{\text{ADD}}_a(s_1,s_2)|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}} \right. \\
&+ \sqrt{\frac{b_a(s_a+1, s_{\bar{a}})}{N_a}} + \frac{b_a(s_1,s_2)^{3/2}}{N_a\sqrt{N_a - b_a(s_1,s_2)}} \\
&+ \frac{b_a(s_1,s_2)}{\sqrt{N_a(N_a - b_a(s_1,s_2))}} \operatorname{sgn}\left(\left| \mathcal{H}^{\text{REM}}_a(s_1,s_2) \right| \right) \sqrt{\frac{|\mathcal{G}_{\bar{a}}(s_a - 1, s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}} \\
&\left. + \sqrt{\frac{N_a - b_a(s_1,s_2)}{N_a}} \sqrt{\frac{|\mathcal{H}^{\text{REM}}_a(s_1,s_2)|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}} + \frac{\sqrt{b_a(s_1,s_2)(N_a - b_a(s_1,s_2))}}{N_a} \right). \quad (6.144)
\end{aligned}$$

Proof. For all Find states we start bounding the norm by splitting the norm by a and sizes of the databases, like in Equation (6.123). Let us now go through the three important modes of operation, i.e. adding, updating, or removing from the database.

The ADD case For the state $|\Psi^{\text{Find}}_{j,1}(\text{ADD})\rangle$ we first analyze its norm squared; Forgetting for now the factors multiplying the whole state, the situation is similar to Equation (6.128) but instead of focusing on the sum over $y^a_{s_a+1}$ we show that the norm squared of $|\Psi^{\text{Find}}_{j,1}(\text{ADD})\rangle$ is a sum of norms of states that differ in the sum over $\vec{y}_{\bar{a}}$. The states on the right hand side have the sum over $\vec{y}_{\bar{a}}$ split according to

$$\sum_{\vec{y}_{\bar{a}} \in \mathcal{G}_{\bar{a}}(\vec{x}_1,\vec{x}_2|\vec{y}_a) \setminus \mathcal{G}_{\bar{a}}(\vec{x}_a \cup \{x\}, \vec{x}_{\bar{a}}|\vec{y}_a)} = \sum_{\vec{y}_{\bar{a}} \in \mathcal{G}_{\bar{a}}(\vec{x}_1,\vec{x}_2|\vec{y}_a)} - \sum_{\vec{y}_{\bar{a}} \in \mathcal{G}_{\bar{a}}(\vec{x}_a \cup \{x\}, \vec{x}_{\bar{a}}|\vec{y}_a)} . \quad (6.145)$$

Note that both states constructed with sums over sets $\mathcal{G}_{\bar{a}}(\vec{x}_1, \vec{x}_2 \mid \vec{y}_a)$ and $\mathcal{G}_{\bar{a}}(\vec{x}_a \cup \{x\}, \vec{x}_{\bar{a}} \mid \vec{y}_a)$ have unit norm. The former state has norm equal to $|\Psi^{\text{Good}}_{j-1}(\text{ADD})\rangle$, the norm of this state does not change when replacing register $D(x)$ with an empty entry of the database. The latter state is just the good state before the application of the adversary's unitary: $\mathsf{U}_j^\dagger |\Psi^{\text{Good}}_j(\text{ADD})\rangle$. This analysis follows the same reasoning as presented in the proof of Claim 6.13. Given that $|\Psi^{\text{Find}}_{j,1}(\text{ADD})\rangle$ without the additional factor $\sqrt{\frac{N_a - b_a(s_a+1,s_{\bar{a}})}{N_a}} \sqrt{\frac{|\mathcal{H}^{\text{ADD}}_a(s_1,s_2)|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}}$ has bounded norm we have the following bound:

$$\left\| |\Psi^{\text{Find}}_{j,1}(\text{ADD})\rangle \right\| \leq \max_{a,s_1,s_2} \sqrt{\frac{N_a - b_a(s_a+1, s_{\bar{a}})}{N_a}} \sqrt{\frac{|\mathcal{H}^{\text{ADD}}_a(s_1,s_2)|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}}. \quad (6.146)$$

The second state has norm bounded in the following way:

$$\left\| |\Psi^{\text{Find}}_{j,2}(\text{ADD})\rangle \right\| \leq \max_{a,s_1,s_2} \sqrt{\frac{b_a(s_a+1, s_{\bar{a}})}{N_a}}. \quad (6.147)$$

This bound holds , because except for the factor in front of the state and register $D_a(x)$ the state is just a good state (one from just before the query we analyze in Equation (6.106)). Moreover register $D(x)$ is normalized (given the fact that η is explicit in the adversary's register).

The UPD **case** In this case we have

$$\left\| |\Psi_{j,1}^{\text{Find}}(\text{UPD})\rangle \right\| \leq \max_{a,s_1,s_2} \frac{b_a(s_1,s_2)^{3/2}}{N_a\sqrt{N_a - b_a(s_1,s_2)}}, \tag{6.148}$$

where we follow the same reasoning as in the proof of Lemma 6.12 and Claim 6.13. For $|\Psi_{j,2}^{\text{Find}}(\text{UPD})\rangle$ we consider the norm square and see that we deal with a difference of two norms with different sets for $\vec{y}_{\bar{a}}$, similarly to $|\Psi_{j,1}^{\text{Find}}(\text{ADD})\rangle$, but the split is done as follows:

$$\sum_{\vec{y}_{\bar{a}} \in \mathcal{G}_{\bar{a}}(\vec{x}_a \setminus \{x\}, \vec{x}_{\bar{a}}|\vec{y}_a) \setminus \mathcal{G}_{\bar{a}}(\vec{x}_1, \vec{x}_2|\vec{y}_a)} = \sum_{\vec{y}_{\bar{a}} \in \mathcal{G}_{\bar{a}}(\vec{x}_a \setminus \{x\}, \vec{x}_{\bar{a}}|\vec{y}_a)} - \sum_{\vec{y}_{\bar{a}} \in \mathcal{G}_{\bar{a}}(\vec{x}_1, \vec{x}_2|\vec{y}_a)} . \tag{6.149}$$

The first state is just $|\Psi_{j,2}^{\text{Bad}}(\text{UPD})\rangle$. The second state is more problematic to deal with , so we just lower bound its norm by 0. We know the bound on $|\Psi_{j,2}^{\text{Bad}}(\text{UPD})\rangle$ so by just taking care of the additional factors we get the bound:

$$\begin{aligned}&\left\| |\Psi_{j,2}^{\text{Find}}(\text{UPD})\rangle \right\| \\ &\leq \max_{a,s_1,s_2} \frac{b_a(s_1,s_2)}{\sqrt{N_a(N_a - b_a(s_1,s_2))}} \text{sgn}\left(\left|\mathcal{H}_a^{\text{REM}}(s_1,s_2)\right|\right) \sqrt{\frac{|\mathcal{G}_{\bar{a}}(s_a-1,s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}},\end{aligned} \tag{6.150}$$

where the sign function ensures that if there is no error in $D_{\bar{a}}$ the norm of the state is 0.

The REM **case** Finally we have

$$\left\| |\Psi_{j,1}^{\text{Find}}(\text{REM})\rangle \right\| \leq \max_{a,s_1,s_2} \sqrt{\frac{N_a - b_a(s_1,s_2)}{N_a}} \sqrt{\frac{|\mathcal{H}_a^{\text{REM}}(s_1,s_2)|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}}, \tag{6.151}$$

that can be derived in the same way as the bound on norm of $|\Psi_{j,1}^{\text{Find}}(\text{ADD})\rangle$. For the second state we have

$$\left\| |\Psi_{j,2}^{\text{Find}}(\text{REM})\rangle \right\| \leq \max_{a,s_1,s_2} \frac{\sqrt{b_a(s_1,s_2)(N_a - b_a(s_1,s_2))}}{N_a} \tag{6.152}$$

and to get it we follow the same reasoning as for $|\Psi_{j,2}^{\text{Find}}(\text{ADD})\rangle$.

We use these bounds and the triangle inequality to bound the second term in Equation (6.96):

$$\left\| \mathsf{J}_R \mathsf{U}_i \mathsf{H} \setminus \mathsf{V}_R \mathsf{U}_{i-1} |\Psi_{i-1}^{\text{Good}}\rangle \right\|$$

$$\leq \max_{a\in\{1,2\},s_1,s_2\leq i-1}\left(\sqrt{\frac{N_a-b_a(s_a+1,s_{\bar{a}})}{N_a}}\sqrt{\frac{|\mathcal{H}_a^{\mathrm{ADD}}(s_1,s_2)|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}}\right.$$
$$+\sqrt{\frac{b_a(s_a+1,s_{\bar{a}})}{N_a}}+\frac{b_a(s_1,s_2)^{3/2}}{N_a\sqrt{N_a-b_a(s_1,s_2)}}$$
$$+\frac{b_a(s_1,s_2)}{\sqrt{N_a(N_a-b_a(s_1,s_2))}}\mathrm{sgn}\left(\left|\mathcal{H}_a^{\mathrm{REM}}(s_1,s_2)\right|\right)\sqrt{\frac{|\mathcal{G}_{\bar{a}}(s_a-1,s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}}$$
$$\left.+\sqrt{\frac{N_a-b_a(s_1,s_2)}{N_a}}\sqrt{\frac{|\mathcal{H}_a^{\mathrm{REM}}(s_1,s_2)|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}}+\frac{\sqrt{b_a(s_1,s_2)(N_a-b_a(s_1,s_2))}}{N_a}\right). \tag{6.153}$$

□

6.4.2 Concrete Relations

In this section we list corollaries important for results that we state in this thesis.

For relations that are defined on a single database (in this thesis we consider preim and coll defined in sections 6.4.2.2 and 6.4.2.1) we can simplify the bound from Lemma 6.11. We omit the differences in $\mathcal{G}_{\bar{a}}$ sets and simplify the terms:

$$\mathrm{preim},\mathrm{coll}:\mathbb{P}\left[\mathrm{Find}:\mathsf{A}[\mathsf{H}\setminus R]\right]$$
$$\leq\sum_{i=1}^{q}\left(\sum_{j=1}^{i-1}\max_{s\leq j-1}\left(3\frac{b(s)}{N}+\frac{b(s)}{\sqrt{N(N-b(s))}}+\frac{b(s+1)}{N}\right)\right.$$
$$\left.+\max_{s\leq i-1}\left(\sqrt{\frac{b(s+1)}{N}}+\frac{b(s)^{3/2}}{N\sqrt{N-b(s)}}+\frac{\sqrt{b(s)(N-b(s))}}{N}\right)\right)^2 \tag{6.154}$$
$$\leq\sum_{i=1}^{q}\left(\sum_{j=1}^{i-1}\max_{s\leq j-1}\left(5\frac{b(s+1)}{\sqrt{N(N-b(q))}}\right)+\max_{s\leq i-1}\left(2\sqrt{\frac{b(s+1)}{N}}+\frac{b(s)^{3/2}}{N\sqrt{N-b(q)}}\right)\right)^2 \tag{6.155}$$
$$\leq\sum_{i=1}^{q}\left(\sum_{j=1}^{i-1}5\frac{b(j)}{\sqrt{N(N-b(q))}}+2\sqrt{\frac{b(i)}{N}}+\frac{b(i)^{3/2}}{N\sqrt{N-b(q)}}\right)^2 \tag{6.156}$$

In the above bound we use the facts that $b(s)$ is a monotonously growing function of s, this is true for all relations we use in this thesis. For the relations that we provide concrete bounds for, the factor $b(s)$ is a linear function of s. To calculate the bounds, we use the following reasoning, properly tailored to each concrete relation:

$$\sum_{j=1}^{i}j^{3/2}\leq\int_1^i\mathrm{d}j\,j^{3/2}\leq\int_0^i\mathrm{d}j\,j^{3/2}=\frac{2}{5}i^{5/2}. \tag{6.157}$$

Moreover $b(s) \leq b(q)$, which we use in the denominator.

In the case that R depends on two databases we need to use the full bound from Lemma 6.11. Whenever the outputs of two databases relate to one another the new entry in the good set is sampled in a way that D is not in R. If outputs of one oracle depend on the inputs of the other, adding a new entry gives a trivial attack, that we exclude. The only scenario that adding a new entry causes errors in the other database is when the other outputs depend on some random function of the new input (that is not accessible for the adversary). This is not the case for the relations that we discuss here, hence we can omit all errors to the other database in the ADD case.

We simplify the additive terms $\sqrt{\frac{|\mathcal{G}_{\bar{a}}(s_a-1,s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}}$, knowing that this factor is larger than 1, the fewer the restrictions from D_a the more good $y^{\bar{a}}$ there are. Moreover $\mathrm{sgn}\left(\left|\mathcal{H}_a^{\mathrm{REM}}(s_1,s_2)\right|\right) = 1$.

An important term that we need to bound is $\sqrt{\frac{|\mathcal{G}_{\bar{a}}(s_a-1,s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}}$. To achieve a constant bound we proceed as follows:

$$\frac{|\mathcal{G}_{\bar{a}}(s_a-1,s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|} = \prod_{k=1}^{s_{\bar{a}}}\left(\frac{N_{\bar{a}} - b_{\bar{a}}(s_a-1,k)}{N_{\bar{a}} - b_{\bar{a}}(s_a,k)}\right) = \prod_{k=1}^{s_{\bar{a}}}\left(1 + \frac{b_{\bar{a}}(s_a,k) - b_{\bar{a}}(s_a-1,k)}{N_{\bar{a}} - b_{\bar{a}}(s_a,k)}\right) \tag{6.158}$$

$$= \exp\left(\sum_{k=1}^{s_{\bar{a}}} \log\left(1 + \frac{b_{\bar{a}}(s_a,k) - b_{\bar{a}}(s_a-1,k)}{N_{\bar{a}} - b_{\bar{a}}(s_a,k)}\right)\right) \tag{6.159}$$

$$\leq \exp\left(\sum_{k=1}^{s_{\bar{a}}} \underbrace{\frac{b_{\bar{a}}(s_a,k) - b_{\bar{a}}(s_a-1,k)}{N_{\bar{a}} - b_{\bar{a}}(s_a,k)}}_{\leq \frac{q^2}{N_{\bar{a}}-q^2}}\right) \leq \exp\left(\frac{q^3}{N_{\bar{a}} - q^2}\right) \leq \exp(2) \leq 3^2, \tag{6.160}$$

where we use the bound $\log(1+x) \leq x$ and the fact that for any of the relations we analyze $\forall i,j \leq q : b_{\bar{a}}(i,j) \leq q^2$.

For bounding the part with $\left|\mathcal{H}_a^{\mathrm{REM}}(s_1,s_2)\right|$ we use the following derivation:

$$\sqrt{\frac{N_a - b_a(s_1,s_2)}{N_a}}\sqrt{\frac{|\mathcal{H}_a^{\mathrm{REM}}(s_1,s_2)|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}}$$

$$= \sqrt{\frac{N_a - b_a(s_1,s_2)}{N_a}}\sqrt{\prod_{k=1}^{s_{\bar{a}}}\left(\frac{N_{\bar{a}} - b_{\bar{a}}(s_a-1,k)}{N_{\bar{a}} - b_{\bar{a}}(s_a,k)}\right) - 1} \tag{6.161}$$

$$= \sqrt{\frac{N_a - b_a(s_1,s_2)}{N_a}}\sqrt{\prod_{k=1}^{s_{\bar{a}}}\left(1 + \frac{b_{\bar{a}}(s_a,k) - b_{\bar{a}}(s_a-1,k)}{N_{\bar{a}} - b_{\bar{a}}(s_a,k)}\right) - 1} \tag{6.162}$$

$$\leq \sqrt{\frac{N_a - b_a(s_1,s_2)}{N_a}} \sqrt{\prod_{k=1}^{s_{\bar{a}}} \left(\underbrace{1 + \frac{\max_{k \leq s_{\bar{a}}} \{b_{\bar{a}}(s_a,k) - b_{\bar{a}}(s_a-1,k)\}}{N_{\bar{a}} - b_{\bar{a}}(s_a,s_{\bar{a}})}}_{\leq \exp\left(\frac{\max_{k \leq s_{\bar{a}}} \{b_{\bar{a}}(s_a,k) - b_{\bar{a}}(s_a-1,k)\}}{N_{\bar{a}} - b_{\bar{a}}(s_a,s_{\bar{a}})}\right)} \right) - 1} \tag{6.163}$$

$$\leq \sqrt{2 \frac{2q \max_{k \leq s_{\bar{a}}} (b_{\bar{a}}(s_a,k) - b_{\bar{a}}(s_a-1,k))}{N_a}}, \tag{6.164}$$

where the last inequality comes from bounding $e^x - 1 \leq 2x$ (valid for $0 \leq x \leq 1$) and assuming that $\frac{N_a - b_a(s_1,s_2)}{N_{\bar{a}} - b_{\bar{a}}(s_1,s_2)} \leq 2$, which is true for the parameters in the problems we analyze.

Finally the bound valid for relations Comp, Rate-1/3, EDM, and EDMD, defined in sections 6.4.2.4, 6.4.2.5, 6.4.2.6, and 6.4.2.7 respectively, is

$$\begin{aligned}
&\textsc{Comp}, \textsc{Rate-1/3}, \text{EDM}, \text{EDMD} : \mathbb{P}\left[\text{Find} : \mathsf{A}[\mathsf{H} \setminus R]\right] \\
&\leq \sum_{i=1}^{q} \left(\sum_{j=1}^{i-1} \max_{a \in \{1,2\}, s_1, s_2 \leq j-1} \left(2\frac{b_a(s_1,s_2)}{N_a} + \frac{b_a(s_a+1,s_{\bar{a}})}{N_a} \right.\right. \\
&\left. + \frac{b_a(s_1,s_2)}{\sqrt{N_a(N_a - b_a(s_1,s_2))}} \sqrt{\frac{|\mathcal{G}_{\bar{a}}(s_a-1,s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}} + \frac{b_a(s_1,s_2)}{N_a} \sqrt{\frac{|\mathcal{G}_{\bar{a}}(s_a-1,s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}} \right) \\
&+ \max_{a \in \{1,2\}, s_1, s_2 \leq i-1} \left(\sqrt{\frac{b_a(s_a+1,s_{\bar{a}})}{N_a}} + \frac{b_a(s_1,s_2)^{3/2}}{N_a \sqrt{N_a - b_a(s_1,s_2)}} \right. \\
&+ \frac{b_a(s_1,s_2)}{\sqrt{N_a(N_a - b_a(s_1,s_2))}} \sqrt{\frac{|\mathcal{G}_{\bar{a}}(s_a-1,s_{\bar{a}})|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}} \\
&\left.\left. + \sqrt{\frac{N_a - b_a(s_1,s_2)}{N_a}} \sqrt{\frac{|\mathcal{H}_a^{\text{REM}}(s_1,s_2)|}{|\mathcal{G}_{\bar{a}}(s_1,s_2)|}} + \frac{\sqrt{b_a(s_1,s_2)(N_a - b_a(s_1,s_2))}}{N_a} \right) \right)^2
\end{aligned} \tag{6.165}$$

$$\begin{aligned}
&\leq \sum_{i=1}^{q} \left(\sum_{j=1}^{i-1} \max_{a \in \{1,2\}, s_1, s_2 \leq j-1} \left(\frac{b_a(s_1,s_2)}{N_a} + \frac{b_a(s_a+1,s_{\bar{a}})}{N_a} \right.\right. \\
&\left. + 3\frac{b_a(s_1,s_2)}{\sqrt{N_a(N_a - b_a(s_1,s_2))}} + \frac{b_a(s_1,s_2)}{N_a} + 3\frac{b_a(s_1,s_2)}{N_a} \right) \\
&+ \max_{a \in \{1,2\}, s_1, s_2 \leq i-1} \left(\sqrt{\frac{b_a(s_a+1,s_{\bar{a}})}{N_a}} + \frac{b_a(s_1,s_2)^{3/2}}{N_a \sqrt{N_a - b_a(s_1,s_2)}} \right. \\
&+ 3\frac{b_a(s_1,s_2)}{\sqrt{N_a(N_a - b_a(s_1,s_2))}}
\end{aligned}$$

$$+\sqrt{2\frac{2q\max_{k\leq s_{\bar{a}}}\left(b_{\bar{a}}(s_a,k)-b_{\bar{a}}(s_a-1,k)\right)}{N_a}}+\frac{\sqrt{b_a(s_1,s_2)(N_a-b_a(s_1,s_2))}}{N_a}\Bigg)\Bigg)^2 \tag{6.166}$$

$$\leq \sum_{i=1}^{q}\left(\sum_{j=1}^{i-1}\max_{a\in\{1,2\},s_1,s_2\leq j-1}\left(9\frac{b_a(s_a+1,s_{\bar{a}})}{\sqrt{N_a(N_a-b_a(q,q))}}\right)\right.$$
$$+\max_{a\in\{1,2\},s_1,s_2\leq i-1}\left(2\sqrt{\frac{b_a(s_a+1,s_{\bar{a}})}{N_a}}+3\frac{b_a(s_1,s_2)}{\sqrt{N_a(N_a-b_a(s_1,s_2))}}\right.$$
$$\left.\left.+\frac{b_a(s_1,s_2)^{3/2}}{N_a\sqrt{N_a-b_a(s_1,s_2)}}+\sqrt{2\frac{2q\max_{k\leq s_{\bar{a}}}\left(b_{\bar{a}}(s_a,k)-b_{\bar{a}}(s_a-1,k)\right)}{N_a}}\right)\right)^2, \tag{6.167}$$

where we use the fact that b_a is a monotonously increasing function: $b_a(s_1,s_2)\leq b_a(s_a+1,s_{\bar{a}})$ and $b_a(s_1,s_2)\leq b_a(q,q)$.

For many of the relations presented below, we do not know quantum algorithms that find a database that is in relation. However, our intuition is that all the distinguishability bounds coming from the application of the O2H lemma (Theorem 6.7) and bounding the probability of Find with our bounds proven in Lemma 6.11 are tight.

In all corollaries stated in the rest of this chapter, to prove them we use the bound from Lemma 6.11 with function b defined before the statement of the corollary. For Corollaries 6.15, 6.16, and 6.17 we take the bound simplified as in Equation (6.156). For Corollaries 6.18, 6.19, 6.20, and 6.21 we take the bound simplified as in Equation (6.167).

6.4.2.1 The Collision Relation

For collisions we consider a single database. The coefficient is then $b(s)=s-1$, a collision might occur with any of the previously output values.

Corollary 6.15. *For any quantum adversary* A *interacting with a punctured oracle* $\mathsf{CStO}_{\mathcal{Y}}\setminus R_{\mathrm{coll}}$*—where* R_{coll} *is defined in Equation* (6.63)*—the probability of* Find *is bounded by:*

$$\mathbb{P}\left[\mathrm{Find}:\mathrm{A}[\mathsf{CStO}_{\mathcal{Y}}\setminus R_{\mathrm{coll}}]\right]\leq\frac{2q^2}{|\mathcal{Y}|}+\frac{4q^{7/2}}{|\mathcal{Y}|\sqrt{|\mathcal{Y}|-q}}+\frac{5q^5}{|\mathcal{Y}|\left(|\mathcal{Y}|-q\right)}, \tag{6.168}$$

where q is the maximal number of queries made by A.

The above bound for $q\in O\left(\sqrt[3]{|\mathcal{Y}|}\right)$ can be bounded by to $\frac{11q^2}{|\mathcal{Y}|}$, so just the classical collision-finding bound. Intuitively, this result is justified by the fact that the coherence needed by the optimal quantum search algorithms (e.g.

the Grover algorithm [Gro96]) is broken by the repeated measurements in the punctured oracle.

The bound proven in Corollary 6.15 is tight. The complexity of a generic collision-finding algorithm is $O\left(|\mathcal{Y}|^{1/3}\right)$ [Zha15a]. Our bound on distinguishing random functions from random injective functions (calculated via the O2H lemma, Theorem 6.7) is a constant close to 1 for the matching $q \in \Omega\left(|\mathcal{Y}|^{1/3}\right)$. Note that the bound from Corollary 6.15 on itself does not tell us much about quantum collision finding. Only with the O2H lemma (that multiplies the bound on Find by q) is makes sense to compare our bounds with known results. It is also important to note that our bound is almost the same as the bound proven in Lemma 9 of [Zha19a]). We do not use Zhandry's result, because we could not verify some steps of the proof.

6.4.2.2 The Preimage Relation

For the preimage relation $b(s) = 1$, because there is just one output $y = 0$ that brings the database to be in R.

Corollary 6.16. *For any quantum adversary* A *interacting with a punctured oracle* $\mathsf{CStO}_{\mathcal{Y}} \setminus R_{\text{preim}}$*—where* R_{preim} *is defined in Equation* (6.62)*—the probability of* Find *is bounded by:*

$$\mathbb{P}\left[\text{Find} : \mathrm{A}[\mathsf{CStO}_{\mathcal{Y}} \setminus R_{\text{preim}}]\right] \leq \frac{9q}{|\mathcal{Y}|} + \frac{30q^2}{|\mathcal{Y}|\sqrt{|\mathcal{Y}|-1}} + \frac{25q^3}{|\mathcal{Y}|\left(|\mathcal{Y}|-1\right)}, \tag{6.169}$$

where q is the maximal number of queries made by A.

The above bound for $q \in O\left(\sqrt[2]{|\mathcal{Y}|}\right)$ simplifies to $\frac{64q}{|\mathcal{Y}|}$, so the classical preimage-finding bound.

6.4.2.3 The Collision and Preimage Relations

We also provide a bound for the combined relations R_{coll} and R_{preim}, in this case $b(s) = s$. The function b is such because in addition to previous outputs, the new value can also be 0 to bring D to R.

Corollary 6.17. *For any quantum adversary* A *interacting with a punctured oracle* $\mathsf{CStO}_{\mathcal{Y}} \setminus (R_{\text{preim}} \cup R_{\text{coll}})$*—where* R_{coll} *is defined in Equation* (6.63) *and* R_{preim} *in Equation* (6.62)*—the probability of* Find *is bounded by:*

$$\begin{aligned} &\mathbb{P}\left[\text{Find} : \mathrm{A}[\mathsf{CStO}_{\mathcal{Y}} \setminus (R_{\text{preim}} \cup R_{\text{coll}})]\right] \\ &\leq \frac{2q^2}{|\mathcal{Y}|} + \frac{4q^{7/2}}{|\mathcal{Y}|\sqrt{|\mathcal{Y}|-q}} + \frac{5q^5}{|\mathcal{Y}|\left(|\mathcal{Y}|-q\right)}, \end{aligned} \tag{6.170}$$

where q is the maximal number of queries made by A.

The bound is the same as in the case of Corollary 6.15, this is because of the simplifications made to find a concise bound, also note how similar b is in both cases.

6.4.2.4 Composition of Compression Functions

The Comp construction is defined in Section 2.5.1. We use the relation

$$R_{\text{Comp}} := \{(D_1, D_2) \in \mathcal{D} : \exists y_1 \in D_1^Y, x_2 \in D_2^X, (y_1, x_2) \in D_2^X\} \tag{6.171}$$

where $\mathcal{D} := \bigcup_{s\in[q+1]} (\mathcal{X}_1 \times \mathcal{Y}_1)^s \times \bigcup_{s\in[q+1]} (\mathcal{X}_2 \times \mathcal{Y}_2)^s$ and $\mathcal{X}_2 := \mathcal{X}_1 \times \mathcal{Y}_1$. The relation captures databases with an output in D_1 that is part of an input in D_2. The full relation we analyze is the union of the above and the collision relation. Both R_{Comp} and R_{coll} are defined on the outputs of H_1. In the following, for $i \in \{1, 2\}$, we define $\mathfrak{U}_i$ to be the uniform distribution over the set of functions from the set $\{\mathbf{h}_i : \mathcal{X}_i \to \mathcal{Y}_i\}$. The coefficients are defined as $b_1(s_1, s_2) = s_1 - 1 + s_2$ and $b_2(s_1, s_2) = 0$, the only source of bad outputs are collisions of outputs of $\mathbf{h}_1$ with other outputs of $\mathbf{h}_1$ and parts of inputs to $\mathbf{h}_2$. With these values we make the following statement, the concrete bound is a result of straightforward simplifications of the bound coming from Equation (6.167).

Corollary 6.18. *For any quantum adversary* A *interacting with a punctured oracle* $(\mathsf{CStO}_{\mathfrak{U}_1}, \mathsf{CStO}_{\mathfrak{U}_2}) \setminus (R_{\text{coll}} \cup R_{\text{Comp}})$*—where* R_{coll} *is defined in Equation* (6.63) *and* R_{Comp} *in Equation* (6.171)*—the probability of* Find *is bounded by:*

$$\begin{aligned}&\mathbb{P}[\text{Find} : \text{A}[(\mathsf{CStO}_{\mathfrak{U}_1}, \mathsf{CStO}_{\mathfrak{U}_2}) \setminus (R_{\text{coll}} \cup R_{\text{Comp}})]] \\ &\leq \frac{36q^2}{|\mathcal{Y}_1|} + \frac{194q^{7/2}}{|\mathcal{Y}_1|\sqrt{|\mathcal{Y}_1| - 2q}} + \frac{297q^5}{|\mathcal{Y}_1|(|\mathcal{Y}_1| - 2q)},\end{aligned} \tag{6.172}$$

where q is the maximal number of queries made by A.

For $q \in O\left(\sqrt[3]{|\mathcal{Y}_1|}\right)$ the bound above is just $O\left(q^2/|\mathcal{Y}_1|\right)$.

6.4.2.5 Rate-1/3 Hash Function Relation

In the indifferentiability proof of the Rate-1/3 construction defined in Section 2.5.3 we lazy sample three functions. The generalization of Lemma 6.11 to $\mathsf{H} = (\mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3)$ can be done in a straight forward way, note that we do not make use of the fact that $a \in \{1, 2\}$ in any place of the proof. We define the relation

$$R_{\text{Rate-1/3}} := \{(D_1, D_2, D_3) \in \mathcal{D} : \exists y_1 \in D_1^Y, y_2 \in D_2^Y, x_3 \in D_3^X, y_1 = y_2 \oplus x_3\}, \tag{6.173}$$

where $\mathcal{D} := \bigcup_{s\in[q+1]} (\mathcal{X}_1 \times \mathcal{Y})^s \times \bigcup_{s\in[q+1]} (\mathcal{X}_2 \times \mathcal{Y})^s \times \bigcup_{s\in[q+1]} (\mathcal{Y} \times \mathcal{Y})^s$. In the following, for $i \in \{1, 2.3\}$, we define $\mathfrak{U}_i$ to be the uniform distribution over the set of functions from the set $\{\mathbf{f}_i : \mathcal{X}_i \to \mathcal{Y}_i\}$. Note however that all of the output sets are the same and we denote them by $\mathcal{Y}$. Moreover $\mathcal{Y} = \{0,1\}^n$. We state a lemma giving a bound on the probability of Find for the triple of compressed oracles $(\mathsf{CStO}_{\mathfrak{U}_1}, \mathsf{CStO}_{\mathfrak{U}_2}, \mathsf{CStO}_{\mathfrak{U}_3})$. The coefficients are $b_1(s_1, s_2, s_3) \leq s_2 \cdot s_3$, $b_2(s_1, s_2, s_3) \leq s_1 \cdot s_3$, and $b_3(s_1, s_2, s_3) = 0$. The b_1 function is such, because for each output of $\mathbf{f}_1$, there are at most $s_2 \cdot s_3$ sums $y_2 \oplus x_3$ of outputs of $\mathbf{f}_2$ and inputs of $\mathbf{f}_3$ that can bring D to be in R. Similarly for b_2. Outputs of $\mathbf{f}_3$ do not cause D to be in relation.

Corollary 6.19. *For any quantum adversary* A *interacting with a punctured oracle* $(\mathsf{CStO}_{\mathfrak{U}_1}, \mathsf{CStO}_{\mathfrak{U}_2}, \mathsf{CStO}_{\mathfrak{U}_3}) \setminus R_{\text{RATE-1/3}}$*, where* $R_{\text{RATE-1/3}}$ *is defined in Equation* (6.173)*, the probability of* Find *is bounded by:*

$$\mathbb{P}\left[\text{Find} : \mathsf{A}[(\mathsf{CStO}_{\mathfrak{U}_1}, \mathsf{CStO}_{\mathfrak{U}_2}, \mathsf{CStO}_{\mathfrak{U}_3}) \setminus R_{\text{RATE-1/3}}]\right] \\ \leq 36\frac{q^3}{|\mathcal{Y}|} + 84\frac{q^5}{|\mathcal{Y}|\sqrt{|\mathcal{Y}| - q^2}} + 70\frac{q^7}{|\mathcal{Y}|\left(|\mathcal{Y}| - q^2\right)}, \tag{6.174}$$

where q *is the maximal number of queries made by* A.

For $q \in O\left(\sqrt[4]{|\mathcal{Y}|}\right)$ the bound above is just $O\left(q^3/|\mathcal{Y}|\right)$.

6.4.2.6 EDM Relation

The EDM construction is defined in Section 2.5.4. We use the relation

$$R_{\text{EDM}} := \{(D_1, D_2) \in \mathcal{D} : \exists (x_1, y_1) \in D_1, x_2 \in D_2^X, y_1 = x_1 \oplus x_2\}, \tag{6.175}$$

where $\mathcal{D} := \bigcup_{s\in[q+1]} (\mathcal{X} \times \mathcal{Y})^s \times \bigcup_{s\in[q+1]} (\mathcal{X} \times \mathcal{Y})^s$ and $\mathcal{X} = \{0,1\}^n$. We state a corollary giving a bound on the probability of Find for the pair of compressed oracles $[\mathsf{H}_1, \mathsf{H}_2]$. For a relation R_{EDM}, the coefficients are $b_1(s_1, s_2) = s_2$ and $b_2(s_1, s_2) = 0$. The coefficient b_1 is such because for an output of $\boldsymbol{\pi}_1$, every input to $\boldsymbol{\pi}_2$ gives rise to a possible collision.

Corollary 6.20. *For any quantum adversary* A *interacting with a punctured oracle* $(\mathsf{CStO}_{\mathcal{Y}}, \mathsf{CStO}_{\mathcal{Y}}) \setminus R_{\text{EDM}}$*—where* R_{EDM} *is defined in Equation* (6.175)*—the probability of* Find *is bounded by:*

$$\mathbb{P}[\text{Find} : \mathsf{A}[(\mathsf{CStO}_{\mathcal{Y}}, \mathsf{CStO}_{\mathcal{Y}}) \setminus R_{\text{EDM}}]] \leq \frac{16q^2}{|\mathcal{Y}|} + \frac{72q^{7/2}}{|\mathcal{Y}|\sqrt{|\mathcal{Y}| - q}} + \frac{98q^5}{|\mathcal{Y}|\left(|\mathcal{Y}| - q\right)}, \tag{6.176}$$

where q *is the maximal number of queries made by* A.

For $q \in O\left(\sqrt[3]{|\mathcal{Y}|}\right)$ the bound above is just $O\left(q^2/|\mathcal{Y}|\right)$.

6.4.2.7 EDMD Relation

The EDMD construction is defined in Section 2.5.5. We use the relation

$$R_{\mathrm{EDMD}} := \{(D_1, D_2) \in \mathcal{D} : \exists (x_1, y_1) \in D_1, x_2 \in D_2^X, y_1 = x_2\}, \tag{6.177}$$

where $\mathcal{D} := \bigcup_{s \in [q+1]} (\mathcal{X} \times \mathcal{Y})^s \times \bigcup_{s \in [q+1]} (\mathcal{X} \times \mathcal{Y})^s$ and $\mathcal{X} = \{0,1\}^n$. We state a corollary giving a bound on the probability of Find for the pair of compressed oracles $[\mathsf{H}_1, \mathsf{H}_2]$. The coefficients are $b_1(s_1, s_2) = s_2$ and $b_2(s_1, s_2) = 0$.

Corollary 6.21. *For any quantum adversary* $\mathcal{A}$ *interacting with a punctured oracle* $(\mathsf{CStO}_{\mathcal{Y}}, \mathsf{CStO}_{\mathcal{Y}}) \setminus R_{\mathrm{EDMD}}$*—where* R_{EDMD} *is defined in Equation* (6.177)*—the probability of* Find *is bounded by:*

$$\mathbb{P}[\mathrm{Find} : \mathcal{A}[(\mathsf{CStO}_{\mathcal{Y}}, \mathsf{CStO}_{\mathcal{Y}}) \setminus R_{\mathrm{EDMD}}]] \leq \frac{16q^2}{|\mathcal{Y}|} + \frac{72q^{7/2}}{|\mathcal{Y}| \sqrt{|\mathcal{Y}| - q}} + \frac{98q^5}{|\mathcal{Y}| (|\mathcal{Y}| - q)}, \tag{6.178}$$

where q *is the maximal number of queries made by* $\mathcal{A}$.

For $q \in O\left(\sqrt[3]{|\mathcal{Y}|}\right)$ the bound above is just $O\left(q^2 / |\mathcal{Y}|\right)$.

6.5 The Framework

Let us first sum up our approach to the general task of bounding the distinguishability advantage between two games. We focus on games that use punctured oracles (Definition 6.5) and for those use the O2H lemma (Theorem 6.7). The crucial point in the O2H lemma is finding a bound on $\mathbb{P}\left[\mathrm{Find}\right]$. Our solution to this problem is presented in Lemma 6.11.

The role of the fundamental game-playing lemma, Lemma 2.22, takes the One-Way to Hiding lemma, Theorem 6.7. In both classical and quantum proofs the most challenging part is bounding the probability of events Bad or Find happening.

We think that this approach is especially useful, when considering lifting the security guarantees from the classical case to the quantum one. That is because a classical proof can be rather easily translated to the quantum world.

The crucial part of games are the bad events, that are the key events that distinguish two games. The role of bad events in quantum proofs take the punctured oracles. The relation we puncture on is the—possibly rephrased—bad event from the classical proof. To correctly translate bad events to relations we identify the set of databases that cause the bad event. When we have the correct relation at hand, the punctured oracle, given $\neg$Find, holds a superposition of databases distributed in exactly the same way as in the classical game. Hence, the classical reasoning can be often reused.

There are of course some operations and lines of reasoning that cannot be repeated in a proof of quantum security. This is an aspect that has to be analyzed in a case-by-case fashion.

CHAPTER

7 QUANTUM INDIFFERENTIABILITY

Chapter contents

In this chapter we prove quantum indifferentiability of multiple constructions. We show how the quantum game-playing framework from Chapter 6 can be applied to constructions defined in Section 2.5. Namely, we first present a classical proof and then a quantum one that follows the same structure but employs the techniques of Chapter 6 to derive the final result.

7.1 Introduction

The quantum security of domain-extension schemes has been the topic of several recent works. In [SY17; CHS19] the authors study domain extension for message authentication codes and pseudorandom functions. For random inner function, [Zha19a] has proven indifferentiability of the Merkle-Damgård construction which hence has strong security in the QROM. For hash functions in the standard model, quantum generalizations of collision resistance were defined in [Unr16b; Ala+18]. For one of them, collapsingness, some domain-extension schemes including the Merkle-Damgård and sponge constructions, have been shown secure [Cza+18; Feh18; Unr16a].

In this chapter we prove a number of indifferentiability results. This strong notion is introduced in Section 2.3.3. We cover multiple cryptographic constructions, described in detail in Section 2.5. The general structure of this chapter is that for every construction we present two proofs of indifferentiability, one classical and one quantum. We always present two proofs to simplify reading the second proof, it follows the same reasoning as the former one. We also want to highlight how similar these proofs are, this similarity is what we consider to be one of the main advantages of our quantum game-playing framework introduced in Chapter 6. In our framework all proofs of quantum indifferentiability can follow the same reasoning and very similar steps as the classical version.

Before we proceed let us remind the reader of the main concepts that are necessary to follow the proof of quantum indifferentiability. The central object of the proof are punctured oracles, defined in Definition. 6.5. They play the role of subroutines that lazy-sample functions and output "True" when a bad event occurs. Readers familiar with the original game-playing framework [BR06] will recognize the crucial subroutines of the classical games. Additionally, punctured oracles are objects that allow to condition probabilistic events on some aspects of quantum queries done by the adversary. This useful feature allows us to sometimes use arguments from the classical proof in the quantum one.

A punctured oracle is built using the compressed-oracle framework and formally includes a quantum database register, as described in detail in Section 6.2. Nonetheless these details are not necessary to follow the contents of this chapter. The only two things to keep in mind are that in general the adversary can make quantum queries to the primitives and that the responses of queries are saved in the adversary's quantum register $|s, v\rangle$, where s is the query and v is any value in the codomain of the queried function.

The reason we use punctured oracles is that they allow to use the One-way To Hiding (O2H) lemma. Which is an extremely useful tool for bounding the distinguishability advantage of two quantum games. We cover this lemma in details in Section 6.3. Technically the most demanding part of using the O2H lemma is bounding the probability of any puncturing measurement succeeding (we call this event Find). We compute a bound on $\mathbb{P}\left[\text{Find}\right]$ useful in the quantum indifferentiability proofs in Section 6.4.2.

The second distinguishability bound that we use is shown in Lemma 6.10. This is a relatively simple statement, that is true for games that are almost identical (Definition 6.9).

We start the chapter with an indifferentiability proof of Comp, a simple construction that was mentioned in [Cor+05], where the authors claim its (classical) indifferentiability. Quantum indifferentiability of Comp is proven in [Zha19a], we comment on this result in the next section. This result is also significant for the discussion in Chapter 8.

In Section 7.3 we focus on Rate-1/3, an interesting construction that uses three non-compressing functions to build a compression function. The construction Rate-1/3 was introduced and proven to be collision-resistant in [SS08].

In sections 7.4 and 7.5 we prove classical and quantum indifferentiability of the EDM and EDMD constructions. These constructions are introduced in [CS16; MN17a] and proven indistinguishable from a random function (given that the internal permutations are random). Our quantum results for the last two constructions are limited to one-way permutations. The reason for that is our inability to quantumly lazy-sample two-way permutations.

In the last section of this chapter we prove quantum indifferentiability of the sponge construction, maintaining the general structure of the classical proof from [Ber+08]. We show quantum security of sponges with random functions as the internal function. We first presented this result in [Cza+19].

We note that when discussing more than two oracles that are punctured with relations that depend on all of them, we use the punctured oracle notation only on those that are directly influenced by the puncturing. The distinguishability bound can be only calculated by considering all of the oracles.

To save space we present multiple algorithms in one. To do that we follow a convention where only the boxed algorithms perform the boxed operations. In case there are actually more than two algorithms, the color of the box also matters. If a line is not surrounded by a box, then all algorithms perform the command. We number the simulators with the number of the game it is first used in.

7.2 Composed Functions

The composition construction is defined as $\textsc{Comp}_{\mathbf{h}_1,\mathbf{h}_2}(x_1, x_2) = \mathbf{h}_2(\mathbf{h}_1(x_1), x_2)$, where $\mathbf{h}_i : \mathcal{X}_i \to \mathcal{Y}_i$ and $\mathcal{Y}_i := [N_i]$ for $i \in \{1, 2\}$. The full definition and additional details are presented in Section 2.5.1.

7.2.1 Classical Indifferentiability

We start with classical indifferentiability, already mentioned in [Cor+05].

Theorem 7.1. *The compression function* $\textsc{Comp}_{\mathbf{h}_1,\mathbf{h}_2}$ *for uniformly random* $\mathbf{h}_1$ *and* $\mathbf{h}_2$ *is* (q, ε)*-classically indifferentiable for* $\varepsilon = 5\left(\frac{(q-1)q}{2N_1} + \frac{q^2}{N_1}\right)$.

Proof. We carry out the proof by starting with the real world and gradually changing the adversary's interface to the ideal world. We define two simulators, the initial S_2 that just lazy samples the compression functions and S_3 that is the actual simulator. In Algorithm 7.1 only the boxed algorithms perform the boxed commands.

Algorithm 7.1 Classical simulators $\boxed{\mathsf{S}_2}$, $\boxed{\mathsf{S}_3}$ for $\textsc{Comp}_{\mathbf{h}_1,\mathbf{h}_2}$

```
procedure $\mathbf{h}_1(x_1)$
    if $x_1 \in D_1^X$ return the corresponding $y_1$
    $y_1 \xleftarrow{\$} \mathcal{Y}_1$
    if $y_1 \in D_1^Y$ then                                  ▷ Collision of $\mathbf{h}_1$
        $\boxed{\text{Set Bad}_1 = 1}$
    if $\exists x_2 : (y_1, x_2) \in D_2^X$ then               ▷ Preimage of $\mathbf{h}_2$
        $\boxed{\text{Set Bad}_2 = 1}$
    Add $(x_1, y_1)$ to $D_1$ and return $y_1$

procedure $\mathbf{h}_2(y_1, x_2)$
    if $(y_1, x_2) \in D_2^X$ return the corresponding $y_2$
    if $\exists (x_1, y_1) \in D_1$ then
        $\boxed{y_2 \xleftarrow{\$} \mathcal{Y}_2, \text{ add } ((y_1, x_2), y_2) \text{ to } D_2}$
        $\boxed{\textbf{return } y_2}$
        $\boxed{\textbf{return } \mathsf{R}(x_1, x_2)}$                         ▷ R: random oracle
    else
        $y_2 \xleftarrow{\$} \mathcal{Y}_2$, add $((y_1, x_2), y_2)$ to $D_2$
        return $y_2$
```

Game 1 The interface in the first game is $(\textsc{Comp}, (\mathbf{h}_1, \mathbf{h}_2))$, where the public interface consists of two uniformly random functions $\mathbf{h}_1$ and $\mathbf{h}_2$. The definition of the game is

$$\textbf{Game 1} := (b = 1 : b \leftarrow \mathsf{A}[\textsc{Comp}, (\mathbf{h}_1, \mathbf{h}_2)])\,. \tag{7.1}$$

Game 2 In the second game we lazy sample the compression functions, the interface is $(\textsc{Comp}, \mathsf{S}_2)$ and the game is defined as

$$\textbf{Game 2} := (b = 1 : b \leftarrow \mathsf{A}[\textsc{Comp}, \mathsf{S}_2])\,. \tag{7.2}$$

This change of the interface is indistinguishable for A:

$$\left|\mathbb{P}\left[\textbf{Game 2}\right] - \mathbb{P}\left[\textbf{Game 1}\right]\right| = 0. \tag{7.3}$$

Game 3 The interface in the third game is $(\textsc{Comp}, \mathsf{S}_3)$, where we introduce the bad events and the random oracle R, note that $\mathbf{h}_2$ is distributed uniformly at random, so adding R does not change the distribution of $\mathbf{h}_2$ at all. The definition of the game is

$$\textbf{Game 3} := (b = 1 : b \leftarrow \mathsf{A}[\textsc{Comp}, \mathsf{S}_3])\,. \tag{7.4}$$

The bad events we introduce are Bad_1 that is 1 when there occurs a collision of $\mathbf{h}_1$ and Bad_2 that happens when $\mathbf{h}_1$ outputs a part of a past query to $\mathbf{h}_2$. The former is important because a collision in $\mathbf{h}_1$ leads to the same output of Comp but—most probably—not in R. The latter event assigns an input to Comp to an already given output. This event is bad because outputs of Comp are supposed to be exchanged to R and if S_3 commits to a uniformly random output it will most probably be different from the output of R. In this case the adversary could easily distinguish the two worlds. Given no bad events we exclude all possibly inconsistent outputs of $\mathbf{h}_2$. The only noticeable change for the adversary are the bad events. To calculate distinguishability we use Lemma 2.22:

$$\left|\mathbb{P}\left[\textbf{Game 3}\right] - \mathbb{P}\left[\textbf{Game 2}\right]\right| \leq \mathbb{P}\left[\mathrm{Bad}\right] \leq \frac{(q-1)q}{2N_1} + \frac{q^2}{N_1}, \tag{7.5}$$

where $\mathrm{Bad} = \mathrm{Bad}_1 \vee \mathrm{Bad}_2$. The first term corresponds to collisions in $\mathbf{h}_1$ and the second to y_1 hitting an input of $\mathbf{h}_2$.

In the following we denote the size of D_2 after i queries by $s_2(i)$. The latter bound above is achieved as follows:

$$\mathbb{P}\left[\mathrm{Bad}_2\right] \leq \sum_{i=1}^{q} \frac{s_2(i)}{N_1} \leq \sum_{i=1}^{q} \frac{q}{N_1} = \frac{q^2}{N_1}, \tag{7.6}$$

where in the first inequality we assume that every query is made to $\mathbf{h}_1$ and use the union bound. The second inequality follows from a bound on the size of D_2, after the i-th query $s_2(i) \leq q$.

Game 4 In the last game the interface is $(\mathsf{R}, \mathsf{S}_3)$, we change the private interface, i.e. the interface giving access to the construction or the random oracle. The definition of the game is

$$\textbf{Game 4} := (b = 1 : b \leftarrow \mathrm{A}[\mathsf{R}, \mathsf{S}_3]) \,. \tag{7.7}$$

The only source of distinguishing advantage for A are the possible bad events in queries to the private interface. We know, however, that if there are no bad events then **Game 3** and **Game 4** are distributed in the same way:

$$\left|\mathbb{P}\left[\textbf{Game 4} \mid \neg\mathrm{Bad}\right] - \mathbb{P}\left[\textbf{Game 3} \mid \neg\mathrm{Bad}\right]\right| = 0. \tag{7.8}$$

Using the above identity we derive the final distinguishing advantage:

$$\left|\mathbb{P}\left[\textbf{Game 4}\right] - \mathbb{P}\left[\textbf{Game 3}\right]\right| \leq 4\mathbb{P}\left[\mathrm{Bad}\right] \leq 4\left(\frac{(q-1)q}{2N_1} + \frac{q^2}{N_1}\right), \tag{7.9}$$

where we use the derivation of Lemma 6.10. To get the first inequality above we consider classical algorithms in place of the quantum ones in Lemma 6.10 and Bad events instead of Find. The event Bad in Equation (7.9) corresponds to Bad in **Game 3**, we bound the bound from the lemma by the bigger of the two probabilities. Probability of Bad in **Game 3** is greater because there are in principle more calls to S_3—the private interface also calls the simulator.

With the last game we have shown that there is a simulator that answers the queries to the public interface such that the adversary has negligible advantage in distinguishing the private interface Comp from a random oracle. □

7.2.2 Quantum Indifferentiability

Now we go to quantum indifferentiability. The theorem has been already proven in [Zha19a]. Zhandry's proof however uses a different approach to bounding $\mathbb{P}\left[\mathrm{Find}\right]$ than we, he just uses the collision finding bound. This approach is somewhat unclear so we use the framework developed in Chapter 6. In Zhandry's original paper, indifferentiability is also proven using oracles that are measured after every query (we call them punctured oracles). There, the distinguishing advantage is derived using techniques that are very similar to finding bounds on quantum query complexity (Lemma 9 in [Zha19a]). Such bounds, however, do not assume measurements performed after every query. In principle this might be a source of an error (possibly coming from using wrong assumptions). Through private communication with Zhandry [Zha20] we learned that it is easy to fill in all the necessary details to make the proof work.

Theorem 7.2. *The compression function* $\textsc{Comp}_{\mathbf{h}_1,\mathbf{h}_2}$ *for uniformly random* $\mathbf{h}_1$ *and* $\mathbf{h}_2$ *is* (q,ε)*-quantumly indifferentiable for any* $q \in O\left(\sqrt[3]{N_1}\right)$ *and* $\varepsilon = \sqrt{527(q+1)\frac{q^2}{N_1}} + 2108\frac{q^2}{N_1}$.

Proof. We carry out the proof by starting with the real world and gradually changing the adversary's interface to the ideal world. We define two simulators, the initial S_2 that just lazy samples the compression functions and S_3 that is the actual simulator. In Algorithm 7.2 only the boxed algorithm performs the boxed commands. In the following we define $\mathfrak{U}_i$ to be the uniform distribution over functions in $\{\mathcal{X}_i \to \mathcal{Y}_i\}$. Even though the relation is defined on both com-

Algorithm 7.2 Quantum simulators $\boxed{\mathsf{S}_2}$, $\boxed{\mathsf{S}_3}$ for $\textsc{Comp}_{\mathbf{h}_1,\mathbf{h}_2}$

```
procedure $\mathbf{h}_1(x_1)$
    Apply $\boxed{\mathsf{CStO}_{\mathfrak{U}_1}}$, $\boxed{\mathsf{CStO}_{\mathfrak{U}_1} \setminus R_{\text{coll}} \cup R_{\textsc{Comp}}}$

procedure $\mathbf{h}_2(y_1, x_2)$
    if $\exists(x_1, y_1) \in D_1$ then
        $\boxed{\text{Apply } \mathsf{CStO}_{\mathfrak{U}_2}}$
        $\boxed{\textbf{return } \mathsf{R}(x_1, x_2)}$
    else
        Apply $\mathsf{CStO}_{\mathfrak{U}_2}$
```

pressed functions, in Algorithm 7.2 we write it only in the compressed oracles that are directly influenced by the puncturing.

Game 1 The interface in the first game is $(\textsc{Comp}, (\mathbf{h}_1, \mathbf{h}_2))$, where $\mathbf{h}_1$ and $\mathbf{h}_2$ are just uniformly random functions. The definition of the game is

$$\textbf{Game 1} := (b = 1 : b \leftarrow \mathsf{A}[\textsc{Comp}, (\mathbf{h}_1, \mathbf{h}_2)]) . \tag{7.10}$$

Game 2 In the second step we lazy sample the compression functions, the interface is $(\textsc{Comp}, \mathsf{S}_2)$. The game is defined as

$$\textbf{Game 2} := (b = 1 : b \leftarrow \mathsf{A}[\textsc{Comp}, \mathsf{S}_2]) . \tag{7.11}$$

Following Theorem 6.3, this change of the interface is indistinguishable for A:

$$\left|\mathbb{P}\left[\textbf{Game 2}\right] - \mathbb{P}\left[\textbf{Game 1}\right]\right| = 0. \tag{7.12}$$

Game 3 The interface in the third game is $(\textsc{Comp}, \mathsf{S}_3)$, where we introduce the punctured oracle and R, introducing the random oracle does not change the distribution of the outputs of $\mathbf{h}_2$ so this change does not add to the distinguishability advantage. The new game is

$$\textbf{Game 3} := (b = 1 : b \leftarrow \mathsf{A}[\textsc{Comp}, \mathsf{S}_3]) . \tag{7.13}$$

We puncture on the same events as in the classical proof. The only noticeable change for the adversary is the punctured oracle. The distinguishing advantage can be bounded by the O2H lemma, Theorem 6.7:

$$\left|\mathbb{P}\left[\textbf{Game 3}\right] - \mathbb{P}\left[\textbf{Game 2}\right]\right| \leq \sqrt{(q+1)\mathbb{P}\left[\text{Find}\right]} \leq \sqrt{(q+1)\frac{527q^2}{N_1}}, \tag{7.14}$$

where the bound on $\mathbb{P}\left[\text{Find}\right]$ comes from Corollary 6.18.

Game 4 In the last step of this proof the interface is $(\mathsf{R}, \mathsf{S}_3)$, we change the private interface, the definition of the game is

$$\textbf{Game 3} := (b = 1 : b \leftarrow \mathsf{A}[\mathsf{R}, \mathsf{S}_3]) . \tag{7.15}$$

Similar to the classical case we have:

$$\left|\mathbb{P}\left[\textbf{Game 4} \mid \neg\text{Find}\right] - \mathbb{P}\left[\textbf{Game 3} \mid \neg\text{Find}\right]\right| = 0. \tag{7.16}$$

Using the above identity we derive the final distinguishing advantage:

$$\left|\mathbb{P}\left[\textbf{Game 4}\right] - \mathbb{P}\left[\textbf{Game 3}\right]\right| \leq 4\mathbb{P}\left[\text{Find}\right] \leq 2108\frac{q^2}{N_1}, \tag{7.17}$$

where we use Lemma 6.10, Find is the event of finding the relation in **Game 3**, we bound the bound from the lemma by the bigger of the two probabilities.

The last game includes the random oracle in the private interface, which concludes the proof. □

7.3 Rate-1/3 Compression Function

The rate-1/3 construction is defined as $\text{Rate-1/3}_{\mathsf{f}_1,\mathsf{f}_2,\mathsf{f}_3}(x_1, x_2) =$ $\mathsf{f}_3\left(\mathsf{f}_1(x_1) \oplus \mathsf{f}_2(x_2)\right) \oplus \mathsf{f}_1(x_1)$, where $\mathsf{f}_1 : \mathcal{X}_1 \to \mathcal{Y}_1$, $\mathsf{f}_2 : \mathcal{X}_2 \to \mathcal{Y}_2$, and $\mathsf{f}_3 : \mathcal{X}_3 \to \mathcal{Y}_3$, moreover $\mathcal{X}_3 = \{0,1\}^n$ and $\mathcal{Y}_i = \{0,1\}^n$ for all $i \in \{1,2,3\}$, $N = 2^n$. The full definition of the construction is presented in Section 2.5.3.

7.3.1 Classical Indifferentiability

Theorem 7.3. *The compression function* $\text{Rate-1/3}_{\mathsf{f}_1,\mathsf{f}_2,\mathsf{f}_3}$ *for uniformly random* f_1, f_2, *and* f_3 *is* (q, ε)*-classically indifferentiable for* $\varepsilon = 10\frac{q^3}{N}$.

Proof. We carry out the proof by starting with the real world and gradually changing the adversary's interface to the ideal world. We define two simulators, the initial S_2 that just lazy samples the compression functions and S_3 that is the actual simulator. In Algorithm 7.3 only the boxed algorithm performs the boxed commands.

Algorithm 7.3 Classical simulators $\boxed{\mathsf{S}_2}$, $\boxed{\mathsf{S}_3}$ for RATE-1/3$_{\mathsf{f}_1,\mathsf{f}_2,\mathsf{f}_3}$

```
procedure f_1($x_1$)
    if $x_1 \in D_1^X$ return the corresponding $y_1$
    $y_1 \xleftarrow{\$} \mathcal{Y}$
    if $\exists y_2 \in D_2^Y, x_3 \in D_3^X : y_1 = y_2 \oplus x_3$ then        ▷ Preimage of f_3
        $\boxed{\text{Set Bad}_1 = 1}$
    Add $(x_1, y_1)$ to $D_1$ and return $y_1$

procedure f_2($x_2$)
    if $x_2 \in D_2^X$ return the corresponding $y_2$
    $y_2 \xleftarrow{\$} \mathcal{Y}$
    if $\exists y_1 \in D_1^Y, x_3 \in D_3^X : y_2 = y_1 \oplus x_3$ then        ▷ Preimage of f_3
        $\boxed{\text{Set Bad}_2 = 1}$
    Add $(x_2, y_2)$ to $D_2$ and return $y_2$

procedure f_3($x_3$)
    if $x_3 \in D_3^X$ return the corresponding $y_3$
    if $\exists y_1 \in D_1^Y, y_2 \in D_2^Y : x_3 = y_1 \oplus y_2$ then
        $\boxed{y_3 \xleftarrow{\$} \mathcal{Y}, \text{add } (x_3, y_3) \text{ to } D_3}$
        $\boxed{\textbf{return } y_3}$
        $\boxed{\textbf{return } \mathsf{R}(x_1, x_2) \oplus y_1}$
    else
        $y_3 \xleftarrow{\$} \mathcal{Y}$, add $(x_3, y_3)$ to $D_3$
        return $y_3$
```

Game 1 The interface in the first game is $(\textsc{Rate-1/3}, (\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3))$, where the public interface holds uniformly random $\mathbf{f}_1$, $\mathbf{f}_2$, and $\mathbf{f}_3$. The definition of the game is

$$\textbf{Game 1} := (b = 1 : b \leftarrow \mathsf{A}[\textsc{Rate-1/3}, (\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3)]) . \tag{7.18}$$

Game 2 In the second game we lazy sample the compression functions, the interface is $(\textsc{Rate-1/3}, \mathsf{S}_2)$, the game is defined as

$$\textbf{Game 2} := (b = 1 : b \leftarrow \mathsf{A}[\textsc{Rate-1/3}, \mathsf{S}_2]) . \tag{7.19}$$

This change of the interface is indistinguishable for A:

$$\left|\mathbb{P}\left[\textbf{Game 2}\right] - \mathbb{P}\left[\textbf{Game 1}\right]\right| = 0. \tag{7.20}$$

Game 3 The interface in the third game is $(\textsc{Rate-1/3}, \mathsf{S}_3)$, where we introduce the bad events and R, note that $\mathbf{f}_3$ is distributed uniformly at random, so adding R does not change the distribution of $\mathbf{f}_3$ at all. The new game is

$$\textbf{Game 3} := (b = 1 : b \leftarrow \mathsf{A}[\textsc{Rate-1/3}, \mathsf{S}_3]) . \tag{7.21}$$

The bad events we introduce happen when either $\mathbf{f}_1$ or $\mathbf{f}_2$ outputs a value that forms a preimage of $\mathbf{f}_3$. The reason why these events are significant is because if we commit to an output of $\mathbf{f}_3$ and after that a query to $\mathbf{f}_1$ or $\mathbf{f}_2$ finishes the chain of values in the construction, then we introduce a discrepancy between the construction and the random oracle. The only noticeable change for the adversary are the bad event. To calculate distinguishability we use the fundamental game-playing lemma, Lemma 2.22:

$$\left|\mathbb{P}\left[\textbf{Game 3}\right] - \mathbb{P}\left[\textbf{Game 2}\right]\right| \leq \mathbb{P}\left[\mathrm{Bad}_1 \vee \mathrm{Bad}_2\right] \leq 2\frac{q^3}{N}, \tag{7.22}$$

where the right hand side follows from the fact that there are at most $s_2(i) \cdot s_3(i)$ pairs (y_2, x_3) that y_1 can collide with:

$$\mathbb{P}\left[\mathrm{Bad}_1\right] \leq \sum_{i=1}^{q} \frac{s_2(i) \cdot s_3(i)}{N} \leq \sum_{i=1}^{q} \frac{q^2}{N} = \frac{q^3}{N}, \tag{7.23}$$

where in the first inequality we use the union bound. The second inequality follows from a bound on the size of D_2 and D_3, after the i-th query $s_2(i), s_3(i) \leq q$. The final bound on $\mathbb{P}\left[\mathrm{Bad}_1 \vee \mathrm{Bad}_2\right]$ comes from the union bound and applying Equation (7.23) to Bad_1 and Bad_2.

Game 4 In the last step of the proof the interface is $(\mathsf{R}, \mathsf{S}_3)$, we change the private interface and the game is

$$\textbf{Game 4} := (b = 1 : b \leftarrow \mathsf{A}[\mathsf{R}, \mathsf{S}_3]) . \tag{7.24}$$

Similar to the proof of Comp we have:

$$\left|\mathbb{P}\left[\textbf{Game 4} \mid \neg\text{Bad}\right] - \mathbb{P}\left[\textbf{Game 3} \mid \neg\text{Bad}\right]\right| = 0. \tag{7.25}$$

Using the above identity we derive the final distinguishing advantage:

$$\left|\mathbb{P}\left[\textbf{Game 4}\right] - \mathbb{P}\left[\textbf{Game 3}\right]\right| \leq 4\mathbb{P}\left[\text{Bad}\right] \leq 8\frac{q^3}{N}, \tag{7.26}$$

where we use the derivation of Lemma 6.10. □

7.3.2 Quantum Indifferentiability

Quantum indifferentiability can proved in a very similar manner to the classical case.

Theorem 7.4. *The compression function* Rate-1/3$_{\mathsf{f}_1,\mathsf{f}_2,\mathsf{f}_3}$ *for uniformly random* f_1, f_2, *and* f_3 *is* (q,ε)*-quantumly indifferentiable for* $q \in O\left(\sqrt[4]{|\mathcal{Y}|}\right)$ *and* $\varepsilon = \sqrt{190(q+1)\frac{q^3}{N}} + 760\frac{q^3}{N}$.

Proof. The proof of quantum indifferentiability mirrors the classical proof. Again we define two simulators, the initial S_2 that just lazy samples the compression functions and S_3 that is the actual simulator. In the following we define $\mathfrak{U}_i$ to be the uniform distribution over functions in $\{\mathcal{X}_i \to \mathcal{Y}_i\}$.

Algorithm 7.4 Quantum simulators $\boxed{\mathsf{S}_2}$, $\boxed{\mathsf{S}_3}$ for Rate-1/3$_{\mathsf{f}_1,\mathsf{f}_2,\mathsf{f}_3}$

```
procedure f_1(x_1)
    Apply [CStO_{U_1}], [CStO_{U_1} \ R_{Rate-1/3}]        ▷ R_{Rate-1/3} defined in (6.173)

procedure f_2(x_2)
    Apply [CStO_{U_2}], [CStO_{U_2} \ R_{Rate-1/3}]

procedure f_3(x_3)
    if ∃y_1 ∈ D_1^Y, y_2 ∈ D_2^Y : x_3 = y_1 ⊕ y_2 then
        [Apply CStO_{U_3}]
        [return R(x_1, x_2) ⊕ y_1]
    else
        Apply CStO_{U_3}
```

Game 1 The interface in the first game is (Rate-1/3, $(\mathsf{f}_1,\mathsf{f}_2,\mathsf{f}_3)$), where f_1, f_2, and f_3 are uniformly random functions. The definition of the game is

$$\textbf{Game 1} := \left(b = 1 : b \leftarrow \mathsf{A}[\text{Rate-1/3}, (\mathsf{f}_1,\mathsf{f}_2,\mathsf{f}_3)]\right). \tag{7.27}$$

Game 2 In the second step we lazy sample the compression functions, the interface is $(\text{Rate-1/3}, \mathsf{S}_2)$. The new game is

$$\textbf{Game 2} := (b = 1 : b \leftarrow \mathsf{A}[\text{Rate-1/3}, \mathsf{S}_2]) \,. \tag{7.28}$$

Following Theorem 6.3, this change of the interface is indistinguishable for A:

$$\left|\mathbb{P}\left[\textbf{Game 2}\right] - \mathbb{P}\left[\textbf{Game 1}\right]\right| = 0. \tag{7.29}$$

Game 3 The interface in the third game is $(\text{Rate-1/3}, \mathsf{S}_3)$, where we introduce the punctured oracle and R, introducing the random oracle does not change the distribution of the outputs of $\mathbf{f}_3$ so this change does not add to the distinguishability advantage. The new game is defined as

$$\textbf{Game 3} := (b = 1 : b \leftarrow \mathsf{A}[\text{Rate-1/3}, \mathsf{S}_3]) \,. \tag{7.30}$$

We puncture on the same events as in the classical proof, relation $R_{\text{Rate-1/3}}$is defined in Equation (6.173). The only noticeable change for the adversary is the punctured oracle. The distinguishing advantage can be bounded by the O2H lemma, Theorem 6.7:

$$\left|\mathbb{P}\left[\textbf{Game 3}\right] - \mathbb{P}\left[\textbf{Game 2}\right]\right| \leq \sqrt{(q+1)\mathbb{P}\left[\text{Find}\right]} \leq \sqrt{190(q+1)\frac{q^3}{N}}, \tag{7.31}$$

where the bound on $\mathbb{P}\left[\text{Find}\right]$ comes from Corollary 6.19.

Game 4 In the last step of the proof, the interface is $(\mathsf{R}, \mathsf{S}_3)$, we change the private interface. The last game is defined as $\textbf{Game 4} := (b = 1 : b \leftarrow \mathsf{A}[\mathsf{R}, \mathsf{S}_3])$. Similar to the proof of Comp we have:

$$\left|\mathbb{P}\left[\textbf{Game 4} \mid \neg\text{Find}\right] - \mathbb{P}\left[\textbf{Game 3} \mid \neg\text{Find}\right]\right| = 0. \tag{7.32}$$

Using the above identity we derive the final distinguishing advantage:

$$\left|\mathbb{P}\left[\textbf{Game 4}\right] - \mathbb{P}\left[\textbf{Game 3}\right]\right| \leq 4\mathbb{P}\left[\text{Find}\right] \leq 760\frac{q^3}{N}, \tag{7.33}$$

where we use Lemma 6.10. □

7.4 Encrypted Davis-Mayer, one-way permutations

We present a proof of quantum indifferentiability of the Encrypted Davis-Mayer construction $\mathrm{EDM}_{\boldsymbol{\pi}_1,\boldsymbol{\pi}_2}(x) := \boldsymbol{\pi}_2\left(\boldsymbol{\pi}_1(x) \oplus x\right)$, where $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2$ are functions $\mathcal{X} \to \mathcal{X}$ and $\mathcal{X} = \{0,1\}^n$. Moreover $N = 2^n$. Further details on the construction can be found in Section 2.5.4.

7.4.1 Classical Indifferentiability

First we prove classical indifferentiability and then follow with the proof of quantum indifferentiability. There is however a limitation to our quantum result. Due to our inability of lazy sampling random permutations, we restrict our result to one-way permutations; The adversary has access to the internal permutations only in the forward direction. Classical indifferentiability is proven for regular permutations and the simulator is constructed in the standard way, that will allow us to translate the full result whenever we learn how to quantum lazy sample random permutations.

Theorem 7.5. *The compression function* $\mathrm{EDM}_{\pi_1,\pi_2}$ *for uniformly random permutations* π_1 *and* π_2 *is* (q,ε)*-classically indifferentiable for* $\varepsilon = 30\frac{q^2}{N}$.

Proof. We carry out the proof by starting with the real world and gradually changing the adversary's interface to the ideal world. We define two simulators, the initial S_2 that just lazy samples the compression functions and S_3 that is the actual simulator.

Algorithm 7.5 Classical simulators $\boxed{\mathsf{S}_2}$ and $\boxed{\mathsf{S}_3}$ for $\mathrm{EDM}_{\pi_1,\pi_2}$, Part I.

```
procedure $\pi_1(x_1)$
    if $\exists y_1 : (x_1, y_1) \in D_1$ return $y_1$
    $y_1 \xleftarrow{\$} \mathcal{X} \setminus D_1^Y$
    if $\exists (x_2, y_2) \in D_2 : y_1 = x_2 \oplus x_1$ then        ▷ Preimage of $\pi_2$
        $\boxed{\text{Set Bad}_1^+ = 1}$
    Add $(x_1, y_1)$ to $D_1$ and return $y_1$

procedure $\pi_1^{-1}(y_1)$
    if $\exists x_1 : (x_1, y_1) \in D_1$ return $x_1$
    $x_1 \xleftarrow{\$} \mathcal{X} \setminus D_1^X$
    if $\exists (x_2, y_2) \in D_2 : x_1 = x_2 \oplus y_1$ then        ▷ Preimage of $\pi_2$
        $\boxed{\text{Set Bad}_1^- = 1}$
    Add $(x_1, y_1)$ to $D_1$ and return $x_1$
```

Game 1 The interface in the first game is $(\mathrm{EDM}, (\pi_1, \pi_2))$. Note that the second interface is in fact four interfaces, the adversary has access to the forward and backward directions of the random permutations π_1 and π_2. The definition of the game is

$$\textbf{Game 1} := (b = 1 : b \leftarrow \mathsf{A}[\mathrm{EDM}, (\pi_1, \pi_2)]) . \tag{7.34}$$

Algorithm 7.6 Classical simulators $\boxed{\mathsf{S}_2}$ and $\boxed{\mathsf{S}_3}$ for $\mathrm{EDM}_{\pi_1,\pi_2}$, Part II.

```
procedure $\boldsymbol{\pi}_2(x_2)$
    if $\exists y_2 : (x_2, y_2) \in D_2$ return $y_2$
    if $\exists (x_1, y_1) \in D_1 :\ x_2 = x_1 \oplus y_1$ then
        $\boxed{y_2 \xleftarrow{\$} \mathcal{X} \setminus D_2^Y \text{, add } (x_2, y_2) \text{ to } D_2}$
        $\boxed{\textbf{return } y_2}$
        if $\exists y_2 \in D_2^Y : \mathsf{R}(x_1) = y_2$ then                ▷ Collision
            $\boxed{\text{Set } \mathrm{Bad}_{2,1}^+ = 1}$
        $\boxed{\textbf{return } \mathsf{R}(x_1)}$
    else
        $y_2 \xleftarrow{\$} \mathcal{X} \setminus D_2^Y$
        if $\exists (x_1, y_1) \in D_1 : y_2 = \mathsf{R}(x_1)$ then          ▷ Collision
            $\boxed{\text{Set } \mathrm{Bad}_{2,2}^+ = 1}$
        Add $(x_2, y_2)$ to $D_2$ and return $y_2$

procedure $\boldsymbol{\pi}_2^{-1}(y_2)$
    if $\exists x_2 : (x_2, y_2) \in D_2$ return $x_2$
    if $\exists (x_1, y_1) \in D_1 :\ \mathsf{R}(x_1) = y_2$ then
        $\boxed{x_2 \xleftarrow{\$} \mathcal{X} \setminus D_2^X \text{, add } (x_2, y_2) \text{ to } D_2}$
        $\boxed{\textbf{return } x_2}$
        if $\exists x_2 \in D_2^X : x_1 \oplus y_1 = x_2$ then          ▷ Not a function
            $\boxed{\text{Set } \mathrm{Bad}_{2,1}^- = 1}$
        $\boxed{\textbf{return } x_1 \oplus y_1}$
    else
        $x_2 \xleftarrow{\$} \mathcal{X} \setminus D_2^X$
        if $\exists (x_1, y_1) \in D_1 : x_2 = x_1 \oplus y_1$ then          ▷ Image of $\boldsymbol{\pi}_1$
            $\boxed{\text{Set } \mathrm{Bad}_{2,2}^- = 1}$
        Add $(x_2, y_2)$ to $D_2$ and return $x_2$
```

Game 2 In the second step we lazy sample the permutations, the interface is $(\mathrm{EDM}, \mathsf{S}_2)$. The new game is defined as

$$\textbf{Game 2} := (b = 1 : b \leftarrow \mathsf{A}[\mathrm{EDM}, \mathsf{S}_2]) \,. \tag{7.35}$$

This change of the interface is indistinguishable for A:

$$\left|\mathbb{P}\left[\textbf{Game 2}\right] - \mathbb{P}\left[\textbf{Game 1}\right]\right| = 0. \tag{7.36}$$

Game 3 The interface in the third game is $(\mathrm{EDM}, \mathsf{S}_3)$, where we introduce the bad events and R. Conditioned on $\neg$Bad the games are identically distributed, introducing the random oracle does not change the distribution of the outputs so this change does not add to the distinguishability advantage. The new game is

$$\textbf{Game 3} := (b = 1 : b \leftarrow \mathsf{A}[\mathrm{EDM}, \mathsf{S}_3]) \,. \tag{7.37}$$

There are two main sources of bad events: First being when a link between $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2$ is created; By this we mean that S_3 had already committed to the input output pair of $\boldsymbol{\pi}_1$ or $\boldsymbol{\pi}_2$ and the output of the queried function fits the construction and can be treated as part fo the input (or output) to the second function. The second source of errors are collisions of y_2 with $\mathsf{R}(x_1)$ or x_2 with $x_1 \oplus y_1$. The only noticeable change for the adversary are the bad event. To calculate distinguishability we use Lemma 2.22:

$$\left|\mathbb{P}\left[\textbf{Game 3}\right] - \mathbb{P}\left[\textbf{Game 2}\right]\right| \leq \mathbb{P}\left[\mathrm{Bad}\right] \leq 6\frac{q^2}{N}, \tag{7.38}$$

where $\mathrm{Bad} = \mathrm{Bad}_1^+ \vee \mathrm{Bad}_1^- \vee \mathrm{Bad}_{2,1}^+ \vee \mathrm{Bad}_{2,2}^+ \vee \mathrm{Bad}_{2,1}^- \vee \mathrm{Bad}_{2,2}^-$. Bound on $\mathbb{P}\left[\mathrm{Bad}\right]$ is derived similarly to Equation (7.6). By the union bound we split the events and bound every term by the greatest probability of a bad event (corresponding to the collision events).

Game 4 In the last game of the proof the interface is $(\mathsf{R}, \mathsf{S}_3)$, we change the private interface. Similar to the proof of Comp we have

$$\left|\mathbb{P}\left[\textbf{Game 4} \mid \neg\mathrm{Bad}\right] - \mathbb{P}\left[\textbf{Game 3} \mid \neg\mathrm{Bad}\right]\right| = 0. \tag{7.39}$$

Using the above identity we derive the final distinguishing advantage:

$$\left|\mathbb{P}\left[\textbf{Game 4}\right] - \mathbb{P}\left[\textbf{Game 3}\right]\right| \leq 4\mathbb{P}\left[\mathrm{Bad}\right] \leq 24\frac{q^2}{N}, \tag{7.40}$$

where we use the derivation of Lemma 6.10. This ends the proof of indifferentiability of the EDM construction. □

7.4.2 Quantum Indifferentiability

Quantum indifferentiability is proved in a very similar manner to the classical one. An important difference from the classical result is that we consider only one-way permutations. That means the adversary gets only forward access to the permutations. The source of this limitation is our inability to quantumly lazy-sample a random permutation.

Theorem 7.6. *The compression function* $\mathrm{EDM}_{\pi_1,\pi_2}$ *for uniformly random one-way permutations* $\boldsymbol{\pi}_1$ *and* $\boldsymbol{\pi}_2$ *is* (q,ε)*-quantumly indifferentiable for* $q \in O\left(\sqrt[3]{|\mathcal{Y}|}\right)$ *and* $\varepsilon = \frac{\pi^2}{3}\frac{(q+2)^3}{N} + 744\frac{q^2}{N} + \sqrt{186(q+1)\frac{q^2}{N}}$.

Proof. The proof of quantum indifferentiability mirrors the classical proof. Again we define two simulators, the initial S_2 that just lazy samples the compression functions and S_3 that is the actual simulator.

Algorithm 7.7 Quantum simulators $\boxed{\mathsf{S}_2}$, $\boxed{\mathsf{S}_3}$ for $\mathrm{EDM}_{\pi_1,\pi_2}$.

```
1: procedure $\boldsymbol{\pi}_1(x_1)$
2:     Apply $\boxed{\mathsf{CStO}_{\mathcal{Y}}}$, $\boxed{\mathsf{CStO}_{\mathcal{Y}} \setminus R_{\mathrm{EDM}}}$          ▷ Preimage of $\boldsymbol{\pi}_2$

3: procedure $\boldsymbol{\pi}_2(x_2)$
4:     if $\exists(x_1, y_1) \in D_1 :\ x_2 = x_1 \oplus y_1$ then
5:         Apply $\boxed{\mathsf{CStO}_{\mathcal{Y}}}$, $\boxed{\textbf{return } \mathsf{R}(x_1)}$
6:     else
7:         Apply $\mathsf{CStO}_{\mathcal{Y}}$
```

Game 1 The interface in the first game is $(\mathrm{EDM}, (\boldsymbol{\pi}_1, \boldsymbol{\pi}_2))$, where $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2$ are uniformly random permutations. The adversary has access only to the forward direction of the permutations. The definition of the game is

$$\textbf{Game 1} := (b = 1 : b \leftarrow \mathsf{A}[\mathrm{EDM}, (\boldsymbol{\pi}_1, \boldsymbol{\pi}_2)]) . \qquad (7.41)$$

Game 2 In the second game we lazy sample the compression functions, the interface is $(\mathrm{EDM}, \mathsf{S}_2)$. The new game is defined as

$$\textbf{Game 2} := (b = 1 : b \leftarrow \mathsf{A}[\mathrm{EDM}, \mathsf{S}_2]) . \qquad (7.42)$$

To swap the internal functions from random permutations to compressed oracles we first swap the permutations to random functions, then lazy sample the random functions. The first step can be bounded by using Theorem 7 in [Zha15a]; The bound is $\frac{\pi^2}{3}\frac{(q+2)^3}{N}$. Compressing a random function is done without error, as proven in Theorem 6.3:

$$\left|\mathbb{P}\left[\textbf{Game 2}\right] - \mathbb{P}\left[\textbf{Game 1}\right]\right| \leq \frac{\pi^2}{3}\frac{(q+2)^3}{N}. \qquad (7.43)$$

Game 3 The interface in the third game is $(\mathrm{EDM}, \mathsf{S}_3)$, where we introduce the punctured oracle and R, introducing the random oracle does not change the distribution of the outputs. The new game is

$$\textbf{Game 3} := (b = 1 : b \leftarrow \mathsf{A}[\mathrm{EDM}, \mathsf{S}_3]) \,. \tag{7.44}$$

We puncture on the same events as in the classical proof, the relation R_{EDM} is defined in Equation (6.175). The only noticeable change for the adversary is the punctured oracle. The distinguishing advantage can be bounded by the O2H lemma, Theorem 6.7:

$$\left|\mathbb{P}\left[\textbf{Game 3}\right] - \mathbb{P}\left[\textbf{Game 2}\right]\right| \le \sqrt{(q+1)\mathbb{P}\left[\mathrm{Find}\right]} \le \sqrt{186(q+1)\frac{q^2}{N}}, \tag{7.45}$$

where the bound on $\mathbb{P}\left[\mathrm{Find}\right]$ comes from Corollary 6.20.

Game 4 In the last game the interface is $(\mathsf{R}, \mathsf{S}_3)$, we change the private interface. The last game is defined as

$$\textbf{Game 4} := (b = 1 : b \leftarrow \mathsf{A}[\mathsf{R}, \mathsf{S}_3]) \,. \tag{7.46}$$

Similar to the proof of Comp we have:

$$\left|\mathbb{P}\left[\textbf{Game 4} \mid \neg\mathrm{Bad}\right] - \mathbb{P}\left[\textbf{Game 3} \mid \neg\mathrm{Bad}\right]\right| = 0. \tag{7.47}$$

Using the above identity we derive the final distinguishing advantage:

$$\left|\mathbb{P}\left[\textbf{Game 4}\right] - \mathbb{P}\left[\textbf{Game 3}\right]\right| \le 4\mathbb{P}\left[\mathrm{Find}\right] \le 4 \cdot 186\frac{q^2}{N}, \tag{7.48}$$

where we use Lemma 6.10. □

7.5 Encrypted Davis-Mayer Dual, one-way permutations

We present a proof of quantum indifferentiability of the Encrypted Davis-Mayer Dual construction. Details on the construction can be found in Section 2.5.5. The setting is the same as in the previous section. We just present the simulators, because all the rest follows exactly the proofs from Section 7.4.

7.5.1 Classical Indifferentiability

Theorem 7.7. *The compression function* $\mathrm{EDMD}_{\boldsymbol{\pi}_1,\boldsymbol{\pi}_2}$ *for uniformly random permutations* $\boldsymbol{\pi}_1$ *and* $\boldsymbol{\pi}_2$ *is* (q,ε)*-classically indifferentiable for* $\varepsilon = 15\frac{q^2}{N}$.

Algorithm 7.8 Classical simulators $\boxed{\mathsf{S}_2}$ and $\boxed{\mathsf{S}_3}$ for EDMD$_{\pi_1,\pi_2}$.

```
1:  procedure $\pi_1(x_1)$
2:      if $\exists y_1 : (x_1, y_1) \in D_1$ return $y_1$
3:      $y_1 \xleftarrow{\$} \mathcal{X} \setminus D_1^Y$
4:      if $\exists (x_2, y_2) \in D_2 : y_1 = x_2$ then                ▷ Preimage of $\pi_2$
5:          $\boxed{\text{Set Bad}_1^+ = 1}$
6:      Add $(x_1, y_1)$ to $D_1$ and return $y_1$

7:  procedure $\pi_1^{-1}(y_1)$
8:      if $\exists x_1 : (x_1, y_1) \in D_1$ return $x_1$
9:      $x_1 \xleftarrow{\$} \mathcal{X} \setminus D_1^X$, add $(x_1, y_1)$ to $D_1$
10:     return $x_1$

11: procedure $\pi_2(x_2)$
12:     if $\exists y_2 : (x_2, y_2) \in D_2$ return $y_2$
13:     $\boxed{y_2 \xleftarrow{\$} \mathcal{X} \setminus D_2^Y, \text{ add } (x_2, y_2) \text{ to } D_2}$
14:     $\boxed{\textbf{return } y_2}$
15:     $\boxed{x_1 \leftarrow \pi_1^{-1}(x_2)}$
16:     if $\mathsf{R}(x_1) \oplus x_2 \in D_2^Y$ then                  ▷ Collision
17:         $\boxed{\text{Set Bad}_2^+ = 1}$
18:     $\boxed{\text{Add } (x_2, \mathsf{R}(x_1) \oplus x_2) \text{ to } D_2 \text{ and } \textbf{return } \mathsf{R}(x_1) \oplus x_2}$

19: procedure $\pi_2^{-1}(y_2)$
20:     if $\exists x_2 : (x_2, y_2) \in D_2$ return $x_2$
21:     $\boxed{x_2 \xleftarrow{\$} \mathcal{X} \setminus D_2^X, \text{ add } (x_2, y_2) \text{ to } D_2}$
22:     $\boxed{\textbf{return } y_2}$
23:     $\boxed{x_1 \xleftarrow{\$} \mathcal{X} \setminus D_1^X,\ y_1 := \mathsf{R}(x_1) \oplus y_2,\ x_2 := \mathsf{R}(x_1) \oplus y_2}$
24:     if $y_1 \in D_1^Y$ or $x_2 \in D_2^X$ then                    ▷ Collision
25:         $\boxed{\text{Set Bad}_2^- = 1}$
26:     $\boxed{\text{Add } (x_2, y_2) \text{ to } D_2 \text{ and } (x_1, y_1) \text{ to } D_1}$
27:     $\boxed{\textbf{return } x_2}$
```

Proof. The simulators are presented in Algorithm 7.8

Game 1 The interface in the first game is $(\mathrm{EDMD}, (\boldsymbol{\pi}_1, \boldsymbol{\pi}_2))$. Note that the second interface is in fact four interfaces, the adversary has access to the forward and backward directions of the uniformly random permutations $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2$. The definition of the game is

$$\textbf{Game 1} := (b = 1 : b \leftarrow \mathsf{A}[\mathrm{EDMD}, (\boldsymbol{\pi}_1, \boldsymbol{\pi}_2)]) . \tag{7.49}$$

Game 2 In the second game we lazy sample the permutations, the interface is $(\mathrm{EDMD}, \mathsf{S}_2)$. The definition of the new game is

$$\textbf{Game 2} := (b = 1 : b \leftarrow \mathsf{A}[\mathrm{EDMD}, \mathsf{S}_2]) . \tag{7.50}$$

This change of the interface is indistinguishable for A:

$$\left|\mathbb{P}\left[\textbf{Game 2}\right] - \mathbb{P}\left[\textbf{Game 1}\right]\right| = 0. \tag{7.51}$$

Game 3 The interface in the third game is $(\mathrm{EDMD}, \mathsf{S}_3)$. We change the source of randomness but as R is uniformly random, the conditional distributions are the same. The new game is

$$\textbf{Game 3} := (b = 1 : b \leftarrow \mathsf{A}[\mathrm{EDMD}, \mathsf{S}_3]) . \tag{7.52}$$

Hence the only source of distinguishing advantage are the bad events. Bad events occur whenever a new output collides with the previous ones. To calculate distinguishability we use Lemma 2.22:

$$\left|\mathbb{P}\left[\textbf{Game 3}\right] - \mathbb{P}\left[\textbf{Game 2}\right]\right| \leq \mathbb{P}\left[\mathrm{Bad}\right] \leq 3\frac{q^2}{N}, \tag{7.53}$$

where $\mathrm{Bad} = \mathrm{Bad}_1^+ \vee \mathrm{Bad}_2^+ \vee \mathrm{Bad}_2^-$. A bound on $\mathbb{P}\left[\mathrm{Bad}\right]$ is derived by using the union bound, note that when querying $\boldsymbol{\pi}_2^{-1}$ a collision might occur in D_1 and D_2.

Game 4 In the last game, defined as

$$\textbf{Game 4} := (b = 1 : b \leftarrow \mathsf{A}[\mathsf{R}, \mathsf{S}_3]) , \tag{7.54}$$

the interface is $(\mathsf{R}, \mathsf{S}_3)$, we change the private interface. Similar to the proof of Comp we have

$$\left|\mathbb{P}\left[\textbf{Game 4} \mid \neg\mathrm{Bad}\right] - \mathbb{P}\left[\textbf{Game 3} \mid \neg\mathrm{Bad}\right]\right| = 0. \tag{7.55}$$

Using the above identity we derive the final distinguishing advantage:

$$\left|\mathbb{P}\left[\textbf{Game 4}\right] - \mathbb{P}\left[\textbf{Game 3}\right]\right| \leq 4\mathbb{P}\left[\mathrm{Bad}\right] \leq 12\frac{q^2}{N}, \tag{7.56}$$

where we use the derivation of Lemma 6.10. □

7.5.2 Quantum Indifferentiability

Quantum indifferentiability is proved in a very similar manner to the classical one. An important difference from the classical result is that we consider only one-way permutations. That means the adversary gets only forward access to the permutations. The source of this limitation is our inability to quantumly lazy-sample a random permutation.

Theorem 7.8. *The compression function* $\mathrm{EDMD}_{\pi_1,\pi_2}$ *for uniformly random one-way permutations* π_1 *and* π_2 *is* (ε, q)*-quantumly indifferentiable for* $q \in O\left(\sqrt[3]{|\mathcal{Y}|}\right)$ *and* $\varepsilon = \frac{\pi^2}{3}\frac{(q+2)^3}{N} + 744\frac{q^2}{N} + \sqrt{186(q+1)\frac{q^2}{N}}$.

Proof. The proof of quantum indifferentiability mirrors the classical proof. Again we define two simulators, the initial S_2 that just lazy samples the compression functions and S_3 that is the actual simulator.

Algorithm 7.9 Quantum simulators $\boxed{\mathsf{S}_2}$, $\boxed{\mathsf{S}_3}$ for $\mathrm{EDMD}_{\pi_1,\pi_2}$.

```
1: procedure $\pi_1(x_1)$
2:     Apply $\boxed{\mathsf{CStO}_{\mathcal{Y}}}$, $\boxed{\mathsf{CStO}_{\mathcal{Y}} \setminus R_{\mathrm{EDMD}}}$        ▷ Preimage of $\pi_2$

3: procedure $\pi_2(x_2)$
4:     if $\exists (x_1, y_1) \in D_1 :\ x_2 = x_1 \oplus y_1$ then
5:         Apply $\boxed{\mathsf{CStO}_{\mathcal{Y}}}$, $\boxed{\textbf{return } \mathsf{R}(x_1)}$
6:     else
7:         Apply $\mathsf{CStO}_{\mathcal{Y}}$
```

Rest of the proof is exactly the same as the proof of Theorem 7.6. □

7.6 Sponges with Random Functions

In the game-playing proofs and Algorithms 7.10 and 7.11 described in this chapter we use the following convention: every version of the algorithm executes the part of the code that is *not boxed* and among the boxed statements only the part that is inside the box in the color corresponding to the color of the name in the definition.

In the case of an adversary querying a random function f we are going to treat the sponge graph as being created one edge per query. The graph G then symbolizes the current state of knowledge of the adversary of the internal function. Note that this dynamical graph can be created efficiently by focusing solely on nodes that appear in the queried edges.

The simulators defined in the proofs in this section are implicitly stateful. They maintain a classical or quantum state containing a database of the adversary's queries and the simulator's outputs. Using that database the simulator can always construct a sponge graph containing all the current knowledge of $\mathbf{f}$.

7.6.1 Classical Indifferentiability

For the proof of indifferentiability we also need an upper bound on the probability of finding a collision in the inner part of outputs of a uniformly random function $\mathbf{f} : \mathcal{A} \times \mathcal{C} \to \mathcal{A} \times \mathcal{C}$. Considering how Sponge is defined we want a bound on finding collisions and zero-preimages. We define the bound as a function of the number of queries q to $\mathbf{f}$:

$$\mathbf{f}_{\text{coll}}(q) := \frac{q(q+1)}{2\,|\mathcal{C}|}, \tag{7.57}$$

the bound can be derived in the standard way. The probability that any classical algorithm finds a collision or a preimage of zero in $[N]$ after q queries is:

$$\mathbb{P}\left[\text{coll} \cup \text{preim} \leftarrow \mathrm{A}\right] \leq \sum_{i=1}^{q} \frac{i}{N} = \frac{q(q+1)}{2N}, \tag{7.58}$$

where we use the union bound and note that after i queries the adversary can either find the preimeage of zero or hit any of the previous outputs, producing a collision. For a more detailed derivation we refer to Appendix A.4 in [KL14].

First we present a slightly modified proof of indifferentiability from [Ber+08]. In a recent paper [Alm+19] the proof of indifferentiability of Sponge was formally verified using the EasyCrypt proof assistant. We modify the original proof to better fit the framework of game-playing proofs. It is not our goal to obtain the tightest bounds nor the simplest (classical) proof. Instead, our classical game-playing proof paves the way to the quantum security proof which is presented in the next section.

Theorem 7.9 (Classical indifferentiability of Sponge). *Sponge* Sponge$_{\mathbf{f}}[\textsc{pad}, \mathcal{A}, \mathcal{C}]$ *calling a random function* $\mathbf{f}$ *is* (q, ε)*-indifferentiable from a random oracle, for* classical *adversaries for any* $q < |\mathcal{C}|$ *and* $\varepsilon = 8\frac{q(q+1)}{2|\mathcal{C}|}$.

Proof. The proof proceeds in six games that we show to be indistinguishable. We start with the real world: the public interface corresponding to the internal function $\mathbf{f}$ is a random transformation and the private interface is Sponge$_{\mathbf{f}}$. Then in a series of games we gradually change the environment of the adversary to finally reach the ideal world, where the public interface is simulated by the simulator and the private interface is a random oracle R. The simulators used in different games of the proof are defined in Algorithm 7.10, the index of

Algorithm 7.10 Classical S_2, $\boxed{\mathsf{S}_3}$, $\boxed{\mathsf{S}_4}$, $\boxed{\mathsf{I}_6}$, functions

```
State: current sponge graph $G$
input: $s \in \mathcal{A} \times \mathcal{C}$
output: $\mathbf{f}(s)$
1:  if $s$ has no outgoing edge then                          ▷ Fresh query
2:      if $\hat{s} \in \mathcal{R} \wedge \mathcal{R} \cup \mathcal{U} \neq \mathcal{C}$ then      ▷ $\hat{s}$-rooted, no saturation
3:          $\hat{t} \xleftarrow{\$} \mathcal{C}$, $\boxed{\boxed{\text{if } \hat{t} \in \mathcal{R} \cup \mathcal{U}, \text{ set Bad} = 1}}$, $\boxed{\hat{t} \xleftarrow{\$} \mathcal{C} \setminus (\mathcal{R} \cup \mathcal{U})}$
4:          Construct a path to $s$: $p := \mathsf{SpPath}(s, G)$
5:          if $\exists x : p = \textsc{pad}(x)$ then
6:              $\bar{t} \xleftarrow{\$} \mathcal{A}$
7:              $\boxed{\boxed{\bar{t} := \mathsf{R}(x)}}$
8:          else
9:              $\bar{t} \xleftarrow{\$} \mathcal{A}$
10:         $t := (\bar{t}, \hat{t})$
11:     else
12:         $t \xleftarrow{\$} \mathcal{A} \times \mathcal{C}$
13:     Add an edge $(s, t)$ to $\mathcal{E}$.
14: Set $t$ to the vertex at the end of the edge starting at $s$
15: return $t$
```

the simulator corresponds to the game in which the simulator is used. Explanations of the simulators follow.

Game 1 We start with the real world where the distinguisher A has access to a random function $\mathbf{f} : \mathcal{A} \times \mathcal{C} \to \mathcal{A} \times \mathcal{C}$ and $\textsc{Sponge}_{\mathbf{f}}$ using this random function. The formal definition of the first game is the event

$$\textbf{Game 1} := (b = 1 : b \leftarrow \mathsf{A}[\textsc{Sponge}_{\mathbf{f}}, \mathbf{f}]) . \tag{7.59}$$

Game 2 In the second game we introduce the simulator S_2—defined in Algorithm 7.10—that lazy-samples the random function $\mathbf{f}$. In Algorithm 7.10 we define all simulators of this proof at once, but note that the behavior of S_2 is not influenced by any of the conditional "if" statements (in lines 1, 2, and 5), because in the end, the output state t is picked uniformly from $\mathcal{A} \times \mathcal{C}$ anyway. The definition of the second game is

$$\textbf{Game 2} := (b = 1 : b \leftarrow \mathsf{A}[\textsc{Sponge}_{\mathsf{S}_2}, \mathsf{S}_2]) . \tag{7.60}$$

Because the simulator S_2 perfectly models a random function and we use the same function for the private interface we have

$$|\mathbb{P}[\textbf{Game 2}] - \mathbb{P}[\textbf{Game 1}]| = 0. \tag{7.61}$$

Game 3 In the next step we modify S_2 to S_3. The game is then

$$\textbf{Game 3} := (b = 1 : b \leftarrow \mathrm{A}[\textsc{Sponge}_{\mathsf{S}_3}, \mathsf{S}_3]) \,. \tag{7.62}$$

We made a single change in S_3 compared to S_2, we introduce the "bad" event Bad that marks the difference between algorithms. The intuition behind the event Bad is that if the new query ends up in $\mathcal{R}$ we found an inner collision or a preimage of zero. This is because all supernodes in $\mathcal{R}$ are the output of a query or are just 0. If on the other hand $\hat{t} \in \mathcal{U}$, the simulator might be caught on an inconsistency. If a node has an outgoing edge and is not in $\mathcal{R}$, then the outer part of the output is a uniformly random value, not an output of H as will be desired in the next game. We use this event as the bad event in Lemma 2.22. With such a change of the simulators we can use Lemma 2.22 to bound the difference of probabilities:

$$|\mathbb{P}[\textbf{Game 3}] - \mathbb{P}[\textbf{Game 2}]| \leq \mathbb{P}[\mathrm{Bad}]. \tag{7.63}$$

We can use the bound because in the worse case (i.e., all previous queries being in $\mathcal{R} \cup \mathcal{U}$) the bad event is equivalent to finding inner-collisions or preimages of zero.

It is quite easy to bound $\mathbb{P}[\mathrm{Bad} = 1]$ as it is the probability of finding a collision or preimage of the root in the set $\mathcal{C}$ having made q random samples. Therefore we have that

$$\mathbb{P}[\mathrm{Bad} = 1] \leq \mathbf{f}_{\mathrm{coll}}(q), \tag{7.64}$$

where f_{coll} is defined in Equation (7.57). The bound is not necessarily tight as not all queries are made to rooted nodes.

Game 4 In this step we introduce the random oracle R but only to generate the outer part of the output of $\mathbf{f}$. The game is defined as

$$\textbf{Game 4} := \Big(b = 1 : b \leftarrow \mathrm{A}[\textsc{Sponge}_{\mathsf{S}_4}, \mathsf{S}_4^{\mathsf{R}}]\Big) \,. \tag{7.65}$$

We observe that if $\mathrm{Bad} = 0$ the outputs are identically distributed.

Claim 7.10. *Given that* $\mathrm{Bad} = 0$ *the mentioned games are the same:*

$$|\mathbb{P}[\textbf{Game 4} \mid \neg\mathrm{Bad}] - \mathbb{P}[\textbf{Game 3} \mid \neg\mathrm{Bad}]| = 0. \tag{7.66}$$

Proof. Note that the inner part is distributed in the same way in both games if $\neg$Bad, so we only need to take care of the outer part of the output. The problem might lie in the outer part, as we modify the output from a random sample to $\mathsf{R}(x)$. If $\mathrm{Bad} = 0$ then $\hat{t}$ is not rooted and has no outgoing edge. Given that Bad was never set to 1, the whole graph G does not contain two paths leading to the same supernode (Lemma 1 in [Ber+08]). Hence, x was not queried before and is uniformly random. This reasoning is made more formal in Lemma 1 and Lemma 2 of [Ber+08]. □

The two games are identical-until-bad, this implies that the probability of setting Bad to one in both games is the same $\mathbb{P}[\text{Bad} = 1 : \textbf{Game 3}] = \mathbb{P}[\text{Bad} = 1 : \textbf{Game 4}]$. Together with the above claim we can derive the advantage:

$$\begin{aligned}|\mathbb{P}[\textbf{Game 4}] - \mathbb{P}[\textbf{Game 3}]| \overset{\text{Claim 7.10}}{=} \Big| & \mathbb{P}[\textbf{Game 4} \mid \neg\text{Bad}] \\ & \cdot \underbrace{(\mathbb{P}[\neg\text{Bad} : \textbf{Game 4}] - \mathbb{P}[\neg\text{Bad} : \textbf{Game 3}])}_{=0} \\ & + \underbrace{\mathbb{P}[\textbf{Game 4} \mid \text{Bad}]}_{\leq 1}\mathbb{P}[\text{Bad}] - \underbrace{\mathbb{P}[\textbf{Game 3} \mid \text{Bad}]}_{\leq 1}\mathbb{P}[\text{Bad}]\Big| \qquad (7.67) \\ \overset{\triangle\text{-inEquation}}{\leq} \quad & 2\mathbb{P}[\text{Bad}]. \qquad (7.68)\end{aligned}$$

Note that the above derivation follows the proof of Lemma 6.10, with $\mathsf{H} \setminus R_1$ replaced by **Game 4**, $\mathsf{G} \setminus R_2$ replaced by **Game 3**, and event Find replaced by Bad.

Game 5 In this stage of the proof we change the private interface to contain the actual random oracle. The simulator is the same as before and the game is

$$\textbf{Game 5} := \left(b = 1 : b \leftarrow \mathsf{A}[\mathsf{R}, \mathsf{S}_4^{\mathsf{R}}]\right). \qquad (7.69)$$

Conditioned on Bad $= 0$, the outputs of the simulator in **Game 4** and **Game 5** are consistent with R. Moreover conditioned on Bad $= 0$ the probabilities of A outputting 1 are the same. Note that the inner states are generated by the same pseudocode and the outer states are distributed in the same way. To calculate the adversary's advantage in distinguishing between the two games we can follow the proof of Lemma 6.10 and Equation (7.68). As the derivation of Lemma 6.10 uses no quantum mechanical arguments and the assumption holds—the games are identical conditioned on Bad $= 0$—the bound holds:

$$|\mathbb{P}[\textbf{Game 5}] - \mathbb{P}[\textbf{Game 4}]| \leq 4\mathbb{P}[\text{Bad}] \leq 4\mathbf{f}_{\text{coll}}(q). \qquad (7.70)$$

Game 6 In the last game we use I_6 (we call it I for *ideal*, that is the world we arrive in the last step of the proof), a simulator that does not check for bad events and samples from the "good" subset of $\mathcal{C}$. The game is

$$\textbf{Game 6} := \left(b = 1 : b \leftarrow \mathsf{A}[\mathsf{R}, \mathsf{I}_6^{\mathsf{R}}]\right) \qquad (7.71)$$

and the advantage is

$$|\mathbb{P}[\textbf{Game 6}] - \mathbb{P}[\textbf{Game 5}]| \leq \mathbb{P}[\text{Bad}] \leq \mathbf{f}_{\text{coll}}(q). \qquad (7.72)$$

following Lemma 2.22. as the only difference is in code but not outputs. We included this last game in the proof because I_6 is clearly a simulator that might fail only if G is saturated but this does not happen if $q < |\mathcal{C}|$. Collecting and adding all the differences yields the claimed $\varepsilon = 8\mathbf{f}_{\text{coll}}(q)$. □

7.6.2 Quantum Indifferentiability

In this subsection we prove quantum indifferentiability of the sponge construction with a uniformly random internal function. Without loss of generality we assume $\mathcal{C} = [|\mathcal{C}|]$ and perform modular addition on the whole state in $\mathcal{A} \times \mathcal{C}$.

In the quantum indifferentiability simulator we want to sample the outer part of outputs of $\mathbf{f}$ and the inner part separately, similarly to the classical one. To do that correctly in the quantum case though we need to maintain two databases: one responsible for the outer part and the other for the inner part. We denote them by $\overline{D}$ and $\widehat{D}$ respectively.

At line 7 of the classical simulator we replace the lazy sampled outer state by the output of the random oracle. In the quantum case we want to do the same. Unlike in the classical case we cannot, however, save the input-output pairs of the random oracle R that were sampled to generate the sponge graph, as they contain information about the adversary's query input. An attempt to store this data would effectively measure the adversary's state and render our simulation distinguishable from the real world. To get around this issue we reprepare the sponge graph at the beginning of each run of the simulator. To prepare the sponge graph we query R on all necessary inputs to $\hat{\mathbf{f}}$, i.e. on the inputs that are consistent with a path from the root to a rooted node. This is done gradually by iterating over the length of the paths. We begin with the length-0 paths, i.e. with all inputs in the database $\widehat{D}$ where the inner part is the all zero string. If the outer part of such an input is equal to a padding of an input, that input is queried to determine the outer part of the output of $\mathbf{f}$, creating an edge in the sponge graph. We can continue with length-1 paths. For each entry of the database $\widehat{D}$, check whether the input register is equal to a node in the current partial sponge graph. If so, the entry corresponds to a rooted node. Using the entry and the edge connecting its input to the root, a possible padded input to SPONGE is created using SpPath. If it is a valid padding, R is queried to determine the outer part of the output of $\mathbf{f}$, etc.

In the proof we will make heavy use of the result of Corollary 6.17. Let us denote the bound on inner collisions by

$$\mathbf{f}_{\mathrm{coll}}^{Q}(q) := 11\frac{q^2}{|\mathcal{C}|}. \tag{7.73}$$

Theorem 7.11 (Quantum indifferentiability of SPONGE)**.** *Sponge* $\mathrm{SPONGE}_{\mathbf{f}}[\mathrm{PAD}, \mathcal{A}, \mathcal{C}]$ *calling a random function* $\mathbf{f}$ *is* (q, ε)*-indifferentiable from a random oracle, for* quantum *adversaries for* $q \in O\left(\sqrt[3]{|\mathcal{C}|}\right)$ *and*

$$\varepsilon = 88\frac{q^2}{|\mathcal{C}|} + \sqrt{11(q+1)\frac{q^2}{|\mathcal{C}|}}. \tag{7.74}$$

Proof. Even though we allow for quantum accessible oracles, the proof we present is very similar to the classical case. The proof follows the same structure, the biggest difference is in the simulators that use the compressed oracle to lazy-sample appropriate answers.

We denote by U_G the unitary that acting on $|0\rangle$ constructs G including edges consistent with queries held by the quantum compressed database from register D. Similarly we define $\mathsf{U}_{\mathcal{R}\cup\mathcal{U}}$ to temporarily create a description of the set of supernodes that are rooted or have an outgoing edge.

In Algorithm 7.11 we describe the simulators we use in this proof. In the quantum simulators we also make use of the graph representation of sponges. Note however that in a single query we only care about the graph before the query. Due to that fact we can apply the compressed oracle defined in Algorithm 6.1 and additionally analyzed in Lemma 6.11. Corollary 6.17 provides a bound of the probability of Find in the case of compressed oracles and relations relevant for the sponge construction.

It is important to note that the "IF" statements are in fact quantum controlled operations. In line 5 we apply a punctured compressed oracle controlled on the input and the database; To correctly perform this operation we postpone the measurement to after uncomputing of G and $\mathcal{R}\cup\mathcal{U}$ in line 15. This procedure is also discussed at the end of Section 6.3.

An illustration of the simulators in the quantum case is depicted in Figure 7.1.

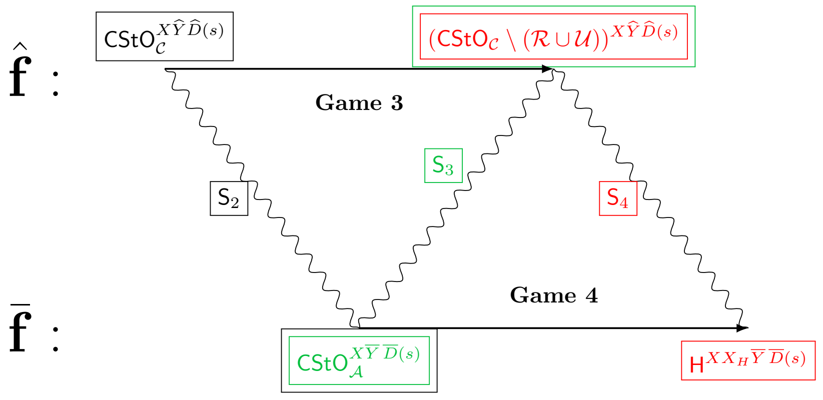

$\widehat{\mathbf{f}}$:
$\mathsf{CStO}_{\mathcal{C}}^{X\widehat{Y}\widehat{D}(s)}$
$(\mathsf{CStO}_{\mathcal{C}}\setminus(\mathcal{R}\cup\mathcal{U}))^{X\widehat{Y}\widehat{D}(s)}$
Game 3
S_3
S_2
S_4
Game 4
$\overline{\mathbf{f}}$:
$\mathsf{CStO}_{\mathcal{A}}^{X\overline{Y}\,\overline{D}(s)}$
$\mathsf{H}^{XX_H\overline{Y}\,\overline{D}(s)}$

Figure 7.1: Schematics of the simulators defined in Algorithm 7.11, horizontal arrows signify the change introduced in the labeled game.

Game 1 We start with the real world where the distinguisher A has quantum access to a random function $\mathbf{f}:\mathcal{A}\times\mathcal{C}\to\mathcal{A}\times\mathcal{C}$ and the $\textsc{Sponge}_{\mathbf{f}}$ construction using this random function. The definition of the first game is

$$\textbf{Game 1} := \left(b=1 : b\leftarrow \mathrm{A}[\textsc{Sponge}_{\mathbf{f}},\mathbf{f}]\right). \tag{7.75}$$

Algorithm 7.11 Quantum $\boxed{\mathsf{S}_2}$, $\boxed{\mathsf{S}_3}$, $\boxed{\mathsf{S}_4}$, functions

```
1:  State: Quantum compressed database register $D$
    input: $|s, v\rangle \in \mathcal{H}_{\mathcal{A}\times\mathcal{C}}^{\otimes 2}$
    output: $|s, v + \mathbf{f}(s)\rangle$
2:  Locate input $s$ in $\overline{D}$ and $\widehat{D}$
3:  Apply $\mathsf{U}_{\mathcal{R}\cup\mathcal{U}} \circ \mathsf{U}_G$ to register $\widehat{D}$ and two fresh registers
4:  if $\hat{s} \in \mathcal{R} \ \wedge \ \mathcal{R}\cup\mathcal{U} \neq \mathcal{C}$ then        ▷ $\hat{s}$-rooted, no saturation
5:      Apply $\boxed{\mathsf{CStO}_{\mathcal{C}}^{X\widehat{Y}\widehat{D}(s)}}$, $\boxed{\boxed{(\mathsf{CStO}_{\mathcal{C}} \setminus (\mathcal{R}\cup\mathcal{U}))^{X\widehat{Y}\widehat{D}(s)}}}$, result: $\hat{t}$
6:      Construct a path to $s$: $p := \mathsf{SpPath}(s, G)$
7:      if $\exists x : p = \textsc{pad}(x)$ then
8:          $\boxed{\boxed{\text{Apply } \mathsf{CStO}_{\mathcal{A}}^{X\overline{Y}\,\overline{D}(s)}}}$, result: $\bar{t}$
9:          Write $x$ in a fresh register $X_R$, $\boxed{\text{apply } \mathsf{R}^{XX_R\overline{Y}\,\overline{D}(s)}}$, uncompute $x$ from
    $X_R$, result: $\bar{t}$
10:     else
11:         Apply $\mathsf{CStO}_{\mathcal{A}}^{X\overline{Y}\,\overline{D}(s)}$, result: $\bar{t}$
12:     $t := (\bar{t}, \hat{t})$, the value of registers $(\overline{D}^Y(s), \widehat{D}^Y(s))$
13: else
14:     Apply $\mathsf{CStO}_{\mathcal{A}\times\mathcal{C}}^{XY\overline{D}(s)\widehat{D}(s)}$, result: $t$
15: Uncompute $G$ and $\mathcal{R}\cup\mathcal{U}$
16: return $|s, v + t\rangle$
```

Game 2 In the second game we introduce the simulator S_2, defined in Algorithm 7.11. This algorithm is essentially a compressed random oracle, the only difference are the if statements, note that the behavior of S_2 is not influenced by any of the conditional "if" statements (in lines 4, and 7), because in the end, the output state t is picked uniformly from $\mathcal{A}\times\mathcal{C}$ anyway. The game is defined as:

$$\textbf{Game 2} := \left(b = 1 : b \leftarrow \mathsf{A}[\textsc{Sponge}_{\mathsf{S}_2}, \mathsf{S}_2]\right). \tag{7.76}$$

Because the simulator S_2 perfectly models a quantum random function and we use the same function for the private interface we have

$$|\mathbb{P}[\textbf{Game 2}] - \mathbb{P}[\textbf{Game 1}]| = 0. \tag{7.77}$$

Game 3 In the next step we modify S_2 to S_3. The game is then

$$\textbf{Game 3} := \left(b = 1 : b \leftarrow \mathsf{A}[\textsc{Sponge}_{\mathsf{S}_3}, \mathsf{S}_3]\right). \tag{7.78}$$

With such a change of the simulators we can use Theorem 6.7 to bound the difference of probabilities. S_3 measures the relation of being an element of $\mathcal{R}\cup$

$\mathcal{U}$. This relation is equivalent to $R_{\text{preim}} \cup R_{\text{coll}}$:

$$|\mathbb{P}[\textbf{Game 3}] - \mathbb{P}[\textbf{Game 2}]| \leq \sqrt{(q+1)\mathbb{P}[\text{Find} : \mathsf{A}[\textsc{Sponge}_{\mathsf{S}_3}, \mathsf{S}_3]]}, \tag{7.79}$$

Using Corollary 6.17 we have that

$$\mathbb{P}[\text{Find} : \mathsf{A}[\textsc{Sponge}_{\mathsf{S}_3}, \mathsf{S}_3]] \leq \mathbf{f}^{Q}_{\text{coll}}(q). \tag{7.80}$$

Game 4 In this step we introduce the random oracle R but only to generate the outer part of the output of $\mathbf{f}$. The game is defined as

$$\textbf{Game 4} := \left(b = 1 : b \leftarrow \mathsf{A}[\textsc{Sponge}_{\mathsf{S}_4}, \mathsf{S}_4^{\mathsf{R}}]\right). \tag{7.81}$$

Thanks to the classical argument we have that S_4 and S_3 are identical until bad, as in Definition 6.9. Then we can use Lemma 6.10 to bound the advantage of the adversary

$$|\mathbb{P}[\textbf{Game 4}] - \mathbb{P}[\textbf{Game 3}]| \leq 4\mathbb{P}[\text{Find} : \mathsf{A}[\textsc{Sponge}_{\mathsf{S}_3}, \mathsf{S}_3]] \leq 4\mathbf{f}^{Q}_{\text{coll}}(q). \tag{7.82}$$

Game 5 In this stage of the proof we change the private interface to contain the actual random oracle. In this game the simulator is still S_4, the definition is as follows:

$$\textbf{Game 5} := \left(b = 1 : b \leftarrow \mathsf{A}[\mathsf{R}, \mathsf{S}_4^{\mathsf{R}}]\right) \tag{7.83}$$

and the advantage is

$$|\mathbb{P}[\textbf{Game 5}] - \mathbb{P}[\textbf{Game 4}]| \leq 4\mathbb{P}[\text{Find} : \mathsf{A}[\textsc{Sponge}_{\mathsf{S}_4}, \mathsf{S}_4^{\mathsf{R}}]] \leq 4\mathbf{f}^{Q}_{\text{coll}}(q). \tag{7.84}$$

Conditioned on $\neg$Find, the outputs of the private interface are the same, then the games are identical-until-bad and we can use Lemma 6.10 to bound the advantage of the adversary.

As long as Find does not occur and the graph is not saturated the adversary cannot distinguish the simulator from a random function except for the distinguishing advantage that we calculated. Saturation certainly does not occur for $q \in O\left(\sqrt[4]{|\mathcal{C}|}\right)$ as the database in every branch of the superposition increases by at most one in every query. Collecting the differences between games yields $\varepsilon = 8\mathbf{f}^{Q}_{\text{coll}}(q) + \sqrt{(q+1)\mathbf{f}^{Q}_{\text{coll}}(q)}$. $\square$

CHAPTER

8 QUANTUM RESOURCE-RESTRICTED INDIFFERENTIABILITY

Chapter contents

In this chapter we present a result related to the notion of quantum resource-restricted indifferentiability. We introduce this notion and present a formal definition in Section 2.3.3.2.

The result we discuss is a quantum version of the challenge-response property of hash functions. This property is defined with a two-stage game and has been used in [RSS11] as an example for why regular indifferentiability is not always applicable to multi-stage games. We prove the same statement as the authors of [RSS11] but giving the adversary access to limited quantum memory. Taking into account the influence of quantum memory is the main challenge of analyzing the quantum version of the game. With this example we bring the discussion about the problem of regular indifferentiability with multi-stage games to the quantum world.

8.1 External Storage Game

The crucial observation made by Ristenpart, Shacham, and Shrimpton in [RSS11] is that in multi-stage games the only way for the indifferentiability simulator to succeed is to maintain state across adversaries in multiple stages. Simulators, however, are defined in a different way: They are algorithms that can "fool" a single adversary. If we talk about multiple adversaries, each interacts with a separate instance of the simulator. The indifferentiability simulators are not "aware" of those multiple instances.

A good example of this situation is presented in this section. We copy the example from [RSS11] but generalize it to the case of quantum adversaries. The example we analyze here shows that there is a multi-stage game that cannot be won by adversaries interacting with a random oracle but is trivial when they interact with the Comp compression function defined in Section 2.5.1. This shows that the composition theorem (Theorem 2.18) does not apply to multi-stage games in general.

Before we go any further, we introduce the hon (honest) and adv (adversarial) interfaces. In games that include adversarial procedures, the interface adv specifies the functionality the adversaries have access to. The game itself has access to the interface hon. A more detailed discussion of these interfaces is presented in [RSS11].

Note that the hon and adv interfaces do not always correspond to the private and public interfaces from the definition of indifferentiability (Definition 2.16). In the indifferentiability game the private interface corresponds to the interface hon. The public interface, however, is the adv interface only in the real world; In the ideal world it is the simulator who provides the public interface and she in turn has access to the adversarial interface of the ideal functionality.

Now we can present a quantum version of the challenge-response (CRP) hash-function property, the classical game can be found in Figure 3 in [RSS11].

In Algorithm 8.1 we consider a quantum adversary $(\mathsf{A}_1, \mathsf{A}_2)$ that does not share (between the stages) any state, except for the explicitly mentioned $|\mathsf{st}\rangle$. The difference between the quantum and classical games is that in the latter the state is a classical string of at most n bits and the adversaries make classical queries to their oracles. In Algorithm 8.1 both the adversary and the game have quantum access to an external oracle $\mathsf{H} = (\mathsf{H}^{\mathrm{adv}}, \mathsf{H}^{\mathrm{hon}})$, where the two elements of the tuple signify the two interfaces of the oracle.

Algorithm 8.1 Game for CRP[H, $(\mathsf{A}_1, \mathsf{A}_2)$]

```
1: $M \xleftarrow{\$} \{0,1\}^p$
2: $|\mathsf{st}\rangle \leftarrow \mathsf{A}_1^{\mathsf{H}^{\mathrm{adv}}}(M)$ such that $|\mathsf{st}\rangle \in (\mathbb{C}^2)^{\otimes n}$
3: $C \xleftarrow{\$} \{0,1\}^s$
4: $Z \leftarrow \mathsf{A}_2^{\mathsf{H}^{\mathrm{adv}}}(|\mathsf{st}\rangle, C)$
5: **return** $Z = \mathsf{H}^{\mathrm{hon}}(M\|C)$
```

8.1.1 The CRP Game with a Random Oracle

We want to prove that for a random oracle and quantum adversaries A_1 and A_2 the CRP game defined in Algorithm 8.1 cannot be won. The proof follows the same path as the proof of Theorem 4.1 in [RSS11]. However, we need a quantum counterpart of a lemma used in the classical proof. In Section 2.1.6 we present a discussion about quantum entropy and prove Lemma 2.4, which plays an important role in the proof here. Let us go over the setting one more time. The two adversaries have quantum access to the adversarial interface of H. If the hash function is a random oracle, i.e. $\mathsf{H} = \mathsf{R}$, the adversarial and honest interfaces are the same and are just R.

Theorem 8.1. *Let $(\mathsf{A}_1, \mathsf{A}_2)$ be quantum algorithms making at most q quantum queries to a random oracle $\mathsf{R} : \{0,1\}^* \to \{0,1\}^r$, then*

$$\mathbb{P}\left[1 \leftarrow \mathrm{CRP}[\mathsf{R}, (\mathsf{A}_1, \mathsf{A}_2)]\right] \leq \sqrt{4q(q+1)\left(\frac{1}{2^s} + \frac{1}{2^{p-n}}\right)} + 4q\left(\frac{1}{2^s} + \frac{1}{2^{p-n}}\right) + \frac{1}{2^r}. \tag{8.1}$$

Proof. Using the semi-classical O2H lemma, Theorem 2.23, we puncture A_1 and A_2 on the relation of querying $(M\|C)$, we denote this relation by R_{query}:

$$\begin{aligned} \mathbb{P}\left[1 \leftarrow \mathrm{CRP}[\mathsf{R}, (\mathsf{A}_1, \mathsf{A}_2)]\right] &\leq \sqrt{(q+1)\mathbb{P}\left[\mathrm{Find}\right]} \\ &+ \mathbb{P}\left[1 \leftarrow \mathrm{CRP}[\mathsf{R}, (\mathsf{A}_1 \setminus R_{\mathrm{query}}, \mathsf{A}_2 \setminus R_{\mathrm{query}})]\right] \\ &\leq \sqrt{(q+1)\mathbb{P}\left[\mathrm{Find}\right]} + \mathbb{P}\left[\mathrm{Find}\right] \end{aligned} \tag{8.2}$$

$$+\,\mathbb{P}\left[1 \leftarrow \mathrm{CRP}[\mathsf{R}, (\mathrm{A}_1 \setminus R_{\text{query}}, \mathrm{A}_2 \setminus R_{\text{query}})] \mid \neg\mathrm{Find}\right]. \tag{8.3}$$

In the first inequality above we introduce the punctured oracle and bound the distinguishing advantage using Theorem 2.23. The second inequality follows from writing a sum of conditional probabilities where we condition on Find and ¬Find and bounding

$$\mathbb{P}\left[\neg\mathrm{Find}\right] \leq 1 \text{ and} \tag{8.4}$$

$$\mathbb{P}\left[1 \leftarrow \mathrm{CRP}[\mathsf{R}, (\mathrm{A}_1 \setminus R_{\text{query}}, \mathrm{A}_2 \setminus R_{\text{query}})] \mid \mathrm{Find}\right] \leq 1. \tag{8.5}$$

Let us first bound the last element in the above inequality. Conditioned on ¬Find the adversary A_2 has never queried $(M\|C)$ to the oracle, hence she has to guess the output of R, therefore the bound on the probability of CRP is $1/2^r$.

Now to bound the probability of Find we use the union bound to distinguish between Find happening in A_1 and in A_2. In the former case we use Corollary 1 from [AHU19] (Lemma 2.24):

$$\mathbb{P}\left[\mathrm{Find}\right] \leq 4q \cdot P_{\max}, \tag{8.6}$$

where $P_{\max} := \max_x \mathbb{P}\left[x \in R_{\text{query}}\right]$. The probability $P_{\max}$ for A_1 is 2^{-s}: this is the probability with which C is sampled.

To bound the probability that A_2 queries $M\|C$ on input C and with the quantum state passed from A_1 we make use of Lemma 2.25. Following the interpretation of min-entropy from [KRS09] we see that $2^{-\mathbf{H}_{\min}(M|S)}$—where M is the random variable corresponding to the string M and S is the quantum register holding $|\mathrm{st}\rangle$—on the c-q state constructed from $|\mathrm{st}\rangle$ is the guessing probability of M. This is the guessing probability that enters the bound in Lemma 2.25. To bound the entropy of the random variable M given $|\mathrm{st}\rangle$, we use Lemma 2.4. The bound on the entropy is $\mathbf{H}_{\min}(M \mid S) \geq p - n$. Finally the bound on Find is

$$\mathbb{P}\left[\mathrm{Find}\right] \leq 4q\left(\frac{1}{2^s} + \frac{1}{2^{p-n}}\right). \tag{8.7}$$

Taking all of the above into account we get the claimed bound. □

8.1.2 The CRP Game with the Comp Construction

Let us now analyze the CRP game where H is instantiated with Comp with the following parameters: the first function is $\mathbf{h}_1 : \{0,1\}^p \to \{0,1\}^n$ and the second is $\mathbf{h}_2 : \{0,1\}^{n+s} \to \{0,1\}^r$. The CRP game can be won by $(\mathrm{A}_1, \mathrm{A}_2)$ that queries M to $\mathbf{h}_1(M) = \mathrm{st}$ and sends the output to A_2. The second adversary easily computes $Z = \mathbf{h}_2(\mathrm{st}, C)$. The probability of winning the game for such adversaries is 1. The crucial point in our discussion is that the composition construction is quantumly indifferentiable from a random oracle, as proven in Theorem 7.2. This fact validates the counterexample from [RSS11] in the quantum world.

8.1.3 Conclusions

As hinted in [RSS11], constructions that follow Definition 2.19 of resource-restricted indifferentiability might compose in multi-stage games. The reason for that is that in a multi-stage indifferentiability security game (that remains to be formally defined), if different instantiations of the simulator could share a state of limited size (similarly to the adversaries in the CRP game) it would be enough for them to provide consistent answers to the adversaries. If there is a construction that is "multi-stage indifferentiable" from a random oracle, then it would also behave as a random oracle in the CRP game. This discussion is just speculations for now, but it will hopefully generate problems for future work. It also remains to be seen if there are constructions that are resource-restricted indifferentiable from a random oracle.

CHAPTER

9

OPEN PROBLEMS

Chapter contents

In this last chapter we present two questions. These are problems that seem to be extremely important in the line of research presented in this thesis. They also seem very difficult and the first problem we discuss lacks an answer.

The first problem that we present is: "Is there a general relation between the statistical distance and the distinguishing advantage of a q-(quantum)-query algorithm A?". This question is motivated by the classical proof techniques that give a positive (and constructive) answer. In the quantum world it is much harder to answer this questions but if we found an answer, we could solve a number of problems in post-quantum security.

The second problem we discuss is quantum lazy-sampling of random permutations. This problem is less general, as it is directly related to the compressed-oracle technique. Nonetheless, seeing how powerful this technique is, lazy-sampling random permutations would open the doors to achieving many important results. To mention one, the SHA3 hash function is defined with public invertible permutations, so a quantum indifferentiability proof (at least one using the quantum game-playing framework) would require lazy-sampling a random invertible permutation. In Section 9.2.2.1 we provide some ideas for a compressed permutation oracle with errors. In the last section of this chapter we analyze the possibility of an error-less way of quantum lazy-sampling random permutations.

9.1 Quantum Indistinguishability of Distributions over Functions

Let us consider two distributions $\mathfrak{D}_1$ and $\mathfrak{D}_2$ over the set of functions $\mathcal{F} := \{\mathbf{f} : \mathcal{X} \to \mathcal{Y}\}$. We discuss algorithms A with oracle access to a function sampled from $\mathcal{F}$ according to $\mathfrak{D}_1$ or $\mathfrak{D}_2$. More concretely we are interested in the problem of A distinguishing between $\mathfrak{D}_1$ and $\mathfrak{D}_2$. The resources the algorithm (also called the adversary or the distinguisher) is given is unlimited computational power and access to q queries to $\mathbf{f}$.

First we describe the setting without specifying the nature (classical or quantum) of the adversary. We define the *distinguishing advantage* for adversaries in a class $\mathcal{A}$ by:

$$\mathbf{Adv}_q^{\mathcal{A}}(\mathfrak{D}_1, \mathfrak{D}_2) := \sup_{\mathrm{A}\in\mathcal{A}} \left| \underset{\mathbf{f}\leftarrow\mathfrak{D}_1}{\mathbb{P}} \left[1 \leftarrow \mathrm{A}^{\mathbf{f}}\right] - \underset{\mathbf{f}\leftarrow\mathfrak{D}_2}{\mathbb{P}} \left[1 \leftarrow \mathrm{A}^{\mathbf{f}}\right] \right|. \tag{9.1}$$

In general bounding the distinguishing advantage is a hard task. There are two important approaches to proving indistinguishability. One is based on stating that the adversary behaves identically in the two worlds (interacting with $\mathfrak{D}_1$ or $\mathfrak{D}_2$) as long as a certain condition is fulfilled. Another finds a connection to the statistical distance between the two distributions and bounds the distance, abstracting from the algorithmic part of distinguishability. Classical examples of the former approaches are Maurer's random systems [Mau02] and Bellare and Rogaway's game-playing framework [BR06]. In the quantum case we have the quantum game-playing framework [AHU19; Zha19a; Cza+19]. The latter approaches are the ones we want to discuss here in more detail.

Before we cover the classical and quantum techniques related to the statistical distance between distributions we define this distance. The assumption on the adversaries that we made is that A makes q (quantum or classical) queries to $\mathbf{f}$ distributed according to $\mathfrak{D}_1$ or $\mathfrak{D}_2$. This implies that she has access to *transcripts* of input-output pairs, where the inputs are the queries made by A and outputs are generated by $\mathbf{f}$. The first parameter of these transcripts that depends on the nature of the queries made by A is their size. In the classical case it is certainly q, but in the quantum case it is less obvious[1]. For that reason let us say that A has access to Q-sized transcripts. As we do not know a priori what queries A makes, let us for now consider the queries $\vec{x} := (x_1, x_2, \ldots, x_Q)$. For these queries a transcript is an array $((x_1, \mathbf{f}(x_1)), (x_2, \mathbf{f}(x_2)), \ldots, (x_Q, \mathbf{f}(x_Q)))$. We denote the distribution of such transcripts by $\mathfrak{D}(\vec{x})$, where $\mathfrak{D}$ is $\mathfrak{D}_1$ or $\mathfrak{D}_2$. The definition of *statistical distance* between distributions over transcripts is

$$\mathbf{d}(\mathfrak{D}_1(\vec{x}), \mathfrak{D}_2(\vec{x})) := \frac{1}{2} \sum_{\vec{y} \in \mathcal{Y}^Q} \left| \underset{\mathbf{f} \leftarrow \mathfrak{D}_1}{\mathbb{P}} \left[\mathbf{f}(\vec{x}) = \vec{y} \right] - \underset{\mathbf{f} \leftarrow \mathfrak{D}_2}{\mathbb{P}} \left[\mathbf{f}(\vec{x}) = \vec{y} \right] \right|. \tag{9.2}$$

Now what we look for is a relation between Equation (9.1) and Equation (9.2). Let us first go over the *classical* case. First of all A makes q classical queries and the transcripts are of size q. For such adversaries the relation that we have is a straight forward inequality:

$$\mathbf{Adv}_q^{\text{class}}(\mathfrak{D}_1, \mathfrak{D}_2) \leq \max_{\vec{x}} \mathbf{d}(\mathfrak{D}_1(\vec{x}), \mathfrak{D}_2(\vec{x})). \tag{9.3}$$

This inequality is analyzed in detail for example in [Nan06]. Equation (9.3) can be proven using Lemma 4 in [Nan06], which states that for a fixed A the distinguishing advantage is upper bounded by the statistical distance of transcripts. Maximizing over A gives the above inequality. Interestingly the inequality in Equation (9.3) is tight. Equality can be achieved by picking an adversary that deterministically chooses the queries (and internal randomness) to maximize her advantage. Moreover A outputs 1 if for the outputs $\vec{y}$ that she gets $\underset{\mathbf{f} \leftarrow \mathfrak{D}_1}{\mathbb{P}} \left[\mathbf{f}(\vec{x}) = \vec{y} \right] > \underset{\mathbf{f} \leftarrow \mathfrak{D}_2}{\mathbb{P}} \left[\mathbf{f}(\vec{x}) = \vec{y} \right]$. This adversary has the distinguishing advantage equal to $\mathbf{d}(\mathfrak{D}_1(\vec{x}), \mathfrak{D}_2(\vec{x}))$, which follows from an equivalent definition of statistical distance $\mathbf{d}(\mathfrak{D}_1(\vec{x}), \mathfrak{D}_2(\vec{x})) = \underset{\mathbf{f} \leftarrow \mathfrak{D}_1}{\mathbb{P}} \left[\mathbf{f}(\vec{x}) \in \mathcal{T}_0 \right] - \underset{\mathbf{f} \leftarrow \mathfrak{D}_2}{\mathbb{P}} \left[\mathbf{f}(\vec{x}) \in \mathcal{T}_0 \right]$, where $\mathcal{T}_0 := \left\{ \vec{y} \in \mathcal{Y}^q : \underset{\mathbf{f} \leftarrow \mathfrak{D}_1}{\mathbb{P}} \left[\mathbf{f}(\vec{x}) = \vec{y} \right] > \underset{\mathbf{f} \leftarrow \mathfrak{D}_2}{\mathbb{P}} \left[\mathbf{f}(\vec{x}) = \vec{y} \right] \right\}$. More details on this fact can be found in [Nan06].

[1]Note that, as discussed in Section 2.4.1, the quantum state of the adversary can depend on up to $2q$ input-output pairs.

Among the most important proof techniques that make use of Equation (9.3) are the H-coefficient technique [Pat08; JN18] and the chi-squared method [DHT17; BN18a]. These techniques are used to bound the statistical distance between transcripts holding the history of the adversary's queries.

Unfortunately most of the classical discussion cannot be repeated in the quantum world. For example, the queries made by quantum adversaries result in a quantum state that can hold a superposition over all possible transcripts. Although A cannot effectively use all of them, it is unclear what relation connects the distinguishing advantage and the statistical distance.

An example of a proof technique useful in the quantum world is Theorem 1.1 from Zhandry's [Zha12]. There he proves that for functions f $\leftarrow \mathfrak{D}^{\mathcal{X}}$ (meaning every input from $\mathcal{X}$ is mapped to a y distributed independently on $\mathcal{Y}$ according to $\mathfrak{D}$), we can bound the distinguishing advantage by $O\left(\sqrt{q^3 \mathbf{d}(\mathfrak{D}_1, \mathfrak{D}_2)}\right)$. Zhandry's technique does not work for general distributions over functions though.

To sum up, the important observation that we want to state is that by Equation (9.3) we move from distinguishing distributions on the distinguishers' outputs to distinguishing random variables of samples of distributions. Distinguishing random variables and distributions is a more fundamental problem, which often can be solved in more ways. Let us denote the class of quantum adversaries making q quantum queries by quant. Finally, the question that we pose is:

Is there a version of Equation (9.3) that applies to $\mathbf{Adv}_q^{\mathrm{quant}}(\mathfrak{D}_1, \mathfrak{D}_2)$?

9.2 Quantum Lazy Sampling Random Permutations

The problem we describe here is quantum lazy-sampling of random permutations and providing superposition access to both directions of the function. In this section we describe the approach introducing additional errors in short and discuss another approach. The second way of defining compressed random permutations does not introduce errors but we have only preliminary results on how to actually implement it.

9.2.1 Sampling Procedure for Random Permutations

Let us first cover the sampling procedure for random permutations. The uniform distribution over permutations is denoted by $\mathfrak{P}$ (read as just "P"). A sampling procedure prepares a state in the appropriate superposition:

$$\mathsf{Samp}_{\mathfrak{P}}(\mathcal{S})|0^{|\mathcal{S}|}\rangle = \sum_{\vec{y}\text{: no collisions}} \frac{1}{\sqrt{N(N-1)\cdots(N-|\mathcal{S}|+1)}}|\vec{y}\rangle, \tag{9.4}$$

where by "no collisions" we mean that the outputs do not collide. In this paragraph we provide a definition of an efficient sampling procedure for random permutations.

To accommodate for random permutations in the compressed-oracle approach we need a quantum operation that prepares an equal superposition over a given set $\mathcal{Y} \setminus \mathcal{S}$. The set $\mathcal{S}$ is stored in a quantum register. Say that the codomain of f is $\mathcal{Y} = [N]$ and the set of outputs with probability zero of appearing is $\mathcal{S}$. We propose that sampling is done first by applying the Quantum Fourier Transform over the set $[L]$ to the state $|0\rangle$, where $L = |\mathcal{Y} \setminus \mathcal{S}|$. After that we spread the values in $[L]$ in a way that values from $\mathcal{S}$ no longer appear in the set. The last operation may be realized by repeating the following unitary

$$\mathsf{V}^{AB}|a\rangle_A|b\rangle_B = \begin{cases} |a\rangle_A|b\rangle_B & \text{if } a > b \\ |a\rangle_A|b+1 \mod N\rangle_B & \text{if } a \le b \end{cases}, \tag{9.5}$$

where by V^{AB} we denote applying V to registers A, B. The operation V as stated above is defined only on $a \in [N]$ and $b \in [N-1]$ (counted from 0), to extend it to the full domain we need to add that $\mathsf{V}|a\rangle_A|N-1\rangle_B := |a\rangle_A|N-1\rangle_B$. The action of V is shifting the states in register B in a way that possible states skip a. Let us show that V is in fact a unitary that is easy to construct. For the construction we need three unitary sub-routines,

$$\mathsf{V}_{\le}|a\rangle_A|b\rangle_B = \begin{cases} |a\rangle_A|b\rangle_B|0\rangle_C & \text{if } a > b \\ |a\rangle_A|b\rangle_B|1\rangle_C & \text{if } a \le b \end{cases}, \tag{9.6}$$

$$\mathsf{V}_{+}|a\rangle_A|b\rangle_B|c\rangle_C = |a\rangle_A|b+c \mod N\rangle_B|c\rangle_C, \tag{9.7}$$

$$\mathsf{V}_{-\le}|a\rangle_A|b\rangle_B|c\rangle_C = \begin{cases} |a\rangle_A|b\rangle_B|c\oplus 0\rangle_C & \text{if } a > b-1 \mod N \\ |a\rangle_A|b\rangle_B|c\oplus 1\rangle_C & \text{if } a \le b-1 \mod N \end{cases}, \tag{9.8}$$

where we additionally need to specify that $\mathsf{V}_{-\le}|a\rangle_A|0\rangle_B|c\rangle = |a\rangle_A|0\rangle_B|c\rangle$ for $a > 0$. Now we define $\mathsf{V} = \mathsf{V}_{-\le}\mathsf{V}_{+}\mathsf{V}_{\le}$, we also discard register C after the three operations. In this approach we need a register holding a description of the set $\mathcal{S}$ but as long as we do, applying V to all registers describing $\mathcal{S}$ will give us the expected result.

We use the unitary V to show how the initial state preparation looks like for a random permutation. For V to be used correctly it needs to be conditioned on a sorted set of values. The full "spreading" unitary is defined as:

$$\mathsf{Spread}^{AB} := \langle 0^{|A|}|_T \mathsf{U}_{\text{sort}}^{AT\dagger} \prod_{i=1}^{|T|} \mathsf{V}^{T_i B} \mathsf{U}_{\text{sort}}^{AT} |0^{|A|}\rangle_T, \tag{9.9}$$

where $\mathsf{U}_{\text{sort}}^{AT}$ writes in register T lexicographically sorted contents of register A.

The definitions of the sampling operation for a uniform distribution on $\mathcal{Y}\backslash\mathcal{S}$, for any $\mathcal{S} \subseteq \mathcal{X}$, is given by:

$$\mathsf{Samp}_{\mathcal{Y}\backslash\mathcal{S}}(x) := \mathsf{Spread}^{SD^Y(x)}\mathsf{QFT}_L^{D^Y(x)\dagger}, \tag{9.10}$$

where S is the quantum register holding elements of $\mathcal{S}$. Note that we apply $\mathsf{QFT}_L^{\dagger}$ to a register of dimension N, we overload the notation but mean $\mathsf{QFT}_L^{\dagger} \oplus \mathbb{1}_{N-L}$. The sampling procedure from Equation (9.10) is used to implement $\mathsf{CStO}_{\mathcal{Y}\backslash\mathcal{S}}$.

The uniform distribution over the set of permutations is denoted by $\mathfrak{P}$. The sampling function is defined using Equation (9.10) with $\mathcal{S}$ defined as previous outputs, held in registers D_i^Y. The definition reads as follows:

$$\mathsf{Samp}_{\mathfrak{P}}(\mathcal{S}) := \prod_{i=1}^{|\mathcal{S}|} \mathsf{Samp}_{\mathcal{Y}\backslash D^Y(\{x_1,\cdots,x_i\})} = \prod_{i=1}^{|\mathcal{S}|} \mathsf{Spread}^{D_{<i}^Y D_i^Y} \mathsf{QFT}_{N-(i-1)}^{D_i^Y\dagger}, \tag{9.11}$$

where $D^Y(\{x_1, \cdots, x_i\})$ denotes the set of outputs y in entries of D corresponding to listed x and $D_{<i}^Y$ is the register consisting of D_j^Y for $1 \le j < i$.

9.2.1.1 Random Permutations in Sponges

In the context of indifferentiability of the sponge construction, we want to lazy sample permutations $\mathbf{f} : \mathcal{A} \times \mathcal{C} \to \mathcal{A} \times \mathcal{C}$ in two stages. First sampling the inner part of the output and then the outer part.

By $\mathcal{D}$ we denote the set of outputs of previous queries. In the language of the sponge graph $\mathcal{D}^{-1}$ is the set of nodes with outgoing edges and $\mathcal{D}$ is the set of nodes with incoming edges. By $\widehat{\mathcal{D}}$ we denote the set of supernodes with all nodes having an incoming edge. By $\overline{\mathcal{D}}(\hat{t})$ we denote the set of nodes in the supernode $\hat{t}$ with an incoming edge.

We need to define a procedure to lazy-sample outputs of a random permutation. The obvious solution of sampling uniformly from $\mathcal{A} \times \mathcal{C} \setminus \mathcal{D}$ is not good enough as we want to sample the inner part before the outer part and retain the step-by-step structure of our proof, similarly to the proof of Theorem 7.9.

Classically we are going to first sample uniformly from $\mathcal{A} \times \mathcal{C} \setminus \mathcal{D}$ but then discard the outer state. The value of the inner state $\hat{t}$ is then effectively sampled from $\mathcal{C}$ with weights $\frac{|\mathcal{A}\backslash\overline{\mathcal{D}}(\hat{t})|}{|\mathcal{A}\times\mathcal{C}\backslash\mathcal{D}|}$. We call this distribution $\mathfrak{C}$. At this point we will be introducing bad events concerning the inner part of the sampled state. To sample the outer state we just sample uniformly from $\mathcal{A} \setminus \overline{\mathcal{D}}(\hat{t})$. We denote this distribution by $\mathfrak{A}(\hat{t})$.

Quantumly the situation is a bit more involved, so we are going to present the sampling procedure in more detail. First we sample pairs from $\mathcal{A} \times \mathcal{C} \setminus \mathcal{D}$ using $\mathsf{Samp}_{\mathcal{A}\times\mathcal{C}\backslash\mathcal{D}}$, defined similarly to the procedure in Equation (9.10). Then we un-sample the outer part, by applying the Hermitian conjugate of $\mathsf{Samp}_{\mathcal{A}\setminus\overline{\mathcal{D}}(\hat{t})}$ to the resulting state. Note that we control the un-sample operation on the inner-part of the initial sample. We start from the following state

$$\begin{aligned}\mathsf{Samp}_{\mathcal{A}\times\mathcal{C}\setminus\mathcal{D}}|0\rangle &= \sum_{t\in\mathcal{A}\times\mathcal{C}\setminus\mathcal{D}} \frac{1}{\sqrt{|\mathcal{A}\times\mathcal{C}\setminus\mathcal{D}|}}|t\rangle \\ &= \sum_{\hat{t}} \sqrt{\frac{\left|\mathcal{A}\setminus\overline{\mathcal{D}}(\hat{t})\right|}{|\mathcal{A}\times\mathcal{C}\setminus\mathcal{D}|}} \sum_{\bar{t}\in\mathcal{A}\setminus\overline{\mathcal{D}}(\hat{t})} \frac{1}{\sqrt{\left|\mathcal{A}\setminus\overline{\mathcal{D}}(\hat{t})\right|}}|\bar{t},\hat{t}\rangle, \end{aligned} \tag{9.12}$$

where the right hand side of the equation follows by just rearranging the sums and noticing that given some $\hat{t}\in\mathcal{C}$ we have $\bar{t}\in\mathcal{A}$ that have no incoming edges in the supernode $\hat{t}$. Now the definition of the second sampling procedure reads

$$\mathsf{Samp}_{\mathcal{A}\setminus\overline{\mathcal{D}}(\hat{t})}|0,\hat{t}\rangle = \sum_{\bar{t}\in\mathcal{A}\setminus\overline{\mathcal{D}}(\hat{t})} \frac{1}{\sqrt{\left|\mathcal{A}\setminus\overline{\mathcal{D}}(\hat{t})\right|}}|\bar{t},\hat{t}\rangle \tag{9.13}$$

and is completed to a unitary acting on every other $\bar{s}\in\mathcal{A}$ under the constraints of Definition 6.2. Note that we use the second register to control the unitary (by providing the set $\overline{D}(\hat{t})$ we exclude from $\mathcal{A}$). By applying $\mathsf{Samp}^{\dagger}_{\mathcal{A}\setminus\bar{\mathcal{D}}(\hat{t})}$ to both sides of Equation (9.13) we see that we can un-compute the outer part $\bar{t}$ from the initial superposition from Equation (9.12) and sample $\hat{t}$ with probability $\frac{|\mathcal{A}\setminus\overline{\mathcal{D}}(\hat{t})|}{|\mathcal{A}\times\mathcal{C}\setminus\mathcal{D}|}$. The sampling procedure $\mathsf{Samp}^{\dagger}_{\mathcal{A}\setminus\overline{\mathcal{D}}(\hat{t})}\circ\mathsf{Samp}_{\mathcal{A}\times\mathcal{C}\setminus\mathcal{D}}$ is used to define $\mathsf{CStO}_{\mathfrak{C}}$. The outer part is then sampled using $\mathsf{Samp}_{\mathcal{A}\setminus\overline{\mathcal{D}}(\hat{t})}$, with which we define $\mathsf{CStO}_{\mathfrak{A}(\hat{t})}$.

In the case of the inverse of the random permutation we can use a similar distribution but in the above definitions we take $\mathcal{D}^{-1}$—i.e. the set of nodes with outgoing edges—in both $\mathfrak{C}$ and $\mathfrak{A}$.

9.2.2 The Compressed Update Procedure

The sampling procedure is not enough to quantumly lazy sample a function. The main problem is that Algorithm 6.1 does not apply here, note that the distribution $\mathfrak{P}$ is not independent. In the following sections we present two approaches to defining the compressed permutation oracle. One that introduces additional errors and one that is a unitary implementation.

9.2.2.1 Compressed Permutation Oracle with Errors

Our idea for the compressed oracle for permutations with errors is to use a compressed oracle for uniformly random functions punctured on collisions.

For queries to the backward direction we invert the database (possible because there are no collisions) and apply the regular update procedure $\mathsf{CStO}_{\mathfrak{U}}$.

In more detail, we provide access to the inverse of the permutation by flipping the database: treat the content of D^Y as inputs and D^X as outputs and sort by D^Y, we call the appropriate unitary Flip. Note that Flip^D works as anticipated only if both in D^Y and D^X there are no collisions; The definition of Flip can be easily extended to a full unitary by, e.g., keeping the order unchanged in a tuple with colliding outputs. We thereby obtain a (superposition) database where the strings $y_i \| x_i$ are lexicographically ordered. A more formal definition of the compressed permutation oracle $\mathsf{CPerO}_{\mathcal{X}}$ is:

$$\mathsf{CPerO}_{\mathcal{X}} := \mathsf{CPhO}_{\mathcal{X}} \setminus R_{\text{coll}}, \tag{9.14}$$

$$\mathsf{CPerO}_{\mathcal{X}}^{-1} := \mathsf{Flip}^D \circ (\mathsf{CPhO}_{\mathcal{X}} \setminus R_{\text{coll}}) \circ \mathsf{Flip}^D. \tag{9.15}$$

A more detailed description of Flip:

1. Controlled on D^Y copy D^X to D'^Y and arrange in lexicographically increasing order according to values of y_i in D^Y.

2. Controlled on D^X and D'^Y copy D^Y to D'^X in the new order.

3. Controlled on D' erase the old D: Take the smallest x_i in D'^Y and subtract it from the first register of D^X, also subtract the corresponding y_i, and so on.

Now the statement without which the above definition is useless is "The full (two-way) permutation oracle is indistinguishable from the compressed permutation oracle".

9.2.2.2 A Unitary Compressed Permutation Oracle

A unitary compressed oracle for random permutations is an approach requiring defining a new version of Algorithm 6.1 suitable to handle sampling procedures that are not independent. Our approach to finding such an algorithm is basically performing an educated guess of the right unitary. The greatest weakness of this approach is of course the hardness of just guessing the solution. Nonetheless we have some preliminary results that might prove useful when combined with other approaches.

One crucial observation concerning a unitary compressed oracle is that a single query conveys a different amount of information depending on the number of previous queries. Note that the first query not only informs us that $\mathbf{f}(x) = y$ but also that the output on all other $x' \neq x$ in not y. The more queries we make, the more "negative" information we have.

The importance of the above observation is confirmed with the indistinguishability bound for a single-query distinguisher. The set of functions we sample from is $\{\mathbf{f} : [N] \to [N]\}$.

Theorem 9.1. *Given quantum access to a function sampled either according to the uniform distribution over functions $\mathfrak{U}$ or the uniform distribution over permutations $\mathfrak{P}$, we have that for all quantum distinguishers making a single quantum query to the oracle, the distinguishability advantage is at most $\frac{1}{N-1}$:*

$$\left|\underset{\mathbf{f}\leftarrow\mathfrak{U}}{\mathbb{P}}\left[1 \leftarrow \mathsf{A}^{|\mathbf{f}\rangle}\right] - \underset{\mathbf{f}\leftarrow\mathfrak{P}}{\mathbb{P}}\left[1 \leftarrow \mathsf{A}^{|\mathbf{f}\rangle}\right]\right| \leq \frac{1}{N-1}. \tag{9.16}$$

Moreover there is a distinguisher B *that has advantage $\frac{1}{2(N-1)}$.*

Proof. Let us inspect the indistinguishability bound for the uniform distribution over functions $\mathfrak{U}$ and the uniform distribution over permutations $\mathfrak{P}$:

$$\left|\underset{\mathbf{f}\leftarrow\mathfrak{U}}{\mathbb{P}}\left[1 \leftarrow \mathsf{A}^{|\mathbf{f}\rangle}\right] - \underset{\mathbf{f}\leftarrow\mathfrak{P}}{\mathbb{P}}\left[1 \leftarrow \mathsf{A}^{|\mathbf{f}\rangle}\right]\right| = \left|\mathrm{Tr}\left(\mathsf{Q}_1\rho^A_{\mathfrak{U}}\right) - \mathrm{Tr}\left(\mathsf{Q}_1\rho^A_{\mathfrak{P}}\right)\right| \tag{9.17}$$

$$\leq \frac{1}{2}\left\|\rho^A_{\mathfrak{U}} - \rho^A_{\mathfrak{P}}\right\|_1 \tag{9.18}$$

where Q_1 is the measurement corresponding to A outputting 1. The inequality above comes from the fact that trace distance gives the optimal measurement for distinguishing two states [NC10]. We also want to stress that register A holds all of the adversary's state, this includes her work register, query register, and any other auxiliary registers she decides to use.

We model the oracle access with the full phase oracle. The adversary's initial state is:

$$|\psi_0\rangle_A := \sum_x \sum_{\eta_x,w} \alpha_{x,\eta_x,w}|x,\eta_x,w\rangle_{A^{XYW}}, \tag{9.19}$$

the states $\rho_{\mathfrak{U}}$ and $\rho_{\mathfrak{P}}$ are defined as the partial trace of $|\psi_0\rangle\langle\psi_0|$ after a single query to $\mathsf{PhO}_{\mathfrak{U}}$ or $\mathsf{PhO}_{\mathfrak{P}}$, respectively. The full definitions of the reduced states are:

$$\rho_{\mathfrak{U}} = \sum_{x,x'} \sum_{\eta_x,\eta'_{x'},w,w'} \alpha\bar{\alpha}' \sum_{y,y_{x'}} \frac{1}{N^2}\omega_N^{\eta_x y_x}\bar{\omega}_N^{\eta'_{x'}y_{x'}}|x,\eta_x,w\rangle\langle x',\eta'_{x'},w'| \tag{9.20}$$

$$\rho_{\mathfrak{P}} = \sum_{x,x'} \sum_{\eta_x,\eta'_{x'},w,w'} \alpha\bar{\alpha}' \sum_{y,y_{x'}\neq y_x} \frac{1}{N(N-1)}\omega_N^{\eta_x y_x}\bar{\omega}_N^{\eta'_{x'}y_{x'}}|x,\eta_x,w\rangle\langle x',\eta'_{x'},w'|, \tag{9.21}$$

where α and $\bar{\alpha}'$ denote $\alpha_{x,\eta_x,w}$ and $\bar{\alpha}_{x',\eta'_{x'},w'}$ respectively.

First of all we see that terms with $x' = x$ are equal, in this case in both states the sum over y_x and $y_{x'}$ simplify to just a single sum over $y_x \in [N]$. The off-diagonal (so for $x' \neq x$) terms are a bit more complicated to calculate. Let us notice that $\sum_{y,y_{x'}\neq y_x} = \sum_{y,y_{x'}} - \sum_{y,y_{x'}=y_x}$, this splits the difference $\rho_{\mathfrak{U}} - \rho_{\mathfrak{P}}$ into two parts:

$$\rho_{\mathfrak{U}} - \rho_{\mathfrak{P}} = \left(\frac{1}{N^2} - \frac{1}{N(N-1)}\right)$$

$$\sum_{x,x'} \sum_{\eta_x,\eta'_{x'},w,w'} \alpha_{x,\eta_x,w}\bar{\alpha}_{x',\eta'_{x'},w'} \sum_{y,y_{x'}} \omega_N^{\eta_x y_x} \bar{\omega}_N^{\eta'_{x'} y_{x'}} |x,\eta_x,w\rangle\langle x',\eta'_{x'},w'|$$
$$+\sum_{x,x'} \sum_{\eta_x,\eta'_{x'},w,w'} \alpha_{x,\eta_x,w}\bar{\alpha}_{x',\eta'_{x'},w'} \underbrace{\sum_{y,y_{x'}=y_x} \frac{1}{N(N-1)} \omega_N^{\eta_x y_x} \bar{\omega}_N^{\eta'_{x'} y_{x'}}}_{=\frac{1}{N-1}\delta_{\eta_x,\eta'_{x'}}} |x,\eta_x,w\rangle\langle x',\eta'_{x'},w'| \tag{9.22}$$

$$= -\frac{1}{N-1}\rho_{\mathfrak{U}} + \frac{1}{N-1}\mathsf{D}_{A^Y}(|\psi_0\rangle\langle\psi_0|), \tag{9.23}$$

where $\mathsf{D}_A(\rho) := \sum_a |a\rangle_A\langle a|\rho|a\rangle_A\langle a|$ denotes the map that outputs the diagonal entries in register A.

Finally we have

$$\left|\underset{\mathsf{f}\leftarrow\mathfrak{U}}{\mathbb{P}}\left[1\leftarrow\mathsf{A}^{|\mathsf{f}\rangle}\right] - \underset{\mathsf{f}\leftarrow\mathfrak{P}}{\mathbb{P}}\left[1\leftarrow\mathsf{A}^{|\mathsf{f}\rangle}\right]\right| \leq \frac{1}{2}\left\|\rho_{\mathfrak{U}}^A - \rho_{\mathfrak{P}}^A\right\|_1 \tag{9.24}$$
$$= \frac{1}{2}\left\|-\frac{1}{N-1}\rho_{\mathfrak{U}} + \frac{1}{N-1}\mathsf{D}_{A^Y}(|\psi_0\rangle\langle\psi_0|)\right\|_1 \tag{9.25}$$
$$\leq \frac{1}{2}\frac{1}{N-1}\|\rho_{\mathfrak{U}}\|_1 + \frac{1}{2}\frac{1}{N-1}\||\psi_0\rangle\langle\psi_0|\|_1 = \frac{1}{N-1}, \tag{9.26}$$

where we use the triangle inequality and the fact that any CPTP (Completely Positive Trace-Preserving) map, like D, can only decrease the norm of a state.

A distinguisher that performs the optimal measurement has advantage $\|\rho_{\mathfrak{U}} - \rho_{\mathfrak{P}}\|$, as stated in Equation (9.18). Now let us consider a simple algorithm B that does not have a work register and prepares the following state:

$$|\psi_0^{\mathsf{B}}\rangle_A := \sum_{x\in\{x_1,x_2\}} \frac{1}{\sqrt{2}}|x,\eta\rangle_{A^{XY}}, \tag{9.27}$$

where $\eta \neq 0$ and x_1 and x_2 are any distinct inputs. We know that in the state after interaction with a random function we have two cases: in the branch of superposition where $x = x'$ the sum $\sum_{y,y_{x'}} \frac{1}{N^2}\omega_N^{\eta_x y_x}\bar{\omega}_N^{\eta'_{x'} y_{x'}} = \delta_{\eta_x,\eta'_{x'}}$, in the branch of superposition where $x \neq x'$ the sum $\sum_{y,y_{x'}} \frac{1}{N^2}\omega_N^{\eta_x y_x}\bar{\omega}_N^{\eta'_{x'} y_{x'}} = \delta_{\eta_x,0}\delta_{\eta'_{x'},0}$. We can observe that the former case equals $\mathsf{D}_{A^X}(|\psi_0\rangle\langle\psi_0|)$ (so the initial state with $x = x'$). As we have set $\eta \neq 0$, the off-diagonal part of $\rho_{\mathfrak{U}}$ (so the case with $x \neq x'$) is 0.

The above discussion leads to the following equality:

$$\left|\underset{\mathsf{f}\leftarrow\mathfrak{U}}{\mathbb{P}}\left[1\leftarrow\mathsf{B}^{|\mathsf{f}\rangle}\right] - \underset{\mathsf{f}\leftarrow\mathfrak{P}}{\mathbb{P}}\left[1\leftarrow\mathsf{B}^{|\mathsf{f}\rangle}\right]\right| = \frac{1}{2}\left\|\rho_{\mathfrak{U}}^A - \rho_{\mathfrak{P}}^A\right\|_1 \tag{9.28}$$
$$= \frac{1}{2}\frac{1}{N-1}\left\|\sum_{x\in\{x_1,x_2\}} \sum_{x'\in\{x_1,x_2\},x'\neq x} \frac{1}{2}|x,\eta\rangle\langle x',\eta|\right\|_1 = \frac{1}{2(N-1)}, \tag{9.29}$$

where the last equality comes from the definition of the trace norm: $\|\mathsf{A}\|_1 = \sum_\lambda |\lambda|$, where λ are the eigenvalues of the matrix A. □

Our general approach to finding the unitary compressed oracle is to calculate the inner product of the full oracle state for different queries and find a compressed state that recovers this behavior. First of all we assume the existence of Dec—a decompression procedure that works perfectly for random permutations. The adversary's state is assumed to be a basis state; To compare two different queries we use $\mathsf{U}_{\vec{x},\vec{\eta}}|0\rangle_A = |\vec{x},\vec{\eta}\rangle$. Finally the goal is to guess Dec out of

$$\begin{aligned}&{}_{AF}\langle 0,\Psi_0|\mathsf{U}^\dagger_{\vec{x}',\vec{\eta}'}\mathsf{PhO}^{s\dagger}\mathsf{U}_{\vec{x}',\vec{\eta}'}\mathsf{U}^\dagger_{\vec{x},\vec{\eta}}\mathsf{PhO}^{s}\mathsf{U}_{\vec{x},\vec{\eta}}|0,\Psi_0\rangle_{AF} = \\ &{}_{AD}\langle 0,0^{2q}|\mathsf{U}^\dagger_{\vec{x}',\vec{\eta}'}\mathsf{CPhO}^{s\dagger}\mathsf{U}_{\vec{x}',\vec{\eta}'}\mathsf{Dec}^\dagger\mathsf{Dec}\mathsf{U}^\dagger_{\vec{x},\vec{\eta}}\mathsf{CPhO}^{s}\mathsf{U}_{\vec{x},\vec{\eta}}|0,0^{2q}\rangle_{AD},\end{aligned} \tag{9.30}$$

where

$$|\Psi_0\rangle_F := \sum_{\mathbf{f}\in\mathcal{I}_{\text{per}}([N])}\frac{1}{\sqrt{N!}}|\mathbf{f}\rangle_F. \tag{9.31}$$

In the following we denote the joint state of A and the oracle when interacting with the full oracle by $|\Psi\rangle$. When dealing with the compressed oracle we write $|\varphi\rangle$.

The First Query First let us analyze the inner product after at most a single query (x,η):

$$\begin{aligned}&\sum_{\mathbf{f}\in\mathcal{I}_{\text{per}}([N])}\frac{1}{\sqrt{N!}}\bar{\omega}_N^{\eta\cdot\mathbf{f}(x)}\langle\mathbf{f}|_F\sum_{\mathbf{f}'\in\mathcal{I}_{\text{per}}([N])}\frac{1}{\sqrt{N!}}\omega_N^{\eta'\cdot\mathbf{f}'(x')}|\mathbf{f}'\rangle_F \\ &=\delta_{x,x'}\frac{1}{N}\sum_{y_1\in[N]}\omega_N^{(\eta'-\eta)\cdot y_1}+(1-\delta_{x,x'})\frac{1}{N(N-1)}\sum_{y_1\in[N]}\sum_{y_2\in[N]\setminus\{y_1\}}\bar{\omega}_N^{\eta\cdot y_1}\omega_N^{\eta'\cdot y_2} \qquad (9.32)\\ &=\delta_{x,x'}\delta_{\eta,\eta'}+(1-\delta_{x,x'})\frac{1}{N-1}\left(N\delta_{\eta,0}\delta_{\eta',0}-\delta_{\eta,\eta'}\right). \qquad (9.33)\end{aligned}$$

Now the question is how to model the above inner product with a compressed state. We can see that η and η' have to be equal for the inner product to be non-zero. That points towards saving η in the database as a basis state. This however means that we should save the information about all other behavior of $\langle\Psi_{x,\eta}|\Psi_{x',\eta'}\rangle$ in the X-type register of the database. Our proposition for that is:

$$\mathsf{CFO}_{\mathfrak{P}}|x,\eta\rangle_A|0,0\rangle_D = |x,\eta\rangle_A\left(\sqrt{\frac{N-1}{N}}|x,\eta\rangle_D-\frac{1}{\sqrt{N(N-1)}}\sum_{\tilde{x}\neq x}|\tilde{x},\eta\rangle_D\right) \tag{9.34}$$

$$= |x, \eta\rangle_A \mathsf{QFT}_N^{X\dagger} \frac{1}{\sqrt{N-1}} \sum_{\xi \neq 0} \omega_N^{x \cdot \xi} |\xi, \eta\rangle \tag{9.35}$$

As we see, even after a single query the quantum database of a random permutation is different from one for a uniformly random function. As already shown in Theorem 9.1.

After two queries As we include more queries it becomes more complicated to figure out the form of the compressed state. Now we want to analyze $\langle \Psi_{\vec{x}', \vec{\eta}'} | \Psi_{\vec{x}, \vec{\eta}} \rangle$ with a different number of distinct queries $s := |\{x'_1, x'_2, x_1, x_2\}|$. Different s correspond to different cross terms in the overall inner product. General observations that we made when researching this subject are:

1. D^X holds the sum in $\langle \Psi_{\vec{x}', \vec{\eta}'} | \Psi_{\vec{x}, \vec{\eta}} \rangle$ reduced to the number of queries done, e.g. if $q = 2$ and $s = 3$ we perform the sum over y_3 and the rest of the formula is in D^X.
2. D^Y holds "half" of the result of the product after reducing to the sum of size q.

By the first point we mean that register D^X needs to take care of the terms in $\langle \Psi_{\vec{x}', \vec{\eta}'} | \Psi_{\vec{x}, \vec{\eta}} \rangle$ that result from distinct inputs giving a non-zero inner product (unlike in the case of the uniform distribution). In the second point we require that the result of the sum (depending on the queries) is encoded in register D^Y.

For compressed oracles after two queries we anticipate the following state will be important in the final result:

$$|\Upsilon^0_{\text{test}}(\eta_1, \eta_2)\rangle := \sqrt{\frac{N}{N-1}} |\eta_1, \eta_2\rangle - \frac{1}{\sqrt{N(N-1)}} \sum_{\tilde{\eta}} |\tilde{\eta}, \eta_1 + \eta_2 - \tilde{\eta}\rangle. \tag{9.36}$$

Additional observations about $|\Upsilon^0_{\text{test}}(\eta_1, \eta_2)\rangle$: It is just the Fourier transform of a partial function state of random permutations:

$$\mathsf{QFT}_N^{D_1 D_2} \sum_{z_1, z_2 \neq z_1} \frac{1}{\sqrt{N(N-1)}} \omega_N^{\eta_1 \cdot z_1} \omega_N^{\eta_2 \cdot z_2} |z_1, z_2\rangle_{D_1 D_2} =$$

$$\mathsf{QFT}_N^{D_1 D_2} \left(\sum_{z_1, z_2} \frac{1}{\sqrt{N(N-1)}} \omega_N^{\eta_1 \cdot z_1} \omega_N^{\eta_2 \cdot z_2} |z_1, z_2\rangle \right.$$

$$\left. - \sum_{z_1} \frac{1}{\sqrt{N(N-1)}} \omega_N^{(\eta_1 + \eta_2) \cdot z_1} |z_1, z_1\rangle \right) = \tag{9.37}$$

$$\sqrt{\frac{N}{N-1}} |\eta_1, \eta_2\rangle - \frac{1}{\sqrt{N(N-1)}} \sum_{\tilde{\eta}} |\tilde{\eta}, \eta_1 + \eta_2 - \tilde{\eta}\rangle \tag{9.38}$$

We also know how to prepare the above state:

$$|\eta_2\rangle_A|\eta_1\rangle_{D_1}|0\rangle_{D_2} \overset{\mathsf{Samp}_{\mathfrak{P}}^{D_1D_2}}{\longmapsto}$$

$$|\eta_2\rangle_A \frac{1}{\sqrt{N}} \sum_{y_1\in[N]} \omega_N^{\eta_1\cdot y_1}|y_1\rangle_{D_1} \sum_{y_2\in[N]\setminus\{y_1\}} \frac{1}{\sqrt{N-1}}|y_2\rangle_{D_2} \overset{\mathsf{PhO}}{\mapsto} \tag{9.39}$$

$$|\eta_2\rangle_A \frac{1}{\sqrt{N}} \sum_{y_1\in[N]} \omega_N^{\eta_1\cdot y_1}|y_1\rangle_{D_1} \sum_{y_2\in[N]\setminus\{y_1\}} \frac{1}{\sqrt{N-1}}\omega_N^{\eta_2\cdot y_2}|y_2\rangle_{D_2} \overset{\mathsf{QFT}_N^{D_1D_2\dagger}}{\mapsto} \tag{9.40}$$

$$\sqrt{\frac{N}{N-1}}|\eta_1,\eta_2\rangle - \frac{1}{\sqrt{N(N-1)}}\sum_{\tilde\eta}|\tilde\eta,\eta_1+\eta_2-\tilde\eta\rangle \tag{9.41}$$

The point of the state $|\Upsilon^0(\eta_1,\eta_2)\rangle$ is that:

$$\langle\Upsilon^0_{\text{test}}(\eta_1',\eta_2')|\Upsilon^0_{\text{test}}(\eta_1,\eta_2)\rangle = \frac{1}{N-1}\left(N\delta_{\eta_1',\eta_1}\delta_{\eta_2',\eta_2} - \delta_{\eta_1'+\eta_2',\eta_1+\eta_2}\right), \tag{9.42}$$

which is the inner product when $s=2$:

$$\langle\Psi_{\vec{x}',\vec{\eta}'}|\Psi_{\vec{x},\vec{\eta}}\rangle = \frac{1}{N(N-1)}\sum_{y_1,y_2\neq y_1}\bar\omega_N^{\eta_1'\cdot y_1}\bar\omega_N^{\eta_2'\cdot y_2}\omega_N^{\eta_1\cdot y_1}\omega_N^{\eta_2\cdot y_2} =$$

$$\frac{1}{N-1}\left(N\delta_{\eta_1',\eta_1}\delta_{\eta_2',\eta_2} - \delta_{\eta_1'+\eta_2',\eta_1+\eta_2}\right) \tag{9.43}$$

Changing the inputs that are equal permutes $(\eta_1',\eta_2',\eta_1,\eta_2)$.

Register D^X might hold some of the following states:

$$|\Xi^0(x_1,x_2)\rangle := |x_1,x_2\rangle, \tag{9.44}$$

$$|\Xi^1_1(x_1,x_2)\rangle := \frac{1}{\sqrt{N-2}}\sum_{\tilde x_2\notin\{x_1,x_2\}}|x_1,\tilde x_2\rangle, \tag{9.45}$$

$$|\Xi^1_2(x_1,x_2)\rangle := \frac{1}{\sqrt{N-2}}\sum_{\tilde x_1\notin\{x_1,x_2\}}|\tilde x_1,x_2\rangle, \tag{9.46}$$

$$|\Xi^2(x_1,x_2)\rangle := \frac{1}{\sqrt{(N-2)(N-3)}}\sum_{\tilde x_1\notin\{x_1,x_2\}}\sum_{\tilde x_2\notin\{x_1,x_2,\tilde x_1\}}|\tilde x_1,\tilde x_2\rangle. \tag{9.47}$$

Let us close this chapter by listing the last two inner products important to the case of $q=2$. For $s=3$ and $x_1'=x_1$ we have:

$$\langle\Psi_{\vec{x}',\vec{\eta}'}|\Psi_{\vec{x},\vec{\eta}}\rangle$$

$$= \frac{1}{N(N-1)(N-2)}\sum_{y_1}\sum_{y_2\neq y_1}\sum_{y_3\in[N]\setminus\{y_1,y_2\}}\bar\omega_N^{\eta_1'\cdot y_1}\bar\omega_N^{\eta_2'\cdot y_2}\omega_N^{\eta_1\cdot y_1}\omega_N^{\eta_2\cdot y_3} = \tag{9.48}$$

$$\frac{1}{(N-1)(N-2)}\left(-N\delta_{\eta_1',\eta_1}\delta_{\eta_2',\eta_2} + 2\delta_{\eta_1'+\eta_2',\eta_1+\eta_2}\right). \tag{9.49}$$

Again, different relations of $\vec{x}$ to $\vec{x}'$ give different permutations of the delta function in the above inner product.

For $s = 4$, so all queries being distinct, we have

$$\langle \Psi_{\vec{x}',\vec{\eta}'} | \Psi_{\vec{x},\vec{\eta}} \rangle = \frac{1}{N(N-1)(N-2)(N-3)}$$

$$\sum_{y_1} \sum_{y_2 \neq y_1} \sum_{y_3 \in [N] \setminus \{y_1, y_2\}} \sum_{y_4 \in [N] \setminus \{y_1, y_2, y_3\}} \bar{\omega}_N^{\eta_1' \cdot y_1} \bar{\omega}_N^{\eta_2' \cdot y_2} \omega_N^{\eta_1 \cdot y_3} \omega_N^{\eta_2 \cdot y_4} \tag{9.50}$$

$$= \frac{1}{(N-1)(N-2)(N-3)}$$

$$\Big(N \delta_{\eta_1', \eta_1} \delta_{\eta_2', \eta_2} + N \delta_{\eta_1', \eta_2} \delta_{\eta_2', \eta_1} - 6 \delta_{\eta_1' + \eta_2', \eta_1 + \eta_2} + N \delta_{\eta_1' + \eta_2', 0} \delta_{\eta_1 + \eta_2, 0} \Big) . \tag{9.51}$$

CHAPTER

10 CONCLUSIONS

This thesis is a significant step in the line of research leading to establishing security guarantees for the sponge construction in the post-quantum world that match the guarantees achievable in the classical world. Proving quantum indifferentiability of the sponge construction was essentially the goal of the author's PhD from when he started it almost four and a half years ago.

More broadly, the security results that we achieve are: proving quantum collision-resistance and designing two matching attacks, proving collapsingness, proving quantum indistinguishability from a random oracle, and proving quantum indifferentiability from a random oracle. On the one hand these results are the most important aspects of this thesis, they improve the security guarantees that we can claim about the cryptographic constructions we use in multiple applications. On the other hand, though, the most important outcome are the methods used to achieve the results, as they propel scientific research forward.

On the way of achieving these results we used, repurposed, and developed a number of techniques. Among these techniques are quantum reductions, the quantum polynomial method, and the quantum game-playing framework. The last proof technique that we mention can be considered the single most important result of this thesis. As illustrated by Chapter 7 it has already proven to be a versatile tool. Even though it heavily builds upon work of others, the framework is much easier to use than the parts that constitute it. Arguably, it can also serve as an important starting point for understanding the general relation between post-quantum and classical security.

A PhD thesis marks the end of an intensive period of research on a particular topic, while at the same time, it forms the stepping stone for more scientific inquiries by the author and others.

BIBLIOGRAPHY

[Ala+18] Gorjan Alagic, Christian Majenz, Alexander Russell, and Fang Song. “Quantum-secure message authentication via blind-unforgeability”. Cryptology ePrint Archive, Report 2018/1150. `https://eprint.iacr.org/2018/1150`. 2018 (cit. on pp. 136, 199).

[Alm+19] José Bacelar Almeida, Cecile Baritel-Ruet, Manuel Barbosa, Gilles Barthe, François Dupressoir, Benjamin Grégoire, Vincent Laporte, Tiago Oliveira, Alley Stoughton, and Pierre-Yves Strub. “Machine-Checked Proofs for Cryptographic Standards: Indifferentiability of Sponge and Secure High-Assurance Implementations of SHA-3”. In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*. Ed. by Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz. ACM, 2019, pp. 1607–1622. ISBN: 978-1-4503-6747-9. DOI: `10.1145/3319535.3363211`. URL: `https://doi.org/10.1145/3319535.3363211` (cit. on p. 218).

[Amb07] Andris Ambainis. “Quantum walk algorithm for element distinctness”. In: *SIAM Journal on Computing* 37.1 (2007), pp. 210–239. DOI: `10.1137/S0097539705447311`. URL: `https://arxiv.org/abs/quant-ph/0311001v9` (cit. on pp. 16, 53, 54).

[AHU19] Andris Ambainis, Mike Hamburg, and Dominique Unruh. “Quantum Security Proofs Using Semi-classical Oracles”. In: *Advances in Cryptology - CRYPTO 2019 - 39th Annual International*

Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II. 2019, pp. 269–295. DOI: `10.1007/978-3-030-26951-7_10`. URL: `https://eprint.iacr.org/2018/904` (cit. on pp. 35, 36, 135, 136, 156, 157, 158, 159, 160, 161, 162, 231, 236).

[And+15] Elena Andreeva, Joan Daemen, Bart Mennink, and Gilles Van Assche. "Security of keyed sponge constructions using a modular proof approach". In: *International Workshop on Fast Software Encryption*. Springer. 2015, pp. 364–384 (cit. on pp. 42, 73, 74, 75, 81).

[AFS05] Daniel Augot, Matthieu Finiasz, and Nicolas Sendrier. "A Family of Fast Syndrome Based Cryptographic Hash Functions". In: *Progress in Cryptology – Mycrypt 2005*. Ed. by Ed Dawson and Serge Vaudenay. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 64–83. ISBN: 978-3-540-32066-1. DOI: `10.1007/11554868_6` (cit. on p. 7).

[Aum+10] Jean-Philippe Aumasson, Luca Henzen, Willi Meier, and María Naya-Plasencia. "Quark: A Lightweight Hash". In: *CHES 2010*. Vol. 6225. LNCS. Springer, 2010, pp. 1–15. DOI: `10.1007/978-3-642-15031-9_1` (cit. on p. 38).

[BBM13] Paul Baecher, Christina Brzuska, and Arno Mittelbach. "Reset Indifferentiability and Its Consequences". In: *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part I*. Ed. by Kazue Sako and Palash Sarkar. Vol. 8269. Lecture Notes in Computer Science. Springer, 2013, pp. 154–173. DOI: `10.1007/978-3-642-42033-7\_9`. URL: `https://doi.org/10.1007/978-3-642-42033-7%5C_9` (cit. on p. 32).

[Bal14] Jaiganesh Balasundaram. "Indifferentiability Results and Proofs for Some Popular Cryptographic Constructions". Cryptology ePrint Archive, Report 2014/533. `https://eprint.iacr.org/2014/533`. 2014 (cit. on p. 29).

[BES18] Marko Balogh, Edward Eaton, and Fang Song. "Quantum collision-finding in non-uniform random functions". In: *International Conference on Post-Quantum Cryptography*. Springer. 2018, pp. 467–486. DOI: `10.1007/978-3-319-79063-3_22`. URL: `https://eprint.iacr.org/2017/688.pdf` (cit. on p. 54).

[Bea+01] Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. "Quantum Lower Bounds by Polynomials". In: *J. ACM* 48.4 (July 2001), pp. 778–797. ISSN:

0004-5411. DOI: 10 . 1145 / 502090 . 502097. URL: https : //doi.org/10.1145/502090.502097 (cit. on p. 33).

[BHK13] Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi. "Instantiating Random Oracles via UCEs". In: *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8043. Lecture Notes in Computer Science. Springer, 2013, pp. 398–415. DOI: 10 . 1007 / 978 - 3 - 642 - 40084 - 1 \ _23. URL: https : //doi.org/10.1007/978-3-642-40084-1%5C_23 (cit. on p. 32).

[BHK14] Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi. "Cryptography from Compression Functions: The UCE Bridge to the ROM". In: *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*. Ed. by Juan A. Garay and Rosario Gennaro. Vol. 8616. Lecture Notes in Computer Science. Springer, 2014, pp. 169–187. DOI: 10 . 1007 / 978 - 3 - 662 - 44371 - 2 \ _10. URL: https://doi.org/10.1007/978-3-662-44371-2%5C_10 (cit. on p. 32).

[BR93] Mihir Bellare and Phillip Rogaway. "Random oracles are practical: A paradigm for designing efficient protocols". In: *Proceedings of the 1st ACM conference on Computer and communications security*. ACM. 1993, pp. 62–73. DOI: 10.1145/168588.168596 (cit. on pp. 8, 27).

[BR06] Mihir Bellare and Phillip Rogaway. "The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs". In: *Advances in Cryptology - EUROCRYPT 2006*. https : / / eprint . iacr . org / 2004 / 331. Springer Berlin Heidelberg, 2006, pp. 409–426. DOI: 10 . 1007 / 11761679 _ 25 (cit. on pp. 34, 35, 199, 236).

[Ben+96] Charles H Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. "Strengths and weaknesses of quantum computing". In: *SIAM journal on Computing* 26.5 (1996), pp. 1510–1523. URL: https://arxiv.org/pdf/quant-ph/9701001.pdf (cit. on p. 8).

[Ber+12a] Thierry P. Berger, Joffrey D'Hayer, Kevin Marquet, Marine Minier, and Gaël Thomas. "The GLUON Family: A Lightweight Hash Function Family Based on FCSRs". In: *Africacrypt 2012*. Springer, 2012, pp. 306–323. DOI: 10 . 1007 / 978 - 3 - 642 - 31410 - 0_19 (cit. on pp. 38, 48).

[BMV78] Elwyn Berlekamp, Robert McEliece, and Henk Van Tilborg. "On the inherent intractability of certain coding problems (corresp.)" In: *IEEE Transactions on Information Theory* 24.3 (1978), pp. 384–386 (cit. on p. 7).

[BBD09] D.J. Bernstein, J. Buchmann, and E. Dahmen. *Post-Quantum Cryptography*. Springer Berlin Heidelberg, 2009. ISBN: 9783540887027 (cit. on p. 6).

[Ber09] Daniel J Bernstein. "Introduction to post-quantum cryptography". In: *Post-quantum cryptography*. Springer, 2009, pp. 1–14 (cit. on p. 16).

[Ber+11a] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. "On the security of the keyed sponge construction". In: *Symmetric Key Encryption Workshop*. Vol. 2011. 2011 (cit. on pp. 42, 73).

[Ber+07] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. "Sponge functions". In: *ECRYPT hash workshop*. Vol. 2007. 9. `https://keccak.team/files/SpongeFunctions.pdf`. Citeseer. 2007 (cit. on pp. 7, 38, 39, 41, 62, 73, 74, 77).

[Ber+08] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. "On the Indifferentiability of the Sponge Construction". In: *Advances in Cryptology – EUROCRYPT 2008*. Ed. by Nigel Smart. Springer Berlin Heidelberg, 2008, pp. 181–197. DOI: `10.1007/978-3-540-78967-3_11` (cit. on pp. 29, 38, 41, 47, 73, 200, 218, 220).

[Ber+10] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. "Sponge-based pseudo-random number generators". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2010, pp. 33–47 (cit. on p. 73).

[Ber+11b] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. "Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications". In: *Selected Areas in Cryptography*. Vol. 7118. Springer. 2011, pp. 320–337. DOI: `10.1007/978-3-642-28496-0_19`. URL: `https://eprint.iacr.org/2011/499` (cit. on p. 40).

[Ber+12b] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. "Permutation-based encryption, authentication and authenticated encryption". In: *Directions in Authenticated Ciphers* (2012). URL: `https://keccak.team/files/KeccakDIAC2012.pdf` (cit. on p. 73).

[Ber+18] Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles Van Assche, Ronny Van Keer, and Benoît Viguier. "KangarooTwelve: Fast Hashing Based on Keccak-p". In: *Applied Cryptography and Network Security*. Ed. by Bart Preneel and Frederik Vercauteren. Cham: Springer International Publishing, 2018, pp. 400–418. ISBN: 978-3-319-93387-0. DOI: `10.1007/978-3-319-93387-0_21` (cit. on p. 38).

[BN18a] Srimanta Bhattacharya and Mridul Nandi. "A note on the chi-square method: A tool for proving cryptographic security". In: *Cryptogr. Commun.* 10.5 (2018), pp. 935–957. DOI: `10.1007/s12095-017-0276-z`. URL: `https://doi.org/10.1007/s12095-017-0276-z` (cit. on p. 237).

[Bin+19] Nina Bindel, Mike Hamburg, Kathrin Hövelmanns, Andreas Hülsing, and Edoardo Persichetti. "Tighter Proofs of CCA Security in the Quantum Random Oracle Model". In: *Theory of Cryptography - 17th International Conference, TCC 2019, Nuremberg, Germany, December 1-5, 2019, Proceedings, Part II*. Ed. by Dennis Hofheinz and Alon Rosen. Vol. 11892. Lecture Notes in Computer Science. Springer, 2019, pp. 61–90. DOI: `10.1007/978-3-030-36033-7_3`. URL: `https://eprint.iacr.org/2019/590` (cit. on p. 35).

[BN18b] Andreas Bluhm and Ion Nechita. "Joint measurability of quantum effects and the matrix diamond". In: *Journal of Mathematical Physics* 59.11 (2018), p. 112202. DOI: `10.1063/1.5049125`. URL: `https://arxiv.org/abs/1807.01508` (cit. on p. 117).

[Bog+13] Andrey Bogdanov, Miroslav Knezevic, Gregor Leander, Deniz Toz, Kerem Varici, and Ingrid Verbauwhede. "SPONGENT: The Design Space of Lightweight Cryptographic Hashing". In: *IEEE Transactions on Computers* 62.10 (2013), pp. 2041–2053. DOI: `10.1109/TC.2012.196` (cit. on p. 38).

[Bon+11] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. "Random Oracles in a Quantum World". In: *Advances in Cryptology – ASIACRYPT 2011*. LNCS 7073. 2011, pp. 41–69. DOI: `10.1007/978-3-642-25385-0_3` (cit. on pp. 8, 16, 27, 47, 74, 137).

[BHT98] Gilles Brassard, Peter HØyer, and Alain Tapp. "Quantum cryptanalysis of hash and claw-free functions". In: *LATIN'98: Theoretical Informatics*. Ed. by Cláudio L. Lucchesi and Arnaldo V. Moura. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 163–169. ISBN: 978-3-540-69715-2. DOI: `10.1007/BFb0054319`. URL: `https://arxiv.org/abs/quant-ph/9705002` (cit. on pp. 8, 54).

[BDS09] Johannes Buchmann, Erik Dahmen, and Michael Szydlo. “Hash-based Digital Signature Schemes”. In: *Post-Quantum Cryptography*. Ed. by Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 35–93. ISBN: 978-3-540-88702-7. DOI: 10.1007/978-3-540-88702-7_3. URL: https://doi.org/10.1007/978-3-540-88702-7_3 (cit. on p. 6).

[Can01] Ran Canetti. “Universally Composable Security: A New Paradigm for Cryptographic Protocols”. In: *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*. IEEE Computer Society, 2001, pp. 136–145. DOI: 10.1109/SFCS.2001.959888. URL: https://eprint.iacr.org/2000/067 (cit. on p. 20).

[Can+12] Anne Canteaut, Thomas Fuhr, Marıéa Naya-Plasencia, Pascal Paillier, Jean-René Reinhard, and Marion Videau. “A Unified Indifferentiability Proof for Permutation- or Block Cipher-Based Hash Functions”. In: *IACR Cryptol. ePrint Arch.* 2012 (2012), p. 363. URL: http://eprint.iacr.org/2012/363 (cit. on p. 29).

[Car+18] Tore Vincent Carstens, Ehsan Ebrahimi, Gelo Noel Tabia, and Dominique Unruh. “On Quantum Indifferentiability”. Cryptology ePrint Archive, Report 2018/257. https://eprint.iacr.org/2018/257. 2018 (cit. on pp. 29, 113, 125, 126).

[CZ83] Gianni Cassinelli and N Zanghi. “Conditional probabilities in quantum mechanics. I. – Conditioning with respect to a single event”. In: *Il Nuovo Cimento B (1971-1996)* 73.2 (1983), pp. 237–245 (cit. on p. 116).

[Cha+12] Dong H. Chang, Morris J. Dworkin, Seokhie Hong, John M. Kelsey, and Mridul Nandi. *A Keyed Sponge Construction with Pseudorandomness in the Standard Model*. NIST. The Third SHA-3 Candidate Conference. 2012 (cit. on p. 42).

[CEV20] Céline Chevalier, Ehsan Ebrahimi, and Quoc Huy Vu. “On the Security Notions for Encryption in a Quantum World”. In: *IACR Cryptol. ePrint Arch.* 2020 (2020), p. 237. URL: https://eprint.iacr.org/2020/237 (cit. on p. 136).

[Chu+20a] Kai-Min Chung, Serge Fehr, Yu-Hsuan Huang, and Tai-Ning Liao. “On the Compressed-Oracle Technique, and Post-Quantum Security of Proofs of Sequential Work”. In: *arXiv preprint arXiv:2010.11658* (2020) (cit. on p. 136).

[Chu+20b] Kai-Min Chung, Siyao Guo, Qipeng Liu, and Luowen Qian. “Tight Quantum Time-Space Tradeoffs for Function Inversion”. In: *Electron. Colloquium Comput. Complex.* 27 (2020), p. 90. URL: `https://arxiv.org/pdf/2006.05650.pdf` (cit. on p. 136).

[CS16] Benoît Cogliati and Yannick Seurin. “EWCDM: An Efficient, Beyond-Birthday Secure, Nonce-Misuse Resistant MAC”. In: *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*. 2016, pp. 121–149. DOI: `10.1007/978-3-662-53018-4_5`. URL: `https://eprint.iacr.org/2016/525` (cit. on pp. 43, 44, 200).

[CLS06] Scott Contini, Arjen K. Lenstra, and Ron Steinfeld. “VSH, an Efficient and Provable Collision-Resistant Hash Function”. In: *Advances in Cryptology - EUROCRYPT 2006: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006. Proceedings*. Ed. by Serge Vaudenay. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 165–182. ISBN: 978-3-540-34547-3. DOI: `10.1007/11761679_11`. URL: `http://dx.doi.org/10.1007/11761679_11` (cit. on p. 8).

[Cor+05] Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. “Merkle-Damgård Revisited: How to Construct a Hash Function”. In: *Advances in Cryptology – CRYPTO 2005*. Springer Berlin Heidelberg, 2005, pp. 430–448. DOI: `10.1007/11535218_26` (cit. on pp. 29, 30, 37, 38, 200, 201).

[Cza21] Jan Czajkowski. *Github repository “joints-counterexample”*. 2021. URL: `https://github.com/czajkowskijan/joints-counterexample` (cit. on p. 127).

[CG21] Jan Czajkowski and Alex B. Grilo. “On-State Commutativity of Measurements and Joint Distributions of Their Outcomes”. In: *arXiv preprint arXiv:2101.08313* (2021). URL: `https://arxiv.org/abs/2101.08313` (cit. on pp. 2, 9).

[Cza+18] Jan Czajkowski, Leon Groot Bruinderink, Andreas Hülsing, Christian Schaffner, and Dominique Unruh. “Post-quantum Security of the Sponge Construction”. In: *Post-Quantum Cryptography*. Springer International Publishing, 2018, pp. 185–204. DOI: `10.1007/978-3-319-79063-3_9` (cit. on pp. 2, 9, 25, 38, 47, 53, 65, 66, 69, 199).

[CHS19] Jan Czajkowski, Andreas Hülsing, and Christian Schaffner. "Quantum Indistinguishability of Random Sponges". In: *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*. 2019, pp. 296–325. DOI: `10.1007/978-3-030-26951-7_11`. URL: `https://eprint.iacr.org/2019/069` (cit. on pp. 2, 9, 38, 48, 199).

[Cza+19] Jan Czajkowski, Christian Majenz, Christian Schaffner, and Sebastian Zur. "Quantum Lazy Sampling and Game-Playing Proofs for Quantum Indifferentiability". In: *CoRR* abs/1904.11477 (2019). arXiv: `1904.11477`. URL: `http://arxiv.org/abs/1904.11477` (cit. on pp. 2, 10, 29, 35, 38, 135, 136, 139, 144, 148, 200, 236).

[CPD19] Jan Czajkowski, Krzysztof Pawłowski, and Rafał Demkowicz-Dobrzański. "Many-body effects in quantum metrology". In: *New Journal of Physics* (2019). URL: `http://iopscience.iop.org/10.1088/1367-2630/ab1fc2` (cit. on p. 2).

[DHT17] Wei Dai, Viet Tung Hoang, and Stefano Tessaro. "Information-Theoretic Indistinguishability via the Chi-Squared Method". In: *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III*. Ed. by Jonathan Katz and Hovav Shacham. Vol. 10403. Lecture Notes in Computer Science. Springer, 2017, pp. 497–523. DOI: `10.1007/978-3-319-63697-9\_17`. URL: `https://doi.org/10.1007/978-3-319-63697-9%5C_17` (cit. on pp. 43, 237).

[Dai+17] Yuanxi Dai, Yannick Seurin, John Steinberger, and Aishwarya Thiruvengadam. "Indifferentiability of Iterated Even-Mansour Ciphers with Non-idealized Key-Schedules: Five Rounds Are Necessary and Sufficient". In: *Advances in Cryptology – CRYPTO 2017*. Springer. 2017, pp. 524–555. DOI: `10.1007/978-3-319-63697-9_18` (cit. on p. 29).

[DS16] Yuanxi Dai and John Steinberger. "Indifferentiability of 8-round feistel networks". In: *Advances in Cryptology – CRYPTO 2016*. Springer. 2016, pp. 95–120. DOI: `10.1007/978-3-662-53018-4_4`. URL: `https://eprint.iacr.org/2015/1069` (cit. on p. 29).

[Dam90] Ivan Bjerre Damgård. "A Design Principle for Hash Functions". In: *Advances in Cryptology — CRYPTO' 89 Proceedings*. Springer New York, 1990, pp. 416–427. DOI: `10.1007/0-387-34805-0_39` (cit. on pp. 7, 37).

[Dem+13] Grégory Demay, Peter Gazi, Martin Hirt, and Ueli Maurer. "Resource-Restricted Indifferentiability". In: *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*. Ed. by Thomas Johansson and Phong Q. Nguyen. Vol. 7881. Lecture Notes in Computer Science. Springer, 2013, pp. 664–683. DOI: 10.1007/978-3-642-38348-9_39. URL: https://doi.org/10.1007/978-3-642-38348-9%5C_39 (cit. on p. 32).

[DCS17] Rafał Demkowicz-Dobrza ński, Jan Czajkowski, and Pavel Sekatski. "Adaptive Quantum Metrology under General Markovian Noise". In: *Phys. Rev. X* 7 (4 Oct. 2017), p. 041009. DOI: 10.1103/PhysRevX.7.041009. URL: https://link.aps.org/doi/10.1103/PhysRevX.7.041009 (cit. on p. 2).

[DH76] Whitfield Diffie and Martin Hellman. "New directions in cryptography". In: *IEEE transactions on Information Theory* 22.6 (1976), pp. 644–654. DOI: 10.1109/TIT.1976.1055638 (cit. on p. 5).

[DP17] J. Ding and A. Petzoldt. "Current State of Multivariate Cryptography". In: *IEEE Security Privacy* 15.4 (2017), pp. 28–36. DOI: 10.1109/MSP.2017.3151328 (cit. on p. 6).

[Ebr18] Ehsan Ebrahimi. "Post-quantum security in the presence of superposition queries". In: *Dissertationes informaticae Universitatis Tartuensis*. 2018. ISBN: 978-9949-77-934-5. URL: https://dspace.ut.ee/bitstream/handle/10062/62687/ebrahimi_ehsan.pdf?sequence=1&isAllowed=y (cit. on pp. 127, 131).

[Ell70] James H Ellis. "The possibility of secure non-secret digital encryption". In: *UK Communications Electronics Security Group* (1970), p. 6. URL: https://cryptocellar.org/cesg/possnse.pdf (cit. on p. 5).

[Feh18] Serge Fehr. "Classical Proofs for the Quantum Collapsing Property of Classical Hash Functions". In: *Theory of Cryptography*. Springer International Publishing, 2018, pp. 315–338. DOI: 10.1007/978-3-030-03810-6_12 (cit. on pp. 25, 199).

[Fin73] Arthur Fine. "Probability and the interpretation of quantum mechanics". In: *The British Journal for the Philosophy of Science* 24.1 (1973), pp. 1–37. DOI: 10.1093/bjps/24.1.1 (cit. on pp. 116, 125).

[Fin82] Arthur Fine. "Joint distributions, quantum correlations, and commuting observables". In: *Journal of Mathematical Physics* 23.7 (1982), pp. 1306–1310. DOI: `10.1063/1.525514` (cit. on pp. 116, 125).

[FR96] Peter Friis and Mikael Rørdam. "Almost commuting self-adjoint matrices-a short proof of Huaxin Lin's theorem". In: *Journal fur die Reine und Angewandte Mathematik* 479 (1996), pp. 121–132. DOI: `10.1515/crll.1996.479.121` (cit. on pp. 117, 131).

[Gal+16] Steven D. Galbraith, Christophe Petit, Barak Shani, and Yan Bo Ti. "On the Security of Supersingular Isogeny Cryptosystems". In: *Advances in Cryptology – ASIACRYPT 2016*. Ed. by Jung Hee Cheon and Tsuyoshi Takagi. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 63–91. ISBN: 978-3-662-53887-6. DOI: `10.1007/978-3-662-53887-6_3`. URL: `https://eprint.iacr.org/2016/859` (cit. on p. 6).

[GPT15] Peter Gaži, Krzysztof Pietrzak, and Stefano Tessaro. "The exact PRF security of truncation: Tight bounds for keyed sponges and truncated CBC". In: *Advances in Cryptology – CRYPTO 2015*. Springer. 2015, pp. 368–387. DOI: `10.1007/978-3-662-47989-6_18` (cit. on p. 73).

[Gri+20] Alex B Grilo, Kathrin Hövelmann, Andreas Hülsing, and Christian Majenz. "Tight adaptive reprogramming in the QROM". In: *arXiv preprint arXiv:2010.15103* (2020). URL: `https://arxiv.org/abs/2010.15103` (cit. on p. 27).

[Gro96] Lov K Grover. "A fast quantum mechanical algorithm for database search". In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. ACM. 1996, pp. 212–219. DOI: `10.1145/237814.237866` (cit. on pp. 16, 192).

[GN02] Stan Gudder and Gabriel Nagy. "Sequentially independent effects". In: *Proceedings of the American Mathematical Society* 130.4 (2002), pp. 1125–1130. DOI: `10.2307/2699560` (cit. on pp. 116, 121).

[GL16] Chun Guo and Dongdai Lin. "Indifferentiability of 3-Round Even-Mansour with Random Oracle Key Derivation". In: *IACR Cryptol. ePrint Arch.* 2016 (2016), p. 894. URL: `http://eprint.iacr.org/2016/894` (cit. on p. 29).

[GPP11] Jian Guo, Thomas Peyrin, and Axel Poschmann. "The PHOTON Family of Lightweight Hash Functions". In: *Crypto 2011*. Springer, 2011, pp. 222–239. DOI: `10.1007/978-3-642-22792-9_13` (cit. on p. 38).

[HM96] Shai Halevi and Silvio Micali. "Practical and Provably-Secure Commitment Schemes from Collision-Free Hashing". English. In: *Crypto '96*. Vol. 1109. LNCS. Springer, 1996, pp. 201–215. ISBN: 978-3-540-61512-5. DOI: 10.1007/3-540-68697-5_16 (cit. on p. 23).

[HM20] Yassine Hamoudi and Frédéric Magniez. "Quantum Time-Space Tradeoffs by Recording Queries". In: *arXiv preprint arXiv:2002.08944* (2020). URL: https://arxiv.org/abs/2002.08944 (cit. on p. 136).

[Has09] Matthew B Hastings. "Making almost commuting matrices commute". In: *Communications in Mathematical Physics* 291.2 (2009), pp. 321–345. DOI: 10.1007/s00220-009-0877-2 (cit. on pp. 113, 114, 116, 117, 131).

[HMZ16] Teiko Heinosaari, Takayuki Miyadera, and Mário Ziman. "An invitation to quantum incompatibility". In: *Journal of Physics A: Mathematical and Theoretical* 49.12 (2016), p. 123001. DOI: 10.1088/1751-8113/49/12/123001. URL: https://arxiv.org/abs/1511.07548 (cit. on p. 117).

[HI19] Akinori Hosoyamada and Tetsu Iwata. "4-Round Luby-Rackoff Construction is a qPRP". In: *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part I*. 2019, pp. 145–174. DOI: 10.1007/978-3-030-34578-5_6. URL: https://eprint.iacr.org/2019/243 (cit. on pp. 136, 155).

[HRS16] Andreas Hülsing, Joost Rijneveld, and Fang Song. "Mitigating Multi-Target Attacks in Hash-based Signatures". In: *Public Key Cryptography – PKC 2016*. Ed. by Giuseppe Persiano and Bo-Yin Yang. Vol. 9614. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 2016, pp. 387–416. DOI: 10.1007/978-3-662-49384-7_15. URL: https://eprint.iacr.org/2015/1256 (cit. on pp. 8, 22, 52).

[JN18] Ashwin Jha and Mridul Nandi. "Applications of H-Technique: Revisiting Symmetric Key Security Analysis". In: *IACR Cryptology ePrint Archive* 2018 (2018), p. 1130. URL: https://eprint.iacr.org/2018/1130 (cit. on p. 237).

[JZM19] Haodong Jiang, Zhenfeng Zhang, and Zhi Ma. "Tighter security proofs for generic key encapsulation mechanism in the quantum random oracle model". Cryptology ePrint Archive, Report 2019/134. https://eprint.iacr.org/2019/134. 2019 (cit. on p. 136).

[Jor75] Camille Jordan. "Essai sur la géométrie à n dimensions". In: *Bulletin de la Société mathématique de France* 3 (1875), pp. 103–174 (cit. on p. 129).

[JM18] Daniel Jost and Ueli Maurer. "Security Definitions for Hash Functions: Combining UCE and Indifferentiability". In: *Security and Cryptography for Networks - 11th International Conference, SCN 2018, Amalfi, Italy, September 5-7, 2018, Proceedings*. Ed. by Dario Catalano and Roberto De Prisco. Vol. 11035. Lecture Notes in Computer Science. Springer, 2018, pp. 83–101. DOI: 10.1007/978-3-319-98113-0_5. URL: https://doi.org/10.1007/978-3-319-98113-0%5C_5 (cit. on p. 32).

[Kap+16] Marc Kaplan, Gaëtan Leurent, Anthony Leverrier, and Marıéa Naya-Plasencia. "Breaking symmetric cryptosystems using quantum period finding". In: *Advances in Cryptology – CRYPTO 2016*. Springer. 2016, pp. 207–237. DOI: 10.1007/978-3-662-53008-5_8. URL: https://arxiv.org/abs/1602.05973 (cit. on pp. 9, 73, 74, 75).

[KL14] J. Katz and Y. Lindell. *Introduction to Modern Cryptography, Second Edition*. Chapman & Hall/CRC Cryptography and Network Security Series. Taylor & Francis, 2014 (cit. on pp. 6, 8, 21, 22, 44, 218).

[Köl+16] Stefan Kölbl, Martin M Lauridsen, Florian Mendel, and Christian Rechberger. "Haraka v2–efficient short-input hashing for post-quantum applications". In: *IACR Transactions on Symmetric Cryptology* (2016), pp. 1–29. DOI: 10.13154/tosc.v2016.i2.1-29. URL: https://eprint.iacr.org/2016/098 (cit. on p. 75).

[KRS09] Robert Konig, Renato Renner, and Christian Schaffner. "The operational meaning of min-and max-entropy". In: *IEEE Transactions on Information theory* 55.9 (2009), pp. 4337–4347. DOI: 10.1109/TIT.2009.2025545 (cit. on p. 231).

[KM07] Hidenori Kuwakado and Masakatu Morii. "Indifferentiability of Single-Block-Length and Rate-1 Compression Functions". In: *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* 90-A.10 (2007), pp. 2301–2308. DOI: 10.1093/ietfec/e90-a.10.2301. URL: https://doi.org/10.1093/ietfec/e90-a.10.2301 (cit. on p. 29).

[KM12] Hidenori Kuwakado and Masakatu Morii. "Security on the quantum-type Even-Mansour cipher". In: *Information Theory and its Applications (ISITA), 2012 International Symposium on*. IEEE. 2012, pp. 312–316 (cit. on pp. 74, 75).

[Lin97] Huaxin Lin. "Almost commuting selfadjoint matrices and applications". In: *Fields Institute Commun.* 13 (1997), pp. 193–233 (cit. on p. 117).

[LLG12] Yiyuan Luo, Xuejia Lai, and Zheng Gong. "Indifferentiability of Domain Extension Modes for Hash Functions". In: *Trusted Systems*. Springer Berlin Heidelberg, 2012, pp. 138–155. DOI: 10.1007/978-3-642-32298-3_10 (cit. on p. 29).

[Maa06] Hans Maassen. "Quantum Probability and Quantum Information Theory". In: *https://www.math.ru.nl/~maassen/lectures/Trieste.pdf* (2006) (cit. on p. 116).

[Maa10] Hans Maassen. "Quantum probability and quantum information theory". In: *Quantum Information, Computation and Cryptography: An Introductory Survey of Theory, Technology and Experiments*. Springer, 2010, pp. 65–108. DOI: 10.1007/978-3-642-11914-9_3 (cit. on p. 116).

[Mah18] U. Mahadev. "Classical Homomorphic Encryption for Quantum Circuits". In: *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*. 2018, pp. 332–338. DOI: 10.1109/FOCS.2018.00039 (cit. on p. 143).

[Mau10] Ueli Maurer. "Constructive Cryptography - A Primer". In: *Financial Cryptography and Data Security, 14th International Conference, FC 2010, Tenerife, Canary Islands, Spain, January 25-28, 2010, Revised Selected Papers*. Ed. by Radu Sion. Vol. 6052. Lecture Notes in Computer Science. Springer, 2010, p. 1. DOI: 10.1007/978-3-642-14577-3_1. URL: https://doi.org/10.1007/978-3-642-14577-3%5C_1 (cit. on p. 26).

[Mau11] Ueli Maurer. "Constructive Cryptography - A New Paradigm for Security Definitions and Proofs". In: *Theory of Security and Applications - Joint Workshop, TOSCA 2011, Saarbrücken, Germany, March 31 - April 1, 2011, Revised Selected Papers*. Ed. by Sebastian Mödersheim and Catuscia Palamidessi. Vol. 6993. Lecture Notes in Computer Science. Springer, 2011, pp. 33–56. DOI: 10.1007/978-3-642-27375-9_3. URL: https://doi.org/10.1007/978-3-642-27375-9_3 (cit. on pp. 26, 32).

[Mau02] Ueli M. Maurer. "Indistinguishability of Random Systems". In: *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings*. Ed. by Lars R. Knudsen. Vol. 2332. Lecture Notes in Computer Science. Springer, 2002, pp. 110–132. DOI: 10.1007/3-540-46035-7_8. URL:

`https://www.iacr.org/archive/eurocrypt2002/23320104/e02proc.pdf` (cit. on p. 236).

[MT07] Ueli M. Maurer and Stefano Tessaro. "Domain Extension of Public Random Functions: Beyond the Birthday Barrier". In: *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*. Ed. by Alfred Menezes. Vol. 4622. Lecture Notes in Computer Science. Springer, 2007, pp. 187–204. DOI: `10.1007/978-3-540-74143-5\_11`. URL: `https://doi.org/10.1007/978-3-540-74143-5%5C_11` (cit. on p. 43).

[MR11] Ueli Maurer and Renato Renner. "Abstract Cryptography". In: *Innovations in Computer Science - ICS 2011, Tsinghua University, Beijing, China, January 7-9, 2011. Proceedings*. Ed. by Bernard Chazelle. Tsinghua University Press, 2011, pp. 1–21. URL: `http://conference.iiis.tsinghua.edu.cn/ICS2011/content/papers/14.html` (cit. on pp. 26, 32).

[MRH04] Ueli Maurer, Renato Renner, and Clemens Holenstein. "Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology". In: *Theory of Cryptography*. Springer Berlin Heidelberg, 2004, pp. 21–39. DOI: `10.1007/978-3-540-24638-1_2` (cit. on pp. 29, 30, 31, 32).

[MN17a] Bart Mennink and Samuel Neves. "Encrypted Davies-Meyer and Its Dual: Towards Optimal Security Using Mirror Theory". In: *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III*. 2017, pp. 556–583. DOI: `10.1007/978-3-319-63697-9_19`. URL: `https://eprint.iacr.org/2017/473` (cit. on pp. 43, 44, 200).

[MN17b] Bart Mennink and Samuel Neves. "Optimal PRFs from Blockcipher Designs". In: *IACR Trans. Symmetric Cryptol.* 2017.3 (2017), pp. 228–252. DOI: `10.13154/tosc.v2017.i3.228-252`. URL: `https://doi.org/10.13154/tosc.v2017.i3.228-252` (cit. on p. 44).

[MRV15] Bart Mennink, Reza Reyhanitabar, and Damian Vizár. "Security of Full-State Keyed Sponge and Duplex: Applications to Authenticated Encryption". In: *Advances in Cryptology – ASIACRYPT 2015*. Springer. 2015, pp. 465–489. DOI: `10.1007/978-3-662-48800-3_19`. URL: `https://eprint.iacr.org/2015/541` (cit. on p. 73).

[MS17] Bart Mennink and Alan Szepieniec. "XOR of PRPs in a Quantum World". In: *Post-Quantum Cryptography*. Springer. 2017, pp. 367–383. DOI: 10.1007/978-3-319-59879-6_21. URL: https://eprint.iacr.org/2017/356 (cit. on p. 77).

[Mer90] Ralph C. Merkle. "A Certified Digital Signature". In: *Advances in Cryptology — CRYPTO' 89 Proceedings*. Springer New York, 1990, pp. 218–238. DOI: 10.1007/0-387-34805-0_21 (cit. on pp. 7, 37).

[ME84] W. de Muynck and J. van den Eijnde. "A derivation of local commutativity from macrocausality using a quantum mechanical theory of measurement". In: *Foundations of Physics* 14.2 (1984), pp. 111–146. DOI: 10.1007/BF00729970 (cit. on pp. 115, 116, 117, 118, 124, 125).

[Nan06] Mridul Nandi. "A Simple and Unified Method of Proving Indistinguishability". In: *Progress in Cryptology - INDOCRYPT 2006*. Ed. by Rana Barua and Tanja Lange. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 317–334. ISBN: 978-3-540-49769-1. DOI: 10.1007/11941378_23 (cit. on p. 236).

[Nat15] National Institute of Standards and Technology (NIST). *Secure Hash Standard (SHS)*. FIPS PUBS 180-4. 2015. DOI: 10.6028/NIST.FIPS.180-4 (cit. on p. 23).

[Nel67] Edward Nelson. *Dynamical theories of Brownian motion*. Vol. 3. Princeton university press, 1967. Chap. 14 (cit. on pp. 116, 125).

[NC10] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. 10th anniversary. Cambridge: Cambridge University Press, 2010. ISBN: 978-1107002173 (cit. on pp. 15, 17, 161, 242).

[NS94] Noam Nisan and Mario Szegedy. "On the degree of Boolean functions as real polynomials". In: *Computational complexity* 4.4 (1994), pp. 301–313. DOI: 10.1007/BF01263419. URL: https://doi.org/10.1007/BF01263419 (cit. on p. 33).

[NIS14] NIST. *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*. Draft FIPS 202. 2014. URL: http://csrc.nist.gov/publications/drafts/fips-202/fips_202_draft.pdf (cit. on pp. 7, 29, 30, 38, 40, 54).

[NIS15] NIST. *Secure Hash Standard (SHS)*. Draft FIPS 180-4. 2015. DOI: 10.6028/NIST.FIPS.180-4 (cit. on pp. 7, 29, 37).

[NIS17] NIST. *Post-Quantum Cryptography Standardization process*. 2017. URL: https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization (cit. on p. 8).

[Pat08] Jacques Patarin. "The "Coefficients H" Technique". In: *Selected Areas in Cryptography, 15th International Workshop, SAC 2008, Sackville, New Brunswick, Canada, August 14-15, Revised Selected Papers*. 2008, pp. 328–345. DOI: `10.1007/978-3-642-04159-4_21`. URL: `https://doi.org/10.1007/978-3-642-04159-4_21` (cit. on p. 237).

[PG97] Jacques Patarin and Louis Goubin. "Trapdoor one-way permutations and multivariate polynomials". In: *Information and Communications Security*. Ed. by Yongfei Han, Tatsuaki Okamoto, and Sihan Qing. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 356–368. ISBN: 978-3-540-69628-5. DOI: `10.1007/BFb0028491` (cit. on p. 7).

[Pei16] Chris Peikert. "A decade of lattice cryptography". In: *Foundations and Trends in Theoretical Computer Science* 10.4 (2016), pp. 283–424. DOI: `10.1561/0400000074` (cit. on p. 6).

[Reg09] Oded Regev. "On lattices, learning with errors, random linear codes, and cryptography". In: *Journal of the ACM (JACM)* 56.6 (2009), pp. 1–40. DOI: `10.1145/1568318.1568324` (cit. on p. 7).

[Ren08] Renato Renner. "Security of quantum key distribution". In: *International Journal of Quantum Information* 6.01 (2008). PhD Thesis, pp. 1–127. DOI: `10.1142/S0219749908003256`. URL: `https://arxiv.org/abs/quant-ph/0512258v2` (cit. on p. 19).

[RSS11] Thomas Ristenpart, Hovav Shacham, and Thomas Shrimpton. "Careful with Composition: Limitations of the Indifferentiability Framework". In: *Advances in Cryptology – EUROCRYPT 2011*. Springer Berlin Heidelberg, 2011, pp. 487–506. DOI: `10.1007/978-3-642-20465-4_27` (cit. on pp. 30, 32, 229, 230, 231, 232).

[RS14] Ronald L. Rivest and Jacob C. N. Schuldt. "Spritz–a spongy RC4-like stream cipher and hash function". In: *Charles River Crypto Day (2014-10-24)* (2014). URL: `https://people.csail.mit.edu/rivest/pubs/RS14.pdf` (cit. on p. 73).

[Ros21] Ansis Rosmanis. "Tight bounds for inverting permutations via compressed oracle arguments". In: *arXiv preprint arXiv:2103.08975* (2021). URL: `https://arxiv.org/abs/2103.08975` (cit. on p. 137).

[SS17] Thomas Santoli and Christian Schaffner. "Using Simon's algorithm to attack symmetric-key cryptographic primitives". In: *Quantum Information & Computation* 17.1–2 (2017), pp. 65–78. DOI: 10 . 5555 / 3179483 . 3179487. URL: https : //arxiv.org/abs/1603.07856 (cit. on pp. 9, 73, 74, 75).

[Sch07] Christian Schaffner. "Cryptography in the bounded-quantum-storage model". In: *arXiv preprint arXiv:0709.0289* (2007). PhD Thesis. URL: https://arxiv.org/abs/0709.0289 (cit. on p. 19).

[Sen17] N. Sendrier. "Code-Based Cryptography: State of the Art and Perspectives". In: *IEEE Security Privacy* 15.4 (2017), pp. 44–50. DOI: 10.1109/MSP.2017.3151345 (cit. on p. 6).

[Sho94] Peter W. Shor. "Algorithms for Quantum Computation: Discrete Logarithms and Factoring". In: *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*. 1994, pp. 124–134. DOI: 10.1109/SFCS.1994.365700. URL: https://doi.org/10.1109/SFCS.1994.365700 (cit. on pp. 6, 16, 73).

[SS08] Thomas Shrimpton and Martijn Stam. "Building a Collision-Resistant Compression Function from Non-compressing Primitives". In: *Automata, Languages and Programming*. Ed. by Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 643–654. ISBN: 978-3-540-70583-3. DOI: 10.1007/978-3-540-70583-3_52. URL: https://eprint.iacr.org/2007/409 (cit. on pp. 43, 200).

[SY17] Fang Song and Aaram Yun. "Quantum Security of NMAC and Related Constructions - PRF Domain Extension Against Quantum attacks". In: *CRYPTO*. Springer, 2017, pp. 283–309. DOI: 10.1007/978-3-319-63715-0_10. URL: https://eprint.iacr.org/2017/509 (cit. on pp. 33, 199).

[Ste+17] Marc Stevens, Elie Bursztein, Pierre Karpman, Ange Albertini, and Yarik Markov. "The First Collision for Full SHA-1". In: *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*. Ed. by Jonathan Katz and Hovav Shacham. Vol. 10401. Lecture Notes in Computer Science. Springer, 2017, pp. 570–596. DOI: 10.1007/978-3-319-63688-7_19. URL: https://shattered.io/static/shattered.pdf (cit. on p. 37).

[Tea17] Sphincs+ Team. *SPHINCS+*. 2017. URL: https://sphincs.org/ (cit. on p. 75).

[Unr14] Dominique Unruh. "Revocable Quantum Timed-Release Encryption". In: *Advances in Cryptology – EUROCRYPT 2014*. Ed. by Elisabeth Nguyen Phong Q.and Oswald. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 129–146. ISBN: 978-3-642-55220-5. DOI: `10.1007/978-3-642-55220-5_8` (cit. on pp. 35, 135, 136, 137, 156).

[Unr16a] Dominique Unruh. "Collapse-Binding Quantum Commitments Without Random Oracles". In: *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II*. Ed. by Jung Hee Cheon and Tsuyoshi Takagi. Vol. 10032. Lecture Notes in Computer Science. 2016, pp. 166–195. ISBN: 978-3-662-53889-0. DOI: `10.1007/978-3-662-53890-6_6`. URL: `http://dx.doi.org/10.1007/978-3-662-53890-6_6` (cit. on pp. 23, 25, 26, 199).

[Unr16b] Dominique Unruh. "Computationally Binding Quantum Commitments". In: *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9666. Lecture Notes in Computer Science. Springer, 2016, pp. 497–527. ISBN: 978-3-662-49895-8. DOI: `10.1007/978-3-662-49896-5_18`. URL: `http://dx.doi.org/10.1007/978-3-662-49896-5_18` (cit. on pp. 22, 23, 25, 26, 199).

[Unr19] Dominique Unruh. "Quantum Relational Hoare Logic". In: *Proc. ACM Program. Lang.* POPL (2019), 33:1–33:31. DOI: `10.1145/3290346` (cit. on p. 136).

[Win99] Andreas Winter. "Coding theorem and strong converse for quantum channels". In: *IEEE Transactions on Information Theory* 45.7 (1999), pp. 2481–2485. DOI: `10.1109/18.796385` (cit. on p. 163).

[YZ20] Takashi Yamakawa and Mark Zhandry. "Classical vs Quantum Random Oracles". In: *IACR Cryptol. ePrint Arch.* 2020 (2020), p. 1270. URL: `https://eprint.iacr.org/2020/1270` (cit. on p. 27).

[Zha12] Mark Zhandry. "How to Construct Quantum Random Functions". In: *FOCS 2013*. Online version is IACR ePrint 2012/182. Los Alamitos, CA, USA: IEEE Computer Society, 2012, pp. 679–687. DOI: `10.1109/FOCS.2012.37` (cit. on pp. 20, 21, 28, 33, 75, 76, 108, 109, 237).

[Zha15a] Mark Zhandry. "A Note on the Quantum Collision and Set Equality Problems". In: *Quantum Information & Computation* 15.7-8 (2015), pp. 557–567. DOI: 10.5555/2871411.2871413. URL: http://arxiv.org/abs/1312.1027v3 (cit. on pp. 8, 22, 51, 54, 55, 73, 76, 105, 106, 192, 213).

[Zha15b] Mark Zhandry. "Secure identity-based encryption in the quantum random oracle model". In: *International Journal of Quantum Information* 13.04 (2015), p. 1550014. DOI: 10.1142/S0219749915500148. URL: http://eprint.iacr.org/2012/076 (cit. on p. 33).

[Zha19a] Mark Zhandry. "How to Record Quantum Queries, and Applications to Quantum Indifferentiability". In: *Advances in Cryptology – CRYPTO 2019*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Cham: Springer International Publishing, 2019, pp. 239–268. ISBN: 978-3-030-26951-7. DOI: 10.1007/978-3-030-26951-7_9 (cit. on pp. 29, 35, 135, 136, 137, 138, 139, 150, 192, 199, 200, 203, 236).

[Zha19b] Mark Zhandry. "Quantum Lightning Never Strikes the Same State Twice". In: *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part III*. Ed. by Yuval Ishai and Vincent Rijmen. Vol. 11478. Lecture Notes in Computer Science. Springer, 2019, pp. 408–438. DOI: 10.1007/978-3-030-17659-4_14. URL: https://doi.org/10.1007/978-3-030-17659-4%5C_14 (cit. on p. 22).

[Zha20] Mark Zhandry. *Private communication*. 2020 (cit. on pp. 136, 203).

SYMBOL INDEX

$\textsc{Coll}$	A quantum algorithm finding collisions in g having at least one collision.	54
Collapse 1	Collapsing game	24
$\textsc{Comp}$	The composition construction	37
$X\|Y$	Concatenation of strings X and Y.	14
CPerO	Compressed Permutation Oracle	241
$\mathsf{CPhO}_{\mathfrak{U}}$	Compressed Phase Oracle	153
CRP	The challenge-response property game	230
$\mathsf{CStO}_{\mathfrak{D}}, \mathsf{CStO}_{\mathcal{Y}\setminus\mathcal{S}}$	Compressed Standard Oracle, for distribution $\mathfrak{D}$ and for a conditionally uniform distribution over $\mathcal{Y}\setminus\mathcal{S}$	239
$D, \Delta,$ Д	Prepared database in the standard basis (and the database register), prepared database in the Fourier basis, and the unprepared databse	142
$\mathbf{d}$	The statistical distance	236
$\mathsf{Dec}_{\mathfrak{D}}$	Decompression procedure	146
D	The distinguisher	30
$\mathfrak{D}$	A distribution.	140
$\mathcal{D}$	The set of databases	239
$\mathcal{E}$	The set of edges of a sponge graph	41
EDM	The Encrypted Davis-Meyer construction	43
EDMD	The Encrypted Davis-Meyer Dual construction	44
ϵ	A negligible function.	14
$\bar{\mathbf{f}}$	The general map between states with its output limited to the set $\mathcal{A}$	39
$\mathbf{f}$	The internal function of $\textsc{Sponge}$	38
$\hat{\mathbf{f}}$	The general map between states with its output limited to the set $\mathcal{C}$	39
Find	Event of measurement of the relation R returning 1	159, 162
$\textsc{Flag}(\nabla\text{-}\mathbf{c}_j^i)$	The flag of ∇-$\mathbf{c} \in \mathbf{p}\text{-}\nabla\text{-}\mathbf{CF}(\mathbf{M}, \mathbf{Z})$, $\textsc{Flag}(\nabla\text{-}\mathbf{c}_j^i) \in \{\boxed{u}, \boxed{f}, \boxed{n}\}$	96
$\textsc{Flag-Assign}$	The algorithm for assigning $\boxed{u}$, $\boxed{f}$, and $\boxed{n}$ flags in ∇-$\mathbf{c}$.	86, 96
Flip	Algorithm flipping the database in the standard basis, inputs are outputs and vice versa.	241

$\lfloor s \rfloor_{\min\{\ell,n\}}$	The first $\min\{\ell, n\}$ symbols of s.	49
FO	Fourier Oracle, $\mathsf{QFT}_N^{YF} \circ \mathsf{StO} \circ \mathsf{QFT}_N^{\dagger YF}$	139
$\mathcal{F}$	The set of functions $\{\mathbf{f} : \mathcal{X} \to \mathcal{Y}\}$	137
$\mathcal{Y}^{\mathcal{X}}$	The set of functions $\{\mathbf{f} : \mathcal{X} \to \mathcal{Y}\}$	14
$\lvert\Psi_q^{\mathrm{Good}}\rangle$	The state with database $D \notin R$.	169
$\mathcal{G}(s)$	The good set of for relation R	164
H, G	Compressed Oracle	37, 156
HT_n	The Hadamard transform	150
$\mathbf{H}_{\min}, \mathbf{H}_{\max}$	Entropy	18
$\mathrm{im}\, P$	Image of function/map P	14
(inner, s, s')	Output of the collision finding reduction: inner collision	49
Inner-Coll	Algorithm for deciding whether there is an inner collision in Sponge$_{\mathbf{f}}$.	58
$\mathcal{I}_{\mathrm{per}}(\mathcal{S})$	The set of bijections (permutations) from $\mathcal{S}$ to $\mathcal{S}$, $\{\mathbf{f} : \mathcal{S} \to \mathcal{S} \mid \mathbf{f} \text{ is a bijection}\}$	106
J_R	Projector on relation R.	158
$\kappa := \sum_{i=1}^{q}(k_i - 1)$	The total number of evaluations of $\mathbf{f}$ in q evaluations of Sponge on $\mathbf{M}^i$ with outputs $\mathbf{Z}^i$.	91
$k_i := \lvert\mathbf{M}^i\rvert + \lvert\mathbf{Z}^i\rvert$	The number of internal states in evaluation of Sponge$(\mathbf{M}^i)$ outputting $\mathbf{Z}^i$.	91
$\lvert\psi\rangle$	A quantum state, a normalized vector in a Hilbert space	16
$\mathcal{M}$	Arbitrary set of messages, e.g. arbitrary-length bit-strings $\{0,1\}^* = \bigcup_{l\geq 0}\{0,1\}^l$.	14
$\mathbf{M}$	Array of messages, $\mathbf{M} = (\mathbf{M}^1, \mathbf{M}^2, \ldots, \mathbf{M}^q)$, where $\forall i, \mathbf{M}^i = M_1^i \| M_2^i \| \ldots \| M_{\lvert\mathbf{M}^i\rvert}^i$ and $\mathbf{M}_j^i \in \mathcal{A}$	40, 82
$(\mathbf{M}, \mathbf{Z})$	Array of input-output pairs, $(\mathbf{M}, \mathbf{Z}) := \{(\mathbf{M}^i, \mathbf{Z}^i)\}_{i\in[2q]}$.	78, 82
$\mu := \sum_{i=1}^{q}\lvert\mathbf{M}^i\rvert$	The number of states with the next step being in the squeezing phase of Sponge.	99
∇-$\mathbf{c}$	A particular assignment of states in Sponge evaluations of $(\mathbf{M}, \mathbf{Z})$.	90

∇-$\mathbf{CF}(\mathbf{M},\mathbf{Z})$	The set of all ∇-$\mathbf{c}$ constrained by $(\mathbf{M},\mathbf{Z})$ with FLAG-ASSIGN run on it.	97
∇-$\mathrm{C}(\mathbf{M},\mathbf{Z})$	The set of all ∇-$\mathbf{c}$ constrained by $(\mathbf{M},\mathbf{Z})$.	92
∇-$\mathbf{cf}$	A particular assignment of states with flags in SPONGE evaluations of $(\mathbf{M},\mathbf{Z})$.	95
$\||\psi\rangle\|$	Norm of a quantum state	14
$\mathcal{N}$	The set of N elements: $\mathcal{N} := \{1,2,\ldots,N\}$	13
$O(n)$	Complexity class "big O"	
(outer, s, s')	Output of the collision finding reduction: outer collision	49
OUTPUT-COLL	Algorithm for deciding whether there is an output collision in $\mathrm{SPONGE}_{\mathbf{f}}$.	58
PAD	Padding function	39
$\mathbf{P}$	Array of padded messages	40
$\mathsf{SpPath}(s,G)$	Function constructing an input to SPONGE leading to a given node	41
$\mathfrak{P}$	Uniform distribution over the set of permutations	239
PhO	Phase Oracle, $\mathsf{QFT}_N^Y \circ \mathsf{StO} \circ \mathsf{QFT}_N^{\dagger Y}$	152
$\boldsymbol{\pi}$	A permutation $\boldsymbol{\pi} : \mathcal{X} \to \mathcal{X}$.	
$\mathbf{p}$-∇-$\mathbf{cf}$	A particular assignment of states in SPONGE evaluations of $(\mathbf{M},\mathbf{Z})$ with additional constraints for $\mathbf{f}$ being a valid function.	98
$\mathbf{p}$-∇-$\mathbf{CF}(\mathbf{M},\mathbf{Z})$	The set of all $\mathbf{p}$-∇-$\mathbf{cf}$ constrained by $(\mathbf{M},\mathbf{Z})$ and $\mathbf{f}$.	97
$\mathbf{p}$-∇-$\mathbf{CF}(\mathbf{M},\mathbf{Z},*)$	$\mathbf{p}$-∇-$\mathbf{CF}(\mathbf{M},\mathbf{Z})$ with specified parameters $*$.	99
$\mathcal{P}(\mathbf{M},\mathbf{Z},\bar{u},u)$	The set of placements	88, 99, 101
QFT_N	The Quantum Fourier Transform	17
$\mathrm{range}\, f$	Range of function f	14
RATE-1/3	The Rate 1/3 Construction	43
$\mathfrak{R}$	Distribution over Random Oracles	27
Rem	Removing $\text{и} = 0$ from the database	148
R	Random Oracle	27
$\mathcal{R}$	The set of rooted supernodes	41
$\mathsf{Samp}_{\mathfrak{D}}(\mathcal{S})$	Algorithm preparing a superposition of samples of outputs of $f \leftarrow \mathfrak{D}$ on inputs from $\mathcal{S}$.	141

$X \xleftarrow{\$} \mathcal{X}$	X chosen uniformly from set $\mathcal{X}$	15
$\boldsymbol{\sigma}$	A permutation.	113
S	Classical and quantum simulators.	219, 224
$\textsc{Sponge}_{\mathbf{f}}[\textsc{pad}, \mathcal{A}, \mathcal{C}]$	Sponge construction with the internal function $\mathbf{f}$, capacity set $\mathcal{C}$, and alphabet $\mathcal{A}$	40
$\textsc{Sponge-Coll}$	A quantum algorithm finding collisions in $\textsc{Sponge}_{\mathbf{f}}[\textsc{pad}, \{0,1\}^r, \{0,1\}^c](\ell, .)$	55
Spread	The full set "spreading" operation.	238
$\mathcal{S}$	$\mathcal{S} := \mathcal{A} \times \mathcal{C}$	39, 77, 78, 107
StO	Standard Oracle	138
$u_{\text{abs}}, u_{\text{squ}}$	The number of "$\boxed{\mathrm{u}}$" states in the absorbing phases and squeezing phases respectively	99
$\bar{u}_{\text{abs}}, \bar{u}_{\text{squ}}$	The number of "$\boxed{\mathrm{u}}$" and "$\boxed{\mathrm{f}}$" states in the absorbing phases and squeezing phases respectively	99
$\mathfrak{U}$	The uniform distribution.	150
$\mathcal{U}$	The set of supernodes with outgoing edges	41
V_R	The unitary outputting $D \in R$.	238
$\mathcal{V}$	The set of vertices of a sponge graph	41
$\mathbf{W}$	The joint distribution describing outputs of a sequence of measurements.	118
$\oplus$	Bitwise XOR	14, 151
$y, \eta, \text{и}$	Values in the Y register of a database in different bases	141
$\mathbf{Z}$	Array of output strings	82
(zero, s)	Output of the collision finding reduction: zero-preimage	49
$\zeta := \sum_{i=1}^{q} (\lvert\mathbf{Z}^i\rvert - 1)$	The number of states with the next state being in the absorbing phase, $\kappa = \mu + \zeta$.	99

INDEX

ABSTRACT

The research covered in this thesis is dedicated to provable post-quantum security of hash functions. Post-quantum security provides security guarantees against quantum attackers. We focus on analyzing the sponge construction, a cryptographic construction used in the standardized hash function SHA3.

Our main results are proving a number of quantum security statements. These include standard-model security: collision-resistance and collapsingness, and more idealized notions such as indistinguishability and indifferentiability from a random oracle. All these results concern quantum security of the classical cryptosystems.

From a more high-level perspective we find new applications and generalize several important proof techniques in post-quantum cryptography. We use the polynomial method to prove quantum indistinguishability of the sponge construction. We also develop a framework for quantum game-playing proofs, using the recently introduced techniques of compressed random oracles and the One-way-To-Hiding lemma.

To establish the usefulness of the new framework we also prove a number of quantum indifferentiability results for other cryptographic constructions. On the way to these results, though, we address an open problem concerning quantum indifferentiability. Namely, we disprove a conjecture that forms the basis of a no-go theorem for a version of quantum indifferentiability.

SAMENVATTING

Post-Quantumveiligheid van Hashfuncties

Het onderzoek in deze dissertatie gaat over bewijsbare post-quantum veiligheid van hashfuncties. Post-quantum veiligheid geeft veiligheidsgaranties tegen quantumaanvallers. We focussen op het analyseren van de sponsconstructie, een cryptografische constructie die wordt gebruikt in de gestandaardiseerde hashfunctie SHA3.

Onze belangrijkste resultaten bestaan uit het bewijzen van een aantal stellingen over quantum veiligheid. Deze omvatten veiligheid in het standaardmodel: botsingbestendigheid en zogeheten *collapsingness*, maar ook geïdealiseerdere concepten zoals ononderscheidbaarheid en ondifferentieerbaarheid van een random orakel. Al deze resultaten gaan over de quantumveiligheid van klassieke cryptografische systemen.

Breder gezien vinden we nieuwe toepassingen en generaliseren we verscheidene belangrijke bewijstechnieken in post-quantumcryptografie. We gebruiken de polynomiaalmethode om quantumononderscheidbaarheid van de sponsconstructie te bewijzen. Ook ontwikkelen we een framework voor bewijzen op basis van quantumspellen, waarbij we gebruik maken van recent geïntroduceerde technieken zoals gecomprimeerde orakels en de *One-way-To-Hidingstelling*.

Om het nut van dit nieuwe framework te bevestigen, bewijzen we ook een aantal resultaten op het gebied van ondifferentieerbaarheid voor andere cryptografische constructies. Tegelijkertijd met het bereiken van deze resultaten lossen we een open probleem op over quantumondifferentieerbaarheid. We ontkrachten namelijk een bestaand vermoeden dat de basis vormt voor een onmogelijkheidsbewijs voor een versie van quantumondifferentieerbaarheid.

Post-Quantum Security of Hash Functions Jan Czajkowski