Cryptographic Primitives that Resist Backdooring and Subversion

Mazaheri, Sogol

(2020)

Cryptographic Primitives that Resist Backdooring and Subversion

Vom Fachbereich Informatik der
Technischen Universität Darmstadt genehmigte

Dissertation

zur Erlangung des Grades
Doctor rerum naturalium (Dr. rer. nat.)
von

Sogol Mazaheri

Referenten:	Prof. Dr. Marc Fischlin
	Prof. Dr. Phillip Rogaway
Tag der Einreichung:	9. Juli 2020
Tag der mündlichen Prüfung:	24. August 2020

Darmstadt, 2020
D 17

List of Publications

[1] Marc Fischlin and Sogol Mazaheri. Notions of deniable message authentication. In *Proceedings of the 14th ACM Workshop on Privacy in the Electronic Society, WPES 2015, Denver, Colorado, USA, October 12, 2015*, pages 55–64, 2015.

[2] Felix Günther and Sogol Mazaheri. A formal treatment of multi-key channels. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III*, pages 587–618, 2017.

[3] Richard Gay, Jinwei Hu, Heiko Mantel, and Sogol Mazaheri. Relationship-based access control for resharing in decentralized online social networks. In *Foundations and Practice of Security - 10th International Symposium, FPS 2017, Nancy, France, October 23-25, 2017, Revised Selected Papers*, pages 18–34, 2017.

[4] Marc Fischlin and Sogol Mazaheri. Self-guarding cryptographic protocols against algorithm substitution attacks. In *31st IEEE Computer Security Foundations Symposium, CSF 2018, Oxford, United Kingdom, July 9-12, 2018*, pages 76–90, 2018. **Part of this thesis.**

[5] Marc Fischlin, Christian Janson, and Sogol Mazaheri. Backdoored hash functions: Immunizing HMAC and HKDF. In *31st IEEE Computer Security Foundations Symposium, CSF 2018, Oxford, United Kingdom, July 9-12, 2018*, pages 105–118, 2018. **Part of this thesis.**

[6] Balthazar Bauer, Pooya Farshim, and Sogol Mazaheri. Combiners for backdoored random oracles. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, pages 272–302, 2018. **Part of this thesis.**

[7] Yevgeniy Dodis, Pooya Farshim, Sogol Mazaheri, and Stefano Tessaro. Towards defeating backdoored random oracles: Indifferentiability with bounded adaptivity. In *TCC 2020: 18th Theory of Cryptography Conference*, 2020. **Part of this thesis.**

Acknowledgments

I am deeply grateful to my supervisor Marc Fischlin for introducing me to the magical world of cryptography, sharing relevant research questions with me, and giving me the opportunity to pursue a PhD in his group. Thank you for your support, patience, and for trusting me to explore freely and find my own path. I am also truly grateful to Phil Rogaway, who has inspired me not just through his immensely successful scientific work but also by being a responsible and socially engaged researcher and sharing his thoughts with the rest of us. Thank you also for being a co-referee of my thesis and for your valuable comments. I would like to further thank Sebastian Faust, Thomas Schneider, and Felix Wolf for serving on my defense committee. Furthermore, I thank Technische Universität Darmstadt and the various projects at our department such as ATHENE, CASED, and CROSSING which have enabled my PhD.

My journey would certainly not have been the same without my kind and intelligent colleagues and friends in Marc's group, with whom I have shared many wonderful memories and discussions about research and life in general. In order of appearance in my life I would like to thank Giorgia Azzurra Marson, Felix Günther, Tommaso Gagliardoni, Arno Mittelbach, Andrea Püchner, Jacqueline Brendel, Christian Janson, Patrick Harasser, Jean Paul Degabriele, Felix Rohrbach, Aishwarya Thiruvengadam, Olga Sanina, Tobias Schmalz, Rune Fiedler, and Shan Chen. I feel specially blessed to have found excellent friends in Giorgia, Felix, Andrea, Jacqueline, and Christian.

I had the pleasure of collaborating with exceptionally talented researchers on the topics included in this thesis. I thank (in alphabetical order) Balthazar Bauer, Yevgeniy Dodis, Pooya Farshim, Marc Fischlin, Christian Janson, and Stefano Tessaro. I owe a great deal of what I learned to them. Very specially I thank Pooya for his friendship, honest advice, and making me a better researcher. I thank Anja Lehmann for enlightening and inspiring discussions during my short visit to IBM and afterwards.

I would like to express my deep gratitude towards everyone who was somehow involved in organizing events such as schools, workshops, and conferences that I attended to. Some of these events have been incredibly influential to my journey. I would further like to thank anonymous reviewers for their feedback on my submissions. Also, this thesis would not have existed without the courage of Edward Snowden.

Last but not least, I am immensely grateful to my family. I thank my husband for making everything in life so much brighter and better. I am also grateful to my dad, my mom, and my sister for their love and support. I am incredibly lucky to have you all.

Abstract

The Snowden revelations of 2013 have shed some light on the extent of state-performed mass surveillance programs that target people all over the world, violate their privacy, and endanger their cyber security. The presumably most expensive of these surveillance programs is the NSA's decryption program, Bullrun, which aims at breaking and sabotaging cryptosystems. This program cost $254.9 million in 2013 alone. Cryptosystems are vulnerable to sabotage in their mathematical specifications, standardization of their parameters, and their implementations. It has been a bitter surprise to realize that the capabilities of adversaries that have been classically considered in cryptographic models often do not even come close to what is attainable for big brother (i.e., state-level) adversaries. Therefore, it is of utmost necessity to rigorously study cryptographic sabotage and to develop resilient cryptosystems. Considering that the anticipated adversary is an extremely powerful one, finding solutions against sabotage requires not only tailoring the existing approaches from various areas in cryptography and other closely related disciplines to new scenarios but at times also entirely new design and proof techniques. As such, this thesis aims at adding new knowledge and techniques to the cryptographic toolbox to help better combat such attacks. In particular, we tackle the problem of disabling *backdoors embedded in the mathematical design* of cryptographic primitives as well as re-establishing security in their *subverted implementations.*

The first part of this thesis is concerned with defeating backdoors in *hash functions*, which are one of the most fundamental and versatile primitives in cryptography. We formulate and study backdoored hash functions, whereby a big brother designs a hash function that despite displaying reasonable functionality and security properties, can be broken using a secret backdoor. We start by modeling backdoored hash functions in the standard model (i.e., a model without idealized primitives), where the backdoor is a key co-designed with the hash function. We then show the feasibility of efficient backdoored hash functions for fixed-length inputs and how iterating such functions in the Merkle–Damgård or the sponge constructions leads to backdoored hash functions for inputs of arbitrary length, where crucial security guarantees break down. On the positive side, we give evidence that the *weak pseudorandomness* property of hash functions, which rely on a secret key, is in fact robust against backdooring. This result allows us to build a backdoor-resilient *iterative pseudorandom function*, more precisely, a variant of HMAC. Furthermore, we show how the *key derivation function* HKDF can be immunized against backdoors at little cost. Unfortunately our findings also suggest that immunizing a hash function against backdoors, without relying on a secret key, is presumably hard. This observation later motivated our study of combining *independent* hash functions as a possible strategy in building secure backdoor-resilient hash functions.

We then introduce a model which we call the *backdoored random-oracle* (BRO) model, whereby a big brother picks a random oracle, i.e., a random function, but he can also obtain *arbitrary information* about the random oracle using a *backdoor oracle*. This model captures not only weaknesses that lead to collision-finding and inversion attacks but also any conceivable weakness that can exist in a hash function. Therefore, adversaries equipped with such a backdoor oracle are powerful to the extent that no security can be achieved based on a single arbitrarily backdoored random oracle. However, when two independent BROs are available, we show that certain security properties, such as one-wayness, pseudorandomness, and collision resistance can be re-established. This is true even when we allow *unrestricted and adaptive access to both backdoor oracles*. To this end we consider three common combiners: concatenation, cascade, and xor. At the core of our results lie new reductions from cryptographic security goals to the *communication complexities* of several two-party tasks. Along the way we establish a communication complexity lower bound for set-intersection for cryptographically relevant ranges of parameters and distributions and where deciding set-disjointness can be easy.

We further study the technique of combining independent BROs in order to construct a hash function from two or more BROs in a way that it can be used in many cryptographic applications that rely on a backdoor-free random oracle. The property that practically allows a hash function construction to replace a random oracle is referred to as *indifferentiability* and was introduced by Maurer, Renner, and Holenstein (TCC 2004). Achieving full indifferentiability in our model seems very challenging at the moment. We however make progress in this direction by showing that the xor combiner goes well beyond security against preprocessing attacks and offers indifferentiability *as long as the number of the adversary's query switches between the backdoor oracles remains logarithmic* in the input size of the underlying BROs. We also show that an extractor-based combiner of three BROs can provide indifferentiability even against adversaries that make a linear number of switches. To prove these results we build on and refine a recent technique by Göös, Lovett, Meka, Watson, and Zuckerman (STOC 2015) for *decomposing high-entropy distributions* into convex combinations of distributions on bit strings that are fixed on some points and highly unpredictable on others. Furthermore, a natural restriction of our definition of indifferentiability in the BRO model gives rise to a notion of *indifferentiability with auxiliary input*, for which we give two positive feasibility results.

The second part of this thesis aims at providing security in face of malicious implementations. We put forward the notion of *self-guarding cryptographic primitives* as a countermeasure to a subclass of so-called *algorithm substitution attacks* (ASAs). These attacks are formalized by Bellare, Paterson, and Rogaway (CRYPTO 2014) as attacks, where a big brother secretly substitutes the genuine implementation of a cryptosystem with a malicious one in order to undermine users' security. The authors also show that randomized symmetric encryption schemes are vulnerable to devastating ASAs that practically allow a big brother to steal secret keys, while to users the input-output behavior of the encryption algorithm remains undetectable from that of a genuine implementation. Detecting ASAs, even if theoretically possible, is unfortunately not an easy task. Our self-guarding primitives, however, do not rely on detection and can still prevent undesirable leakage by subverted algorithms, usually for a bounded time, if one has the guarantee that the system has been *working properly during an initial phase*. This secure initial phase is justified for instance, before a malicious software update is performed or before a malicious internal state is reached. We present constructions of basic self-guarding primitives for symmetric and asymmetric encryption and for signatures. We also argue that the model captures attacks with malicious hardware tokens and show how to self-guard a key exchange protocol that is based on a physical uncloneable function (PUF).

Zusammenfassung

Die Snowden-Enthüllungen aus dem Jahr 2013 gaben Aufschluss über das Ausmaß staatlich durchgeführter Massenüberwachungsprogramme, die Menschen auf der ganzen Welt ausspionieren, ihre Privatsphäre verletzen und ihre Cyber-Sicherheit gefährden. Das vermutlich teuerste dieser Programme, das Entschlüsselungsprogramm Bullrun der NSA, zielt darauf ab, Kryptosysteme zu brechen und sogar zu sabotieren. Allein im Jahr 2013 kostete Bullrun 254,9 Millionen Dollar. Kryptosysteme sind anfällig für Sabotage in ihren mathematischen Spezifikationen, der Standardisierung ihrer Parameter und ihrer Implementierungen. Es war eine bittere Überraschung zu realisieren, dass die Fähigkeiten von Angreifern, die in klassischen kryptographischen Modellen betrachtet werden, oft nicht annähernd an das herankommen, was für einen Big Brother, d.h. einen Angreifer auf Staatsebene, machbar ist. Eine gründliche Untersuchung der kryptographischen Sabotage und die Entwicklung resistenter Kryptosysteme sind deshalb von größter Notwendigkeit. Dadurch, dass der von uns berücksichtigte Angreifer äußerst mächtig ist, erfordern Lösungen in solchen Szenarien nicht nur eine Anpassung der bestehenden Ansätze aus diversen Bereichen der Kryptographie und anderen eng verwandten Disziplinen an neue Szenarien, sondern manchmal auch völlig neue Design- und Beweistechniken. Daher setzt sich diese Dissertation als Ziel, neue Erkenntnisse und Techniken zum kryptographischen Werkzeugkasten beizutragen, damit man solche Angriffe besser bekämpfen kann. Wir befassen uns insbesondere mit der Deaktivierung von *Hintertüren im mathematischen Design* kryptographischer Primitiven sowie der Wiederherstellung der Sicherheit in ihren *unterwanderten Implementierungen.*

Der erste Teil dieser Dissertation beschäftigt sich mit dem Vernichten von Hintertüren in *Hashfunktionen*, die eine der grundlegendsten und vielseitigsten Primitiven in der Kryptographie sind. Wir formulieren und untersuchen Hashfunktionen mit eingebetteten Hintertüren, wobei ein Big Brother eine Hashfunktion entwirft, die zwar vernünftige Funktionalität und Sicherheitseigenschaften vorzeigt, aber mit einer geheimen Hintertür gebrochen werden kann. Wir beginnen mit der Modellierung von Hashfunktionen mit Hintertüren im Standardmodell (d.h. ein Modell ohne idealisierte Primitiven), wobei die Hintertür ein mit der Hashfunktion entworfener Schlüssel ist. Wir zeigen die Realisierbarkeit effizienter Hashfunktionen mit fester Eingabelänge und eingebetteten Hintertüren und wie das Iterieren dieser Funktionen durch die Merkle-Damgård- oder die Sponge-Kontruktionen zu Hashfunktionen mit beliebiger Eingabelänge und Hintertüren führt, bei denen wesentliche Sicherheitsgarantien versagen. Auf der positiven Seite zeigen wir, dass die *schwache Pseudozufälligkeitseigenschaft* der Hashfunktionen, die einen geheimen Schlüssel verwenden, in der Tat robust gegen Hintertüren ist. Dieses Resultat erlaubt es uns, eine hintertürresistente *iterative Pseudozufallsfunktion* zu bauen, genauer gesagt eine Variante von HMAC. Wir zeigen auch, wie die *Schlüsselableitungsfunktion*, HKDF, mit geringem Aufwand gegen Hintertüren immunisiert werden kann. Leider deuten unsere Resultate auch darauf hin, dass das Immunisieren einer Hashfunktion, ohne sich dabei auf einen

geheimen Schlüssel zu verlassen, vermutlich schwierig ist. Diese Beobachtung motivierte später unsere Idee zum Kombinieren von *unabhängigen* Hashfunktionen als eine Strategie für die Konstruktion sicherer und hintertürresistenter Hashfunktionen.

Wir führen anschließend ein Modell ein, das wir *Backdoored-Random-Oracle* (BRO) nennen, auf Deutsch Zufallsorakel mit Hintertür, wobei ein Big Brother eine zufällige Hashfunktion, d.h. ein Zufallsorakel, auswählt, aber dennoch *beliebige Funktionen* seiner Tabelle mit Hilfe eines *Hintertürorakels* sehen kann. Dieses Modell erfasst zusätzlich zu den gewöhnlichen Angriffen wie Kollisionsfindung oder Invertieren auch jegliche denkbare Schwäche, die eine Hashfunktion haben kann. Ausgestattet mit einem solchen Hintertürorakel sind Angreifer daher so mächtig, dass keine auf einem einzelnen BRO mit allmächtigem Hintertürorakel basierende Sicherheit erreichbar ist. Wir zeigen jedoch, dass bestimmte Sicherheitseigenschaften wie One-Way-Sicherheit, Pseudozufälligkeit und Kollisionsresistenz gerettet werden können, wenn zwei unabhängige BROs zur Verfügung stehen. Dies gilt, auch bei *uneingeschränktem und adaptivem Zugriff auf beide Hintertürorakel.* Zu diesem Zweck berücksichtigen wir drei weitverbreitete Kombinierer: Konkatenation, Kaskade und XOR. Im Mittelpunkt unserer Resultate stehen neue Reduktionen von kryptographischen Sicherheitszielen auf die *Kommunikationskomplexitäten* verschiedener Zweiparteienaufgaben. Dabei etablieren wir auch eine neue Unterschranke für die Kommunikationskomplexität von Schnittmengenberechnung für kryptographisch relevante Bereiche von Parametern und Verteilungen, wobei das Problem der Mengendisjunktheit einfach sein kann.

Wir untersuchen die Technik der Kombination unabhängiger BROs weiter, um eine Hashfunktion aus zwei oder mehr BROs zu bilden, so dass diese Konstruktion in vielen kryptographischen Anwendungen verwendet werden kann, die auf ein hintertürfreies Zufallsorakel setzen. Die Eigenschaft, die es einer Hashfunktionskonstruktion praktisch erlaubt, ein Zufallsorakel zu ersetzen wird als *Undifferenzierbarkeit* bezeichnet und wurde von Maurer, Renner und Holenstein (TCC 2004) eingeführt. Das Erreichen vollständiger Undifferenzierbarkeit in unserem Modell scheint im Moment sehr herausfordernd zu sein. Wir machen jedoch Fortschritte in dieser Richtung, indem wir zeigen, dass der XOR-Kombinierer weit über die Sicherheit gegen Vorverarbeitungsangriffe hinausgeht und Undifferenzierbarkeit bietet, *solange die Anzahl der Abfragewechsel des Angreifers zwischen den Hintertürorakeln logarithmisch in der Eingabelänge der einzelnen BROs bleibt.* Wir zeigen auch, dass ein auf einem Zufallsextrahierer basierender Kombinierer für drei BROs kann Undifferenzierbarkeit sogar gegen Angreifer mit einer linearen Anzahl von Abfragewechsel anbieten. Um diese Behauptungen zu beweisen, verwenden und verfeinern eine neue Technik von Göös, Lovett, Meka, Watson und Zuckerman (STOC 2015) für die *Zerlegung von Verteilungen mit hoher Entropie* in konvexe Kombinationen von Verteilungen über Bitstrings, die in einigen Punkten fixiert und in den restlichen Punkten sehr unvorhersehbar sind. Eine natürliche Einschränkung unserer Definition von Undifferenzierbarkeit im BRO-Modell führt außerdem zu einer Definition von *Undifferenzierbarkeit mit Hilfseingabe*, für die wir zwei positive Machbarkeitsergebnisse liefern.

Der zweite Teil dieser Dissertation befasst sich damit, Sicherheit in der Gegenwart von unterwanderten Implementierungen zu bieten. Wir stellen die Idee des *Selfguarding der kryptographischen Primitiven* als eine Gegenmaßnahme zu einer Unterklasse sogenannter *Algorithm-Substitution-Attacks* (ASAs), auf Deutsch Algorithmensubstitutionsangriffe, vor. Solche Angriffe werden von Bellare, Paterson und Rogaway (CRYPTO 2014) als Angriffe formalisiert, bei denen ein Big Brother heimlich die sichere Implementierung eines Kryptosystems durch ein Bösartiges ersetzt, um die Sicherheit der

Nutzer zu unterminieren. Die Authoren zeigen darüber hinaus, dass randomisierte symmetrische Verschlüsselungsverfahren anfällig für solch verheerende ASAs sind, die es einem Big Brother praktisch ermöglichen, geheime Schlüssel zu extrahieren. Dabei bleibt das Ein-Ausgabeverhalten des Verschlüsselungsalgorithmus ununterscheidbar von dem einer aufrichtigen und sicheren Implementierung. Die Erkennung von ASAs, auch wenn theoretisch möglich, ist leider keine leichte Aufgabe. Unsere Selfguarding-Primitiven sind nicht auf Erkennbarkeit angewiesen und können trotzdem unerwünschte Informationsleck durch unterwanderte Algorithmen unterbinden, meist für eine begrenzte Zeit, wenn man die Garantie hat, dass *das System während einer Anfangsphase richtig funktioniert hat.* Diese sichere Anfangsphase ist beispielsweise gerechtfertigt, bevor eine böswillige Software-Update durchgeführt wird oder ein bösartiger interner Zustand erreicht wird. Wir präsentieren Konstruktionen für symmetrische und asymmetrische Verschlüsselung und für digitale Signaturen. Wir argumentieren auch, dass das Modell Angriffe mit bösartigen Hardware-Tokens erfasst und zeigen, wie ein Schlüsselaustauschprotokoll, das auf einer Physical-Unclonable-Function (PUF) basiert, Selfguarding werden kann.

Short Contents

Contents

Primary Abbreviations

AI	auxiliary input
ASA	algorithm substitution attack
BND	Bundesnachrichtendienst
BPRG	backdoored pseudorandom generator
BRO	backdoored random oracle
CIA	Central Intelligence Agency
CR	collision resistance
DISJ	set-disjointness
DPT	deterministic polynomial time
DRBG	deterministic random bit generator
EC	elliptic curve
EUF-CMA	existential unforgeability against chosen-message attack
FBI	Federal Bureau of Investigation
FDH	full domain hash
GF	Galois field
GGM	generic group model
GOST	(Russian: ГОСТ) GOsudarstvennyy STandart
HKDF	hash-based key derivation function
HMAC	hash-based message authentication code
IACR	International Association for Cryptologic Research
IETF	Internet Engineering Task Force
iff	if and only if

IND-CCA	indistinguishability against chosen-ciphertext attack
IND-CPA	indistinguishability against chosen-plaintext attack
INT	set-intersection
IPSec	Internet protocol security
IU	image uniformity
IV	initialization vector
KDF	key derivation function
MAC	message authentication code
MD	Merkle–Damgård
NIST	National Institute of Standards and Technology
NMAC	nested message authentication code
NOBUS	nobody but us
NSA	National Security Agency
NUMS	nothing up my sleeves
OAEP	optimal asymmetric encryption padding
oPRG	oblivious pseudorandom generator
OT	oblivious transfer
OW	one-wayness
PPT	probabilistic polynomial time
PRF	pseudorandom function
PRG	pseudorandom generator
PSK	pre-shared key
PSS	probabilistic signature scheme
PUF	physical uncloneable functions
RC	randomized cascade
resp	respectively
RFC	request for comments
RO	random oracle

rPre	random preimage-resistance
RSA	Rivest–Shamir–Adleman
SHA	secure hash algorithms
SKDF	salted key derivation function
SPR	second-preimage resistance
SSL	secure sockets layer
TDF	trapdoor function
TDP	trapdoor permutation
TLS	transport layer security
VPN	virtual private network
wPRF	weak pseudorandom function

Chapter 1

Introduction

Cryptography is about designing and analyzing systems that enable certain tasks, such as communication, *in presence of adversaries.* Assessing the security of a cryptosystem crucially relies on precise and rigorous *models* and profound *assumptions.* Meaningful models give an accurate description of reality, while still abstracting from less relevant details, in order to be applicable to conceptually similar scenarios. To demonstrate our confidence in the security of a cryptosystem, we need to exclude the possibility that adversaries with realistic capabilities can succeed in breaking the desired security goals in the considered model. This is done by means of mathematical *proofs.* If the assumptions that we rely on happen not to hold, or if the considered model is inadequate for the intended application, the practical consequences can be devastating.

A particularly dangerous threat which has been unjustly neglected by our community is that of *big brother* (i.e., state-level) adversaries who perform mass surveillance [Rog15]. It is generally illusive to assume that this type of adversaries is already captured in traditional security models as adversaries that simply have more computing power than classical ones, and that it suffices to increase the security level, e.g., key sizes, to protect against them. This assumption falsely suggests that big brothers operate more or less passively and overlooks the possibility of an *active* involvement during the design, implementation, and standardization of software and hardware. Below we discuss several historic examples which show that such an assumption can be very problematic.

Attempts at embedding government-mandated backdoors in everyday cryptosystems intensified in the 1990s with the NSA's Clipper chip: a backdoored encryption device which was going to be included in every cell phone. We emphasize that regardless of the term used to describe backdoors, e.g., key escrow, golden keys, or front doors, they introduce a vulnerability in the system and are inherently in conflict with security, not only against the big brother responsible for the backdoor but also against other adversaries. Luckily, Clipper chip was defunct by 1996. However, sabotaging cryptography in various shapes and forms has been an ongoing operation. Tech-companies are pressured to embed backdoors in their products as was showcased, starting February 2016, in the case of the FBI demanding that Apple enables unlocking of iPhones [Con20]. Furthermore, the Swiss company Crypto AG, which was secretly co-owned by the CIA and the BND from 1970 to 1993 and then solely by the CIA until its liquidation in 2018, was selling backdoored products to many governments all around the world [Mil20]. Moreover, the specification of cryptosystems and their standardization process are also vulnerable to manipulation, as the prominent case of the NIST-standardized Dual_EC_DRBG pseudorandom generator indicates: the entity picking the elliptic curve parameters used in Dual_EC_DRBG can not only distinguish its outputs from random but also predict future outputs. Although Shumow and Ferguson [SF07] warned about this issue already in 2007, compelling evidence that Dual_EC_DRBG was indeed backdoored by the NSA as part of their Bullrun program was given about six years later in 2013 by some of the Snowden revelations [PLS13, Men13, BLN16].

More generally, the Snowden revelations exposed several ongoing surveillance programs that undermine security and privacy of foreigners as well as citizens [Arc13]. These programs include, among others, bulk collection of communication contents and meta data obtained from undersea fiber-optic cables or from Internet companies, as well as inserting vulnerabilities and embedding backdoors in computer systems and standards [BBG13, MBH^{+}13, PLS13, Men13, GP13, Gre14]. The NSA program that aims at breaking and even sabotaging cryptography is called Bullrun, which with a cost of $254.9 million in 2013 alone is believed to be by far the most expensive [BBG13].

Making mass surveillance difficult to perform is not merely a task for cryptographers. It rather requires sincere effort in various other disciplines, from computer science in general to sociology and law. Nonetheless, cryptography is a necessary part of the solution. In a 2014 statement, the International Association for Cryptologic Research (IACR) called for "expediting research and deployment of effective techniques to protect personal privacy against governmental and corporate overreach" [Res14]. Despite the Snowden revelations, still very little is known about intelligence agencies' concrete abilities and attacks on cryptography. Nonetheless, knowing that they can be and are often involved in the design, standardization, and implementation of cryptographic schemes increases the responsibility of cryptographers more than ever to design secure schemes while keeping such threats in mind. Motivated by such risks, this thesis aims at developing cryptographic primitives that resist attacks that manipulate *mathematical designs* and *concrete implementations.*

1.1 Sabotage-Resilient Cryptography

Recent revelations have drawn our community's attention more than ever before to the realness and the gravity of attacks that sabotage cryptography on a massive scale and the increasing importance of their rigorous treatment. The damage that attacks of such kind can effectively cause depends on several factors, e.g., the resources needed to undermine the system, how easy it is to detect or circumvent the weakness by users or for other attackers to exploit it, whether the attack can be plausibly denied, who is affected by the attack, and so on. A comprehensive overview and discussions on this topic is given by Schneier et al. [SFKR15].

Our focus here is on deliberate but at the same time non-trivial classes of sabotage that enable a big brother adversary, who is aware of the secret sabotage mechanism, to easily break the system. Consequently we do not consider generally insecure schemes, such as export restrictions, which restricted US companies in the 1990s to only export weak cryptographic technologies, e.g., systems with a symmetric key size of less than 40 bits. However, we emphasize that attempts of any kind at weakening cryptography put users in unforeseen risks, even if the exploitation with little effort seems to have been made exclusive (by cryptographic means) to those aware of the backdooring strategy [AAB^{+}97]. In NSA's terminology such non-trivial vulnerabilities, which decrease the risk of detection or exploitation by others, are referred to as *nobody-but-us* (NOBUS) attacks [Buc17].

We consider two important types of NOBUS sabotage in cryptographic primitives: a) tampering with the publicly available *mathematical design* (a.k.a. specification) and b) tampering with the actual *implementation.* Throughout this thesis, we use the term *backdoor* to refer to a design-specific property that enables an adversary to compromise the security of a cryptosystem. We assume that the specification on its own is public and, therefore, a (NOBUS) backdoored scheme is hard to break without access to the backdoor. Furthermore, any implementation that truthfully follows the specification of a backdoored system also implements the backdoor strategy and is, hence, insecure.

We refer to attacks that target the implementation of cryptosystems as *algorithm substitution attacks* (ASAs), following the terminology of Bellare, Paterson, and Rogaway [BPR14]. Subverted implementations are assumed to be accessed in a black-box manner, i.e., users are not provided access to the source code. Contrary to backdoored primitives, access to the source code of a subverted algorithm may even make it immediately possible for anyone to exploit its vulnerability. Furthermore, subverted algorithms may keep a secret state, even if the specification of those algorithms do not. Such a state may be used to gradually leak secrets or delay the malicious behavior to a later point in time, e.g., to pass some initial security evaluation tests.

Further Related Work

In the realm of backdoored primitives, Dodis et al. [DGG$^+$15] formally studied backdoored pseudorandom generators (BPRGs), capturing the well-known Dual_EC_DRBG scheme, and showed that they can be immunized by applying a non-trivial function (e.g., a pseudorandom function or a seeded extractor) to their outputs. They also showed the equivalence of BPRGs and public-key encryption schemes. Their notion of BPRGs was extended by Degabriele et al. [DPSW16] in order to investigate backdoorability of forward-secure pseudorandom generators and robust pseudorandom generators with continuously refreshed states. Other notable works in the context of the Dual_EC_DRBG-related incidents include analyses of its usage in ScreenOS, which is the operating system of Juniper Networks' VPN routers [CMG$^+$16] as well as the practical exploitability of the TLS protocol when using Dual_EC_DRBG [CNE$^+$14] by Checkoway et al. Furthermore, Bernstein et al. [BCC$^+$15] analyzed the cost of breaking elliptic-curve cryptography by standardizing sabotaged or broken elliptic curves.

Albertini et al. [AAE$^+$14] investigated backdoorability of hash functions due to the designers' freedom in choosing the round constants. They illustrated the plausibility of malicious hash function designs by providing a malicious version of SHA-1, under which two colliding messages can be found with an approximate complexity of 2^{48} calls, while adaptively choosing the round constants for this SHA-1 variant. In comparison, the complexity of finding collisions for the standard SHA-1 is believed to be over 2^{63} calls to the hash function. Aumasson [Aum11] also presented a backdoored version of BLAKE (SHA-3 competition's finalist) where the designer adaptively modifies the operators used in the finalization function in order to find a colliding pair of messages. Furthermore, AlTawy and Youssef [AY15] gave a backdoored version of Streebog which is a GOST-standardized hash function, and Morawiecki [Mor15] studied a malicious variant of Keccak (the winner of the SHA-3 competition). Both papers modify the round constants and generate collisions using differential cryptanalysis.

The study of tampering with the implementation of cryptosystems with the intent of stealing secret information was already initiated over two decades ago by Young and Yung in a line of work referred to as *kleptography*, using cryptography against cryptography [YY96, YY97a, YY97b, YY04, YY06]. Kleptography expands on subliminal channels, introduced by Simmons, which are used to communicate secret messages covertly within other channels [Sim83]. In a kleptographic setting, an implementation is subverted in a way that it leaks some secret information, e.g., users' secret keys, to the adversary. At the same time, the subverted implementation may even be undetectable to users with black-box access to its algorithms. Such attacks are of added concern when using closed-source software.

Motivated by the Snowden revelations, Bellare, Paterson, and Rogaway [BPR14] formalized algorithm substitution attacks (ASAs), which are undetectable kleptographic attacks, whereby a big brother subverts implementations of cryptosystems with the goal of performing mass surveillance. They showed that randomized symmetric encryption schemes are highly vulnerable to such attacks. The subverting adversary can exfiltrate secret keys from ciphertexts, while the input-output behavior of the subverted encryption algorithm looks completely innocent to users. Several recent work further investigated such attacks [DFP15, BJK15, AP19b, AP19a] and suggested countermeasures against them [MS15, AMV15, DMS16, BKR16, RTYZ16, CMY$^+$16, RTYZ17, AFMV19, HPRV19]. Subversion of public parameters (e.g., prime numbers and elliptic curves) has also been considered by Auerbach, Bellare, and Kiltz [ABK18] in the context of public-key encryption and key encapsulation mechanisms.

Proposed countermeasures against ASAs often rely more or less on the ability to detect whether implementations adhere to the mathematical specifications they are supposed to follow. A noteworthy exception is the preventive approach of *reverse firewalls*, introduced by Mironov and Stephens-Davidowitz [MS15], where a user's outgoing communication is routed through an untrusted firewall which may take further cryptographic steps, in order to prohibit leakage of secret information. Furthermore, by considering a restricted type of subversion, Russel et al. [RTYZ18] showed that so-called subverted random oracles (i.e., random functions) that are subverted on a negligible portion of their outputs, can be transformed into functions which can effectively replace random oracles. Their solution is based on the domain-efficient salting developed by Coretti et al. [CDGS18], which is in turn based on a technique by Maurer [Mau92].

Another important capability of adversaries, especially big brothers, that should not be underestimated, is the ability to perform off-line preprocessing computations on public schemes and parameters in order to considerably expedite on-line attacks. This line of research includes classical inversion attacks with consideration of memory-time tradeoff by Hellman [Hel80] and Fiat and Naor [FN91], as well as a more recent line of work by Unruh [Unr07], Dodis, Guo, and Katz [DGK17], Coretti et al. [CDGS18, CDG18], and Corrigan-Gibbs and Kogan [CK18], which gives security proofs of primitives in ideal models with auxiliary input. Furthermore, motivated by the recent Logjam attack on TLS [ABD$^+$15], Auerbach, Giacon, and Kiltz [AGK20] introduce a framework to assess how attacks on public-key encryption schemes scale when performed on masses.

1.2 Contributions and Structure of this Thesis

Despite an increased attention to the topic of sabotage-resilient cryptography in the last few years, there are still various directions to explore and primitives to study in this new setting. This thesis strives after cryptographic primitives that resist backdoors in their specification as well as subversion of their implementations. To this end we present models that accurately reflect these settings and search out assumptions that are reasonable to make. In order to advance our understanding of sabotage in cryptography, we also explore some attacks. As countermeasures, we construct several primitives that provably resist backdooring or subversion attacks. Modern cryptography already presents an extensive and impressive set of tools and techniques that can be applied in analyzing or providing security in numerous new settings. Nonetheless, it should not come as a surprise that achieving security against incredibly powerful state-level adversaries is not an easy task. We were challenged to not only rethink and refine existing design and proof techniques, but at times also to come up with entirely new ideas.

This thesis is organized into two parts based on the type of sabotage. The first part is concerned with backdoors in cryptographic hash functions and consists of Chapters 3 to 7. The second part is concerned with subverted implementations and consists of Chapter 8. We further include basic notations and preliminaries in Chapter 2 and conclude with open problems in Chapter 9. The results presented in this thesis are the product of inspiring collaborations with several incredible individuals: Balthazar Bauer, Yevgeniy Dodis, Pooya Farshim, Marc Fischlin, Christian Janson, and Stefano Tessaro. As research is a highly collaborative task, it is nearly impossible to give a detailed account of each author's contributions. For the purpose of this thesis, however, I will break down the individual scientific contributions to the best of my memory and include them in the beginning of their corresponding chapters.

While it is incredibly important to put effort in detecting potential backdoors and ASAs, detection can be infeasible due to various reasons, including the complexity of designs and implementations, lack of trust in the process of generating parameters and constants, and lack of access to the source code (in case of ASAs). Instead, a high-level approach, which is present throughout this thesis, is to bootstrap security by means of simple extra operations and based on assumptions that do not rely on detection. In the first and main part of this thesis, we formally study backdoored hash functions, providing different strategies that disable their backdoors and re-establish crucial security properties. In the second part, we investigate the possibility of reviving security in subverted implementations of cryptographic primitives if we have the guarantee that a trustworthy implementation was available during a short setup phase. An overview of the most significant technical contributions of this thesis is given below.

Modeling Backdoored Hash Functions

Hash functions are one of the most essential primitives in cryptography. Security of many cryptographic tasks, such as digital signatures, pseudorandom generation, password protection, and blockchains crucially relies on the security of the underlying hash functions. We are interested in protecting hash functions against a variety of attacks that may arise due to built-in backdoors, cryptanalytic advances, or preprocessing attacks. We also remark that hash functions are very foundational and conceptually simple primitives and, hence, solutions for them are likely to be helpful in developing solutions that defeat similar attacks in other primitives. In Chapter 3 we model backdoored hash functions with the intuition that a big brother designs a hash function in a way that he has a considerable advantage in breaking it. We take two different approaches in modeling backdoored hash functions: *standard-model* backdoored hash functions (i.e., without any idealized primitives) and backdoored hash functions in a model, where hash functions without backdoors are *random oracles*, i.e., ideal hash functions. We then define meaningful variants of established security notions for hash functions, including one-wayness, pseudorandomness, second-preimage resistance, and collision resistance, in a setting with backdoors.

In our definition of standard-model backdoored hash functions, a *backdoor key* enables violating the security of a backdoored hash function, in the sense of one-wayness, second-preimage resistance, or collision resistance. Such a backdoor key is modeled as a short bit string co-designed with a family of backdoored hash functions. It can work either for some adversarially chosen hash function or for any hash function from the family.

We also introduce a model for the analysis of backdoored hash functions which substantially weakens the traditional random-oracle (RO) model. In our *backdoored random-oracle* (BRO) model, besides access to a random oracle (i.e., a random function, or a very well-behaved hash function) H, adversaries are provided with a *backdoor oracle* that can compute arbitrary leakage functions f of the function table of H. Thus an adversary would be able to invert arbitrary points, find collisions in hash outputs, test for membership in certain sets, and more. The BRO model allows for a parameterization of the class of backdoor functions that can be computed by the backdoor oracle. However we work with respect to the full set of functions. As we discuss shortly, the assumption that hash functions are ideal if no access to the backdoor oracle is given, allows us to define simple constructions and provide proofs of security for them without assuming restricted backdoor capabilities.

Understanding the Threat of Backdoored Hash Functions

In Chapter 4 we showcase the feasibility of efficient Merkle–Damgård-based as well as sponge-based backdoored hash functions by iterating a (fixed input-length) backdoored hash function, which we build based on trapdoor one-way functions, studied by Bellare et al. [BHSV98], where some images have an exponential number of preimages. We discuss concrete attacks on such hash functions, whereby an adversary can encode a backdoor key in inputs to the hash function in order to trigger a misbehavior and easily find preimages and collisions, while the description of the hash function does not reveal the backdoor key. Unfortunately we also show that although the HMAC construction of an iterative pseudorandom function by Krawczyk, Bellare, and Canetti [KBC97] enjoys a secret key, it is not automatically immune to such attacks. In other words, a potential backdoor in the hash function underlying HMAC can entirely compromise HMAC's pseudorandomness and security for the purpose of message authentication. Overall, protecting standard-model hash functions (without secret keys) against backdoors seems to be a hard task to achieve.

Defeating Backdoors in Pseudorandom and Key Derivation Functions

In Chapter 5 we raise the question of whether defeating backdoors in hash functions that have a *secret key* is possible in the standard model. We particularly focus on immunizing arbitrarily backdoored versions of HMAC by Krawczyk et al. [KBC97] as a pseudorandom function and the hash-based key derivation function HKDF by Krawczyk and Eronen [KE10], both of which are widely deployed in critical protocols such as TLS. Luckily we identify a property of common secret-keyed hash functions, which are not based on public-key primitives, that is robust against backdooring. This property is *weak pseudorandomness* which intuitively guarantees that outputs of a hash function, which uses a secret key, on random inputs look random. This positive result allows us to build a backdoor-resilient variant of HMAC using the randomized cascade approach by Maurer and Tessaro [MT08], and to show that HKDF can be immunized against backdoors at little cost, using an approach by Halevi and Krawczyk [HK06].

Simple Combiners for Backdoored Random Oracles

Backdoored random oracles make the task of bootstrapping cryptographic hardness somewhat challenging. Indeed, with only a single BRO, which allows for arbitrary queries to the backdoor oracle, no hardness can be found, as any construction can be broken in any reasonable sense. However, we show in Chapter 6 that when two (or more) independent hash functions are available, hardness can emerge. Combining hash functions is a classical approach to providing protection against failures of hash functions and re-establishing security as long as one of the underlying hash functions is secure [BB06, FL07, FLP14] or all hash functions have some known weaknesses, such as invertibility [HS08]. However, we aim at building backdoor-resilient hash functions by combining hash functions that have *arbitrary* adversarial weaknesses and where *no* secure hash function is on hand. We show in Chapter 6 that certain security properties, such as one-wayness, pseudorandomness, and collision resistance can be re-established by combining two independent BROs, even if the adversary has *unrestricted and adaptive access to both backdoor oracles.* To this end we consider three well-known combiners: concatenation, cascade, and xor. We give several positive results for these combiners depending on the size of their domains and co-domains.

Our security proofs rely on new reductions from cryptographic goals to the *communication complexities* of several two-party tasks. The communication complexity of a problem is roughly speaking the number of bits that are communicated by the best protocol that solves that problem. Two well-known problems in this area are *set-disjointness* and *set-intersection.* In the set-disjointness problem, two parties need to decide whether their sets intersect or not, whereas in the set-intersection problem, they need to find a common element. On a high level, we show that security of the aforementioned combiners as pseudorandom generators (PRGs) goes down to the hardness of solving set-disjointness, while one-way security goes down to the hardness of solving the set-intersection problem. We further define a variant of set-intersection, which we call multi-set double-intersection, and reduce collision resistance of combiners to its conjectural hardness.

Communication complexity lower bound for set-intersection. Worst-case lower bounds for the communication complexity of set-disjointness and set-intersection are quite well-studied. However, we are (as usually is the case in cryptography) interested in the *average-case* (a.k.a. distributional) hardness of solving these problems, i.e., hardness for a randomized choice of inputs from a certain, often uniform, distribution. In Section 6.4 of Chapter 6, we give communication complexity lower bounds for set-disjointness and set-intersection for cryptographically relevant ranges of parameters. While the result for set-disjointness is a generalization of known bounds [MB12, GC13], the one for set-intersection is, to the best of our knowledge, new. More precisely, we give a lower bound for the distributional communication complexity of set-intersection on input of two independent sets, where the elements in each set are chosen, independently of other elements, from a Bernoulli distribution. Furthermore, our lower bound for set-intersection holds for protocols with arbitrary error and also for parameters where the hardness of set-intersection is not implied by the hardness of set-disjointness, i.e., deciding set-disjointness can be easy.

Indifferentiability-Combiners for Backdoored Random Oracles

In Chapter 7 we further develop the technique of combining two or more independent BROs to make their backdoors ineffective in a variety of applications. More precisely, we study the question of combining BROs in the *indifferentiability* framework introduced by Maurer, Renner, and Holenstein [MRH04] in order to construct a hash function that can practically replace a conventional backdoor-free random oracle in numerous scenarios. Achieving full indifferentiability seems to be very challenging in this model. We, however, make progress in this direction by showing that the xor combiner goes well beyond security against preprocessing attacks and offers indifferentiability as long as the number of times where the adversary's backdoor queries switch back and forth between the backdoor oracles remains logarithmic in the input size of the underlying BROs. We also show that an extractor-based combiner of three BROs achieves indifferentiability with better bounds and even against adversaries that make a linear number of switches between the backdoor oracles. To prove these results we build on a technique by Göös et al. [GLM$^+$15] for *decomposing distributions* with high min-entropy into convex combinations of distributions with more structure, where some blocks (i.e., substrings inside bit strings taken from that distribution) are fixed, while any mix of the remaining blocks still maintains a high min-entropy, i.e., the rest is highly unpredictable. We refine this technique such that it can be applied in a more involved setting, where we can not only decompose distributions of random oracles after responding to one backdoor query but also after *adaptive* backdoor queries on a function taken from a decomposed distribution.

Indifferentiability with auxiliary input. The BRO model extends the auxiliary-input random-oracle model studied by Unruh [Unr07], Dodis et al. [DGK17], and Coretti et al. [CDGS18]. It can, therefore, model arbitrary *preprocessing* attacks (a.k.a. non-uniform attacks) as any auxiliary information about the hash function can be computed via a one-time oracle access to the backdoor oracle at the onset. We define this natural restriction of our definition of indifferentiability in the BRO model and give rise to a notion of *indifferentiability with auxiliary input*. We give two positive feasibility results for this notion in Section 7.5 of Chapter 7, one based on salting (i.e., using a public random string as part of the input to the hash function) and the other based on combiners. We emphasize, however, that salting does not help in the BRO model, since the adversary has permanent access (in particular also after becoming aware of the chosen salt) to the backdoor oracle.

Self-Guarding Primitives against Subverted Implementations

The second part of this thesis, consisting of Chapter 8, is concerned with providing protection against a class of algorithm substitution attacks, where the cryptosystem at hand runs properly and securely during a short initial period. At a later point in time, however, the implementation starts to behave maliciously. This can happen for instance, due to a malicious software update or triggered when reaching a malicious internal state or receiving a malicious input. We introduce the concept of *self-guarding cryptographic primitives* that can resist such attacks by relying on an initial secure phase to immunize their future executions. In constructing self-guarding primitives, it is important to rely on simple operations and not to trivialize the protection mechanism by implementing the scheme under attack from scratch.

Self-guarding primitives proactively thwart attacks without the need to detect them first, and unlike reverse firewalls they do not assume an on-line external party. On the downside, the number of secure executions of self-guarded schemes is usually limited if the subverted scheme maintains an internal state. We present constructions of basic primitives for public-key (i.e., asymmetric) and private-key (i.e. symmetric) encryption and for digital signatures. We also argue that our model captures attacks with malicious hardware tokens and show how to self-guard a key exchange protocol that employs a physical uncloneable function (PUF) against attacks that subvert the PUF in transmission.

Chapter 2

Preliminaries

This chapter contains basic notations and definitions as well as known results that are either directly used in following chapters or are overall helpful in understanding this thesis.

2.1 General Notation

Strings. We let $\mathbb{N}$ denote the set of non-negative integers. We denote the set of bit strings of length $n \in \mathbb{N}$ by $\{0,1\}^n$ and the set of bit strings of length at most n by $\{0,1\}^{\leq n}$. The set of bit strings of arbitrary length is denoted by $\{0,1\}^*$. Special symbols $\epsilon_0 \in \{0,1\}^*$ and $\perp \notin \{0,1\}^*$ are chosen to indicate the empty string and an error, respectively. By $(\{0,1\}^n)^+$ we denote the set of bit strings with a length that is a non-zero multiple of n. We let $[n]$ denote the set $\{0,\ldots,n-1\}$. When we write $[N]$ for any uppercase letter N, we use the convention that N is an integer that is a power of two, i.e., $N = 2^n$ for some $n \in \mathbb{N}$. The length of a bit string $s \in \{0,1\}^*$ is denoted by $|s|$ and the concatenation of two bit strings s_1 and s_2 by $s_1 \| s_2$. Let $s \in \{0,1\}^*$ be a non-empty bit string and $i, j \in \mathbb{N}$ be integers with $0 \leq i \leq |s| - 1$ and $i \leq j$. Then by $s_{[i,j]}$ we denote the substring of s (from left to right) starting from the i-th bit and ending with the j-th bit, where the first index of a string, i.e., $i = 0$, corresponds to the position of its most significant bit.

Functions. By $\mathrm{Fun}[n,m]$ we denote the set of all functions $f : \{0,1\}^n \to \{0,1\}^m$. We denote by $\mathsf{dom}(f)$ a function which returns the domain of a function f. We sometimes write $\mathrm{Fun}[\mathsf{dom}(f), m]$ to denote the set of all functions $g : \mathsf{dom}(f) \to \{0,1\}^m$. Let log denote the binary logarithm function. In fact all logarithms used in this thesis are to the base 2. We use $[M]^N$ to denote the set of all bit strings of length $N \cdot \log M$, which we often use as a compact representation of the set of all functions $f : [N] \to [M]$, i.e., $\mathrm{Fun}[\log N, \log M]$. In such a representation, we see the x-th $\log M$-bit block of a bit string $f \in [M]^N$ as the image of x under the corresponding function, i.e., with a slight abuse of notation $f(x)$. Let $f \in [M]^N$ and $P \subseteq [N]$. By $f(P)$ we denote the set of images of all $x \in P$ under f. Furthermore, by $\mathsf{img}(f) := f([N])$ we denote the set of all possible images under f. A function $p \in [N]^N$ with $\mathsf{img}(p) = [N]$ is called a permutation. We use f^- to denote the inverse of f such that for all $x \in [N]$ and $y \in [M]$ with $f(x) = y$ we have $x \in f^-(y)$. In other words, $f^-(y)$ is a set that contains all preimages of y under f. For a set $I \subseteq [N]$ we denote by $f_I \in [M]^{|I|}$ a string which constitutes the projection of f onto the points in I. In particular, for $x \in [N]$, $f_{\{x\}}$ can be understood as the image of x under f. Finally, we sometimes use $e^{-x} := \lim_{n\to\infty}(1 - x/n)^n$, where e is the Euler's number.

Data structures. Let $A \subseteq \{(a,b) \mid (a,b) \in [N] \times [M]\}$ be a set of assignments. We let $A_{.1} \subseteq [N]$ (resp. $A_{.2} \subseteq [M]$) denote the set containing the first (resp. second) coordinates of all elements in A. Furthermore, a queue Q is an abstract collection of ordered elements. A new element x can be added to the queue using an enqueue function $\mathsf{enq}(Q, x)$, and the oldest element in the queue can be accessed and at the same time removed from it using a dequeue function $x \leftarrow \mathsf{deq}(Q)$. This makes queues a first-in-first-out collection. Using the predicate $\mathsf{is\text{-}empty}(Q) \in \{0,1\}$ we can check whether the queue is empty. We denote an empty queue by $\square$.

Probability distributions. We denote the uniform distribution over an arbitrary finite set S by $\mathcal{U}_S$. By $s \twoheadleftarrow S$ (which is a simplification of $s \twoheadleftarrow \mathcal{U}_S$) we denote the uniform sampling from the set S. The Bernoulli distribution $\mathrm{Ber}(p)$ takes value 1 with probability p and 0 with probability $1-p$, where $0 \leq p \leq 1$. The binomial random variable $\mathrm{Bin}(n,p)$ constitutes a sequence of n independent $\mathrm{Ber}(p)$ samples. Let μ be a probability density function over the domain $[M]^N$, in other words over functions from $[N] \to [M]$. We write $F \sim \mu$ to represent F as the corresponding random variable to μ. Let $g \in [M]^N$. Then, we write $\mu(g) := \Pr_{f \twoheadleftarrow \mu}[f = g]$ as the probability that g is hit when sampling from μ. Let $D \subseteq [M]^N$. Then we define $\mu(D) := \Pr_{f \twoheadleftarrow \mu}[f \in D]$ as the probability that a sample drawn from μ falls into the domain D. By $\mu|_D$ we denote the density μ conditioned on the fact that samples f fall into D. For a function $g : [M]^N \to \{0,1\}^\ell$ and a string $z \in \{0,1\}^\ell$, by $\mu|_{g(\cdot)=z}$ we denote μ conditioned on the fact that for any sample f we have that $g(f) = z$. Recall that for a function $f \in [M]^N$ and a set $I \subseteq [N]$ we denote by f_I the projection of f onto the points in I. Similarly, for a random variable F over $[M]^N$ we denote by F_I a random variable resulting from the projection of all its values onto I. For a set of assignments $A \subseteq \{(a,b) \mid (a,b) \in [N] \times [M]\}$, by $\mu|_A$ we denote μ conditioned on $f_{\{a\}} = b$ for all $(a,b) \in A$ and $f \twoheadleftarrow \mu$. A convex combination of distributions $\mu_1, \ldots, \mu_n$ is a distribution that can be written as $\alpha_1 \cdot \mu_1 + \ldots + \alpha_n \cdot \mu_n$, where $\alpha_1, \ldots, \alpha_n$ are non-negative real numbers that sum up to 1.

Random variables and entropy metrics. Let X and Y be two random variables over the domains D_X and D_Y, respectively. We let $\mathrm{supp}(X)$ denote the support of X, i.e., the set of values that have a non-zero probability of happening. We write $X = x$ to denote the event that a value sampled from X is equal to x. The Shannon entropy of X is defined by

$$\mathbf{H}(X) := -\sum_{x \in D_X} \Pr[X = x] \cdot \log \Pr[X = x] \ .$$

The mutual information between X and Y is defined by

$$\mathbf{I}(X;Y) := \sum_{x \in D_X} \sum_{y \in D_Y} \Pr[X = x \wedge Y = y] \cdot \log \frac{\Pr[X = x \wedge Y = y]}{\Pr[X = x]\Pr[Y = y]} \ .$$

It holds that $\mathbf{I}(X;Y) = \mathbf{I}(Y;X) = \mathbf{H}(X) - \mathbf{H}(X|Y) = \mathbf{H}(X) + \mathbf{H}(Y) - \mathbf{H}(X,Y)$, where conditional entropy $\mathbf{H}(X|Y)$ and joint entropy $\mathbf{H}(X,Y)$ of X and Y are defined as expected.
The min-entropy $\mathbf{H}_\infty(X)$ of X is defined as

$$\mathbf{H}_\infty(X) := -\log \max_{x \in D_X} \Pr[X = x]$$

and can be understood as a measure of unpredictability, which makes it a very useful metric in cryptography.

Algorithms. For a natural number $n \in \mathbb{N}$ we denote its unary representation by 1^n. Two of the most common mathematical models of computation are *Turing machines* and *circuits*. For the purpose of this thesis it suffices to think of algorithms as Turing machines. We use pseudocode to describe algorithms, using notations and conventions that are common to well-known programming languages, e.g., keywords such as `return`, `if`, `else`, and `for`. Regarding the efficiency of algorithms, we use PPT to denote probabilistic polynomial-time and denote by $\mathsf{poly}(n)$ an unspecified polynomial in the security parameter 1^n, which is given as input (often implicitly for compactness) to algorithms. For a probabilistic algorithm A, the random variable $A(x)$ describes its output when run on input x, and we write $y \twoheadleftarrow A(x)$ for the sampling. Adversaries are usually modeled as probabilistic algorithms. We write $A^O(x)$ to denote the run of an algorithm on input x when given oracle access to an algorithm called O. This in particular means that A does not have to run O internally and O's time and space complexity is irrelevant for A. We use DPT to denote deterministic polynomial-time algorithms. For a deterministic algorithm we simply write $y \leftarrow A(x)$ to denote y as the output of the algorithm on x. Functions can be thought of DPT algorithms. We denote by $A[\mathit{param}](\mathit{input})$ a call of the algorithm A with (constant) parameters *param* and variable inputs *input*. This is to provide clarity, among multiple calls to the algorithm, about the main input, while the parameters remain unchanged.

2.2 Cryptographic Security

There are two main approaches to formalize security of cryptographic schemes: *game-based* and *simulation-based*. The former approach formalizes a security property as a game, where an adversary interacts with the cryptographic scheme in a specific way and with the goal of violating a desired security goal. If the adversary wins the game, the scheme is deemed broken. The latter approach declares a scheme secure if there exists an algorithm called the *simulator* which can, roughly speaking, simulate the scheme in a way that the adversary cannot distinguish between the real world and the simulated one. The simulated scheme does not know any secrets, nor does it show any weaknesses. In other words, it describes an ideal notion of security. Hence, if the real world and the simulated world are indistinguishable, the (real) scheme is secure. Throughout the thesis we almost always use game-based security notions, since they are arguably more natural and easier to work with. However, in Chapter 7, we use the notion of *indifferentiability* (see Section 2.4.1), which is simulation-based.

Advantage of adversaries and security proofs. Intuitively a cryptographic scheme is secure if the probability that any adversary (with some assumed capabilities, often PPT) breaks it is not significantly higher than the success probability of a naive strategy. This intuition is captured by the notion of *advantage* of adversaries: the smaller the advantage is, the less successful is the adversary. Consider for instance the security of *one-way functions*. The one-wayness game for a function $f \in \mathrm{Fun}[n, m]$ starts with picking an element from the domain $x \twoheadleftarrow \{0,1\}^n$ uniformly at random. Then the image $y \leftarrow f(x)$ under f is computed and the adversary is challenged to find some x' (which may or may not be different than the original x) which is a preimage of y, i.e., $f(x') = y$. Here, the advantage of the adversary is its probability of winning (i.e., finding a preimage), where the probability is over all random choices made, i.e., sampling x, as well as the internal coin tosses of the adversary. We denote the advantage of an adversary $\mathcal{A}$ in breaking a scheme Σ with respect to a security game called Game by $\mathsf{Adv}_{\Sigma}^{\mathrm{game}}(\mathcal{A})$.

We differentiate between two important types of security games: *unpredictability*, where the adversary has to predict a certain value (e.g., a preimage), and *indistinguishability*, where the adversary has to distinguish between two possible situations (e.g., pseudorandom vs. truly random or sampled from X vs. sampled from Y). Indistinguishability games are actually a specific type of unpredictability games, where just one bit is to be predicted. As a rule of thumb, the advantage in unpredictability games is usually defined as the probability of correct prediction, i.e., the probability of winning $\Pr[\mathcal{A} \text{ wins}]$. The advantage in indistinguishability games is usually defined as $2\Pr[\mathcal{A} \text{ wins}] - 1$ (alternatively, $\Pr[\mathcal{A} \text{ wins}] - \frac{1}{2}$), where $\frac{1}{2}$ is the probability of guessing a bit at random. Furthermore, the probability of an adversary winning in a game is formally described as the game outputting bit 1 after interacting with the adversary.

To bound the advantage of an adversary in breaking some security goal, we often need to rely on assumptions about certain problems being hard to solve. We then relate the security of our scheme to the hardness of the underlying problems by means of a *reduction*. In computational complexity theory, a reduction from a problem P to a problem Q is an algorithm $\mathcal{R}$ that can solve P using any algorithm that solves Q. This means that if solving P is hard for efficient algorithms and the reduction is efficient, then solving Q is hard. In cryptographic terminology, suppose that a scheme S is secure if an assumption A holds. Then, we can say that the security of S reduces to the hardness of A, or alternatively, (the problem of) breaking A reduces to (the problem of) breaking S.

Unproven assumptions. Security of many cryptosystems relies on unproven assumptions about the computational hardness of algebraic or number-theoretic problems. Hardness of factoring, discrete-logarithm, Diffie–Hellman, and finding the shortest vector in a lattice are some of the most prominent, well-studied, and unbroken problems that cryptography, in particular public-key cryptography, has been relying on for years. In contrast, information-theoretically secure cryptosystems do not rely on unproven assumptions about computational hardness of problems. Their security solely relies on proven information-theoretic results and the adversaries not having sufficient information to break the system.

Negligible functions. In asymptotic cryptography we usually say that the advantage of the adversary in breaking a secure scheme should be *negligible*. A function $\varepsilon : \mathbb{N} \to \mathbb{R}$ is called *negligible*, often denoted as $\varepsilon(n) \approx 0$, if for any polynomial $p : \mathbb{N} \to \mathbb{R}^+$, there exists some $N \in \mathbb{N}$ such that for all $n \geq N$ it holds that $\varepsilon(n) \leq \frac{1}{\mathsf{poly}(n)}$. In simple words, a negligible function vanishes faster than the inverse of any polynomial.

Weak security. Some constructions only achieve a weak security goal, where the advantage of adversaries can be bounded away from 1, i.e., despite not being negligible, it is also not *overwhelming*. We call the corresponding notions *weak*. However, note that weak pseudorandom functions (wPRF) are an exception to this rule and there weak refers to the weakened ability of the adversary in that evaluation of the wPRF on chosen messages is prohibited and only possible on random ones.

Indistinguishability. Two distributions X and Y over a common domain D are called computationally indistinguishable if no efficient algorithm can distinguish them with non-negligible probability.

More precisely, for any PPT adversary $\mathcal{A}$ it holds that the advantage

$$\mathsf{Adv}^{\mathrm{ind}}_{X,Y}(\mathcal{A}) := \left|\Pr[\mathcal{A}(z) = 1 \mid z \twoheadleftarrow X] - \Pr[\mathcal{A}(z) = 1 \mid z \twoheadleftarrow Y]\right|$$

is negligible. The probability is over $\mathcal{A}$'s coin tosses and the random choice of z. We say X and Y are statistically indistinguishable if their *statistical distance* is negligible. For two random variables X and Y over a common support D, their statistical distance is defined as

$$\mathsf{SD}(X,Y) := \frac{1}{2}\sum_{z\in D}\left|\Pr[X = z] - \Pr[Y = z]\right| .$$

When two random variables have a statistical distance of at most γ, we refer to them as γ-close.

2.3 Basic Cryptographic Primitives

Here we briefly go over a few of the basic cryptographic primitives that are of interest in this thesis. As the motivation of the thesis suggests, we will later build upon these definitions, modify, and extend them to capture the setting of backdooring or subversion in order to enable the study of what resilience means and what is required to make schemes resilient to such threats. Therefore, we delay some of the formal definitions and their extension to the sabotaged setting to later chapters.

2.3.1 Hash Functions

A hash function is an efficiently computable function which takes an input, often referred to as message, and outputs a usually shorter random-looking string, often referred to as digest. To name a few applications, hash functions are used in message authentication codes (MACs) such as HMAC, signature schemes, pseudorandom generators, randomness extraction such as HKDF, password protection, and proofs of work.

To bridge the gap between *keyed* hash functions in theory and *unkeyed* hash functions in practice, we adopt the more general notion of hash functions as families of keyed functions. The keys are public unless explicitly stated otherwise. Therefore a key here can be thought of as an index specifying which particular hash function from the family is being considered. In practical unkeyed hash functions, the key is set to some constant.

Formally, $\{\mathsf{H}_{\mathrm{k}} : X \to Y \mid \mathrm{k} \in K\}$ is a family of hash functions with associated key space K, message space X, and digest space Y. A hash function H_{k} from the family is identified by its key $\mathrm{k} \twoheadleftarrow K$ chosen at random from the key space. Practical hash functions usually work for messages of different lengths and are constructed by *iterating* a *monolithic* (i.e., fixed input-length) hash function on message blocks to extend the domain to variable lengths. Monolithic hash functions which produce outputs that are shorter than their inputs are often referred to as *compression functions*. The Merkle–Damgård [Dam90, Mer90] and the sponge [BDPVA11] constructions are two widely used domain-extenders for hash functions. They operates by iterating a compression function on blocks of messages while always using the previous (intermediate) digest and the current message block to compute the next (intermediate) digest, until the entire message is processed and a final digest is computed. For formal definitions we refer to Sections 4.2.1 and 4.3.1.

Depending on concrete cryptographic applications, hash functions are required to meet certain security requirements, among which *collision resistance*, *one-wayness*, *second-preimage resistance*, and *pseudorandomness* are the most common ones. Roughly speaking, collision resistance means that it is infeasible to find any two distinct messages which will be mapped to the same digest. One-wayness, also known as preimage resistance, concerns the infeasibility of finding a message that hashes to a given digest of a random message from the domain. Second-preimage resistance indicates that given a random message it is infeasible to find a second distinct message that collides with the first message. Finally, pseudorandomness requires that it must be hard to distinguish the output of the function on a random message from a random value in the co-domain. For relations and separations among security notions for hash functions we refer to [RS04] by Rogaway and Shrimpton. We formalize the above security notions (covering the setting of backdoors) later in the standard model in Section 3.2 (Figure 3.1) and in the ideal model in Section 3.3 (Figure 3.2).

Hash function combiners. It is possible to combine multiple hash functions in order to retain some level of security in case a number of them turn out to be insecure. For k hash functions, we recall below four common combiners: *concatenation*, *cascade*, *xor*, and *pairwise inner-product.* The concatenation combiner $\mathsf{C}_{\|}$ on some input x simply concatenates the outputs of hash functions on x. The cascade combiner $\mathsf{C}_{\circ}$ is basically the composition of hash functions, e.g., computing $\mathsf{H}_2(\mathsf{H}_1(x))$ (or $\mathsf{H}_2 \circ \mathsf{H}_1(x)$). The xor combiner $\mathsf{C}_{\oplus}$ computes the *exclusive or* of the outputs of hash functions on given inputs. Finally, the pairwise inner-product combiner C_{pip} on some input x is defined as the inner-product (denoted by $\cdot$) of all pairs of images of hash functions on x.

$$\mathsf{C}_{\|}^{\mathsf{H}_1,\ldots,\mathsf{H}_k}(x) := \mathsf{H}_1(x)\|\ldots\|\mathsf{H}_k(x) \ , \qquad \mathsf{C}_{\circ}^{\mathsf{H}_1,\ldots,\mathsf{H}_k}(x) := \mathsf{H}_k(\ldots(\mathsf{H}_1(x))) \ ,$$

$$\mathsf{C}_{\oplus}^{\mathsf{H}_1,\ldots,\mathsf{H}_k}(x) := \mathsf{H}_1(x) \oplus \ldots \oplus \mathsf{H}_k(x) \ , \qquad \mathsf{C}_{\text{pip}}^{\mathsf{H}_1,\ldots,\mathsf{H}_k}(x) := \sum_{1 \le i < j \le k} \mathsf{H}_i(x) \cdot \mathsf{H}_j(x) \ ,$$

where $\mathsf{H}_i \in \text{Fun}[n, n+s_i]$ in the first construction, $\mathsf{H}_1 \in \text{Fun}[n, n+s_1]$ and for $i > 1$ we assume $\mathsf{H}_i \in \text{Fun}[n+s_{i-1}, n+s_{i-1}+s_i]$ in the second construction, and $\mathsf{H}_i \in \text{Fun}[n, n+s]$ in the third and also the fourth construction. Each s_i intuitively defines the stretch of the corresponding hash function. The stretch values are integers and can assume negative values (compressing), positive values (expanding), or be zero (length-preserving).

2.3.2 Randomness Extractors

A random variable X is called a (weak) *k-source* if $\mathbf{H}_\infty(X) \ge k$, i.e., for all x we have $\Pr[X = x] \le 2^{-k}$. The min-entropy of a source typically determines how many bits can be *extracted* from it which are statistically close to uniform. An efficiently computable function $\mathsf{Ext} : [S] \times [N] \to [M]$ is a (k, ε)-*extractor* if for all k-sources X (over $[N]$) and a uniform distribution of seeds $\mathcal{U}_{[S]}$ we have $\mathsf{SD}(\mathsf{Ext}(\mathcal{U}_{[S]}, X), \mathcal{U}_{[M]}) \le \varepsilon$. An extractor is called strong if the stronger condition of $\mathsf{SD}(\mathsf{Ext}(\mathcal{U}_{[S]}, X) \mid \mathcal{U}_{[S]}, \mathcal{U}_{[M]} \mid \mathcal{U}_{[S]}) \le \varepsilon$ holds, i.e., the extracted value is close to uniform even when the seed is known. Some extractors do not require a random seed but rather rely on multiple weak sources.

Definition 2.1 (Multi-source extractors). *An efficient function* $\mathsf{Ext} : [N_1] \times \ldots \times [N_t] \to [M]$ *is a* $(k_1, \ldots, k_t, \varepsilon)$*-extractor if for all* k_i*-sources* X_i *over* $[N_i]$*, we have:*

$$\mathsf{SD}\big(\mathsf{Ext}(X_1, \ldots, X_t), \mathcal{U}_{[M]}\big) \leq \varepsilon \ ,$$

where ε *is usually defined as a function of* $k_1, \ldots, k_t$*. We call* Ext *an s-out-of-t* $(k_1, \ldots, k_t, \varepsilon)$*-extractor if* $\mathsf{Ext}(X_1, \ldots, X_t)$ *is* ε*-close to uniform even if only s arbitrary sources fulfill their min-entropy condition.*

A function family $\{f_{\mathrm{k}} : [N] \to [M] \mid \mathrm{k} \in [K]\}$ is called $1/M$*-universal* (or just universal) if for any two distinct values $x, x' \in [N]$ and a k chosen uniformly at random we have $\Pr[f_{\mathrm{k}}(x) = f_{\mathrm{k}}(x')] \leq 1/M$. A simple and well-known universal hash function is $f_{\mathrm{k}}(x) = (\mathrm{k} \cdot x)_{[0,m-1]}$, where $\cdot$ is the multiplication of elements in the Galois field over N, i.e., $\mathrm{GF}(N)$, and the output takes the first m bits of the multiplication. $1/M$-universal hash functions are strong $(\log M + 2 \log \varepsilon^{-1}, \varepsilon)$-extractors with k as their seed.

2.3.3 Other Primitives

Here we briefly describe a few other cryptographic primitives used in this thesis. The desired security goals are defined later in relevant chapters, when these primitives are put in the context of backdoors or subversion attacks.

Trapdoor One-Way Functions

A *trapdoor one-way function* (TDF) is a function which can be efficiently evaluated but is hard to invert, unless a *trapdoor* key is available. In other words, a TDF is a one-way function which is easy to invert for an adversary that knows its trapdoor. Bellare et al.~[]C:BHSV98 investigated TDFs and their relations to other primitives. It was already well known that TDFs with *polynomial* preimage sizes (in particular trapdoor one-way permutations, TDPs) imply public-key encryption [Yao82, GM84]. However, Bellare et al. showed that non-injective TDFs where the number of preimages of at least one value is *super-polynomial* in the input size of the function can be constructed from any one-way function. Therefore, it is widely considered infeasible to base public-key encryption solely on such TDFs. Their construction of a TDF with super-polynomial preimage size from a one-way function is elegantly simple. Given a one-way function f, for a randomly chosen β in f's range, build a trapdoor one-way function g, with a trapdoor x^* such that $f(x^*) = \beta$, as below.

$$g(y, x, v) := \begin{cases} y & \text{if } f(x) = \beta \\ f(v) & \text{otherwise} \ . \end{cases}$$

Observe that g is one-way (because of f being one-way) unless a preimage x^* of β under f (i.e., a trapdoor) is known, in which case for any v the triple (y, x^*, v) is a valid preimage of any y under g.

Pseudorandom Functions and Message Authentication Codes

A function family $\{f_{\mathrm{k}} \in \mathrm{Fun}[n, m] \mid \mathrm{k} \in \{0,1\}^k\}$ is *pseudorandom* if no efficient adversary can distinguish (with non-negligible probability) a random function of the family from a function chosen uniformly at random from the set of all functions in $\mathrm{Fun}[n, m]$. A *message authentication code* (MAC) scheme is a symmetric-key scheme consisting of three PPT algorithms $(\mathsf{KGen}, \mathsf{MAC}, \mathsf{Vf})$, which can be used to convince a receiver of the authenticity of a message. A MAC scheme is correct (with some probability) if for any key $\mathrm{k} \twoheadleftarrow \mathsf{KGen}(1^n)$ and message x we have $\mathsf{Vf}(\mathrm{k}, \mathsf{MAC}(\mathrm{k}, x)) = 1$, i.e., successful verification. A secure MAC must be hard to forge without knowing the secret key, even if MAC tags for other messages under the same key are available. A pseudorandom function family can be easily used to build a MAC scheme, by picking a key $\mathrm{k} \twoheadleftarrow \{0,1\}^k$ uniformly at random, one can generate MAC tags as $t \leftarrow f_{\mathrm{k}}(x)$ and verify them by checking $f_{\mathrm{k}}(x) = t$. Intuitively, since the outputs of a pseudorandom function f_{k} (with long enough outputs) are indistinguishable from random, they are also unpredictable and, hence, unforgeable.

Encryption Schemes

Encryption enables confidential communications. An encryption scheme consists of three PPT algorithms $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ for key generation, encryption, and decryption. A message or *plaintext* is encrypted using a key to generate a *ciphertext*, which can be decrypted to the same plaintext. There are symmetric (secret-key) and asymmetric (public-key) encryption schemes. In symmetric encryption schemes, the key $\mathrm{k} \twoheadleftarrow \mathsf{KGen}(1^n)$ used for encryption and decryption is the same and must be kept secret. In asymmetric encryption schemes, a secret and a public key are generated $(\mathrm{sk}, \mathrm{pk}) \twoheadleftarrow \mathsf{KGen}(1^n)$. A person's public key can be used by anyone to send a confidential message to her, which only she can decrypt knowing the secret key matching that specific public key. Correctness of an encryption scheme requires that ciphertexts decrypt to the plaintexts they encrypt, i.e., for any message m we should have $\mathsf{Dec}(\mathrm{k}, \mathsf{Enc}(\mathrm{k}, m)) = m$ (resp. $\mathsf{Dec}(\mathrm{sk}, \mathsf{Enc}(\mathrm{pk}, m)) = m$) with high probability.

Indistinguishability against chosen-message attacks (IND-CPA, defined in Figure 8.3) and *indistinguishability against chosen-ciphertext attacks* (IND-CCA) are two classical security notions for encryption schemes. Roughly speaking, IND-CPA requires that no efficient adversary can decide whether an encryption oracle always encrypts the left or the right message chosen by the adversary in pairs. The IND-CCA security notion, which is stronger than IND-CPA, gives the adversary, in addition to the encryption oracle, also access to a decryption oracle which decrypts ciphertexts that have not been returned by the encryption oracle in the past.

Homomorphic encryption schemes are asymmetric encryption schemes that allow one to perform operations on encrypted messages by computing directly on their ciphertexts. Below we recall the formal definition. We assume that the message space M with some efficiently computable operation "$\circ$" forms a group (where the message space usually depends on the security parameter or the public key, but we omit this reference for sake of simplicity). Analogously, we assume that the ciphertext space C forms a group with some efficiently computable operation "$\diamond$". Furthermore, inverses in a cyclic group are efficiently computable.

Definition 2.2 (Homomorphic Encryption Scheme)**.** *A homomorphic public-key encryption scheme* $\mathsf{HE} := (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *with associated message group* $(M, \circ)$ *and ciphertext group* $(C, \diamond)$ *consists of three algorithms:*

$\mathsf{KGen}(1^n) \twoheadrightarrow (\mathrm{sk}, \mathrm{pk})$*: On input the security parameter* 1^n *this PPT algorithm generates a secret key* sk *and a public key* pk*.*

$\mathsf{Enc}(\mathrm{pk}, m) \twoheadrightarrow c$: *On input a public key* pk *and a message* $m \in M$ *this PPT algorithm outputs a ciphertext* $c \in C$.

$\mathsf{Dec}(\mathrm{sk}, c) \rightarrow m$: *On input a secret key* sk *and a ciphertext* $c \in C$, *this DPT algorithm outputs a message* $m \in M$.

For any $n \in \mathbb{N}$, *any* $(\mathrm{sk}, \mathrm{pk}) \twoheadleftarrow \mathsf{KGen}(1^n)$, *any messages* $m, m' \in M$ *the following conditions hold:*

Correctness: $\mathsf{Dec}(\mathrm{sk}, \mathsf{Enc}(\mathrm{pk}, m)) = m$.

Homomorphism: $\mathsf{Enc}(\mathrm{pk}, m \circ m')$ *has the same distribution as* $\mathsf{Enc}(\mathrm{pk}, m) \diamond \mathsf{Enc}(\mathrm{pk}, m')$.

A classical example of a homomorphic encryption scheme is the ElGamal encryption scheme [ElG84], where ciphertexts $c = (g^r, \mathrm{pk}^r \cdot m)$ are pairs of elements from a cyclic group $\mathcal{G}$ with a generator g and of prime order q, and messages are from $\mathcal{G}$ as well. The operations are multiplication in $\mathcal{G}$ for messages, and component-wise multiplication in $\mathcal{G}$ for ciphertexts.

Signature Schemes

A *digital signature* scheme is a public-key primitive, which provides a way to verify the authenticity of messages. It consists of three PPT algorithms $(\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$. The key generation algorithm on input of a security parameter 1^n produces a secret signing key sk and a matching public verification key pk. Suppose this key pair belongs to a party called Alice. She can then sign any message m and produce a signature $\sigma \twoheadleftarrow \mathsf{Sig}(\mathrm{sk}, m)$. A receiver, Bob, can verify σ by calling $\mathsf{Vf}(\mathrm{pk}, \sigma, m)$. If the output is 1 (and not 0), Bob is convinced that m indeed comes from Alice and has not been altered during transmission. No efficient adversary should be able to forge signatures on behalf of Alice without knowing her secret key. This property is referred to as *unforgeability*. More precisely, we often use *existential unforgeability against chosen-message attacks* (EUF-CMA), which requires that it should be hard to forge a new signature that verifies under a given public key, even if the adversary can see other valid signatures on many messages of its choice. See Figure 8.6 for a formal definition of this game. Practical signature schemes usually follow the *hash-then-sign* approach, where messages are first hashed to short digests of fixed length, for efficiency reasons, and then signed. The employed hash function is usually required to be collision resistant.

Key Exchange

In a *key exchange* (a.k.a. key agreement) protocol, two or more parties interact to derive a shared secret key, which they subsequently can use for example to establish a confidential channel. More formally, a two-party key exchange protocol can be defined as a pair of algorithms (*Alice*, *Bob*) which run on some inputs and interact with each other in a number of communication rounds. From their interaction, Alice derives a key k_A and Bob a key k_B. The protocol is correct if $\mathrm{k}_A = \mathrm{k}_B$ (with high probability). There are various security goals that can be defined for a key exchange protocol, based on two general flavors: unpredictability and indistinguishability. Roughly speaking, the former type captures security against *key recovery*, saying that no efficient adversary, passively listening on the interaction or actively modifying the communication, should be able to derive the same key as Alice and Bob. The latter, *key indistinguishability*, requires that such adversaries cannot even distinguish the real exchanged key from a random bit string of same size. A formal definition of key indistinguishability is given in Figure 8.10. One of the earliest practical key exchange protocols was proposed by Diffie and Hellman [DH76].

Physical Uncloneable Functions

A physical uncloneable function (PUF) is an at least partially physical entity that is easy to evaluate, if one is in possession of the PUF, and hard to predict otherwise. A PUF can be stimulated with so-called challenges to which it responds with slightly noisy values, called responses. For the sake of simplicity, we assume that PUFs deterministically return consistent answers. In practice a fuzzy extractor [DRS04] can be applied to responses in order to eliminate the noise. Moreover, we only consider PUFs that have exponential challenge and response spaces and, hence, cannot be efficiently learned.

Due to uncontrollable variations in the manufacturing process, it is even for the manufacturer practically infeasible to clone a good PUF. This property is referred to as *unclonability.* Only a party (honest or malicious) that is in possession of the PUF can evaluate it. Besides the parties, another PUF may be in possession of the PUF, called encapsulated PUFs [BKOV17] or PUF-inside-PUFs [Rüh16]. In this case the outer PUF may then exclusively evaluate the inner PUF. The possibility to encapsulate PUFs allows for example to perform simple checks, such as challenge-response validation, before evaluating the PUF on the actual data; a malicious PUF may switch only to a skewed mode after the checks. Alongside unclonability, PUFs can have various other properties that make them attractive for cryptographic schemes. A property that we take advantage of is *pseudorandomness* of PUF responses [AMSY16]. This means that the PUF approximates a random function. A simplified and intuitive game for (computational) pseudorandomness in our terminology that suffices for this thesis can be found in Figure 8.9 in Chapter 8. For a more comprehensive and formal definition of PUFs and their security properties we refer to [BFSK11] by Brzuska et al. and [AMSY16] by Armknecht et al.

2.4 Random Oracle Methodology

From a provable security perspective, a particularly successful methodology to use hash functions in protocols has been the introduction of the random-oracle (RO) model [FS87, BR93]. In this model all parties, honest or otherwise, are given oracle access to a uniformly chosen random function $\mathsf{H} \twoheadleftarrow \mathrm{Fun}[n, m]$. This formalizes the intuition that the outputs of a well-designed hash function look random. The output of a random oracle on any input is uniformly random and independent of the rest of the random oracle. It is often convenient in proofs to fill in the table of the random oracle gradually and upon request, rather than fixing it completely beforehand. This technique is called *lazy sampling*, where an output is sampled at random upon receiving a fresh query and stored for future references. The strong randomness properties inherent in the random oracle, in turn, facilitate the conjectured security analyses of many practical protocols such as RSA-PSS and ElGamal (hash-then-sign) signatures and RSA-OAEP encryption [BR93, BR95, PS96, BR06].

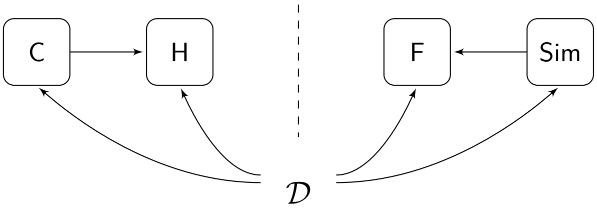
The random-oracle model is an *idealized* model, i.e., it assumes an ideal or perfect hash function. Other important idealized models used in cryptography are the ideal-cipher model (i.e., idealized block ciphers), random-permutation model (i.e., idealized permutations), and generic-group model (i.e., idealized cyclic groups). The term *standard model* (or plain model) is used to refer to a cryptographic model that does not make use of ideal objects.

2.4.1 Indifferentiability

A common paradigm in the design of hash functions is to start with some underlying primitive, and through some construction build a more complex one. The provable security of such constructions have been analyzed through two main approaches. One formulates specific goals (such as collision resistance) and goes on to show that the construction satisfies them if its underlying primitives satisfy their own specific security properties. Another is a general approach, whose goal is to show that a wide class of security goals are simultaneously met.

The latter has been formalized in a number of frameworks, notably in the universal composability (UC) framework of Canetti [Can01], the reactive systems framework of Pfitzmann and Waidner [PW01], and the indifferentiability framework of Maurer, Renner, and Holenstein [MRH04]. The indifferentiability framework is by now considered a standard methodology to study the soundness of cryptographic constructions, particularly symmetric ones such as hash functions [CDMP05, BDPV08] and block ciphers [CPS08, HKT11, ABD$^+$13, DSSL16] in idealized models of computation.

In the indifferentiability framework, a public primitive H is available and the goal is to build another primitive, say a random oracle, from H through a construction C^{H}. Indifferentiability formalizes a set of necessary and sufficient conditions for the construction C^{H} to securely replace its ideal counterpart F in a wide range of environments: as depicted below, for a *simulator* Sim, the systems $(\mathsf{C}^{\mathsf{H}}, \mathsf{H})$ and $(\mathsf{F}, \mathsf{Sim}^{\mathsf{F}})$ should be indistinguishable. Note that the adversary (or distinguisher) $\mathcal{D}$ in the real world has access to the underlying primitive H.



Below we restate a composition theorem proven by Maurer et al. but in a terminology similar to the one by Coron et al. [CDMP05], which is applied to hash functions and is also more widely used. A central corollary of this composition theorem is that indifferentiability implies any single-stage security goal (i.e., a security game where either only one adversarial procedure is considered or multiple ones that share full state). Single-stage security goals include one-wayness, collision resistance, PRG security, PRF security, and more. In RO-indifferentiability, the ideal primitive F in the theorem below is a random oracle.

Theorem 2.1 (Composition). *Let $\mathcal{G}$ be a cryptosystem with oracle access to an ideal primitive F. Let C be a construction such that C^{H} is indifferentiable from F. Then, for any single-stage security notion, the cryptosystem $\mathcal{G}$ when using C^{H} is at least as secure as when it uses F.*

2.5 Communication Complexity

The communication cost [Yao79] of a two-party deterministic protocol π on inputs x and y is the number of bits that are transmitted in a run of the protocol $\pi(x, y)$. We denote this by $\mathsf{CC}(\pi(x, y))$. The worst-case communication complexity of π is $\max_{(x,y)} \mathsf{CC}(\pi(x, y))$. A protocol π computes a task (here, a function) $f : X \times Y \to Z$ if the last message of $\pi(x, y)$ corresponds to $f(x, y)$. The communication complexity of a task f is the minimum communication complexity of any protocol π that computes f. Protocols can also be randomized and, thus, might err for given inputs (x, y) with probability $\Pr[\pi(x, y) \neq f(x, y)]$.

In the cryptographic setting, we are interested in distributional (a.k.a. average-case) communication complexity measured by averaging the communication cost over random choices of inputs and coins. A standard coin-fixing argument shows that in the *distributional* setting any protocol can be derandomized with no change in communication complexity, and thus we can focus on deterministic protocols. For a given distribution μ over the inputs (x, y), the protocol error and correctness are computed by taking the probability over the choice of (x, y). Following cryptographic conventions, we denote protocol correctness by $\mathsf{Adv}^{\mathsf{f}}_{\mu}(\pi)$, where f is a placeholder for the name of the task f and μ is the distribution of the inputs. We define the distributional communication cost of a deterministic protocol π as the expected communication complexity according to the distribution, i.e.,

$$\mathsf{D}_{\mu}(\pi) := \mathbb{E}_{(x,y) \twoheadleftarrow \mu}[\mathsf{CC}(\pi(x, y))] \ .$$

The distributional communication complexity of a task f with error ε is

$$\mathsf{D}^{\varepsilon}_{\mu}(f) := \min_{\pi} \mathsf{D}_{\mu}(\pi) \ ,$$

where the minimum is taken over all deterministic protocols π which err with probability at most ε. In this thesis we need to slightly generalize functional tasks to relational tasks $R(x, y) \subseteq Z$ and define error as $\Pr[\pi(x, y) \notin R(x, y)]$.

Two well-studied problems in this area are the *set-disjointness* and *set-intersection* problems (see [CP10] by Chattopadhyay and Pitassi for a survey). In these problems two parties hold sets S and T respectively. In set-disjointness, their goal is to decide whether or not $S \cap T = \emptyset$; in set-intersection they want to compute at least one element in this intersection. In this thesis we will rely on distributional lower bounds for set-disjointness (Theorem 6.10) and set-intersection (Theorem 6.11). For more on communication complexity, we refer the reader to the books [KN97] by Kushilevitz and Nisan and [RY20] by Rao and Yehudayoff.

Cut-and-paste. There is a lemma called *cut-and-paste* from communication complexity, which we will use later in proving Theorems 6.10 and 6.11. We restate this lemma below for the sake of completeness. But before we do so, we need a few basics. First, the Hellinger distance between random variables X and Y over a common domain D is defined as

$$\Delta_{\mathsf{Hel}}(X, Y) := \sqrt{1 - \textstyle\sum_{z \in D} \sqrt{\Pr[X = z] \cdot \Pr[Y = z]}} \ .$$

Furthermore, note that we can represent a set $S \subseteq [N]$ as an N-bit string X where its i-th bit x_i is 1 iff $i \in S$. We also need an understanding of *combinatorial rectangles* and why they are useful in communication complexity. A set $R \subseteq R_1 \times R_2$ is called a combinatorial rectangle iff whenever $(x, y) \in R$ and $(x', y') \in R$, then we also have $(x', y) \in R$ and $(x, y') \in R$. Deterministic protocols have a rectangle property, which means that any of their transcripts τ corresponds to some rectangle $R_\tau = S_\tau \times T_\tau$. In other words, all combinations of inputs from S_τ for Alice and T_τ for Bob lead to the same transcript τ.

Lemma 2.2 (Cut-and-Paste)**.** *Let $\Pi(X, Y)$ denote a random variable for the transcripts of a deterministic protocol on input bit strings sampled from independent random variables (X, Y) such that the corresponding sets S and T are drawn from a product distribution μ, i.e., $(S, T) \twoheadleftarrow \mu$ where the sets are independently chosen. Let $a, b \in \{0, 1\}$ and define $\Pi^i_{a,b}(X, Y) := \Pi(X, Y) \mid x_i = a \wedge y_i = b$, where x_i is the i-th bit of X and analogously y_i is the i-th bit of Y. Then for each i, it holds that*

$$\Delta^2_{\mathsf{Hel}}(\Pi^i_{0,0}(X, Y), \Pi^i_{1,1}(X, Y)) = \Delta^2_{\mathsf{Hel}}(\Pi^i_{0,1}(X, Y), \Pi^i_{1,0}(X, Y)) \ .$$

Proof. By definition of Δ_{Hel} it suffices to show that for each i and any transcript τ,

$$\Pr[\Pi^i_{0,0}(X, Y) = \tau] \cdot \Pr[\Pi^i_{1,1}(X, Y) = \tau] = \Pr[\Pi^i_{0,1}(X, Y) = \tau] \cdot \Pr[\Pi^i_{1,0}(X, Y) = \tau] \ .$$

By the rectangle property of protocols, any transcript τ corresponds to a rectangle $R_\tau = S_\tau \times T_\tau$. Thus

$$\Pr[\Pi^i_{a,b}(X, Y) = \tau] = \Pr[X^{-i}_a \in S_\tau \wedge Y^{-i}_b \in T_\tau] \ ,$$

where X^{-i} (resp. Y^{-i}) is the input with i-th coordinate removed. By the independence of X and Y we can write

$$\Pr[X^{-i}_a \in S_\tau \wedge Y^{-i}_b \in T_\tau] = \Pr[X^{-i}_a \in S_\tau] \Pr[Y^{-i}_b \in T_\tau]$$

That is

$$\Pr[\Pi^i_{a,b} = \tau] = A^i_a(\tau) B^i_b(\tau)$$

for some A^i_a and B^i_b. Thus we obtain

$$\begin{aligned}\Pr[\Pi^i_{0,0} = \tau] \Pr[\Pi^i_{1,1} = \tau] &= A^i_0(\tau) B^i_0(\tau) A^i_1(\tau) B^i_1(\tau) \\ &= A^i_0(\tau) B^i_1(\tau) A^i_1(\tau) B^i_0(\tau) \\ &= \Pr[\Pi^i_{0,1} = \tau] \Pr[\Pi^i_{1,0} = \tau] \ .\end{aligned}$$

□

2.6 Information-Theoretic Inequalities

In this section we recall a few information-theoretic inequalities and known results that will mostly be of use in Chapter 6, more specifically, when giving lower bounds for the communication complexity of set-disjointness (Theorem 6.10) and set-intersection (Theorem 6.11) problems.

First, we recall that statistical distance ($\mathsf{SD}(X,Y) := \frac{1}{2}\sum_{z\in D}\left|\Pr[X=z]-\Pr[Y=z]\right|$) and Hellinger distance ($\Delta_{\mathsf{Hel}}(X,Y) := \sqrt{1-\sum_{z\in D}\sqrt{\Pr[X=z]\cdot\Pr[Y=z]}}$) of two random variables X and Y over D are related via

$$\Delta^2_{\mathsf{Hel}}(X,Y) \le \mathsf{SD}(X,Y) \le \sqrt{2}\cdot\Delta_{\mathsf{Hel}}(X,Y)\ .$$

To see this, let $p_z := \Pr[X=z]$ and $q_z := \Pr[Y=z]$. For the first inequality, we proceed as follows, where all the sums are over $z \in D$:

$$\begin{aligned}
\Delta^2_{\mathsf{Hel}}(X,Y) &= 1-\sum\sqrt{p_zq_z}\\
&= (1/2)\cdot\sum(p_z+q_z)-\sum\sqrt{p_zq_z}\\
&= (1/2)\cdot\sum(\sqrt{p_z}-\sqrt{q_z})^2\\
&= (1/2)\cdot\sum|\sqrt{p_z}-\sqrt{q_z}||\sqrt{p_z}-\sqrt{q_z}|\\
&\le (1/2)\cdot\sum|\sqrt{p_z}-\sqrt{q_z}||\sqrt{p_z}+\sqrt{q_z}|\\
&\le (1/2)\cdot\sum|p_z-q_z|\\
&= \mathsf{SD}(X,Y)\ .
\end{aligned}$$

For the second inequality, again taking all the sums over $z \in D$, we have

$$\begin{aligned}
\mathsf{SD}^2(X,Y) &= (1/4)\cdot\Big(\sum|p_z-q_z|\Big)^2\\
&= (1/4)\cdot\Big(\sum|\sqrt{p_z}-\sqrt{q_z}||\sqrt{p_z}+\sqrt{q_z}|\Big)\\
&\le (1/4)\cdot\Big(\sum(\sqrt{p_z}-\sqrt{q_z})^2\Big)\Big(\sum(\sqrt{p_z}+\sqrt{q_z})^2\Big) \qquad (2.1)\\
&= (1/2)\cdot\Delta^2_{\mathsf{Hel}}(X,Y)\cdot\Big(2+2\sum\sqrt{p_zq_z}\Big)\\
&= \Delta^2_{\mathsf{Hel}}(X,Y)\cdot(2-\Delta^2_{\mathsf{Hel}}(X,Y))\\
&\le 2\cdot\Delta^2_{\mathsf{Hel}}(X,Y)\ .
\end{aligned}$$

The first inequality (i.e., Line 2.1) is an application of the Cauchy–Schwarz inequality, which states that $\left(\sum uv\right)^2 \le \sum u^2\sum v^2$.

Lemma 2.3. *Let Π be any random variable and $x_1,\ldots,x_N,y_1,\ldots,y_N$ be independent random variables. Then we have*

$$\mathbf{I}(x_1,\ldots,x_N,y_1,\ldots,y_N;\Pi) \ge \sum_{i=1}^{N}\mathbf{I}(x_i,y_i;\Pi)\ .$$

Proof. We have

$$\begin{aligned}
&\mathbf{I}(x_1,\ldots,x_N,y_1,\ldots,y_N;\Pi)\\
&\qquad = \mathbf{H}(x_1,\ldots,x_N,y_1,\ldots,y_N)+\mathbf{H}(\Pi)-\mathbf{H}(x_1,\ldots,x_N,y_1,\ldots,y_N,\Pi)
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{i=1}^{N} \mathbf{H}(x_i, y_i) + \mathbf{H}(\Pi) - (\mathbf{H}(\Pi) + \sum_{i=1}^{N} \mathbf{H}(x_i, y_i \mid x_1, \ldots, x_{i-1}, y_1, \ldots, y_{i-1}, \Pi)) && (2.2)\\
&= \sum_{i=1}^{N} (\mathbf{H}(x_i, y_i) - \mathbf{H}(x_i, y_i \mid x_1, \ldots, x_{i-1}, y_1, \ldots, y_{i-1}, \Pi))\\
&\geq \sum_{i=1}^{N} (\mathbf{H}(x_i, y_i) - \mathbf{H}(x_i, y_i \mid \Pi)) && (2.3)\\
&= \sum_{i=1}^{N} \mathbf{I}(x_i, y_i; \Pi) \ ,
\end{aligned}
$$

where the independence is used in Line 2.2. Note that $\mathbf{H}(x_i, y_i \mid x_1, \ldots, x_{i-1}, y_1, \ldots, y_{i-1}, \Pi) \leq \mathbf{H}(x_i, y_i \mid \Pi)$ holds since although x_i and y_i's are independent, Π may still depend on them. □

The next lemma relates the mutual information of two random variables with their Hellinger distance. In the proof we need the Kullback–Leibler (KL) divergence between probability distributions P and Q on domain D, which is defined as

$$D_{\mathrm{KL}}(P \| Q) := -\sum_{z \in D} P(z) \cdot \log \frac{Q(z)}{P(z)} \ .$$

Lemma 2.4. *Let X and Y be random variables over D_X and D_Y, respectively. Let $Y_x := Y|X = x$, i.e., Y conditioned on $X = x$. Then it holds that*

$$\mathbb{E}[\Delta^2_{\mathsf{Hel}}(Y, Y_x)] \leq \mathbf{I}(X; Y) \ .$$

Proof. Using shorthand notations $P(y|x) := \Pr[Y = y|X = x]$, and $P(x, y) := \Pr[X = x \wedge Y = y]$, we have

$$
\begin{aligned}
\mathbb{E}[\Delta^2_{\mathsf{Hel}}(Y, Y_x)] &= \mathbb{E}\big[1 - \sum_{y} \sqrt{P(y|x) \cdot P(y)}\big]\\
&= \sum_{x} P(x) \cdot \big(1 - \sum_{y} \sqrt{P(y|x) \cdot P(y)}\big)\\
&= \sum_{x} P(x) - \sum_{x,y} P(x)\sqrt{P(y|x) \cdot P(y)}\big)\\
&= 1 - \sum_{x,y} \sqrt{P(y|x) \cdot P(x) \cdot P(x) \cdot P(y)}\\
&= 1 - \sum_{x,y} \sqrt{P(x,y) \cdot P(x) \cdot P(y)}\\
&= \Delta^2_{\mathsf{Hel}}(P(X,Y), P(X)P(Y))\\
&\leq D_{\mathrm{KL}}(P(X,Y) \| P(X)P(Y)) && (2.4)\\
&= \mathbf{I}(X; Y) \ , && (2.5)
\end{aligned}
$$

where the sums are over $x \in D_Y$ and $y \in D_Y$ and Line 2.4 follows from Lemma 2.5 stated below and Line 2.5 follows from the fact that $\mathbf{I}(X; Y) = D_{\mathrm{KL}}(P(X, Y) \| P(X)P(Y))$. □

The final lemma in this section relates KL-divergence and Hellinger distance.

Lemma 2.5. *For any probability distributions P and Q over a domain D, we have*

$$D_{\mathrm{KL}}(P\|Q) \geq \Delta^2_{\mathsf{Hel}}(P,Q) \ .$$

Proof. We have

$$\begin{aligned}
\Delta^2_{\mathsf{Hel}}(P,Q) &= 1 - \sum_{z\in D}\sqrt{P(z)\cdot Q(z)} \\
&\leq -\log \sum_{z\in D}\sqrt{P(z)\cdot Q(z)} \qquad (2.6)\\
&\leq -\log \sum_{z\in D}\big(P(z)\cdot Q(z)\big) \\
&= -\log \sum_{z\in D}\big(P(z)\cdot \frac{Q(z)}{P(z)}\big) = -\log \mathbb{E}_{z\in D}\frac{Q(z)}{P(z)} \\
&\leq -\mathbb{E}_{z\in D}\log\frac{Q(z)}{P(z)} \qquad (2.7)\\
&= -\sum_{z\in D}P(z)\cdot\log\frac{Q(z)}{P(z)} \\
&= D_{\mathrm{KL}}(P\|Q) \ ,
\end{aligned}$$

where 2.6 holds because $1-x \leq -\log x$ and 7.2 is an application of Jensen's inequality, which we restate below. □

Jensen's inequality. Let X be a random variable and f a concave function. Then it holds that $\mathbb{E}[f(X)] \leq f(\mathbb{E}(X))$. For a convex function the direction of the inequality reverses.

PART I

Backdoors in Hash Functions

Chapter 3

Modeling Backdoored Hash Functions

In this chapter we formally define backdoored hash functions and adapt common security properties for hash functions to the setting with backdoors. We consider a scenario, where a big brother designs a hash function that can be broken easily with access to some secret backdoor, but the hash function retains some security against adversaries without knowledge of that backdoor. This formalization is justified, since it intuitively matches the NOBUS property to not only keep other adversaries out but also to keep users oblivious towards the existence of any backdoors. We take two different approaches in modeling backdoored hash functions: backdoored standard-model hash functions and backdoored hash functions where the hash function without the backdoor is ideal, i.e., a random oracle.

My Scientific Contribution in this Chapter

The formalization of backdoored standard-model hash functions is a part of [FJM18a], which is a joint work with Marc Fischlin and Christian Janson. This model was devised jointly by Christian, Marc, and me. The one presented in this thesis, in Section 3.2, differs mostly syntactically from the original publication. I chose to modify the model to obtain more intuitive definitions and enable a better comparison with the backdoored random-oracle model.

The backdoored random-oracle model described in Section 3.3 and the infeasibility of black-box $(0, k)$-combiners in the standard model, discussed in Section 3.4, both appeared in [BFM18a], which is a joint work with Balthazar Bauer and Pooya Farshim. Pooya and I jointly devised the idea of combining two independently backdoored primitives to build a backdoor-resilient one. Despite our strong intuition that this goal should be achievable, our attempts at proving positive results in the standard model were not successful. Pooya then suggested that we move to a backdoored version of an idealized model of computation (e.g., the random-oracle model), which ultimately lead to our positive results in Chapter 6 (and a follow-up work described in Chapter 7). Pooya and I jointly developed the BRO model in Section 3.3. The impossibility result in Section 3.4 was conducted mostly by Pooya with my help.

3.1 Introduction

Hash functions are one of the most fundamental building blocks in cryptography. For this reason, both the cryptanalysis and provable security of hash functions have been active areas of research in recent years. While cryptanalytic validation can strengthen our confidence in the security of hash functions, as such analyses improve, also weaknesses are discovered, which can lead to partial or total break of their security. For example, the first known instances of collisions and chosen-prefix collisions in SHA-1 were demonstrated recently by Stevens et al. [SBK^{+}17] and Leurent and Peyrin [LP19], respectively. Still, these analyses might fail to detect *intentional* weaknesses that may have been built into hash functions. Backdoors might even be themselves constructed using sophisticated cryptographic techniques, which make them hard to detect. It is of utmost importance to understand the risks that are posed by backdoors in hash functions and develop solutions to mitigate them. An inevitable starting point is to adequately model the syntax and security of hash functions in such adversarial settings.

We explore two different directions in modeling backdoors in hash functions: backdoors in standard-model hash functions and backdoors in the random-oracle model. Both approaches are useful for studying different aspects of attacks as well providing security under different assumptions, as we show in Chapter 4 for attacks, and in Chapter 5 for defeating standard-model backdoors, as well as in Chapters 6 and 7 for defeating backdoored random oracles. In both approaches we assume that hash functions are designed (and potentially even standardized) by malicious authorities. These hash functions may even display good behaviors, but they are insecure against adversaries that have access to their backdoors.

In the first part of this chapter, we formalize backdoored hash functions in the standard model. We often refer to backdoored hash functions in the standard model simply as backdoored hash functions, when it is clear from the context that the underlying model is standard. Here a big brother generates a backdoored hash function family together with a short bit string corresponding to a so-called *backdoor key*. The adversarial influence in the design is captured by this definition in that the hash function family and its internal constants can be chosen in a way that a short backdoor key co-designed with that family enables bypassing some of the security properties. Meanwhile, we assume that a backdoored hash function retains security against adversaries that do not hold a backdoor key, reflecting the fact that a rational malicious designer would want the backdoor to be NOBUS.

We include the possibility of having backdoored hash functions in four central security requirements, which are one-wayness, second-preimage resistance, collision resistance, and pseudorandom-function security. Looking ahead, in Chapter 4 we analyze constructions of backdoored hash functions, where the backdoor undermines one-wayness, second-preimage resistance, and collision resistance. In Chapter 5 we discuss a technique to neutralize backdoors in pseudorandom functions.

In the second part of this chapter, we introduce an extension to the random-oracle model, which we call the backdoored random-oracle (BRO) model. The BRO model substantially weakens the traditional RO model by letting the adversary additionally access an oracle called the *backdoor oracle* which can be used to break the random oracle in an arbitrary sense. Simply put, the backdoor oracle can be queried on arbitrary functions, upon which it computes the queried function on the entire table of the random oracle and returns the result. Contrary to the standard model, we cannot reasonably capture breaking a variety of security notions if we model the backdoor as a short backdoor key, since random oracles do not even have efficient descriptions. A short backdoor key in this setting rather resembles an auxiliary input (e.g., a few points) dependent on the random oracle, than an actual backdoor.

The BRO model is simple and does not make assumptions about the backdoor capability. It might seem contradictory to assume RO-like behavior from a backdoored hash function. However keep in mind that we assume this *without* access to the backdoor. This approach opens a door to formalizing concepts such as independence of multiple hash functions and it enables building simple backdoor-resilient hash functions with meaningful proofs of security. Such positive results are very challenging and often infeasible to prove in the standard model.

We again revisit a few essential security notions of hash functions to reflect the existence of backdoors. Here, we choose one-wayness, pseudorandom-generator security, and collision resistance. Looking ahead, in Chapter 6 we give positive results for these three security notions in a model where two independent BROs are available. Hence, we define these notions for a combiner rather than a single BRO.

Finally, we explore the difficulty of constructing a secure hash function by combining only backdoored hash functions in the standard model. We confirm this intuition by giving an impossibility result which rules out fully black-box reductions for such combiners in the standard model.

3.2 Backdoored Hash Functions

A standard-model backdoored hash function is a hash function which is designed by an adversary together with a short backdoor key, whose knowledge enables violating the security of the hash function. More precisely, we consider an efficient algorithm BDHGen, which on input of three integers k, n, and m outputs a hash function family H, a key space K, and a backdoor key bk. The hash function family H consists of hash function with key length k and output length m. Regarding the input, the hash functions either accept variable-length inputs of up to a polynomial length $\mathsf{poly}(n)$ or only inputs of a fixed-length $\mathsf{poly}(n)$. The key space is a subset of all strings of size k, i.e., $K \subseteq \{0,1\}^k$. If $K \neq \{0,1\}^k$, then the backdoor is key-dependent and works only for instances H_k, where $\mathrm{k} \in K$.

Definition 3.1 (Backdoored Hash Function Generator)**.** *A* PPT *algorithm* BDHGen *is called a* backdoored hash function generator*, if on input of parameters* $k, n, m \in \mathbb{N}$*, it outputs a hash function family* H*, a key space* $K \subseteq \{0,1\}^k$*, and a backdoor key* $\mathrm{bk} \in \{0,1\}^*$*. The hash function family is either defined for variable-length inputs of up to a polynomial* $\mathsf{poly}(n)$ *length, in which case we have* $\mathsf{H} := \{\mathsf{H}_\mathrm{k} : \{0,1\}^{\leq \mathsf{poly}(n)} \to \{0,1\}^m \mid \mathrm{k} \in \{0,1\}^k\}$*, or for fixed-length inputs, in which case we have* $\mathsf{H} := \{\mathsf{H}_\mathrm{k} : \{0,1\}^{\mathsf{poly}(n)} \to \{0,1\}^m \mid \mathrm{k} \in \{0,1\}^k\}$*.*

3.2.1 Security Notions in the Standard Model

We give definitions for four of the most commonly used security notions of hash functions: one-way security, second-preimage resistance, collision resistance, and security of pseudorandom functions. These games are formalized in Figure 3.1. Furthermore, to avoid confusion about the secrecy of the hash key in security notions, we often use iv (instead of k) to denote a publicly known key, also called the initialization vector. In all but the PRF game, the key iv is public and is provided to the adversary. However, in the PRF game, not only is the key k secret, but it is also chosen uniformly at random from an unrestricted set of possible keys, i.e., from $\{0,1\}^k$ instead of K. In defining these games, we deviate slightly from common formulations and potentially give the adversary the backdoor key. Our definitions are thus general enough to capture standard security notions without backdoors ($bd = 0$) as well as security against backdooring adversaries ($bd = 1$). To sample an element of the domain, which we need in the one-wayness and second-preimage resistance games, or to sample a whole function with a certain domain, which we need in the PRF game, we use the function $\mathsf{dom}(\mathsf{H}_\mathrm{iv})$ to get the domain of H_iv. Recall also that we use $\mathrm{Fun}[\mathsf{dom}(\mathsf{H}_\mathrm{iv}), m]$ to denote the

```
Game $\mathrm{OW}^{\mathcal{A},bd}_{\mathsf{BDHGen}}(1^k, 1^n, 1^m)$
$(\mathsf{H}, K, \mathrm{bk}) \twoheadleftarrow \mathsf{BDHGen}(1^k, 1^n, 1^m)$
if $bd = 0$ then $\mathrm{bk} \leftarrow \bot$
$\mathrm{iv} \twoheadleftarrow K$
$x \twoheadleftarrow \mathsf{dom}(\mathsf{H}_{\mathrm{iv}})$
$y \leftarrow \mathsf{H}_{\mathrm{iv}}(x)$
$x' \twoheadleftarrow \mathcal{A}(\mathsf{H}, \mathrm{iv}, \mathrm{bk}, y)$
return $(\mathsf{H}_{\mathrm{iv}}(x') = y)$
```

```
Game $\mathrm{SPR}^{\mathcal{A},bd}_{\mathsf{BDHGen}}(1^k, 1^n, 1^m)$
$(\mathsf{H}, K, \mathrm{bk}) \twoheadleftarrow \mathsf{BDHGen}(1^k, 1^n, 1^m)$
if $bd = 0$ then $\mathrm{bk} \leftarrow \bot$
$\mathrm{iv} \twoheadleftarrow K$
$x \twoheadleftarrow \mathsf{dom}(\mathsf{H}_{\mathrm{iv}})$
$x' \twoheadleftarrow \mathcal{A}(\mathsf{H}, \mathrm{iv}, \mathrm{bk}, x)$
return $(x \neq x' \wedge \mathsf{H}_{\mathrm{iv}}(x') = \mathsf{H}_{\mathrm{iv}}(x))$
```

```
Game $\mathrm{CR}^{\mathcal{A},bd}_{\mathsf{BDHGen}}(1^k, 1^n, 1^m)$
$(\mathsf{H}, K, \mathrm{bk}) \twoheadleftarrow \mathsf{BDHGen}(1^k, 1^n, 1^m)$
if $bd = 0$ then $\mathrm{bk} \leftarrow \bot$
$\mathrm{iv} \twoheadleftarrow K$
$(x_1, x_2) \leftarrow \mathcal{A}(\mathsf{H}, \mathrm{iv}, \mathrm{bk})$
$y_1 \leftarrow \mathsf{H}_{\mathrm{iv}}(x_1)$
$y_2 \leftarrow \mathsf{H}_{\mathrm{iv}}(x_2)$
return $(x_1 \neq x_2 \wedge y_1 = y_2)$
```

```
Game $\mathrm{PRF}^{\mathcal{A},bd}_{\mathsf{BDHGen}}(1^k, 1^n, 1^m)$
$(\mathsf{F}, K, \mathrm{bk}) \twoheadleftarrow \mathsf{BDHGen}(1^k, 1^n, 1^m)$
if $bd = 0$ then $\mathrm{bk} \leftarrow \bot$
$\mathrm{k} \twoheadleftarrow \{0,1\}^k$
$\mathsf{F}_0 \leftarrow \mathsf{F}_{\mathrm{k}}$
$\mathsf{F}_1 \twoheadleftarrow \mathrm{Fun}[\mathsf{dom}(\mathsf{F}_{\mathrm{k}}), m]$
$b \twoheadleftarrow \{0,1\}$
$b' \twoheadleftarrow \mathcal{A}^{\mathsf{F}_b}(\mathsf{F}, \mathrm{bk})$
return $(b = b')$
```

Figure 3.1: The one-wayness, second-preimage resistance, collision resistance, and pseudorandom function security games for a hash function generator BDHGen.

set of all functions $f : \mathsf{dom}(\mathsf{H}_{\mathrm{iv}}) \to \{0,1\}^m$. The advantage terms corresponding to the games of Figure 3.1 are

$$\mathsf{Adv}^{\mathrm{ow},bd}_{\mathsf{BDHGen}}(\mathcal{A}, k, n, m) := \Pr[\mathrm{OW}^{\mathcal{A},bd}_{\mathsf{BDHGen}}(1^k, 1^n, 1^m)],$$

$$\mathsf{Adv}^{\mathrm{spr},bd}_{\mathsf{BDHGen}}(\mathcal{A}, k, n, m) := \Pr[\mathrm{SPR}^{\mathcal{A},bd}_{\mathsf{BDHGen}}(1^k, 1^n, 1^m)],$$

$$\mathsf{Adv}^{\mathrm{cr},bd}_{\mathsf{BDHGen}}(\mathcal{A}, k, n, m) := \Pr[\mathrm{CR}^{\mathcal{A},bd}_{\mathsf{BDHGen}}(1^k, 1^n, 1^m)] \ ,$$

$$\mathsf{Adv}^{\mathrm{prf},bd}_{\mathsf{BDHGen}}(\mathcal{A}, k, n, m) := 2 \cdot \Pr[\mathrm{PRF}^{\mathcal{A},bd}_{\mathsf{BDHGen}}(1^k, 1^n, 1^m)] - 1 \ ,$$

where all probabilities are over the internal randomness used in the game, BDHGen, and by $\mathcal{A}$. We say that a hash function generator is one-way, second-preimage resistant, or collision resistant (with backdoor, in case of $bd = 1$), if the corresponding advantage of all PPT adversaries $\mathcal{A}$ is negligible.

Below, we formally define what it means for a hash function to be backdoored in a way that knowledge of a backdoor key allows for breaking certain security properties. More formally, we denote by a parameter c a backdoor capability, i.e., the security property of hash functions that a backdoor targets. For instance we can have $c \in \{\mathrm{cr}, \mathrm{ow}, \mathrm{spr}, \mathrm{prf}\}$ for the security games defined above or any other security goal for hash functions. Intuitively, a c-backdoored hash function is a hash function with a backdoor key that enables breaking the property c with a *significant* (a.k.a. large) probability. The larger the probability, the more powerful is the backdoor. In particular the attacks that we study in Chapter 4 have a success probability of 1. Note that a backdoor-resilient hash function is, however, a hash function that is hard to break with a *non-negligible* probability for any efficient adversary.

Definition 3.2 (c-Backdoored Hash Function)**.** *Let* BDHGen *be a* PPT *backdoored hash function generator and let* c *denote the backdoor capability. We call* BDHGen *a* c-backdoored hash function generator *(and its output a* c-backdoored hash function*), if there is a* PPT *adversary* $\mathcal{A}$ *such that the advantage* $\mathsf{Adv}^{c,1}_{\mathsf{BDHGen}}(\mathcal{A}, k, n, m)$ *is significant. At the same time, however, for all* PPT *adversaries* $\mathcal{A}$ *without* bk *the advantage* $\mathsf{Adv}^{c,0}_{\mathsf{BDHGen}}(\mathcal{A}, k, n, m)$ *is negligible.*

Restricted key space. As mentioned before, the key space K returned by BDHGen can be a strict subset of $\{0, 1\}^k$. Then the backdoor bk is supposed to break only those instances of the hash function with iv's included in the set K. In particular, this set can even contain just a single element, in other words, a fixed public iv, which captures a very common setting in hash functions used in practice. However, we show later in Chapter 4 that backdoored hash functions according to the stronger notion of *key-independent* backdoors exist, which enable attacks for keys chosen uniformly at random from the set of all possible strings, i.e., where $K = \{0, 1\}^k$.

3.3 Backdoored Random Oracles

The random oracle (RO) methodology [BR93] has been very influential in designing secure and practical cryptosystems (see Section 2.4 for some background). In this section we introduce a substantially weakened RO model where an adversary, on top of hash values, can also obtain *arbitrary functions* of the table of the hash function, which is modeled as a random oracle. We formalize this capability via access to a *backdoor oracle* BD that on input a function f returns $f(\mathsf{H})$, i.e., arbitrary auxiliary information about the function table of the hash function H. In particular the queries to BD do not have to be efficiently computable. We call this the *backdoored random-oracle* (BRO) model. Here, a family of backdoored hash functions is simply the set of *all* possible functions (on a predefined domain and co-domain) together with their backdoor oracles.

Our model captures backdoors that are powerful enough to allow for point inversions—simply hardwire the point y that needs to be inverted into a function f_y that searches for a preimage of y under H—or finding collisions. But they can go well beyond such attacks. For example, although Liskov [Lis07] proves one-way security of the combiner $\mathsf{H}(0\|x_1\|x_2)\|\mathsf{H}(1\|x_2\|x_1)$ under *random* inversions, this construction becomes insecure when inverted points are not assumed to be random: given $y_1\|y_2$ simply look for an inverse $0\|x_1'\|x_2'$ for y_1 such that $1\|x_2'\|x_1'$ also maps to y_2.

Backdoor functions. A backdoor function for $\mathsf{H} \in \mathrm{Fun}[n, m]$ is a function $f : \mathrm{Fun}[n, m] \to \{0, 1\}^\ell$. A backdoor capability class $\mathcal{F}$ is a set of such backdoor functions. The unrestricted class contains all functions from $\mathrm{Fun}[n, m]$ to $\{0, 1\}^\ell$. But the capability class can be also restricted, for example, functions f_y for $y \in \{0, 1\}^m$ whose outputs x are restricted to be in $\mathsf{H}^-(y)$, where $\mathsf{H}^-(y)$ is the set of all preimages of y under H. If randomness is desired in computations, it can be hardwired in the function description.

The basic BRO model. In the BRO model, a random function $\mathsf{H} \twoheadleftarrow \mathrm{Fun}[n, m]$ is sampled. All parties are provided with oracle access to H. Adversarial parties are additionally given access to the oracle

$$\underline{\mathsf{BD}(f)}: \ \textbf{return } f(\mathsf{H})$$

for $f \in \mathcal{F}$. Formally, we denote this model by $\mathrm{BRO}[n, m, \mathcal{F}]$ but will omit $[n, m, \mathcal{F}]$ when it is clear from the context. When $\mathcal{F} = \emptyset$, we recover the conventional RO model. In the BRO model, we only consider functions with fixed input lengths, i.e., a domain of $\{0, 1\}^n$ for some $n \in \mathbb{N}$. One can easily extend the model to other domains.

In a setting, where the adversarial parties call the backdoor oracle only once and before any hash queries, we recover random oracles with auxiliary input, i.e., the AI-RO model [CDGS18, Definition 2]. Hence, BRO also models oracle-dependent auxiliary input or pre-computation attacks on hash functions, which is an active area of research [Unr07, DGK17, CDGS18, CDG18], as special cases. Since BD calls can be adaptive, salting the BROs does not help in our setting at all. Indeed, with a single hash function which provides arbitrary backdoor capabilities no secure construction can exist, as any construction C^{H} can be easily inverted by a function that sees the entire H and searches for inversions under the construction C^{H}.

The k-BRO model. In practice it is natural to assume that independent hash functions are available. We can easily model this by an extension to the *k-BRO* model, whereby k independent ROs and their respective backdoor oracles are made available. The interpretation in our setting is that different authorities have, independently of each other, designed and made public hash functions that display good (i.e., RO-like) behaviors, but their respective backdoors enable computing any function of the hash tables.

Our positive results in Chapters 6 and 7 are in the k-BRO model, i.e., they rely on combining two or more independent BROs. In the k-BRO model (with the implicit parameters $[n_i, m_i, \mathcal{F}_i]$ for $i = 1, \ldots, k$) access to k independent random oracles $\mathsf{H}_i \in \mathrm{Fun}[n_i, m_i]$ and their respective backdoor oracles BD_i with capabilities $\mathcal{F}_i$ are provided. That is, procedure $\mathsf{BD}_i(f)$ returns $f(\mathsf{H}_i)$. In this thesis we are primarily interested in the 1-BRO, 2-BRO, and 3-BRO models with *unrestricted* $\mathcal{F}$.

We observe that the $2\text{-BRO}[n, m, \mathcal{F}_1, n, m, \mathcal{F}_2]$ model is identical to the $1\text{-BRO}[n+1, m, \mathcal{F}]$ model where for $\mathsf{H} \in \mathrm{Fun}[n+1, m]$ we define $\mathsf{H}_1(x) := \mathsf{H}(0\|x)$, $\mathsf{H}_2(x) := \mathsf{H}(1\|x)$ and $\mathcal{F}$ to consist of two types of functions: those in $\mathcal{F}_1$ and dependent on values $\mathsf{H}(0\|x)$, that is the function table of H_1, only, and those in $\mathcal{F}_2$ and dependent on values of $\mathsf{H}(1\|x)$, that is the function table of H_2, only. Thus the adversary in the unrestricted 2-BRO model has less power than in the unrestricted 1-BRO model.

3.3.1 Security Notions in the BRO Model

We recall the basic notions of one-wayness, pseudorandomness, and collision resistance for a construction $\mathsf{C}^{\mathsf{H}_1, \mathsf{H}_2}$ in the 2-BRO model in Figure 3.2. These security notions can be easily defined in the more general k-BRO model. For $k > 2$ we simply give the adversary access to the additional H_i and BD_i oracles and for $k = 1$ we remove access to H_2 and BD_2. These notions can also be defined in the random-oracle model without backdoors by simply removing access to the backdoor oracles

```
Game $\mathrm{OW}_{\mathsf{C}}^{\mathcal{A}}$
for $i = 1, 2$ do
    $\mathsf{H}_i \twoheadleftarrow \mathrm{Fun}[n_i, m_i]$
$x \twoheadleftarrow \{0,1\}^n$
$y \leftarrow \mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}(x)$
$x' \twoheadleftarrow \mathcal{A}^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{BD}_1,\mathsf{BD}_2}(y)$
return $(\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}(x') = y)$
```

```
Game $\mathrm{PRG}_{\mathsf{C}}^{\mathcal{A}}$
for $i = 1, 2$ do
    $\mathsf{H}_i \twoheadleftarrow \mathrm{Fun}[n_i, m_i]$
$y_0 \twoheadleftarrow \{0,1\}^m$
$x \twoheadleftarrow \{0,1\}^n$
$y_1 \leftarrow \mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}(x)$
$b \twoheadleftarrow \{0,1\}$
$b' \leftarrow \mathcal{A}^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{BD}_1,\mathsf{BD}_2}(y_b)$
return $(b' = b)$
```

```
Game $\mathrm{CR}_{\mathsf{C}}^{\mathcal{A}}$
for $i = 1, 2$ do
    $\mathsf{H}_i \twoheadleftarrow \mathrm{Fun}[n_i, m_i]$
$(x_1, x_2) \leftarrow \mathcal{A}^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{BD}_1,\mathsf{BD}_2}$
$y_1 \leftarrow \mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}(x_1)$
$y_2 \leftarrow \mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}(x_2)$
return $(x_1 \neq x_2 \wedge y_1 = y_2)$
```

Figure 3.2: The one-way, pseudorandomness, and collision resistance games for a function $\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2} \in \mathrm{Fun}[n, m]$.

BD_1 and BD_2. The advantage terms are

$$\mathsf{Adv}^{\mathrm{ow}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{A}) := \Pr[\mathrm{OW}^{\mathcal{A}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}], \qquad \mathsf{Adv}^{\mathrm{prg}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{A}) := 2 \cdot \Pr[\mathrm{PRG}^{\mathcal{A}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}] - 1,$$

$$\mathsf{Adv}^{\mathrm{cr}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{A}) := \Pr[\mathrm{CR}^{\mathcal{A}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}] \ .$$

The above probabilities are also taken over random choices of H_1 and H_2. Informally, $\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}$ is OW, PRG, or CR if the advantage of any adversary $\mathcal{A}$ querying its oracles, such that the total length of the received responses remains "reasonable", is "small". *Weak* security in each case means that the corresponding advantage is less than 1, i.e., not overwhelming. Contrary to the standard-model security notions, it is not necessary here to additionally require security without backdoors, since the security of reasonable combiners, and in particular those that we will consider in this thesis, is well-known (and often trivial) in the RO model.

We denote by $\mathsf{Q}(\mathcal{A})$ the number of oracle queries made by the adversary $\mathcal{A}$ to H_1 and H_2 as well as to BD_1 and BD_2. Note that if one only considers backdoor functions with 1-bit output lengths, the total length of the backdoor oracle responses directly translates to the number of backdoor queries made by $\mathcal{A}$.

Indifferentiability in k-BRO. Next we define indifferentiability in the k-BRO model. Looking ahead, in Chapter 7 we give positive results in the 2-BRO and the 3-BRO models. For the classical definition of indifferentiability we refer to Section 2.4.1. In the k-BRO model the underlying honest interfaces are k random oracles H_i and their respective adversarial interfaces BD_i. A simulator needs to simulate both type of interfaces. We emphasize that the simulators do not get access to any backdoor oracles. This ensures that any attack against a construction with backdoors translates to one against the random oracles *without* any backdoors. In the backdoored setting, we define the indifferentiability advantage of an adversary $\mathcal{D}$ with respect to a construction $\mathsf{C}^{\mathsf{H}_1,\ldots,\mathsf{H}_k}$ and a simulator $\mathsf{Sim} := (\mathsf{SimH}_1, \ldots, \mathsf{SimH}_k, \mathsf{SimBD}_1, \ldots, \mathsf{SimBD}_k)$ as

$$\mathsf{Adv}^{\mathrm{indiff}}_{\mathsf{C}^{\mathsf{H}_1,\ldots,\mathsf{H}_k},\mathsf{Sim}}(\mathcal{D}) := \left| \Pr\left[\mathcal{D}^{\mathsf{C}^{\mathsf{H}_1,\ldots,\mathsf{H}_k},\mathsf{H}_1,\ldots,\mathsf{H}_k,\mathsf{BD}_1,\ldots,\mathsf{BD}_k}\right] - \Pr\left[\mathcal{D}^{\mathsf{RO},\mathsf{SimH}_1^{\mathsf{RO}},\ldots,\mathsf{SimH}_k^{\mathsf{RO}},\mathsf{SimBD}_1^{\mathsf{RO}},\ldots,\mathsf{SimBD}_k^{\mathsf{RO}}}\right] \right| ,$$

where RO is a random oracle whose domain and co-domain match those of C.

3.3.2 Further Remarks on the Model

Backdoors as weaknesses. A number of works [Lis07, HS08, LW15, NIT08, KNTX10] introduce an intermediate weakened RO model, where hash functions are vulnerable to strong forms of attack, but are otherwise random. This is an approach that we also adopt in defining the BRO model, albeit allowing arbitrary weaknesses. One of the main motivations for the works of Liskov [Lis07] and Hoch and Shamir [HS08] is the study of design principles for symmetric schemes that can offer protections against weaknesses in their underlying primitives. For example, Hoch and Shamir study the failure-friendly double-pipe hash construction of Lucks [Luc05]. Similarly, Liskov shows that his *zipper hash* is indifferentiable from a random oracle even with an inversion oracle for its underlying compression function. Proofs of security in the unrestricted BRO model would strengthen these results as they place weaker assumptions on the types of weaknesses that are discovered. Furthermore, Katz, Lucks, and Thiruvengadam [KLT15] study the construction of collision-resistant hash functions from ideal ciphers that are vulnerable to differential related-key attacks. We leave the study of backdoored ideal ciphers for future work.

Auxiliary inputs. As mentioned above, a closely related model to BRO is the AI-RO model, introduced by Unruh [Unr07] and recently refined by Dodis, Guo, and Katz [DGK17] and Coretti et al. [CDGS18], and also adapted to other ideal models by Coretti et al. [CDG18]. Here the result of a one-time preprocessing attack with access to the full table of the random oracle is available to the adversary. The BRO and AI-RO models are similar in that they both allow for arbitrary functions of the random oracle to be computed. However, BRO allows for adaptive, instance-dependent auxiliary information, whereas the AI-RO model only permits a one-time access at the onset. Thus AI-RO is identical to BRO when only a single BD query at the onset is allowed. Extension to multiple ROs can also be considered for AI-ROs, where independent preprocessing attacks are performed on the hash functions. A corollary is that any positive result in the k-BRO model would also hold in the k-AI-RO model. Results in k-AI-RO model can be proven more directly using the decomposition of high-entropy densities as the setting is non-interactive. In more details, we study the relationship between BRO and AI-RO models with the goal of achieving indifferentiability from a conventional RO later in Section 7.5.

Feasibility in 1-BRO. As already discussed, any construction in the 1-BRO model is insecure with respect to arbitrary backdoors. We can, however, consider a model where backdoor capabilities are restricted. Security in such models will depend on the exact specification of backdoor functionalities $\mathcal{F}$. For example, under random inversions positive results can be established using standard lazy sampling techniques. But another natural choice is to consider functions which output possibly adversarial preimages, i.e., functions f_y whose outputs are restricted to those x for which $\mathsf{C}^{\mathsf{H}}(x) = y$. As we saw for Liskov's construction, under such generalized inversions provably secure constructions can fail. Moreover, proving security under general inversions seems to require techniques from communication complexity as we do in this thesis. A very interesting approach is taken by Golovnev et al. [GGH$^+$19, GGH$^+$20], where instead of a backdoor oracle, the (on-line) adversary is given access to some auxiliary information, which depends on a very large portion of the table of the random oracle but not all of it. Their model is stronger than the 1-BRO model, since the backdoor oracle not only sees the entire table of the function, but it can also be accessed by the adversary at any time during a security game. Golovnev et al. use these restrictions on the adversarial power together with the assumed (i.e., not unconditional) hardness of a preprocessing variant of the 3SUM problem to build a one-way function which is secure against adversaries with massive auxiliary information about the random oracle.

3.4 Infeasibility of Securely Combining Backdoored Hash Functions in Standard Model

A common approach to building a good hash function from a number of possibly "faulty" hash functions is to combine them [Leh10]. For instance, given k hash functions $\mathsf{H}_1, \ldots, \mathsf{H}_k$, the classical concatenation combiner is guaranteed to be collision resistant as long as at least one of the k hash functions is collision resistant. More formally, a black-box collision-resistance combiner C is a pair of oracle algorithms $(\mathsf{C}^{\mathsf{H}_1,\ldots,\mathsf{H}_k}, \mathsf{R}^{\mathcal{A}})$ where $\mathsf{C}^{\mathsf{H}_1,\ldots,\mathsf{H}_k}$ is the construction and $\mathsf{R}^{\mathcal{A}}$ is a reduction that given oracle access to any algorithm $\mathcal{A}$ that finds a collision for $\mathsf{C}^{\mathsf{H}_1,\ldots,\mathsf{H}_k}$, returns collisions for *all* of the underlying hash functions $\mathsf{H}_1, \ldots, \mathsf{H}_k$. We are, however, interested in a setting, where *none* of the available hash functions is secure. Under this assumption, a secure hash function must be built from scratch, implying that the source of cryptographic hardness must lie elsewhere.

Here, we briefly explore the difficulty of building a secure construction from backdoored hash functions in the standard model. Following the model of backdoored hash functions from Section 3.2, we assume that our hash functions are weak due to the existence of backdoor keys co-designed with the hash functions. We denote by iv the initialization vector (i.e., the publicly known hash key) used for hashing and by bk the backdoor key which enables an unspecified backdoor capability (such as finding preimages or collisions). Our hardness assumption is that the hash function with IV iv is collision resistant without access to bk. However, when bk is available, no security is assumed. In this setting the definition of a combiner can be simplified: instead of requiring the existence of a reduction $\mathsf{R}^{\mathcal{A}}$ as above, we can proceed in the standard way and require that the advantage of any adversary $\mathcal{A}(B)$ that gets any subset $B \subset \{\mathrm{bk}_1, \ldots, \mathrm{bk}_k\}$ of size $|B| < k$ to be small.[1] Let us call a combiner which is secure against any set of at most b backdoors a $(k-b, k)$-combiner.

It is trivial to see that a $(0, k)$-combiner is also a $(1, k)$-combiner. It is also easy to see that a black-box combiner is a $(1, k)$-combiner. We are, however, interested in the feasibility of $(0, k)$-combiners. In this setting constructions that have to work with a *provided* hash function family (and potentially a given set of iv's) seem hard. Without this restriction, a trivial construction exists: generate a good backdoor-free hash function family, sample a fresh hash function from it and simply use that one. In practice, however, this requires designing a hash function from scratch and it is unclear if the designer can be trusted. Thus we assume that for any sampled hash function its backdoor is also available. We next give a simple impossibly result that formalizes this intuition under fully black-box constructions.

[1] The classical (backdoor-free) setting can be viewed as one where bk's are *fixed*, which leads to a difficulty when the new definition is used: a combiner (formally speaking) can "detect" which hash functions are the good ones and use them. Since this detection procedure is not considered practical, one instead asks for the existence of a reduction R as discussed above.

Theorem 3.1. *For any positive $k \in \mathbb{N}$, there are no fully black-box constructions of compressing collision resistant $(0, k)$-combiners.*

Proof idea. Let H be a hash function family such that $\mathsf{H}(\mathrm{iv}, \cdot)$ implements a random function and let $\mathcal{A}$ be an algorithm which on inputs C, $\mathrm{iv}_1, \dots, \mathrm{iv}_k$, and $\mathrm{bk}_1, \dots, \mathrm{bk}_k$ operates as follows. It first interprets C as the description of a combiner. It then checks that each bk_i indeed enables generating collisions under $\mathsf{H}(\mathrm{iv}, \cdot)$. If so, it (inefficiently) finds a random collision for $\mathsf{C}^{\mathsf{H}(\mathrm{iv}_1,\cdot),\dots,\mathsf{H}(\mathrm{iv}_k,\cdot)}$ and returns it. An efficient reduction R is given oracle access to $\mathcal{A}$ and H as well as a key iv^* (without its backdoor bk^*). It should find a collision for $\mathsf{H}(\mathrm{iv}^*, \cdot)$ while making a small (below birthday bound) number of queries to the two oracles $\mathcal{A}$ and H. We show that any such reduction R must have a negligible success probability.

We distinguish between two cases based on whether the reduction R uses the provided oracle $\mathcal{A}$ or not. Without the use of $\mathcal{A}$, the reduction would break collision-resistance for iv^* on its own, contradicting the collision-resistance of iv^* beyond the birthday bound. To use $\mathcal{A}$, the reduction has to provide it with k keys iv_i and some other backdoor keys bk_i that enable finding collisions (since $\mathcal{A}$ checks this). However, none of the provided keys iv_i can be iv^*, since R must also provide some $\tilde{\mathrm{bk}}^*$ that enables finding collisions under iv^*, which means that R can directly use $\tilde{\mathrm{bk}}^*$ to compute a collision for $\mathsf{H}(\mathrm{iv}^*, \cdot)$, once again contradicting the assumed collision resistance of iv^* beyond the birthday bound. Thus R does not use iv^*. A random oracle $\mathsf{H}(\mathrm{iv}^*, \cdot)$, however, is collision resistant even in the presence of random collisions for $\mathsf{H}(\mathrm{iv}, \cdot)$ for $\mathrm{iv} \neq \mathrm{iv}^*$. This means that R, which places a small number of oracle queries, will have a negligible success probability. □

There is room to circumvent this result by considering non-black-box constructions. In Chapter 6 we will study basic security properties of the concatenation, cascade, and xor combiners for hash functions in the *unrestricted* 2-BRO model, where the hash oracles model access to different iv and the backdoor oracles model access to the corresponding bk's. As mentioned in the previous section, the approach of modeling weaknesses through a break oracle to a random oracle has also been adopted in a number of prior work, both from a provable security as well as a cryptanalytic view [HS08, Mit13, KNTX10, LW15, Jou04].

Chapter 4

Threat of Backdoored Hash Functions

This chapter starts with disproving a misconception about backdoored hash functions. One may believe that building a backdoored hash function, which is secure without the backdoor key but insecure with it, would imply public-key primitives and is, hence, not only slow but also highly suspicious. In other words, this hypothesis declares any symmetric hash function as inherently backdoor-free. We contradict this belief by giving constructions of a compression function based on many-to-one trapdoor one-way functions with an exponential preimage size as studied by Bellare et al. [BHSV98]. We then show that constructions of hash functions with variable-length inputs which iterate such a backdoored compression function using the Merkle–Damgård or the sponge transforms are also backdoored. Furthermore, we show at the example of HMAC that using a secret key does not automatically prohibit such attacks. These examples serve a better understanding of how backdoors may look like and which security implications they may have. This is an important step towards defeating backdoors, not only by showcasing a concrete backdoor type and suspicious design choices to watch for but more so because they guide us in searching for countermeasures.

My Scientific Contribution in this Chapter

The material in this chapter is published in [FJM18a] and its full version [FJM18b], which are joint work with Marc Fischlin and Christian Janson. The idea of basing backdoored compression functions on trapdoor functions with exponential preimage size, described in Section 4.1, was suggested my me. Christian and I then jointly applied this idea to hash functions and analyzed the attacks. I focused mainly on the Merkle–Damgård-based construction in Section 4.2. Christian mainly worked on the sponge-based construction in Section 4.3, which appeared in the full version [FJM18b, Section 8].

4.1 Introduction

It may appear that a backdoored hash function implies trapdoor permutations (or equivalently public-key encryption), especially as it does in case of backdoored pseudorandom generators studied by Dodis et al. [DGG$^+$15]. In particular one might believe that backdoored hash functions that allow for finding collisions are the same as *chameleon* hash functions, which were introduced by Krawczyk and Rabin [KR98] and are believed to require public-key primitives [BR14]. If such an equivalence were true, one could safely assume that common constructions of hash functions, which are symmetric-key confusion/diffusion-style primitives, are backdoor-free. We show, however, that for hash functions (without secret keys) and (strong) pseudorandom functions this is not necessarily true. To this end we give backdoored constructions based on standard collision-resistant and one-way functions used to construct a many-to-one trapdoor function with an exponential preimage size, which are studied by Bellare et al. [BHSV98]. For a definition of such trapdoor functions see Section 2.3.3.

Our constructions show clearly that backdoored hash functions can exist outside cryptomania, i.e., a world described by Impagliazzo [Imp95], where public-key cryptography exists. From an efficiency point of view, backdoored hash functions can even be as fast as non-backdoored functions.

On a high level, a malicious designer can build a backdoored compression function (a.k.a. a compressing hash function that works on fixed-length inputs) from an arbitrary and secure compression function, where the backdoored function basically “mimics” the behavior of the healthy function unless a backdoor key, which is a special bit string, is embedded as part of the input. For this exceptional input the altered function returns something trivial, e.g., a part of its input. In other words, the backdoor key triggers a malicious behavior in the backdoored compression function. It is noteworthy that the backdoor can only be triggered by an adversary with prior knowledge of the backdoor key, since it is cryptographically protected in the construction.

Furthermore we show that iterating such a backdoored function in the Merkle–Damgård or sponge construction leads to a backdoored hash function with variable input lengths. Since the inputs to hash functions can be chosen by the adversary during the attacks, finding collisions, preimages, and second preimages becomes easy by triggering the backdoor. Furthermore, we show that even though HMAC uses a secret key, it is not a secure pseudorandom function against an adversary that can exploit the backdoor for the underlying hash function. This enables the adversary to find collisions in the inner hash chain, which is exactly what makes it possible to forge MAC tags and distinguish outputs from random.

We do not claim that any of today’s widely used hash functions such as SHA-1, SHA-2, or SHA-3 actually have backdoors. The constructions discussed in this chapter are mainly theoretical results meant to increase our understanding of backdoors in hash functions and their complexity, as well as eliminating a common misconception that backdoored hash functions necessarily have to be based on public-key primitives. We note that, although it is infeasible to derive the backdoor key from the specification, our constructions do look suspicious. As discussed in Section 1.1, other types of backdoors in hash functions have been studied, e.g., by Albertini et al. which are less obvious but can only give one single collision [AAE$^+$14].

4.2 Merkle–Damgård-based Hash Functions and HMAC

In this section we demonstrate the feasibility of embedding a backdoor in ordinary hash functions, in a way that the adversary in possession of the backdoor is able to undermine the most crucial security properties of the hash function. What makes the threat even bigger, is that the hash function retains all those security properties against adversaries that do not know the backdoor key. Hence, the black-box behavior of the hash function does not raise any suspicion about existence of a backdoor.

Moreover, the specification only uses symmetric-key primitives and still resists reverse engineering attempts by cryptographically hiding the backdoor key. Our construction resembles many-to-one trapdoor one-way functions with an exponential preimage size. Bellare et al. [BHSV98] studied such trapdoor one-way functions and showed that they can be built from ordinary one-way functions (see Section 2.3.3) and, hence, building secure public-key encryption from them is presumably hard.

We discuss a backdoored Merkle–Damgård-based hash function which iterates a backdoored compression function, i.e., a backdoored compressing hash function for fixed-length inputs. This backdoored compression function behaves like a secure compression function unless the backdoor is triggered by a special key as part of the input message to the function. The construction sadly gives evidence of how easy it is to embed a backdoor into the building block of hash functions and violate their properties. We also investigate whether our construction has any weakening impact when used in HMAC (where HMAC is used as is, without our immunization modifications). Unfortunately we have to answer this in the affirmative and show that even though HMAC uses a secret key, it is not secure, since the adversary has full control over the input message and can trigger the backdoor for the underlying hash function.

4.2.1 The Merkle–Damgård Construction

The Merkle–Damgård construction [Dam90, Mer90] is one of the most commonly used approaches for building a full-fledged hash function with arbitrary input length. Such constructions are also referred to as domain extenders. The MD-construction works by iterating a compression function, processing a single block of the input message in each iteration and using padding techniques to make the entire input message length comply with the block length. In terms of security, for appropriate paddings, the collision resistance of the iterated compression function is preserved. The MD domain extender is extensively used in practice for hash functions including the MD family, SHA-1 and SHA-2, while each one employs a different compression function.

Let $\mathsf{h}\colon \{0,1\}^\ell \times \{0,1\}^b \to \{0,1\}^\ell$ be a compression function. The MD-based hash function family $\mathsf{H}^{\mathsf{h}}_{\mathrm{md}} := \{\mathsf{H}^{\mathsf{h}}_{\mathrm{md,iv}} : \{0,1\}^{\leq 2^p} \to \{0,1\}^\ell \mid \mathrm{iv} \in \{0,1\}^\ell\}$ iterating h is described in Figure 4.1. Here, an input message x is first padded such that its length becomes a multiple of the block size b that is processable by the compression function. The padded message is then partitioned into blocks $x_0, x_1, \ldots, x_{n-1}$, where each message block is of size b. Below we discuss the padding function in more detail. Then, the compression function h is iterated in such a way that the output of the previous compression function and the next message block become the input to the next compression function. The iteration starts with an initialization value $\mathrm{iv} \twoheadleftarrow \{0,1\}^\ell$ and the first message block x_0 and the hash digest is the last output of h after consuming all message blocks.

```
$\mathsf{H}^{\mathsf{h}}_{\mathrm{md,iv}}(x)$
$x \leftarrow x \| \mathsf{lpad}(x, b, p)$
parse $x$ as $x_0 \| \ldots \| x_{n-1}$ where $|x_i| = b$ for all $0 \leq i < n$
$y_0 \leftarrow \mathrm{iv}$
for $i = 0 \ldots n-1$ do
    $y_{i+1} \leftarrow \mathsf{h}(y_i, x_i)$
$y \leftarrow y_n$
return $y$
```

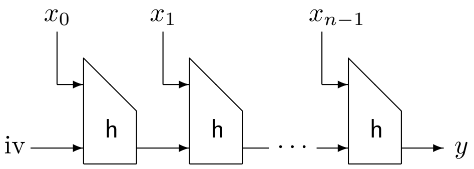


Figure 4.1: Merkle–Damgård construction from a compression function h.

Length padding. The padding algorithm used with a domain extender must itself be collision free. Length padding is typically used in MD-based hash functions. It appends the length of the message to the end, while making sure that the length of the padded message is a multiple of the block size b required by the compression function. We consider a compact length padding function lpad that uses p bits to represent the message length, where $p < b$ so that the length fits into one block and also $p \leq \ell$, which is also common for practical hash functions (e.g., $p = 64$ for SHA-256). Hence, the padded message contains the message length in its last b-bits block possibly together with some of the least significant bits of the message. Such a length padding function is commonly used in practical MD-based hash functions, such as MD5, SHA-1 and SHA-2. A similar padding that additionally prepends the length of the message with a bit of 1 is used in BLAKE. Let $\mathsf{binary}(x, y)$ be the binary representation of x in y bits, then lpad is defined as:

$$\mathsf{lpad}(x, b, p) := 1 \| 0^{(b-|x|-p-1) \bmod b} \| \mathsf{binary}(|x|, p).$$

A less compact and mostly theoretical variant of length padding uses exactly b bits for representing the message length, which is then encoded in a separate block.

4.2.2 Backdoored MD-based Hash Functions

Let $\mathsf{h}\colon \{0,1\}^\ell \times \{0,1\}^b \to \{0,1\}^\ell$ be a collision-resistant compression function with $b \geq 2\ell$. Consider an algorithm BDHGen which creates a backdoored compression function $\tilde{\mathsf{h}}\colon \{0,1\}^\ell \times \{0,1\}^b \to \{0,1\}^\ell$ and its backdoor key $\mathrm{bk} \in \{0,1\}^\ell$, as defined below. In order for $\tilde{\mathsf{h}}$ to be collision resistant against adversaries without knowledge of the backdoor, $\tilde{\mathsf{h}}$ behaves essentially like h, unless the backdoor is triggered. The backdoor bk is defined in a way that $\mathsf{h}(\mathrm{bk}, a) = \beta$, for fixed values $a \in \{0,1\}^b$ and $\beta \in \{0,1\}^\ell$. The values bk and a can for instance be chosen randomly by the malicious designer of $\tilde{\mathsf{h}}$, from which β is determined.

compression function :

$$\tilde{\mathsf{h}}\colon \{0,1\}^\ell \times \{0,1\}^b \to \{0,1\}^\ell$$

$$\tilde{\mathsf{h}}(y, x) := \begin{cases} x_{[\ell, 2\ell-1]}, & \text{if } \mathsf{h}(x_{[0,\ell-1]}, a) = \beta \\ \mathsf{h}(y, x), & \text{otherwise .} \end{cases}$$

backdoor key :

any $\mathrm{bk} \in \{0,1\}^\ell$
such that $\mathsf{h}(\mathrm{bk}, a) = \beta$.

Figure 4.2: Backdoored compression function $\tilde{\mathsf{h}}$ and its backdoor key bk, built from any compression function $\mathsf{h}\colon \{0,1\}^\ell \times \{0,1\}^b \to \{0,1\}^\ell$ with $b \geq 2\ell$.

The backdoor key is a special point bk of admissible inputs of h. The malicious compression function $\tilde{\mathsf{h}}$ simply evaluates h on most inputs, except if it detects a backdoor key in the input. Let us take a closer look at the definition of $\tilde{\mathsf{h}}$. If the first ℓ bits of the processed b-bit block correspond to the backdoor value bk (or another preimage of β under $\mathsf{h}(\cdot, a)$), then $\tilde{\mathsf{h}}$ sets the output to the next ℓ bits of that input string. This allows to program the output of any iteration of $\tilde{\mathsf{h}}$ since the input message block under adversarial control in many relevant security games. For all other inputs, $\tilde{\mathsf{h}}$ simply calls the underlying compression function h on the given inputs. We describe the attacks in more detail shortly.

Proposition 4.1. *The compression function* $\tilde{\mathsf{h}}$ *given in Figure 4.2 is collision resistant if the underlying* h *is collision resistant and* $\mathsf{h}(\cdot, a)$ *is one-way (for parameter* ℓ*) for* $a \twoheadleftarrow \{0,1\}^b$.

The idea is that a collision finder can only take advantage of the embedded case if it finds a preimage for β for $\mathsf{h}(\cdot, a)$. Else, it needs to find a collision from scratch.

Proof. Suppose to the contrary that there exists a PPT adversary $\mathcal{A}$ which finds collisions for $\tilde{\mathsf{h}}$ with a non-negligible probability, i.e., outputs $(y, x) \neq (y', x')$, such that $\tilde{\mathsf{h}}(y, x) = \tilde{\mathsf{h}}(y', x')$. We make a case distinction:

- Assume that $\mathsf{h}(x_{[0,\ell-1]}, a) = \beta$ or $\mathsf{h}(x'_{[0,\ell-1]}, a) = \beta$. Then it is straightforward to build an adversary against the one-way security of $\mathsf{h}(\cdot, a)$, since $x_{[0,\ell-1]}$ resp. $x'_{[0,\ell-1]}$ constitutes a preimage for β.
- According to the other case we thus have $\tilde{\mathsf{h}}(y, x) = \mathsf{h}(y, x) = \mathsf{h}(y', x') = \tilde{\mathsf{h}}(y', x')$ such that $(y, x) \neq (y', x')$. This, however, contradicts the collision resistance of h.

In summary, an adversary $\mathcal{A}$ successfully attacking collision resistance of $\tilde{\mathsf{h}}$ can be used to build an adversary that can either find preimages for $\mathsf{h}(\cdot, a)$ or find collisions under h (in the same time). Hence, $\mathcal{A}$'s success probability is bounded by the sum of these cases. □

With a similar argument we can show that the same holds for the other properties:

Proposition 4.2. *The compression function* $\tilde{\mathsf{h}}$ *given in Figure 4.2 is one-way if the underlying* h *is one-way for parameter* $\ell + b$ *and if* $\mathsf{h}(\cdot, a)$ *is one-way for parameter* ℓ *for* $a \twoheadleftarrow \{0,1\}^b$.

As in the proof for collision resistance, this holds because an adversary $\mathcal{A}$ against one-way security either needs to find a preimage for parameter ℓ (i.e., a backdoor key), or under the original compression function h for the entire parameter $\ell + b$.

Proposition 4.3. *The compression function* $\tilde{\mathsf{h}}$ *given in Figure 4.2 is second-preimage resistant for parameter* $\ell + b$ *if the underlying* h *is second-preimage resistant for* $\ell + b$ *and* $\mathsf{h}(\cdot, a)$ *is one-way for parameter* ℓ *for* $a \twoheadleftarrow \{0,1\}^b$.

Next we build from $\tilde{\mathsf{h}}$ a backdoored hash function $\mathsf{H}^{\tilde{\mathsf{h}}}_{\mathrm{md}}$ using the standard MD domain extender, which iterates the backdoored compression function $\tilde{\mathsf{h}}$. Intuitively, with the backdoor key an adversary can trigger one or more iterations of the compression function to land in a "weak mode" and then abuse it to break the hash function, i.e., find collisions, preimages, and second preimages.

Deniability. Although it is hard to find the backdoor itself given the description of the compression function, the fact that the above construction is indeed backdoored is fairly obvious and cannot be plausibly denied. However, a big brother adversary may employ techniques such as practical obfuscation to better hide the (implicit) if-clause and the constants used for backdooring. The complexity of permutations and compression functions used in practice can make detection even harder. Furthermore, the attacker may also aim at weakening the implementation and insert malicious code in a software update. The subverted implementation remains indistinguishable from its specification via black-box testing.

Exposure of the backdoor key. As discussed, a backdoor can enable adversaries to break security of a hash function. The same backdoored construction is unexploitable by an adversary who does not know the backdoor key. Attempts at detecting a potential backdoor via black-box testing or finding the backdoor key by reverse engineering the code may easily fail. However, observe that every collision, preimage, or second preimage found using the backdoor key, encodes the backdoor key in the message. Therefore, using the backdoor may put the adversary in risk of being exposed. It is unclear whether constructions of backdoored compression functions are possible that do not expose their backdoor key in adversarial inputs and do not rely on indistinguishability obfuscation to hide a secret key in the compression function and use it to internally decrypt malicious triggers.

Attack Description

We describe how a backdoor key bk as described in Figure 4.2 enables breaking collision resistance, one-way security and second-preimage resistance of $\mathsf{H}^{\tilde{\mathsf{h}}}_{\mathrm{md}}$. Definitions of these properties were given in Section 3.2. Interestingly, for finding collisions and preimages, the adversary does not need to actually evaluate the hash function at all. For finding second preimages, the adversary evaluates the hash function only once when computing the digest of the challenge message. Here, we consider the compact length padding lpad. However, we remark that backdoored compression functions, similar to the one that we discuss below, do exist for other padding functions.

Finding collisions. The adversary's goal is to find two messages $x, x' \in \{0,1\}^{\leq 2^p}$, that are mapped to the same digest. We argue that any two messages x and x' of the same length that have an arbitrary prefix of the same size, which is a multiple of the block size b, followed by a common suffix starting with bk, collide. It immediately follows from the exponential size of the message space that the backdoor enables finding an exponential number of collisions. More precisely, for all prefixes $v, v' \in (\{0,1\}^b)^+$ with $|v| = |v'|$ and suffixes $w \in \{0,1\}^*$ the colliding messages are:

$$x = v\|\mathrm{bk}\|w \quad \text{and} \quad x' = v'\|\mathrm{bk}\|w.$$

Observe that, after padding, both messages contain the backdoor bk at the beginning of the same block. This triggers the "bad" case of the backdoored compression function, which then outputs the following ℓ bits of the message, ignoring the intermediate chaining value obtained from the previous iteration. Since the messages do not differ after the backdoor and neither does the padding of the equal length messages, their digests eventually coincide.

In a similar attack, the adversary is able to encode an targeted digest $y \in \{0,1\}^\ell$ directly in the messages. That is, for all prefixes $v, v' \in (\{0,1\}^b)^+$ of the same size (i.e., $|v| = |v'|$), and short suffixes $u, u' \in \{0,1\}^{\leq b-2\ell-p-1}$ the following messages are both mapped to the same value by the backdoored hash function:

$$x = v\|\mathrm{bk}\|y\|u \quad \text{and} \quad x' = v'\|\mathrm{bk}\|y\|u'.$$

If $b \geq 2\ell + p + 1$, the lengths of the messages fit in the same block as bk, y, and u resp. u'. Hence, the encoded length $\mathsf{binary}(|x|, p)$ resp. $\mathsf{binary}(|x'|, p)$ is simply ignored by the backdoored compression function and y is output as the final digest for both. Here, the messages do not even have to be of the same size. However, if $2\ell \leq b < 2\ell + p + 1$ the binary encoding of the message lengths cannot be included in the same block as bk and y, hence, the above messages will collide (mapped to a different digest than y) if their last blocks after padding (in particular their lengths) are equal.

Finding preimages. Let $y \in \{0,1\}^\ell$ be the digest which an adversary is challenged to invert. We distinguish two cases based on the length of message blocks. First we consider $b \geq 2\ell + p + 1$, where the backdoor key, the challenge digest, and the padded message length can all fit in a single block. Second, we consider $2\ell \leq b < 2\ell + p + 1$, where at least the backdoor key and the message length can fit into the same block, however not together with the challenge digest.

If $b \geq 2\ell + p + 1$, the adversary can encode y in the message directly after the backdoor key bk, while still leaving space in the last block for the message length to be encoded in by the padding. More precisely, for all prefixes $v \in (\{0,1\}^b)^+$ and all short suffixes $u \in \{0,1\}^{\leq b-2\ell-p-1}$ the following message x is a valid preimage for y:

$$x = v \| \mathrm{bk} \| y \| u.$$

Here, after the malicious case of the compression function is triggered by bk, the digest y will be output, and the rest of the last block, containing the length is simply ignored.

The attack is slightly more subtle in case of $2\ell \leq b < 2\ell + p + 1$. However, one can still trigger the backdoor in the last message block, causing the next ℓ bits to be output by the backdoored compression function. Consider the message $x = v \| \mathrm{bk} \| u$ now with $u \in \{0,1\}^{\leq b-\ell-p-1}$. If the ℓ bits of the padded message immediately following the backdoor key bk, i.e., u and a prefix of $\mathsf{lpad}(x, b, p)$, correspond to the challenge digest y, then x is a valid preimage.

Finding second-preimages. Finding second preimages is very similar to finding preimages. The adversary simply performs the above attacks to find a second preimage x' for a given message x, after setting $y = \mathsf{H}^{\tilde{\mathsf{h}}}_{\mathrm{md,iv}}(x)$. Note that since the adversary can find an exponential number of preimages by choosing different prefixes and suffixes, she can easily find a preimage x' of y that different from the challenge message x.

4.2.3 The HMAC Construction

Message authentication codes (MACs) provide message integrity, i.e., they prevent adversaries from tampering with a communication without being detected by the receiver. The widely used HMAC scheme [BCK96a] is built on a cryptographic hash function. It has been standardized in IETF RFC [KBC97] and NIST [FIP02], and is widely deployed in various security protocols such as TLS and IPSec. HMAC (to be precise, its theoretical counterpart NMAC) is provably a pseudorandom function under the assumption that its underlying compression function is a pseudorandom function [Bel15]. Note that PRF security implies the standard notion of unforgeability for MAC schemes: if real (and reasonably long) MAC tags cannot be distinguished from random strings, it is also infeasible to forge them.

Definition 4.1 (HMAC). *Let* $\mathsf{H}^{\mathsf{h}}_{\mathrm{md}} := \{\mathsf{H}^{\mathsf{h}}_{\mathrm{md,iv}} : \{0,1\}^{\leq 2^p} \to \{0,1\}^\ell \mid \mathrm{iv} \in \{0,1\}^\ell\}$ *be a Merkle–Damgård-based hash function family. On input of a key* $\mathrm{k} \in \{0,1\}^b$ *and an IV* $\mathrm{iv} \in \{0,1\}^\ell$*, the hash-based message authentication* $\mathsf{HMAC}^{\mathsf{h}}$ *with associated secret key space* $\{0,1\}^b$ *is a DPT algorithm* *defined as*

$$\mathsf{HMAC}^{\mathsf{h}}(\mathrm{k}, \mathrm{iv}, x) := \mathsf{H}^{\mathsf{h}}_{\mathrm{md,iv}}\big((\mathrm{k} \oplus \mathsf{opad}) \,\|\mathsf{H}^{\mathsf{h}}_{\mathrm{md,iv}}((\mathrm{k} \oplus \mathsf{ipad}) \,\|x)\big) \ ,$$

where ipad *and* opad *are fixed, distinct b-bit constants.*

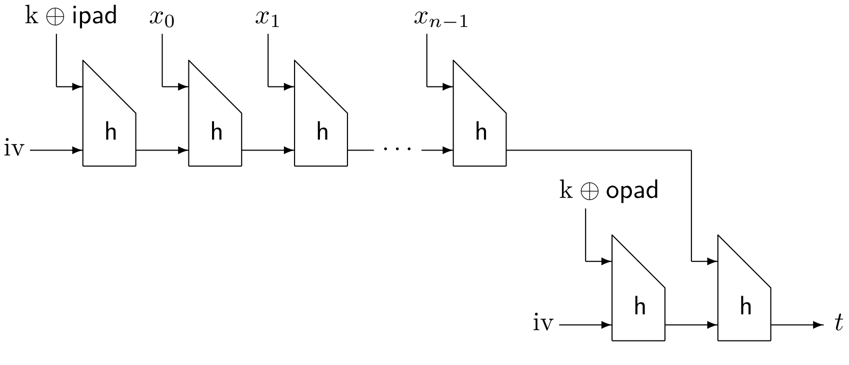


Figure 4.3: Illustration of HMAC for inputs $x \in \{0,1\}^{n \cdot b}$.

4.2.4 Backdoored HMAC

Next we show that building HMAC upon the backdoored MD-based hash function $\mathsf{H}^{\tilde{\mathsf{h}}}_{\mathrm{md}}$ of Section 4.2.2 yields a backdoored HMAC scheme, which is forgeable using the backdoor key. More precisely, the backdoored HMAC scheme is defined as $\mathsf{HMAC}^{\tilde{\mathsf{h}}}$ with a key space $\{0,1\}^b$. However, note that $\tilde{\mathsf{h}}$ is still a PRF against adversaries that do not know a backdoor key, as we prove below. Therefore, the resulting HMAC construction $\mathsf{HMAC}^{\tilde{\mathsf{h}}}$ is still a PRF against such weak adversaries while broken when a backdoor key is known.

Lemma 4.4. *The compression function* $\tilde{\mathsf{h}}$ *from Section 4.2.2 is a PRF if the underlying function* h *is a PRF and* $\mathsf{h}(\cdot, a)$ *is one-way (for parameter* ℓ*) for* $a \twoheadleftarrow \{0,1\}^b$*.*

Proof. Assume that there exist an adversary $\mathcal{A}$ with a non-negligible advantage $\mathsf{Adv}^{\mathrm{prf},0}_{\tilde{\mathsf{h}}}(\mathcal{A}, \ell, b, \ell)$ in distinguishing $\tilde{\mathsf{h}} : \{0,1\}^\ell \times \{0,1\}^b \to \{0,1\}^\ell$ from a random function with the same domain and range. We use $\mathcal{A}$ to build an adversary $\mathcal{B}$ against the PRF-security of h as follows. By definition $\mathcal{B}$ gets access to an oracle which either implements $\mathsf{h}(\mathrm{k}, \cdot)$, for a random key k, or a truly random function $f \twoheadleftarrow \mathrm{Fun}[b, \ell]$.

Initially, $\mathcal{B}$ picks random values bk, a and sets $\beta := \mathsf{h}(\mathrm{bk}, a)$. Upon receiving a query $y \in \{0,1\}^b$ from $\mathcal{A}$, our new adversary $\mathcal{B}$ simply forwards this query to its oracle and returns the answer unless $\mathsf{h}(x_{[0,\ell-1]}, a) = \beta$ is met, in which case $x_{[\ell,2\ell-1]}$ is returned. When the adversary $\mathcal{A}$ terminates with output b, then so does $\mathcal{B}$.

For the analysis note that the only difference in the answers handed to $\mathcal{A}$ lie in the exceptional case that $\mathsf{h}(x_{[0,\ell-1]}, a) = \beta$, i.e., in case it is possible to compute a preimage of β under $\mathsf{h}(\cdot, a)$ with the help of $\mathcal{A}$'s queries. This straightforwardly leads to a contradiction to the one-way security of h, via the construction of some algorithm $\mathcal{C}$ against one-way security based on $\mathcal{A}$. Otherwise, the behavior of $\mathcal{A}$ and $\mathcal{B}$ are identical. Therefore,

$$\mathsf{Adv}_{\mathsf{h}}^{\mathrm{prf},0}(\mathcal{B}, \ell, b, \ell) \geq \mathsf{Adv}_{\tilde{\mathsf{h}}}^{\mathrm{prf},0}(\mathcal{A}, \ell, b, \ell) - \mathsf{Adv}_{\mathsf{h}}^{\mathrm{ow},\ell}(\mathcal{C}, \ell, b, \ell)$$

Since h is supposed to be one-way, this contradicts the PRF security of h. Furthermore, note that it is also unlikely that computing $\tilde{\mathsf{h}}(\mathrm{iv}, \mathrm{k} \oplus \mathsf{ipad})$ or $\tilde{\mathsf{h}}(\mathrm{iv}, \mathrm{k} \oplus \mathsf{opad})$ in HMAC triggers the exceptional branch. The reason is that this could only happen in the unlikely case that the key parts constitute a preimage of the backdoor value β. Thus, the claim holds. □

Attack Description

Recall that the backdoor bk, defined in Figure 4.2, allows an adversary to find collisions for the underlying hash function $\mathsf{H}_{\mathrm{md}}^{\tilde{\mathsf{h}}}$. Finding collisions for the inner hash chain of the backdoored HMAC construction is precisely what makes forging MAC tags easy. First, the adversary queries $\mathsf{HMAC}^{\tilde{\mathsf{h}}}$ on a message $x = v \| \mathrm{bk} \| w$, where $v \in (\{0,1\}^b)^+$ and $w \in \{0,1\}^*$. After receiving the corresponding tag t, the adversary returns the pair (x^*, t) as a forgery, where $x^* := v' \| \mathrm{bk} \| w$, with any $v' \in (\{0,1\}^b)^+$, such that $v \neq v'$, and $|v| = |v'|$.

Such messages x and x^* lead to collisions under $\mathsf{H}_{\mathrm{md}}^{\tilde{\mathsf{h}}}$. Since their prefixes v and v' of equal length can be arbitrary, they can in particular start with a block of b-bits equal to $\mathrm{k} \oplus \mathsf{ipad}$. Hence it is easy to see that after the messages are prepended by $\mathrm{k} \oplus \mathsf{ipad}$, they still lead to a collision in the inner hash chain of $\mathsf{HMAC}^{\tilde{\mathsf{h}}}$. Since the outer chain is equal for all messages, x and x^* both have the same tag t. Put differently, HMAC constructions do not automatically resist backdoors just because they use a secret key. In summary, since an adversary holding a backdoor can forge a tag for a new message, $\mathsf{HMAC}^{\tilde{\mathsf{h}}}$ is forgeable and, hence, not pseudorandom.

4.3 Sponge-based Hash Functions

In this section, after a brief review of the sponge construction, we discuss how backdooring the underlying permutation can lead to a backdoored sponge-based hash function.

4.3.1 The Sponge Construction

The *sponge* construction was introduced by Bertoni et al. [BDPVA11]. As illustrated in Figure 4.4, the sponge construction is, similar to the MD construction, of iterative nature, but it works in two phases: *absorbing* messages and *squeezing* the digest. The sponge construction can be used to build a function with variable-length input and output based on a permutation p operating on a fixed number of bits b, called width. The sponge construction operates on a state of size $b = r + c$ bits. The value r is the bit rate and denotes the length of the first part of the state, where message blocks of length r are absorbed. The value c is called the capacity and describes the length of the remaining state, and generally it holds that $r > c$. The construction starts with the state being initialized with a string of length b. The message x is first padded to a multiple of r bits and then divided up into blocks of this length. In the absorbing phase, the message blocks are simply xored into the first r bits of the state and then the permutation p is applied to the entire state. After all message blocks

```
$\mathsf{H}^{\text{sponge}}_{p,\text{iv}}(x)$
$s_0 \leftarrow \text{iv}$
$x \leftarrow x \| \mathsf{pad}_{10^*}(x, r)$
parse $x$ as $x_0 \| \dots \| x_{n-1}$ where $|x_i| = r$ for all $0 \le i < n$
for $i = 0 ... n-1$ do
    $\widetilde{s_i} \leftarrow s_i \oplus (x_i \| 0^c)$
    $s_{i+1} \leftarrow p(\widetilde{s_i})$
$y \leftarrow s_{n_{[0,r]}}$
while $|y| < \ell$ do
    $s_n \leftarrow p(s_n)$
    $y \leftarrow y \| s_{n_{[0,r]}}$
return $y_{[0,\ell-1]}$
```

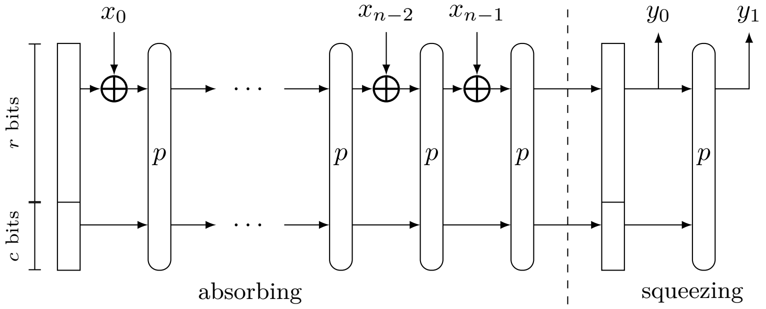


Figure 4.4: Sponge-based hash function with output-length ℓ for inputs $x \in \{0,1\}^{n \cdot r}$. Illustration is a modification of [Jea16].

were processed, the construction switches to the second phase, the so-called squeezing. Here the first r bits of the state are returned as output blocks and the permutation p is applied to the state. The squeezing process is repeated until the desired number of blocks are returned. We define sponge constructions of a fixed (but arbitrary) output length ℓ. Note that the capacity of the state is never directly affected by the input message blocks and is not returned during the squeezing phase.

The sponge construction is used in building the SHA-3 hash function which is part of the cryptographic hash function family known as Keccak [BDPVA09]. A sponge-based hash function family $\mathsf{H}^p_{\text{sponge}} := \{\mathsf{H}^p_{\text{sponge,iv}} : \{0,1\}^{\le \mathsf{poly}(b)} \to \{0,1\}^\ell \mid \text{iv} \in \{0,1\}^b\}$, as formally described in Figure 4.4, is constructed by iterating a permutation $p \colon \{0,1\}^b \to \{0,1\}^b$ as follows. The initial state s of the sponge is set to an initialization vector $\text{iv} \twoheadleftarrow \{0,1\}^b$. However, note that we use a more general terminology and usually in SHA-3 the state is simply initialized with the zero bit string of length b. On the input of a message x, the padding follows the standard convention and appends to the message a bit string of the form 10^* to form a message with a length that is a multiple of the block size r. More formally the padding function is defined as

$$\mathsf{pad}_{10^*}(x, r) := 10^{(r-1-|x|) \bmod r} \ .$$

A length padding is not necessary. Similar to Section 4.2.1, we remark that such a padding function is itself collision free. The padded message is divided into blocks of size r. Finally, the permutation p is iteratively applied to the state which is composed by xoring the current state with the current message block (and a zero string of length c to form an admissible input for the permutation). As soon as the whole message is absorbed, the first r bits of the state can be returned as the digest in the squeezing phase. In case the length of the digest is smaller than required, the permutation is applied further to produce r more bits. This is repeated until the required length of ℓ bits is obtained, and in case the digest is a few bits too long one can simply truncate the string to exactly ℓ bits.

4.3.2 Backdoored Sponge-based Hash Functions

A backdoored sponge-based hash function cannot directly apply the same “trick” as backdoored MD-based hash functions from Section 4.2, where both the backdoor and the digest could enter the same message block. Since the message blocks in the sponge construction can potentially be of the same length or even smaller than the digest, we use a different encoding here. We consider a backdoor key which is a string of length r that contains a “sub-backdoor” of length $c < r$ in its prefix. The basic idea is that after the adversary has triggered the backdoor in a message block, the backdoored function simply programs the sub-backdoor into the lowest c bits of the output state. This signals to the next iteration of the sponge that the absorbing phase is about to be finalized and, hence, enables the backdoored function to encode the desired digest into the next message block (upper state). Whenever, a backdoor is recognized in the lower state of its input, the backdoored function simply let’s the upper r bits of the state pass through. In other words, the adversary can precisely encode an arbitrary digest into the message blocks which will be simply output after the backdoor was triggered and, hence, enables finding collisions. Below we formally define the iterated backdoored function and in the next section we describe the attacks in more detail.

Let $p\colon \{0,1\}^b \to \{0,1\}^b$ be a one-way permutation where $b := r + c$. Let BDHGen create a backdoored one-way function $\tilde{p}\colon \{0,1\}^b \to \{0,1\}^b$ and its backdoor $\mathrm{bk} \in \{0,1\}^r$ as described in Figure 4.5. In more detail, we denote a c-bit long prefix of bk as $\mathrm{bk}_\beta := \mathrm{bk}_{[0,c-1]}$. The backdoor bk is defined in terms of the conditions $p(\mathrm{bk}\|a_\alpha) = \alpha$ and $p(a_\beta\|\mathrm{bk}_\beta) = \beta$ to hold for arbitrary but fixed constants $a_\alpha \in \{0,1\}^c, a_\beta \in \{0,1\}^r$ and $\alpha, \beta \in \{0,1\}^b$. We remark that a_α, a_β, and bk can be chosen by a malicious designer of $\tilde{p}$ and α and β are computed as mentioned. Note that an admissible input to $\tilde{p}$ is of the following form $s = s_{[0,b-1]} = s_{[0,r-1]}\|s_{[r,b-1]}$.

function :

$\tilde{p}\colon \{0,1\}^b \to \{0,1\}^b$

$$\tilde{p}(s) := \begin{cases} p(s)_{[0,r-1]}\|s_{[0,c-1]}, & \text{if } p(s_{[0,r-1]}\|a_\alpha) = \alpha \\ s_{[0,r-1]}\|p(s)_{[r,b-1]}, & \text{if } p(a_\beta\|s_{[r,b-1]}) = \beta \\ p(s), & \text{otherwise} \end{cases}$$

backdoor key :

any $\mathrm{bk} \in \{0,1\}^r$ with
$\mathrm{bk}_\beta := \mathrm{bk}_{[0,c-1]}$
such that $p(\mathrm{bk}\|a_\alpha) = \alpha$
and $p(a_\beta\|\mathrm{bk}_\beta) = \beta$

Figure 4.5: Backdoored function $\tilde{p}$ build from any permutation $p\colon \{0,1\}^b \to \{0,1\}^b$ with a corresponding backdoor key bk.

Let us take a closer look at the definition of $\tilde{p}$ in Figure 4.5. If the first r bits of the input $s_{[0,r-1]}$ correspond to bk (or another partial preimage of α under $p(\cdot\|a_\alpha)$), then $\tilde{p}$ sets the first r bits of the output to correspond to the correct execution of p while the remaining c bits are set to correspond to the sub-backdoor bk_β. This step can be used when attacking the sponge construction to signal that the absorbing phase is about to be finalized and hence the adversary has one iteration left to program the output. In the second step, the first r bits of the input correspond to the sabotaged digest the adversary wishes the function to output, while the lower c bits correspond to the sub-backdoor from the previous iteration (or another partial preimage of β under $p(a_\beta\|\cdot)$), then $\tilde{p}$ outputs the first r bits from the input and the remaining c bits are set to the correct execution of p. Knowing a backdoor key, these two steps allow for programing the output, when the message blocks (and in particular the last two) are under adversarial control, as in many security games. For all other inputs that do not trigger a backdoor, the third case of $\tilde{p}$ applies, where p is simply evaluated on the provided input. Note that the backdoor itself belongs to the set of admissible inputs of the underlying one-way permutation, and without its knowledge it is hard to find collisions for $\tilde{p}$, since the underlying p is a permutation.

Given the malicious construction $\tilde{p}$, a backdoored sponge-based hash function $\mathsf{H}^{\tilde{p}}_{\text{sponge}}$ can be built.

Attack Description

Next we describe how an adversary equipped with the backdoor key is able to find collisions, preimages, and second preimages in the backdoored sponge-based hash function family $\mathsf{H}^{\tilde{p}}_{\text{sponge}}$. To some degree, the attacks are similar to those described in Section 4.2.2 for MD-based hash functions but the details are specific to the sponge construction.

Finding collisions. The adversary aims to output two messages $x, x' \in \{0,1\}^*$ that map to the same hash value. For all equal-length prefixes $v, v' \in (\{0,1\}^r)^{n-3}$ the colliding messages are:

$$x := v \| \tilde{\text{bk}} \| \tilde{y} \quad \text{and} \quad x' := v' \| \tilde{\text{bk}}' \| \tilde{y}'.$$

Note that the adversary cannot input the real backdoor key directly. The two blocks $\tilde{\text{bk}}$ and $\tilde{y}$ (analogously also $\tilde{\text{bk}}'$ and $\tilde{y}'$) fulfill the following conditions (for fixed but arbitrary $y \in \{0,1\}^r$), where $\tilde{s}_{n-2}$ and $\tilde{s}_{n-1}$ are the intermediate states for computing the digest of x (resp. x') as defined in Figure 4.4:

$$\tilde{s}_{n-2[0,r-1]} \oplus \tilde{\text{bk}} = \text{bk} \quad \text{and} \quad \tilde{s}_{n-1[0,r-1]} \oplus \tilde{y} = y. \tag{4.1}$$

Recall that collisions can be found by triggering the backdoor key in the inputs. However, the sponge construction always uses as input to the permutation the first r bits of the state xored with the current message block. If the message block simply equals a backdoor key, it is unlikely that after xoring with the state the input to the function is still a valid backdoor key. But the adversary can choose a message block of a form such that after the xor operation is performed the first r bits of the state correspond to the backdoor key and then after being triggered enables the adversary to efficiently find collisions.

Observe that when the above equations are satisfied, the first case of $\tilde{p}$ is triggered, which writes the sub-backdoor into the lower state for the next call. Then, the sub-backdoor causes the second case of $\tilde{p}$ to be triggered. This ensures that the first r bits of the state which correspond to the digest y are simply passed through as the final output in the squeezing phase. The same applies to the second message x' and hence we have a collision on y.

Finding preimages. Given a digest y such that $|y| = r$, the adversary can use its backdoor by following the above strategy to encode a message of the form $x = v \| \tilde{\text{bk}} \| \tilde{y}$ for some arbitrary prefix

$v \in (\{0,1\}^r)^{n-3}$. If this message satisfies both equations from (4.1) as specified above for a given digest then the backdoor was successfully triggered and x is a valid preimage.

In case the digest is smaller than r bits, say $|y| = \ell \leq r$, the attacker can use the same message, however, the second equation needs to satisfy a slightly different relation. Here, we require first to pad the digest to the appropriate length of r bits by simply appending a bit string to complete the block. Hence, we require that the relation

$$\tilde{s}_{n-1[0,r]} \oplus \tilde{y} = y \| 1 \| 0^{r-\ell-1}$$

needs to be satisfied. The final output will then be the padded digest of length r. However, since we require the digest to be of length ℓ we just strip the padding off the digest by truncating it to the appropriate size, i.e., $y_{[0,\ell-1]}$ and, therefore, the adversary has successfully found a valid preimage x.

Finding second-preimages. The goal of finding a second preimage is similar to finding a preimage and follows the same strategy that have been put forward in Section 4.2.2 and simply apply the attack to $\mathsf{H}^{\tilde{p}}_{\text{sponge}}$.

Long digests. The attacks on the presented backdoored sponge-based hash function are somewhat restricted. They only deal with the case where digests are at most of the same size as a single message block, i.e., the digest is at most r bits long, which corresponds to squeezing the sponge only once. Still many applications (e.g. computing a MAC tag [BDPVA09]) are vulnerable to these attacks since they only require short digests. As soon as the produced digest length $|y| > r$, the sponge applies $\tilde{p}$ again to its state to obtain the next r bits corresponding to the digest block y_1. Since $\tilde{p}$ behaves like p on an overwhelming number of inputs, it is harder to find preimages or collisions for long digests. However, this does not help to immunize the construction against backdoors, since even though the second digest block (and the following ones) are harder to program, the security of the hash function is still very much compromised.

4.4 Practical Implications

The backdoored hash function designs discussed in this chapter use an *if-then-else* construct. Such constructs, or derivatives thereof, are often implicit in the design of rounds inside compression functions for nonlinearity reasons. For instance, SHA-1 and SHA-2 both use the function $\mathrm{Ch}(x,y,z) = (x \wedge y) \oplus (\neg x \wedge z)$ on 32-bit words in the round functions, implementing a bit-wise "if x_i then y_i else z_i" simultaneously over words. In SHA-3 the χ operation $\chi(a,b,c) = c \oplus (\neg a \wedge b)$ on 64-bit words can be viewed to implement "if a_i then c_i else $b_i \oplus c_i$" for each bit in the words.

We stress, however, that we are not claiming that SHA-1, SHA-2, or SHA-3 actually have built-in backdoors. In particular, embedding such type of backdoors would introduce additional complications since one has less control over the inputs when the operations Ch and χ are applied in the iterations of the round functions underlying the compression functions. Our constructions serve merely to raise awareness by demonstrating that embedding very powerful backdoors in hash functions is possible in principle, does not imply public-key primitives, and that only mild operations are necessary to exploit the backdoor.

If a practical hash function is vulnerable to such backdoors, security of countless practical schemes that rely on a secure hash function may be compromised. For instance, hash-then-sign signature schemes are forgeable by an adversary that can find a second preimage for a digest of a message that it has obtained a signature for. Such an adversary can potentially forge public-key certificates. Furthermore, the so-called *nothing-up-my-sleeve numbers* (NUMS) are supposed to be hard-to-invert hash digests, which are widely used in practical cryptographic designs. Constants in cryptographic algorithms are often hashed in order to destroy any potential structure that might give some advantage to the authority that has chosen those constants. However, if the NUMS are generated using a backdoored hash function, an adversary can use the backdoor to find a preimage and, therefore, manipulate the constants.

On a final note, we again remark that every collision, preimage, or second preimage found using the backdoor key for the constructions discussed in this chapter, encodes the backdoor key in the message. Therefore, using the backdoor may put the adversary and the backdoor key in risk of being exposed. An interesting open question here is to investigate whether constructions of backdoored compression functions exist that do not expose their backdoor key in adversarial inputs and do not rely on indistinguishability obfuscation to hide a secret key in the compression function and use it to internally decrypt malicious triggers.

Chapter 5

Defeating Backdoors in Pseudorandom Functions and Key Derivation Functions

In the next three chapters, we develop different strategies to defeat backdoors in hash functions. The attacks discussed in Chapter 4 give evidence of the difficulty of achieving this goal for hash functions in the standard model and without secret keys. In this chapter we show positive results for hash functions that use a secret key. More precisely, we show how to build a secure backdoor-resilient pseudorandom function and a key derivation function from any backdoor-free weak pseudorandom function. The central contribution of this chapter lies in the justification of our reliance on a backdoor-free weak pseudorandom function as the underlying primitive. We provide this by proving that any backdoored weak pseudorandom function must necessarily rely on expensive public-key primitives. Therefore, it is safe to assume that a symmetric-key weak pseudorandom function is backdoor-free.

My Scientific Contribution in this Chapter

The results in this chapter are published in [FJM18a], which is a joint work with Marc Fischlin and Christian Janson. The observation that weak pseudorandomness of secret-keyed functions that are not based on public-key primitives is preserved under backdooring attempts was made by me. Marc and I then jointly developed the proof in Section 5.2. Christian, Marc, and I had many discussions around potential immunization strategies, especially for HMAC, and were all involved in applying the final strategies to HMAC and HKDF, which were suggested by Marc. Christian and I mainly focused on immunizing HMAC using the randomized cascade idea in Section 5.3, while Marc focused on the immunization of HKDF in Section 5.4. The proof of salted NMAC being a secure KDF is included in the full version [FJM18b]. The high-level immunization of TLS against potentially backdoored hash functions, in Section 5.5, was conducted by Marc.

5.1 Introduction

Withstanding backdoor attacks can be very challenging. As we demonstrated in Chapter 4, fast backdoored hash functions, such that the knowledge of a backdoor key widely compromises security in crucial aspects, can indeed exist. Our input-triggered constructions of backdoored hash functions do not necessarily constitute the only possible type of backdoored hash functions. Nonetheless, some evidence that this specific type of backdoors can be immunized was recently presented by Russel et al. [RTYZ18] in a model called *subverted random-oracle* model. They show namely that if a hash function is subverted on a negligible fraction of its outputs, with the rest being perfectly random (i.e., a random oracle), one can build, using an improved salting due to Coretti et al. [CDGS18], a hash function that is indifferentiable from a random oracle. Overall, the quest for defeating backdoors in standard-model hash functions without severely restricting the considered backdooring strategy is still ongoing.

Here, we search for cryptographic properties of hash functions that can be used towards a successful immunization against backdoors. Fortunately, we are able to identify a promising candidate property of secret-keyed compression functions which cannot be weakened by a backdoor. This property is *weak pseudorandomness*, intuitively saying that a compression function's outputs on uniformly random inputs look random. Contrary to the (strong) PRF game, in the weak PRF game the adversary is not allowed to query its oracle on arbitrary inputs. Instead, it obtains some random inputs and the corresponding outputs from an oracle which is either a random function from the considered family or a uniformly random function from the set of all functions (for predetermined domain and co-domain sets).

The reason for our optimism is that we can show that distinguishing outputs (on random inputs) from random *with a backdoor* implies public-key encryption. In other words, placing a backdoor in a hash function design, in a way that weak pseudorandomness is violated, implicitly needs to embed a tedious public-key scheme and makes the design look suspicious. Of course there are instances, where at the cost of additional computational overhead, one prefers to use a provably secure (by means of security reductions to an assumed hard problem) hash function which implies public-key primitives. However, this is not true for the overwhelming majority of constructions and applications of today's hash functions. Overall, unless there is surprising progress in the efficiency of public-key schemes, fast compression functions will not be built from public-key tools and, hence, will remain weakly pseudorandom, even with knowledge of the backdoor. This result resembles an idea of Pietrzak and Sjödin [PS08] for building key agreement from secret-coin (i.e., the randomness used to sample the inputs is not revealed) weak pseudorandom functions, as well as backdoored pseudorandom generators implying public-key encryption, shown by Dodis et al. [DGG^{+}15].

Using the assumption of weak pseudorandomness we are able to provide an immunization strategy for HMAC based on the *randomized cascade* construction introduced by Maurer and Tessaro [MT08]. On a high level, this construction makes use of a prefix-free encoding to map blocks of the input message to (honestly chosen) random strings and can be used to build a full-fledged PRF from a weak PRF. We argue that since the randomized cascade construction yields a PRF, it can be used in the inner HMAC chain showing that such a modified HMAC is indeed a backdoor-resilient PRF. However, there is a small caveat in terms of efficiency since the underlying transformation in the randomized cascade construction from a weak PRF to a PRF can be expensive in terms of the number of compression function evaluations.

We further investigate whether there exist simpler immunization solutions for key derivation functions (KDFs) based on hash functions, especially HKDF based on HMAC. Fortunately, we answer this in the affirmative and show that an idea by Halevi and Krawczyk [HK06] for strengthening hash-and-sign schemes via input randomization can be used to immunize HMAC when used as a KDF. This result again relies on the weak pseudorandomness of the underlying compression function. We also briefly discuss how such immunized primitives can be used in the TLS 1.3 protocol for pre-shared keys, possibly enabling backdoor-resilient key exchange protocols.

5.2 On the Implausibility of Backdoored Weak Pseudorandom Functions

In this section we argue that it is reasonable to assume that a backdoored PRF, which is secure in the standard sense against distinguishers who do not know the backdoor key, remains a weak PRF even against distinguishers who do know the backdoor key. We prove that if a backdoor allows for distinguishing outputs of a weak PRF on random inputs from random bit strings, then that weak PRF family implies public-key encryption. Put differently, any such backdoored function would need to already contain some form of public-key encryption. Such constructions are, however, significantly slower than symmetric-key based pseudorandom functions and hence, in most practical applications using them would raise suspicion.

Koblitz and Menezes point out that the availability of some auxiliary information (like a backdoor key) may reduce the security of a pseudorandom function like HMAC noticeably [KM13]. The attack, however, relies on the possibility to query the pseudorandom function on chosen inputs. Gazi et al. [GPR14, Footnote 2] remarked that the attack is not known to be applicable to weak PRFs. Our results, when viewing the backdoor as some kind of auxiliary information, gives some argument why this is the case: it would mean that the weak PRF has already embedded a public-key encryption scheme where the auxiliary information acts as a (non-efficiently computable) secret key.

5.2.1 Weak Pseudorandom Functions

A family of functions is weakly pseudorandom if no efficient adversary can decide which of the two following behaviors is implemented by an oracle RoR (i.e., real-or-random) that it is given access to: choosing random elements in the domain and returning them together with their image either (a) under a function chosen uniformly at random from the function family or (b) under a uniformly random function with the same domain and co-domain. This game is formalized in Figure 5.1. Compared to the (strong) PRF game of Figure 3.1, the adversary in the wPRF game does not get to choose the inputs to its oracle. The advantage of an adversary $\mathcal{A}$ in the wPRF game, optionally using the backdoor bk (based on a bit *bd*), is defined as

$$\mathsf{Adv}^{\mathrm{wprf},bd}_{\mathsf{BDHGen}}(\mathcal{A},k,n,m) := 2 \cdot \Pr[\mathrm{wPRF}^{\mathcal{A},bd}_{\mathsf{BDHGen}}(1^k,1^n,1^m)] - 1 \ ,$$

where the probability is taken over the internal coin tosses in the game and by the adversary $\mathcal{A}$.

Note that if the backdoor key is not given to the adversary, we obtain the standard security notion for weak PRFs. We say that F is a *backdoored weak PRF* family if F is a weak PRF against adversaries without the backdoor, while with the backdoor there exists a PPT distinguisher $\mathcal{A}$ against its weak pseudorandomness that has a significant advantage.

When saying that a compression function $\mathsf{h} : \{0,1\}^\ell \times \{0,1\}^b \to \{0,1\}^\ell$ is weakly pseudorandom we implicitly consider a generator returning the function family $\{\mathsf{h}_\mathrm{k} : \{0,1\}^b \to \{0,1\}^\ell \mid \mathrm{k} \in \{0,1\}^\ell\}$, i.e., the key is chosen from the set $\{0,1\}^\ell$.

```
Game $\mathrm{wPRF}^{\mathcal{A},bd}_{\mathsf{BDHGen}}(1^k, 1^n, 1^m)$
$(\mathsf{F}, K, \mathrm{bk}) \twoheadleftarrow \mathsf{BDHGen}(1^k, 1^n, 1^m)$
if $bd = 0$ then $\mathrm{bk} \leftarrow \perp$
$\mathrm{k} \twoheadleftarrow \{0,1\}^k$
$\mathsf{F}_0 \leftarrow \mathsf{F}_{\mathrm{k}}$
$\mathsf{F}_1 \twoheadleftarrow \mathrm{Fun}[\mathsf{dom}(\mathsf{F}_{\mathrm{k}}), m]$
$b \twoheadleftarrow \{0,1\}$
$b' \twoheadleftarrow \mathcal{A}^{\mathrm{RoR}(b,\mathsf{F}_0,\mathsf{F}_1)}(\mathsf{F}, \mathrm{bk})$
return $(b = b')$
```

```
Oracle $\mathrm{RoR}(b, \mathsf{F}_0, \mathsf{F}_1)$
$x \twoheadleftarrow \mathsf{dom}(\mathsf{F}_b)$
$y \leftarrow \mathsf{F}_b(x)$
return $(x, y)$
```

Figure 5.1: weak-PRF security game for a backdoored hash generator BDHGen.

5.2.2 Backdoored Weak PRFs Imply Public-Key Encryption

In the following we construct a public-key encryption scheme from a backdoored weak PRF. More precisely, we show that a backdoor key which can be used to distinguish truly random from pseudorandom images (of random inputs), can be used to recover an encrypted bit with some probability bounded away from $\frac{1}{2}$ (and we can amplify the success probability via repetitions). Since one cannot distinguish random outputs of a weak PRF from uniform outputs without the backdoor key, we obtain a secure public-key bit-encryption scheme.

We give the construction and proof in terms of concrete security but occasionally refer to the common asymptotic setting. As for asymptotic behavior, we note that we get an infinitely-often public-key encryption scheme, where infinitely often means that the decryption algorithm works for infinitely many security parameters.

Theorem 5.1. *Let* BDHGen *be a generator for a backdoored weak pseudorandom function family. Then we can build an IND-CPA-secure public-key bit-encryption scheme from* BDHGen.

Proof. Let $\mathcal{A}$ be a PPT adversary against wPRF-security of BDHGen and suppose without loss of generality that $\mathcal{A}$ makes exactly $q \in \mathbb{N}$ queries to its RoR-oracle. Suppose that the advantage of $\mathcal{A}$ with the backdoor key is non-negligible, i.e., $\mathsf{Adv}^{\mathrm{wprf},1}_{\mathsf{BDHGen}}(\mathcal{A}, k, n, m) =: \varepsilon \not\approx 0$. In particular, let $\varepsilon \geq \frac{1}{\mathsf{poly}(k,n,m)}$ infinitely often. Furthermore, without loss of generality, let $\mathsf{BDHGen}(1^k, 1^n, 1^m)$ generate function families on the domain $\{0,1\}^n$, instead of the more general case of $\{0,1\}^{\mathsf{poly}(n)}$.

Then we can construct a public-key bit-encryption scheme with overwhelming correctness as follows.[1] For sake of simplicity we first construct an encryption scheme $\mathcal{E} := (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ with correctness $\frac{1}{2} + \frac{\varepsilon}{2}$ and explain afterwards how to boost the correctness bound. Below oracle O is a sub-algorithm used by the decryption algorithm to simulate the RoR-oracle from the wPRF game for the adversary. This oracle simply returns one point (i.e., an input-output pair) from a set C (which is a part of the ciphertext) after the other. Depending on the encrypted bit, the images from the points in C are either pseudorandom or truly random.

[1] Joseph Jaeger pointed out to us that the correctness of a previous version of our public-key bit encryption scheme could not be amplified by a simple majority decision and suggested masking the plaintext bit. To see the issue with directly encrypting the plaintext without masking it, consider an adversary $\mathcal{A}$ that given a pseudorandom function outputs 1 with probability 1, and given a truly random function outputs 1 with probability $1 - \varepsilon$, for a noticeable ε, and note that a majority decision would then lead to incorrect decryption of ciphertexts of 0.

```
$\mathsf{KGen}(1^k, 1^n, 1^m)$
$(\mathsf{F}, K, \mathrm{bk}) \twoheadleftarrow \mathsf{BDHGen}(1^k, 1^n, 1^m)$
$\mathrm{pk} \leftarrow \mathsf{F}$
$\mathrm{sk} \leftarrow \mathrm{bk}$
return $(\mathrm{pk}, \mathrm{sk})$
```

```
$\mathsf{Enc}(\mathrm{pk}, b)$
$d \twoheadleftarrow \{0,1\};\ b' \leftarrow d \oplus b$
$\mathrm{k} \twoheadleftarrow \{0,1\}^k$
$C \leftarrow \emptyset$
for $i = 1 \ldots q$ do
  $x_i \twoheadleftarrow \{0,1\}^n$
  if $b' = 0$ then
    $y_i \twoheadleftarrow \{0,1\}^m$
  else
    $y_i \leftarrow \mathsf{F}_{\mathrm{k}}(x_i)$
  $C \leftarrow C \cup \{(x_i, y_i)\}$
$c \leftarrow (d, C)$
return $c$
```

```
$\mathsf{Dec}(\mathrm{sk}, \mathrm{pk}, c)$
$(d, C) \leftarrow c$
$b' \twoheadleftarrow \mathcal{A}^{\mathrm{O}(C)}(\mathrm{pk}, \mathrm{sk})$
return $b' \oplus d$
```

```
Oracle $\mathrm{O}(C)$
$(x, y) \twoheadleftarrow C$
$C \leftarrow C \setminus \{(x, y)\}$
return $(x, y)$
```

Observe that a ciphertext can be correctly decrypted if $\mathcal{A}$ successfully distinguishes random outputs of F_{k} from uniformly random bit strings. Hence, we obtain the following correctness guarantee:

$$\begin{aligned}\Pr[\mathsf{Dec}(\mathrm{sk}, \mathrm{pk}, \mathsf{Enc}(\mathrm{pk}, b)) = b] &= \Pr[\mathcal{A}^{\mathrm{O}(\mathsf{Enc}(\mathrm{pk},b))}(\mathrm{pk}, \mathrm{sk}) = b] \\ &= \frac{\mathsf{Adv}^{\mathrm{wprf},1}_{\mathsf{BDHGen}}(\mathcal{A}, k, n, m) + 1}{2} \\ &= \frac{1}{2} + \frac{\varepsilon}{2}\ .\end{aligned}$$

In other words, the decryption error is noticeably smaller than $\frac{1}{2}$ for infinitely many security parameters.

It remains to show that $\mathcal{E}$ is indistinguishable under chosen-plaintext attacks. Suppose to the contrary that there exists a PPT adversary $\mathcal{B}$ against the security of $\mathcal{E}$. Since we are concerned with security of bit encryption, this means that $\mathcal{B}$ can decrypt a ciphertext with a non-negligible advantage ε'. We can build from $\mathcal{B}$ a PPT adversary $\mathcal{C}$ against the weak-PRF security of BDHGen, i.e., the generated family F (when not holding a backdoor key). The adversary $\mathcal{C}$ queries its oracle q times to obtain a set C of input-output pairs. It then runs $\mathcal{B}$ and outputs the returned guess b as its own. We obtain

$$\mathsf{Adv}^{\mathrm{wprf},0}_{\mathsf{BDHGen}}(\mathcal{C}, k, n, m) = \mathsf{Adv}^{\text{IND-CPA}}_{\mathcal{E}}(\mathcal{B}, k, n, m) = 2 \cdot \Pr[\mathcal{B}(\mathrm{pk}, \mathsf{Enc}(\mathrm{pk}, b)) = b] - 1 = \varepsilon'.$$

Thus our adversary $\mathcal{C}$, who does not know a backdoor key, has a non-negligible advantage against the weak pseudorandomness of F. This contradicts our assumption of F being weakly pseudorandom (without the backdoor key).

As a final step we note that it is possible to reduce the decryption error by standard techniques. For this, we repeat the above basic encryption step a polynomial number of times, letting the sender always generate pseudorandom or truly random strings in each of the repetitions. The decrypter outputs a majority decision for all the decrypted bits. If we use $(k + n + m) \cdot \varepsilon^2 \le (k + n + m) \cdot \mathsf{poly}(k, n, m)^2$ repetitions, where $\varepsilon \ge \frac{1}{\mathsf{poly}(k,n,m)}$ is the lower bound for the distinguisher's advantage, then the

Hoeffding-Chernoff bound implies that the decryption error is upper-bounded by $e^{-(k+n+m)}$. At the same time, the security of the public-key encryption scheme remains intact for a polynomial number of repetitions. □

5.3 Backdoor-Resilient HMAC

According to the result presented in the previous section, we may assume that the compression function used in an HMAC construction preserves weak pseudorandomness in the presence of backdoors. In the following we use the *randomized cascade* (RC) construction introduced by Maurer and Tessaro [MT08] in order to immunize HMAC under this weak pseudorandomness assumption. In basic terms, the RC construction is an iterated construction of a PRF from a (constant-query) weak PRF. The first construction of a PRF from a weak PRF is due to Naor and Reingold [NR99], and a further construction was later proposed by Maurer and Sjödin [MS07]. In our case, we are interested in an iterated construction of a PRF, where the candidate for a weak PRF is a compression function. Maurer and Tessaro note that both constructions [MS07, NR99] may be turned easily into iterative versions with the drawback that the number of calls to the underlying function would increase significantly. In contrast, the randomized cascade construction is more efficient and requires for input length b approximately $b/\log s$ (for some $s \geq 2$) many calls to the underlying function and also only requires the weaker underlying assumption of an *s-query weak PRF*[2] instead of a weak PRF.

Let us now review the idea of the RC construction which is based on the cascade construction for hash functions introduced by Bellare, Canetti and Krawczyk [BCK96b]. The RC construction requires a prefix-free encoding of the input message. Let $X \subseteq \{0,1\}^{\mathsf{poly}(b)}$ denote the message space. We say an efficiently computable encoding $\mathsf{encode}\colon X \to \{0,\dots,s-1\}^+$ is prefix-free if for all distinct inputs $x, x' \in X$ and $\mathsf{encode}(x)$ is not a prefix of $\mathsf{encode}(x')$. On a high level, the RC construction resembles the Merkle–Damgård construction (cf. Figure 4.1) with some additional randomness whereby the underlying building block is a s-weak PRF.

The RC construction with parameter s and message space X for the compression function $\mathsf{h}\colon \{0,1\}^\ell \times \{0,1\}^b \to \{0,1\}^\ell$ and a prefix-free encoding encode as described above is a mapping $\mathsf{RC}^{\mathsf{h}}\colon \{0,1\}^\ell \times \{0,1\}^{s\cdot b} \times X \to \{0,1\}^\ell$. The mapping uses as input a secret key k of length ℓ and a $(s \cdot b)$-bit long *public* string r which can be interpreted as the concatenation of s many b-bit strings $r_0, \dots, r_{s-1}$ and a message $x \in X$. The message x is first padded (following Section 4.2.1) such that the length becomes a multiple of b and is then further processed with the above prefix-free encoding which outputs a sequence $(m_0, \dots, m_{n-1}) \in \{0, \dots, s-1\}^+$. Then for $i = 0, \dots, n-1$ the cascade is computed as $y_{i+1} \leftarrow \mathsf{h}(y_i, r_{m_i})$ with $y_0 \leftarrow \mathrm{k}$. First let us remark that in each iteration the r_{m_i}'s are chosen according to the outputted sequence from the encoding. Maurer and Tessaro formally prove that if h is a s-weak PRF, then the resulting RC construction is a PRF. The proof relies on the encoding being done via a tree structure, where it is argued that by the definition of s-weak

[2] We say that a function $f\colon \{0,1\}^k \times \{0,1\}^i \to \{0,1\}^o$ for some constant s with $k < s \cdot o$ is an *s-query weak PRF* if $f(\mathrm{k}, \cdot)$ (under a secret key k) is indistinguishable from a random function when evaluated at s independent known random inputs. This notion is weaker than a (regular) weak PRF where we require indistinguishability for polynomially many random inputs.

PRF whenever we evaluate the function under some secret key at s independent random inputs, it produces s pseudorandom outputs and in particular sets all vertices in the tree to be pseudorandom.

Now let $\mathsf{HMAC}^{\mathsf{h}}$ be a backdoored HMAC construction. Our goal is to replace the Merkle–Damgård construction with the randomized cascade construction from above and argue that the resulting variant of HMAC resists backdoors. The idea is that we first pad the input message x with the usual length padding function and then use a prefix-free encoding obtaining a sequence m. According to this sequence, the correct random string will be chosen and only then used as the input to the compression function. More formally it follows that

$$\mathsf{HMAC}^{\mathsf{h}}_{\mathsf{rc},r,\mathrm{iv}}(\mathrm{k}, x) = \mathsf{H}^{\mathsf{h}}_{\mathrm{iv}}\left((\mathrm{k} \oplus \mathsf{opad}) \| \mathsf{H}^{\mathsf{h}}_{\mathrm{iv}}((\mathrm{k} \oplus \mathsf{ipad}) \| r_{m_0} \| \ldots \| r_{m_{n-1}})\right).$$

The above HMAC construction is a secure PRF, since its inner hash chain is a PRF even against backdooring adversaries. In particular, the first iteration in the inner chain (i.e. $\mathsf{h}(\mathrm{iv}, \mathrm{k} \oplus \mathsf{ipad})$) is computationally indistinguishable from a uniformly-distributed random string, assuming as a weak dual PRF [Bel06]. This guarantees that the first chaining value is pseudorandom and, hence, can be used as a “good” key in the RC construction. The same argument applies for the first chaining value in the outer chain. The last iteration in HMAC receives as input from both chains a pseudorandom input and, hence, the output is still pseudorandom and thus the above construction of HMAC is secure.

On using randomized cascade for immunizing hash functions without secret keys. It may seem that the idea of encoding a message by a sequence of random (but public and chosen after the design) strings $r = (r_0, \ldots, r_{s-1})$ before processing it by a compression function can also immunize MD-based hash functions, since in the backdoored compression functions that we described in Section 4.2.1 the backdoor key must be input directly to trigger the malicious behavior and it is unlikely that such a backdoor key is included in r. However, we remark that although RC may be useful for immunization of this particular type of backdoors, their usefulness as a general immunization mechanism for hash functions (without secret keys) does not seem to have a solid foundation.

5.4 Backdoor-Resilient HKDF

The above transformation from a weak PRF to a full-fledged PRF can be expensive in terms of the number of compression function evaluations. More precisely, this number depends on the parameter s, in a way that $\log s$ determines the length of the message blocks that need an individual encoding. This means that for a message x, RC requires $|x|/\log s$ many calls to the underlying compression function. For instance, in the extreme case of $s = 2$, the encoding function would encode each bit of x by either r_0 or r_1. Hence, for each bit of x, our backdoor-resilient HMAC makes one call to the compression function (plus three more calls for the initial inner and outer chain of HMAC). In this section we argue that for key derivation functions based on hash functions, in particular HKDF based on HMAC, there exists a simpler solution.

The HMAC-based key derivation function HKDF [Kra10, KE10] consists of two steps: an *extraction* step to smooth the entropy in some input key material like a Diffie–Hellman key, and an *expansion* step where sufficient key material is generated. The extraction step may use some (public) salt *extsalt* (if not present, it is set to 0) and produces a pseudorandom key PRK from the input key material IKM by calling HMAC. The expansion step takes the key PRK, some context information *info* like a transcript from a key exchange protocol, and the requested output length *len* (in octets). It iterates HMAC on PRK, the previous value, *info*, and a counter, encoded as an octet, to generate sufficient key material. The last key part in the output may be truncated to match the requested output length. Formally,

```
$\mathsf{HKDF\text{-}Extract}(extsalt, \mathrm{IKM})$
$\mathrm{PRK} \leftarrow \mathsf{HMAC}(extsalt, \mathrm{IKM})$
return PRK
```

```
$\mathsf{HKDF\text{-}Expand}(\mathrm{PRK}, info, len)$
$i \leftarrow 1$
$\mathrm{K}, \mathrm{K}_0 \leftarrow \epsilon_0$
while $|\mathrm{K}| < len$ do
    $\mathrm{K}_i \leftarrow \mathsf{HMAC}(\mathrm{PRK}, \mathrm{K}_{i-1} \| info \| i)$
    $\mathrm{K} \leftarrow \mathrm{K} \| \mathrm{K}_i$
    $i \leftarrow i + 1$
$\mathrm{K} \leftarrow \mathrm{K}_{[0, len-1]}$
return K
```

Immunizing HKDF boils down to hardening HMAC and, therefore, the compression function h. The security of HKDF relies on the pseudorandomness of h, which does not hold for backdoored functions according to the attack described in Section 4.2.4. As argued in Section 5.2, the assumption that h is still a *weak* PRF in the presence of a backdoor appears to be more reasonable. Hence our goal is to tweak HKDF to base its security on the weak-PRF security of its underlying compression function h.

We use the idea of Halevi and Krawczyk [HK06] to strengthen hash-and-sign schemes via input randomization. They propose to pick a fresh random string r with each signature generation and then compute the hash as $\mathsf{H}((x_0 \oplus r) \| \cdots \| (x_{n-1} \oplus r))$, i.e., xoring the random string to each message block. This alleviates the necessary assumption for the compression function h from collision resistance to some kind of second-preimage resistance. We stress that this strategy does not work to immunize hash functions against backdoors, as our attacks in Section 4.2.2 show that a backdoored compression function would even allow to break second-preimage resistance, the malicious behavior can be triggered in the input by the adversary. In fact, one can show that a backdooring adversary can still break the randomized hash-and-sign scheme.

The idea of Halevi and Krawczyk does apply, nonetheless, in the case of HMAC *when used as a key derivation function*, if we allow for a random value *salt* in the computation. Suppose that, when computing keying material, one is allowed to pick a random string *salt* of size b. Then, instead of using the compression function h in the HMAC computations, we use the function $\mathsf{h}_{salt}(x, y) = \mathsf{h}(x, y \oplus salt)$. Note that this means that we add *salt* to each input in each of the iteration steps. When outputting the key material IKM, one can also generate the salt. Note that the salt sometimes even needs to be published, for instance, in a key exchange protocol where the other party should be able to derive the same key.

```
Game $\mathrm{SKDF}^{\mathcal{A},bd}_{\mathsf{BDKDFGen}}(1^k, 1^b)$
$(\mathsf{KDF}, K_{\mathrm{kdf}}, \mathrm{bk}) \twoheadleftarrow \mathsf{BDKDFGen}(1^k, 1^b)$
if $bd = 0$ then $\mathrm{bk} \leftarrow \perp$
$\mathrm{k} \twoheadleftarrow K_{\mathrm{kdf}}$
$g \twoheadleftarrow \{0,1\}$
$g' \twoheadleftarrow \mathcal{A}^{\mathsf{KDF}_{\mathrm{k}}(\cdot,\cdot),\mathrm{RoR}(g,\mathsf{KDF}_{\mathrm{k}},\cdot,\cdot)}(\mathsf{KDF}, \mathrm{bk})$
return $(g = g')$
```

```
Oracle $\mathrm{RoR}(g, \mathsf{KDF}_{\mathrm{k}}, \mathit{info}, \mathit{len})$
if $g = 0$ then
    $(\mathit{salt}, \mathrm{K}) \twoheadleftarrow \mathsf{KDF}_{\mathrm{k}}(\mathit{info}, \mathit{len})$
else
    $\mathit{salt} \twoheadleftarrow \{0,1\}^b$
    $\mathrm{K} \twoheadleftarrow \{0,1\}^{len}$
return $(\mathit{salt}, \mathrm{K})$
```

Figure 5.2: salted-KDF-security for a generator $\mathsf{BDKDFGen}$.

The downside of our construction is that each HMAC call in the expansion requires a fresh salt. However, usually only a few iterations in the expansion step are required. For example, the cipher suite `AES_256_CBC_SHA256` in TLS 1.2 requires 128 key bytes such that four iterations, each with 256 bits output, suffice.

5.4.1 Security of Salted Key Derivation

To define a security notion for salted key derivation functions we adopt the approach of Krawczyk [Kra10], demanding that the KDF provides pseudorandom outputs even when the adversary can ask to see derived keys on different information *info*. Since we use a fresh salt for each KDF call, we can even allow the adversary to query the same information *info* multiple times and still demand indistinguishability from fresh random key material.

In the SKDF security game, described in Figure 5.2, we consider a PPT generator $\mathsf{BDKDFGen}$ which on input of parameters $(1^k, 1^b)$ outputs a family of key derivation functions $\mathsf{KDF} := \{\mathsf{KDF}_{\mathrm{k}} : (\{0,1\}^{\mathsf{poly}(b)})^2 \to \{0,1\}^b \times \{0,1\}^{\mathsf{poly}(b)} \mid \mathrm{k} \in \{0,1\}^k\}$ and possibly a backdoor key $\mathrm{bk} \in \{0,1\}^*$. We assume that the function $\mathsf{KDF}_{\mathrm{k}}$ takes two inputs, a context information *info* and a length *len*, and returns a random value *salt* (of size b) and keying material of size *len*. To claim indistinguishability from random we consider an oracle $\mathrm{RoR}(\cdot,\cdot)$ which on any input pair $(\mathit{info}, \mathit{len})$ returns a fresh random value *salt* and a random string K of size *len*. The advantage of an adversary $\mathcal{A}$ in the SKDF game, optionally given the backdoor bk, is defined by:

$$\mathsf{Adv}^{\mathrm{skdf},bd}_{\mathsf{BDKDFGen}}(\mathcal{A}, k, b) := 2 \cdot \Pr[\mathrm{SKDF}^{\mathcal{A},bd}_{\mathsf{BDKDFGen}}(1^k, 1^b)] - 1 \ ,$$

where the probability is over the random choices of the game, the generator $\mathsf{BDKDFGen}$ and $\mathcal{A}$.

5.4.2 HKDF Expansion based on NMAC

We first discuss the case of expansion being based on NMAC instead of HMAC and argue afterwards that the result can be lifted to HMAC and the extraction step, making some additional assumptions. Recall that there are two differences between NMAC and its “practical cousin” HMAC. First, NMAC takes two independent keys $\mathrm{k}_{\mathrm{in}}, \mathrm{k}_{\mathrm{out}} \in \{0,1\}^{\ell}$ instead of using correlated keys $\mathrm{k} \oplus \mathsf{ipad}, \mathrm{k} \oplus \mathsf{opad}$ as in HMAC. Second, the keys in NMAC are used directly as a substitute for the initialization vector iv, instead of making an extra iteration to first compute $\mathsf{h}(\mathrm{iv}, \mathrm{k} \oplus \mathsf{ipad})$ resp. $\mathsf{h}(\mathrm{iv}, \mathrm{k} \oplus \mathsf{opad})$ as done in HMAC.

Let $\mathsf{sNMAC}((\mathrm{k}_{\mathrm{in}}, \mathrm{k}_{\mathrm{out}}), \cdot)$ (for salted NMAC) be the probabilistic algorithm which, on being called, picks a fresh $\mathit{salt} \twoheadleftarrow \{0,1\}^b$ and then computes NMAC on keys $\mathrm{k}_{\mathrm{in}}, \mathrm{k}_{\mathrm{out}}$ for the salted compression function $\mathsf{h}_{\mathit{salt}}$. It outputs the result of the computation together with the salt. By the construction of $\mathsf{HKDF\text{-}Expand}$ we can assume that the adversary in the SKDF experiment only queries the key derivation function for length values $\mathit{len} = \ell$ equal to the output size of NMAC. This would already allow the adversary to assemble the full key material by sequentially making the corresponding queries.

Theorem 5.2. *The* sNMAC *construction is a secure salted KDF, i.e., for any adversary $\mathcal{A}$ against* sNMAC *we obtain an adversary $\mathcal{B}$ against the weak-PRF property for any generator* BDHGen *of the underlying compression function such that*

$$\mathsf{Adv}^{\mathrm{skdf},bd}_{\mathsf{sNMAC}}(\mathcal{A}, k, b, \ell) \leq 2nq \cdot \mathsf{Adv}^{\mathrm{wprf},bd}_{\mathsf{BDHGen}}(\mathcal{B}, k, b, \ell) \ ,$$

where B is the maximal number of message blocks and each of them has at most q key derivation queries, and $n := B + 2 \cdot \lceil \ell/b \rceil + 3$. Furthermore, the run times of $\mathcal{A}$ and $\mathcal{B}$ are essentially the same.

The proof idea is that each computation of sNMAC starts with an evaluation of the backdoored compression function for $x_1 = \mathsf{h}(\mathrm{k}_{\mathrm{in}}, y_0 \oplus \mathit{salt})$, where y_0 is the first input block according to the adversary's query and salt is a fresh random value, picked independently after y_0 has been determined. This means that the input pair is random, such that we can conclude by the weak pseudorandomness that the output value x_1 looks random, too. The argument then applies to the next iteration step as well, since the next input $(x_1, y_1 \oplus \mathit{salt})$ to the compression function is (indistinguishable from) random. The approach can be set forth to show that all final answers in the computations look random, where the formal way to show this is via a hybrid argument. Since we pick a fresh salt in each computation, the result also holds for multiple queries.

Proof. The proof strategy is to first show that in case $\mathcal{A}$ has access to two sNMAC oracles (i.e., can evaluate the KDF and the RoR oracle for $g = 0$), then we can replace both oracles by two (independent) oracles of the type RoR with $g = 0$. This will be indistinguishable by the wPRF property of the compression function. Then we can switch back the left oracle to sNMAC because the right random oracle is easy to simulate, concluding again that this is indistinguishable by the wPRF property. So let $\mathcal{A}$ be an adversary against two sNMAC oracles, making at most q queries to both oracles together, each input information of at most B blocks. This means that, together with the counter i, the ℓ-bit value k_{i-1} of the previous iteration, and the padding, we evaluate h at most $B + \lceil \ell/b \rceil + 2$ times in the inner NMAC computation. We make at most another $\lceil \ell/b \rceil + 1$ iterations for the outer computation.

For the proof it is instructive to write down all pairs $(x_i^j, y_i^j \oplus \mathit{salt}_i)$ inserted into the compression function h during the experiment, where i denotes the number of the query, salt_i is the i-th chosen salt value, and j the iteration within a full NMAC computation. We order these elements in a table, where we put the iteration of the computation in the rows. The columns then correspond to the iteration round, where different queries may have a different number of iterations. We put all the outer computations in the final columns. In particular, since this number only depends on the hash function parameters, the numbers of columns required for the outer computation are identical over all queries. If $\mathcal{A}$ makes at most q queries we thus evaluate the compression function h on a table of the following form:

	→ inner sNMAC computation →				outer sNMAC
query	column 0	column 1	column 2	...	column $n-1$
1	$(x_1^0, y_1^0 \oplus \mathit{salt}_1),$	$(x_1^1, y_1^1 \oplus \mathit{salt}_1),$	$(x_1^2, y_1^2 \oplus \mathit{salt}_1),$	...	$(x_1^{n_1-1}, y_1^{n_1-1} \oplus \mathit{salt}_1)$
2	$(x_2^0, y_2^0 \oplus \mathit{salt}_2),$	$(x_2^1, y_2^1 \oplus \mathit{salt}_2),$	$(x_2^2, y_2^2 \oplus \mathit{salt}_2),$	...	$(x_2^{n_2-1}, y_2^{n_2-1} \oplus \mathit{salt}_2)$
		$\vdots$			
q	$(x_q^0, y_q^0 \oplus \mathit{salt}_q),$	$(x_q^1, y_q^1 \oplus \mathit{salt}_q),$	$(x_q^2, y_q^2 \oplus \mathit{salt}_q),$	...	$(x_q^{n_q-1}, y_q^{n_q-1} \oplus \mathit{salt}_q)$

where $n_i \leq n$ is the number of evaluations in the i-th query. Note that $x_i^0 = \mathrm{k_{in}}$ is always the inner key and in the part referring to the inner computations of NMAC we always have $x_i^{j+1} = \mathsf{h}(x_i^j, y_i^j \oplus \mathit{salt}_i)$ for all i, j. When proceeding to the outer computation, one has $x_i^j = \mathrm{k_{out}}$ for the outer key, and y_i^j is determined by the final hash value $\mathsf{h}(x_i^{j-1}, y_i^{j-1} \oplus \mathit{salt}_i)$ of the inner computation. By construction, the column where we progress to the outer computation is identical for all queries.

Our claim is now that the final output values in the last n-th column all look random. This follows by a slightly involved but, nonetheless, standard hybrid argument over the columns of the table. Since the argument is fairly straightforward to formalize we only explain the main idea.

First note that in the first column of the table we apply h for the same key $\mathrm{k_{in}} = x_i^0$ on random inputs $y_i^0 \oplus \mathit{salt}_i$ for uniformly chosen salt_i. Intuitively, we can thus replace the output $x_i^1 = \mathsf{h}(x_i^0, y_i^0 \oplus \mathit{salt}_i)$ by a random value by the weak PRF property of the (backdoored) function h. The formal argument is via a black-box reduction, where an algorithm $\mathcal{B}$ against the weak PRF property simulates $\mathcal{A}$'s attack against sNMAC. Adversary $\mathcal{B}$ initially receives as input q pairs (a_i, b_i), where each a_i is random and each b_i is either $\mathsf{h}(\mathrm{k}, a_i)$ or random, too. It uses the b_i's as replacements for the values x_i^1, and sets $\mathit{salt}_i = a_i \oplus y_i^0$ for all i. Note that this makes salt_i a correctly distributed uniform value. Algorithm $\mathcal{B}$ performs the remaining computations of the table as before, using the now derived value salt_i in each row and picking the outer key $\mathrm{k_{out}}$ itself. If the b_i's are pseudorandom, then this corresponds exactly to the original computation, whereas for truly random b_i's we simulate the slightly changed game.

Given that the values x_i^1 are random now, we can set the argument forth, noting that we can pick the salt values salt_i afresh in the next hybrid step (because the values $x_i^0, y_i^0 \oplus \mathit{salt}_i$ have become irrelevant). The argument in this step is, nonetheless, slightly more involved since we may have different x_i^1's, but some of these values may coincide. This can be resolved by handing over multiple values $(a_{i,j}, b_{i,j})$ to $\mathcal{B}$ for $i, j = 1, 2, \ldots, q$, where $b_{i,j} = \mathsf{h}(\mathrm{k}_j, a_{i,j})$ for q independent keys k_j, or all $b_{i,j}$'s are random. By another hybrid argument it follows from the pseudorandomness of h that the two cases are indistinguishable. Algorithm $\mathcal{B}$ can then "consume" sufficiently many values for the simulation for identical x_i^1.

In the formal hybrid argument, $\mathcal{B}$ picks a column k among the n ones at random and injects values from the q^2 input pairs $(a_{i,j}, b_{i,j})$ as above. For this injection strategy it sets all values x_i^j for "earlier" columns $j < k$ to be independent random values, except for the x_i^j's corresponding to the inner and outer key (which are set to be an equal random value). This results in a security loss equal to the number n of columns, and the number q of input sequences $(a_{i,j}, b_{i,j})_{j=1,\ldots,q}$ for $\mathcal{B}$. This yields the claimed bound, taking into account that we derive the factor 2 by switching the left oracle back to sNMAC. □

5.4.3 Lifting the Result to HKDF

To extend the above argument to also cover the extraction step we need to assume, as in the original security proof of HMAC [Bel06], that the compression function h is a weak dual PRF. This means that $\mathsf{h}(\cdot, \mathrm{IKM})$ is weakly pseudorandom for the input keying material IKM (with sufficient entropy). This appears to be a widely accepted assumption, but in our case this should also hold for backdoored compression functions. With a similar argument as in the wPRF case we can argue that a weak dual PRF remains secure (for fixed extraction salt *extsalt*) even when having a backdoor, or else one can again construct a public-key encryption scheme. The argument is as before, putting *extsalt* as part of the public-key and using the backdoor to distinguish random values (for encryptions of a bit b with $d \oplus b = 0$) from $\mathsf{h}(\mathit{extsalt}, \mathrm{IKM})$ values (for encryptions with $d \oplus b = 1$, where the sender chooses IKM).

Similarly, we need to argue that using $\mathrm{k_{in}} = \mathsf{h}(\mathrm{iv}, \mathrm{k} \oplus \mathsf{ipad})$ and $\mathrm{k_{out}} = \mathsf{h}(\mathrm{iv}, \mathrm{k} \oplus \mathsf{opad})$ in the HMAC computation, instead of random values $\mathrm{k_{in}}, \mathrm{k_{out}}$ as in sNMAC, does not endanger the security, even for backdoored h. The argument that the backdoored case should not make a difference is as before: pseudorandomness of the (correlated values) $\mathsf{h}(\mathrm{iv}, \mathrm{k} \oplus \mathsf{ipad})$ and $\mathsf{h}(\mathrm{iv}, \mathrm{k} \oplus \mathsf{opad})$ should also hold in the backdoored case, unless the backdooring already embeds a public-key encryption scheme.

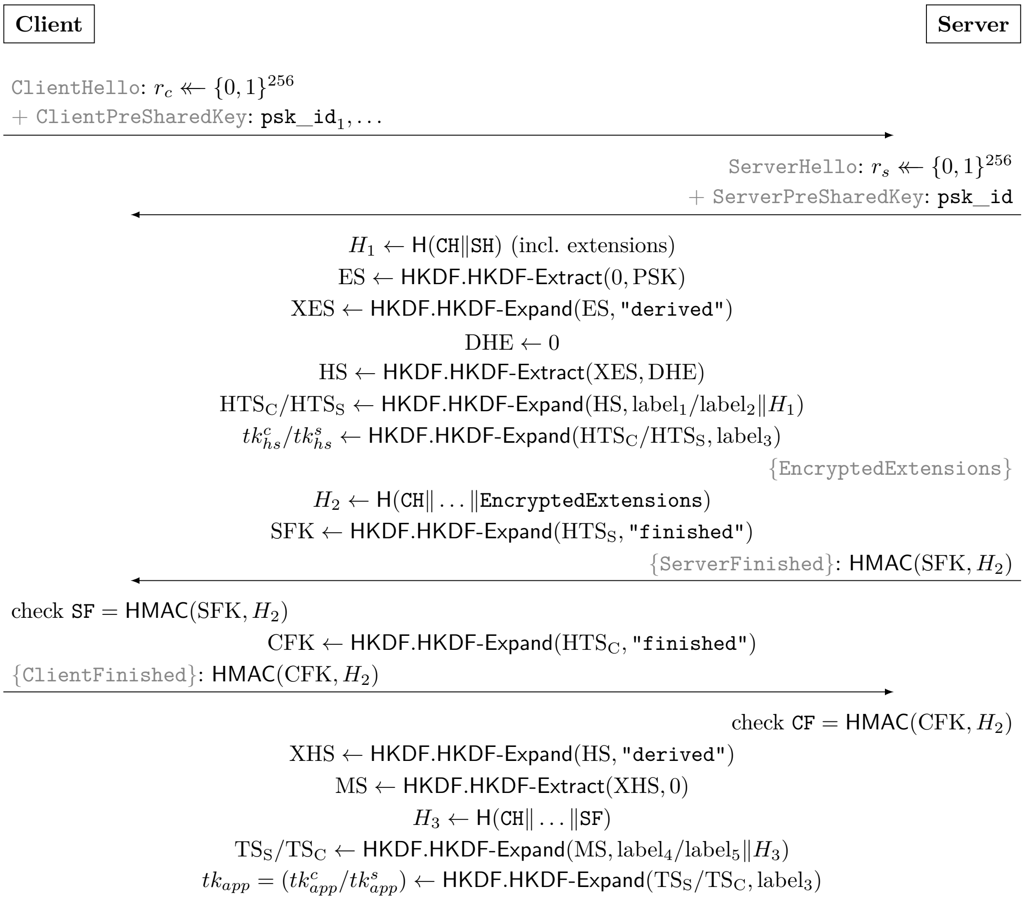
5.5 Backdoor-Resilient TLS-like Key Exchange

Once we have immunized HMAC and HKDF the next question is how we can use these building blocks in higher-level protocols to make them resilient against backdoors. We discuss this here briefly for the case of the TLS 1.3 protocol [Res18], and especially for the pre-shared key (PSK) mode. The PSK mode covers the case in which client and server already hold a shared key and do not need to run a Diffie–Hellman key exchange sub-protocol; immunizing the latter would be beyond our work's scope. The PSK protocol only relies on a cryptographic hash function but used in different contexts: as a collision-resistant hash function, as a MAC via HMAC, and as a key derivation function via HKDF.

The PSK mode is displayed in Figure 5.3. We follow the presentation in [DFGS15, DFGS16]. In the protocol the client starts with the `ClientHello` message, containing a nonce r_c, and specifies identifiers for shared keys via the `ClientPreSharedKey` message. The server replies with the `ServerHello` message, also containing a nonce r_s, and the choice of key identifier in `ServerPreSharedKey`. The server then starts deriving keys via HKDF on the pre-shared key PSK and the transcript hashes. It sends the encrypted extension information {`EncryptedExtensions`}. The server also computes a finished message `ServerFinished` which is an HMAC over the derived keys and the transcript hash. The client subsequently computes the keys, checks the HMAC, and sends its finished message `ClientFinished`. Both parties once more use HKDF and transcript hashes to derive the shared session key.

5.5.1 Towards a Backdoor-Resilient PSK Mode

Not surprisingly, we are not able to show that the PSK mode of TLS 1.3, as is, can be immunized against backdoors. There are both security-related as well as functional reasons. In terms of security,



Client **Server**

`ClientHello`: $r_c \twoheadleftarrow \{0,1\}^{256}$
$+$ `ClientPreSharedKey`: $\texttt{psk\_id}_1, \ldots$

`ServerHello`: $r_s \twoheadleftarrow \{0,1\}^{256}$
$+$ `ServerPreSharedKey`: `psk_id`

$H_1 \leftarrow \mathsf{H}(\texttt{CH}\|\texttt{SH})$ (incl. extensions)
$\mathrm{ES} \leftarrow \mathsf{HKDF.HKDF\text{-}Extract}(0, \mathrm{PSK})$
$\mathrm{XES} \leftarrow \mathsf{HKDF.HKDF\text{-}Expand}(\mathrm{ES}, \texttt{"derived"})$
$\mathrm{DHE} \leftarrow 0$
$\mathrm{HS} \leftarrow \mathsf{HKDF.HKDF\text{-}Extract}(\mathrm{XES}, \mathrm{DHE})$
$\mathrm{HTS_C}/\mathrm{HTS_S} \leftarrow \mathsf{HKDF.HKDF\text{-}Expand}(\mathrm{HS}, \mathrm{label}_1/\mathrm{label}_2\|H_1)$
$tk^c_{hs}/tk^s_{hs} \leftarrow \mathsf{HKDF.HKDF\text{-}Expand}(\mathrm{HTS_C}/\mathrm{HTS_S}, \mathrm{label}_3)$
{`EncryptedExtensions`}
$H_2 \leftarrow \mathsf{H}(\texttt{CH}\|\ldots\|\texttt{EncryptedExtensions})$
$\mathrm{SFK} \leftarrow \mathsf{HKDF.HKDF\text{-}Expand}(\mathrm{HTS_S}, \texttt{"finished"})$
{`ServerFinished`}: $\mathsf{HMAC}(\mathrm{SFK}, H_2)$

check $\texttt{SF} = \mathsf{HMAC}(\mathrm{SFK}, H_2)$
$\mathrm{CFK} \leftarrow \mathsf{HKDF.HKDF\text{-}Expand}(\mathrm{HTS_C}, \texttt{"finished"})$
{`ClientFinished`}: $\mathsf{HMAC}(\mathrm{CFK}, H_2)$

check $\texttt{CF} = \mathsf{HMAC}(\mathrm{CFK}, H_2)$
$\mathrm{XHS} \leftarrow \mathsf{HKDF.HKDF\text{-}Expand}(\mathrm{HS}, \texttt{"derived"})$
$\mathrm{MS} \leftarrow \mathsf{HKDF.HKDF\text{-}Extract}(\mathrm{XHS}, 0)$
$H_3 \leftarrow \mathsf{H}(\texttt{CH}\|\ldots\|\texttt{SF})$
$\mathrm{TS_S}/\mathrm{TS_C} \leftarrow \mathsf{HKDF.HKDF\text{-}Expand}(\mathrm{MS}, \mathrm{label}_4/\mathrm{label}_5\|H_3)$
$tk_{app} = (tk^c_{app}/tk^s_{app}) \leftarrow \mathsf{HKDF.HKDF\text{-}Expand}(\mathrm{TS_S}/\mathrm{TS_C}, \mathrm{label}_3)$

Protocol flow legend

`MSG`: Y	TLS 1.3 message `MSG` containing Y
$+$ `MSG`	message sent as extension within previous message
{`MSG`}	message `MSG` AEAD-encrypted with tk^c_{hs}/tk^s_{hs}
a/b	alternative usage of a or b for server and client
label_i	specific label in derivation step

Figure 5.3: The TLS 1.3 [Res18] PSK handshake protocol.

the main problem is that the protocol crucially relies on the collision resistance of the hash function to compute the transcript hashes. As we discussed in Chapter 4, planting backdoors in collision-resistant hash functions is rather easy, such that we may not get immunity for the given protocol. Nonetheless, the transcript hashes are only used to enable the parties to store the intermediate hash values instead of the entire transcript. In terms of security, one can easily forgo the transcript hashes and feed the full transcript into the backdoor-resilient version of HMAC resp. HKDF.

Another obstacle to use our immunization strategy via salting of HKDF is that the salt needs to be picked independently of the input to the hash function. This can only be done by the party which evaluates the hash function next, e.g., when the server computes

$$\mathrm{HTS_C}/\mathrm{HTS_S} \leftarrow \mathsf{HKDF.HKDF\text{-}Expand}(\mathrm{HS}, \mathrm{label}_1/\mathrm{label}_2\|H_1)$$

over the transcript hash $H_1 \leftarrow \mathsf{H}(\texttt{CH}\|\texttt{SH})$, or rather the full transcript $H_1 \leftarrow \texttt{CH}\|\texttt{SH}$, to send the encrypted {`EncryptedExtensions`} message, then the entire input is only determined when the server is deriving the keys. The same holds on the client side for the finished message key CFK. Hence, we require that both parties at some point pick a random salt in a trustworthy way and therefore can only cover backdooring attacks against outsiders, eavesdropping on the communication and not being able to modify the salt. Still, we preserve active security against adversaries which cannot tamper with the cryptographic primitives.

Another problem with TLS 1.3 in its current form is that it is not clear how to embed the salt in the protocol flow. The extensions currently do not offer a variable field for this. Hence, one would need to change the specification to enable the inclusion of such extra data, as well as the algorithm specifiers to capture the salted versions.

With all the modifications above, one obtains a PSK mode which only relies on the backdoor-resilient modified primitives HMAC and HKDF. We omit a formal analysis as it would require to define security of key exchange protocols, which is beyond the scope here.

Chapter **6**

Simple Combiners for Backdoored Random Oracles

This chapter explores the possibility of re-establishing a number of security guarantees in hash functions without secret keys and in a setting where all available hash functions are backdoored. We show several positive results by combining two independently backdoored hash functions in the 2-BRO model using concatenation, cascade, and xor as combiners. We show that one-wayness, pseudorandomness, and collision resistance can hold in this setting against adversaries that have unrestricted and adaptive access to not just one but both backdoor oracles. To this end we give new reductions from cryptographic security of these combiners to the communication complexities of set-disjointness, set-intersection, and a new variant of set-intersection. We also establish a new lower bound for the communication complexity of set-intersection for cryptographically relevant ranges of parameters and distributions.

My Scientific Contribution in this Chapter

Most of the material in this chapter is published in [BFM18a], which is a joint work with Balthazar Bauer and Pooya Farshim. Pooya and I formalized the security notions rPre, oPRG, and IU, and connected rPre and oPRG to their classical relatives OW and PRG security (see the results in Section 6.2). Balthazar and Pooya jointly developed the proofs for the IU security of random functions in Section 6.2.1, while Pooya and I focused on proving the IU security of combiners, stated in Lemma 6.1. Pooya and I analyzed the attacks (for certain parameter regimes) of concatenation and cascade combiners, which are included in Sections 6.3.1 and 6.3.2, respectively. These attacks were part of the full version of our paper [BFM18b, Appendix D]. Pooya later noticed that a multi-collision attack applies when combining iterative hash functions; my formal analysis of this attack in Section 6.3.3 does not appear elsewhere. Pooya found out that refined lower bounds for the communication complexity of set-disjointness and set-intersection (from Section 6.4) could follow those of [MB12, GC13] for set-disjointness. Pooya had the idea behind extending the set-intersection lower bound for large errors. Pooya and I then jointly developed the proofs of Theorems 6.10 and 6.11. Balthazar established a refined lower bound for set-disjointness which extends Theorem 6.10 to $\delta \geq 0.8$ (instead of $\delta \geq 1$). This refinement is not included in this thesis and can be found in the full version [BFM18b, Appendix C.3]. The high-level approach and intuition of why OW, PRG, and CR should hold for our combiners in the 2-BRO model was suggested by Pooya. Marc Fischlin was involved in the early stages of this project, mainly in the discussions around the question of what type of lower bounds can be useful in our setting. Balthazar, Pooya, and I were all involved in the security results given in Sections 6.5, 6.6, and 6.7. My main focus was on concatenation, and xor and the problem instances, to which we can reduce the targeted security notions. Balthazar and Pooya developed the solutions of ReDist and HashSam to fix the problems with non-binomial distributions in the proofs of CR, in particular Theorem 6.16 and OW security of cascade, i.e., 6.18, respectively.

6.1 Introduction

We revisit a classical question on protections against failures of hash functions. Numerous works in this area have studied if, and to what extend, one can re-establish security guarantees by *combining* different hash functions; see for instance [BB06, FL07, FLP14, HS08] for theoretical treatments and [MRS09, LW15, Din16] for cryptanalytic work. However, most prior work focuses on unintentional failures to protect against cryptanalytic advances. We continue with our more adversarial view of hash function failures and ask if well-designed but possibly *backdoored* hash functions can be used to build backdoor-resilient hash functions.

Hash-function combiners in the works mentioned above typically convert two or more hash functions into a new one that is secure as long as *at least one* of the underlying hash functions is secure. For example, the concatenation combiner builds a collision-resistant hash function given k hash functions as long as one function is collision resistant. Multi-property combiners for other notions, such as PRG, MAC or PRF security, also exist [FLP14].

As discussed in Section 3.4, typical black-box combiners are not necessarily expected to offer protection when *all* hash functions fail. Intuitively, our goal is more challenging as we assume that all “sources of hardness” have been rendered useless. We observe that the result of Hoch and Shamir [HS08] can be seen as one building a collision-resistant hash function in the 2-BRO model assuming backdoor oracles that allow for random inversions only. Since our goal is to protect against *adversarial* weaknesses (a.k.a. backdoors), we place no assumptions on hash-function weaknesses—they can go well beyond computing random preimages or collisions. In other words, we assume unrestricted backdoor capability classes.

We ask whether BROs can be combined in a way that renders their backdoors useless. From a high-level point of view, our main result shows that in the 2-BRO model cryptographic hardness *can* be bootstrapped, even with access to *both* backdoor oracles and even when *arbitrary* backdoor capabilities are provided. In other words, there are secure constructions in the 2-BRO model that can tolerate arbitrary weaknesses in all underlying hash functions. At the core of our results lies new links with hard problems in the area of communication complexity. For a brief introduction on communication complexity we refer to Section 2.5.

6.1.1 Connection to Communication Complexity

We focus on three of the most important hash function combiners: concatenation, cascade, and xor of two hash functions H_1 and H_2:

$$\mathsf{C}_{\|}^{\mathsf{H}_1,\mathsf{H}_2}(x) := \mathsf{H}_1(x)\|\mathsf{H}_2(x)\ , \qquad \mathsf{C}_{\circ}^{\mathsf{H}_1,\mathsf{H}_2}(x) := \mathsf{H}_2(\mathsf{H}_1(x))\ ,$$

$$\mathsf{C}_{\oplus}^{\mathsf{H}_1,\mathsf{H}_2}(x) := \mathsf{H}_1(x) \oplus \mathsf{H}_2(x)\ .$$

Here $\mathsf{H}_1 \in \mathrm{Fun}[n, n+s_1]$ and $\mathsf{H}_2 \in \mathrm{Fun}[n, n+s_2]$ in the concatenation combiner, $\mathsf{H}_1 \in \mathrm{Fun}[n, n+s_1]$ and $\mathsf{H}_2 \in \mathrm{Fun}[n+s_1, n+s_1+s_2]$ in the cascade combiner, and $\mathsf{H}_1, \mathsf{H}_2 \in \mathrm{Fun}[n, n+s]$ in the xor combiner.

Consider the one-way security of the concatenation combiner in the 2-BRO model, described in

Figure 3.2. An adversary is given a point $y^* := y_1^* \| y_2^* := \mathsf{H}_1(x^*) \| \mathsf{H}_2(x^*)$ for a random x^*. It has access to the backdoor oracles BD_1 and BD_2 for functions H_1 and H_2 respectively. Its goal is to compute a preimage x for y^* under the construction $\mathsf{C}_{\|}^{\mathsf{H}_1,\mathsf{H}_2}$. This is the case iff $\mathsf{H}_1(x) = y_1^*$ and $\mathsf{H}_2(x) = y_2^*$. Now define two sets $S := \mathsf{H}_1^-(y_1^*)$, the set of preimages of y_1^* under H_1, and $T := \mathsf{H}_2^-(y_2^*)$, the set of preimages of y_2^* under H_2. *Thus the adversary wins in the OW game iff $x \in S \cap T$.*

The two backdoor oracles respectively know S and T as they are part of the descriptions of the two hash functions. This allows us to convert a successful one-way adversary to a two-party protocol that computes an element x of the intersection $S \cap T$, as we described below. Put differently, if the communication complexity of set-intersection for sets that are distributed as above has a high lower bound, then the adversary has to place a large number of queries, which, in turn, allows us to conclude that the concatenation combiner is one-way in the 2-BRO model.

On a high level, the reduction works as follows. Alice is given a set S and Bob is given a set T. Their goal is to find a common element $x \in S \cap T$. Alice and Bob sample some $y_1 \twoheadleftarrow \{0,1\}^{n+s_1}$ and $y_2 \twoheadleftarrow \{0,1\}^{n+s_2}$ uniformly at random. Then, Alice locally builds a random oracle H_1 such that $\mathsf{H}_1^-(y_1) = S$ holds, while for all $x \notin S$ she sets $\mathsf{H}_1(x) = y$ for some $y \twoheadleftarrow \{0,1\}^{n+s_1} \setminus \{y_1\}$. Analogously, Bob builds a random oracle H_2 using y_2 and T. Knowing the entire table of H_1 (resp. H_2), Alice (resp. Bob) can compute any information about the truth table of H_1 (res. H_2). Recall that Alice and Bob are unbounded in their local running time are only concerned about the cost of their communication. Let $\mathcal{A}$ be any algorithm that invert images under the concatenation combiner in the 2-BRO model. Alice and Bob run $\mathcal{A}$ in tandem on the input $y_1 \| y_2$, answering its queries using the simulated H_1 and H_2. That is, Alice answers BD_1 queries until a BD_2 query is made. Then, she hands the execution of $\mathcal{A}$ with its current state consisting of BD_1 responses over to Bob. Bob resumes $\mathcal{A}$ using the responses received from Alice, and answering its further BD_2 queries, until a BD_1 query is made, in which case he hands the execution with all new BD_2 responses over to Alice, and so on. Once $\mathcal{A}$ returns with some value x, either Alice or Bob communicates x as a last message and they both terminate. If $\mathcal{A}$ is successful and x is indeed a preimage of $y_1 \| y_2$, then it is also in the intersection $S \cap T$. The communication complexity of the protocol run by Alice and Bob is determined by number backdoor queries made by $\mathcal{A}$ and the size of the corresponding responses. We give a precise description of this protocol in Section 6.5.1.

Now the question is: for which sets S and T is set-intersection hard? Suppose the hash functions $\mathsf{H}_1, \mathsf{H}_2 \in \mathrm{Fun}[n, m]$ are compressing and $m = n - s$, for a positive integer $s \in \mathbb{N}$. Then, on average the sets S and T would each have 2^s elements. In a naive protocol, one party can of course communicate its entire set in $\mathcal{O}(2^s)$ bits and have the other party find a common element. However, the cost of this attack when s is linear in n (or even super-logarithmic in n) becomes prohibitive. This raises the question if set-intersection is hard for, say, $s = n/2$ and where the distribution over (S, T) is induced by the two hash functions, where except a single element in common (guaranteed to exist by the rules of the one-way game) all others are sampled uniformly and independently at random and included in the sets.

We observe that hardness of the set-disjointness problem implies hardness of set-intersection as the parties can verify that a given element is indeed in both their sets.[1] Set-disjointness is a better studied problem. To the best of our knowledge two results on lower bounds of set-disjointness

[1] On the other hand, for sufficiently large sets that intersect with high probability, set-disjointness is easy whereas set-intersection can remain hard.

with parameters and distributions close to those in our setting have been proven. First, a classical (and technical) result of Babai, Simon and Frankl [BFS86] which shows an $\Omega(\sqrt{N})$ lower bound for random and independent sets S and T of size *exactly* $\sqrt{N}$ in a universe of size N. Second, a result based on information-theoretic arguments due to Bar-Yossef et al. [BYJKS02], for dependent sets S and T, which has been adapted to *binomial product distributions* in lectures by Moshkovitz and Barak [MB12, Lecture 9] and Guruswami and Cheraghchi [GC13, Lecture 21]. The distribution is as follows: for each of the N elements in the universe, independent $\mathrm{Ber}(1/\sqrt{N})$ bits are sampled. (The probability of 1 is $1/\sqrt{N}$.) The sets then consist of all elements for which the bit is set to 1. Note that the expected size of such sets is $N/\sqrt{N} = \sqrt{N}$, but this size can deviate from the mean and this distribution is *not* identical to that by Babai et al. [BFS86]. The authors again prove an $\Omega(\sqrt{N})$ lower bound (which is tight up to logarithmic factors). We note that both these results only hold for protocols that err with probability at most $\varepsilon \leq 1/100$. However, we only found incomplete proofs of set-disjointness for product binomial distributions, and thus have included a self-contained proof here. We also prove a distributional communication complexity lower bound for set-*intersection* for parameters where set-disjointness can be *easy*. Both theorems can be found in Section 6.4.

The results with binomial product distribution are better suited for our purposes as the size restriction in [BFS86] would restrict us to regular random oracles. Indeed, the distribution induced on the preimages of y_1^* (resp. y_2^*) by the hash function outside the common random point *is* binomial, i.e., Bernoulli with the probability: $\Pr[\mathsf{H}_1(x) = y_1^*] = 1/2^m$ (resp. $\Pr[\mathsf{H}_2(x) = y_2^*] = 1/2^m$) for any x and independently for all values of x, where the probability is over the random choice of H_1 resp. H_2. We use this fact to show that set-intersection and set-disjointness problems are, respectively, sufficient to prove it is hard to invert random co-domain points (a property that we call random preimage resistance, rPre) or even decide if a preimage exists (which we call oblivious PRG, oPRG). The main benefit of these games is that they do away with the common point guaranteed to exist by the rules of one-way game (and also similar technicalities associated with the standard PRG game). These games can then be related to the one-way and PRG games via cryptographic reductions.

6.1.2 Security of Combiners in the 2-BRO Model

Our lower bound for set-intersection allows us to prove strong one-way security for some parameters, while the set-disjointness bound only enables proving weak PRG security. Using amplification techniques we can then convert the weak results to strong one-way functions [Gol01] or strong PRGs [MT10]. Note that the reductions for all these results are fully black-box and thus would relativize [RTV04]. This implies that the same proofs also hold in the presence of backdoor oracles. Construction of other primitives in minicrypt also relativize. This means we also obtain backdoor-resilient PRFs, MACs, PRPs, and symmetric encryption schemes in our model. The resulting constructions, however, are often too inefficient to be of any practical use. The bottleneck for PRG efficiency here is the proven lower bounds for set-disjointness. New lower bounds that give trade-offs between protocol error and communication complexity will enable more efficient/secure constructions. We show in Section 6.4 why the current proof does not permit this.

Recall that collision resistance can *not* be based on one-way functions [Sim98]. The concatenation combiner, on the other hand, appears to be collision resistant as *simultaneous collisions* seem hard to find, even with respect to arbitrary backdoors for each hash function. Indeed, an analysis of collision resistance for this combiner reveals a natural *multi-set* analogue of the set-intersection problem for finding two elements, a problem that we call *multi-set double-intersection* (or multi-set 2INT), which to the best of our knowledge has not been studied yet. We conjecture that this problem has a high communication complexity and then show that, assuming the hardness of this problem, we get collision resistance. We note that fully black-box amplification for collision resistance also exists [CRS$^+$07], and it is sufficient to prove hardness for small values of protocol error ε (should this be the case as in the setting of single-instance set-disjointness).

We carry out similar analyses for the *cascade* and *xor* combiners, for which different choices of parameters lead to security. Although the overall approach remains the same, we need to deal with some difficulties in the security proofs of these combiners. For the cascade combiner these arise from the fact that one of the sets is the *image* of a hash function. The latter distribution is somewhat different to binomial sets (as elements are not chosen independently). We show, however, that by addition of noise one-way and PRG security can be based on *known* lower bounds. For collision resistance we give a reduction to multi-set double-intersection. For the xor combiner, the communication complexity problem that directly underlies our reduction for one-wayness is multi-set: both parties hold multiple sets and need to find an intersection in two sets of their own choice. We are, however, able to relate this problem to the standard set-intersection by generating one larger set for each party which labels and contains all smaller sets.

We do not treat PRF security of combiners. Nonetheless, we remark that the weak PRF security of a combiner intuitively goes down to the problem of deciding whether there is at least one pair of keys k_1 and k_2 that can simultaneously explain the obtained values for H_1 and H_2, which is a variant of set-disjointness. We also leave out discussions on second-preimage resistance of combiners. The reason is that in this chapter we concentrate on achieving security and whenever we achieve one-wayness, second-preimage resistance follows. Similar to our detour in proving one-way security through rPre-security, in any case, we would also have to obtain second-preimage resistance through rPre-security.

We summarize our results in Table 6.1. The parameters for collision resistance and the weak PRG security of xor are conjectural. Recall that strong security demands that the advantage of adversaries in the corresponding security game is negligible, while for weak security it suffices that the advantage is not overwhelming. In the table, concatenation is with respect to hash function $\mathsf{H}_1 \in \mathrm{Fun}[n, n+s_1]$ and $\mathsf{H}_2 \in \mathrm{Fun}[n, n+s_2]$. Cascade is with respect to hash function $\mathsf{H}_1 \in \mathrm{Fun}[n, n+s_1]$ and $\mathsf{H}_2 \in \mathrm{Fun}[n+s_1, n+s_1+s_2]$. The xor combiner is with respect to hash functions $\mathsf{H}_1, \mathsf{H}_2 \in \mathrm{Fun}[n, n+s]$. The stretch values s_1, s_2, and s can assume negative values (compressing), positive values (expanding), or be zero (length-preserving). The range of secure stretches are described using a variable $0 \leq \sigma \leq 1$.

As a final remark, note that proofs in the random-oracle model often proceed via direct information-theoretic analyses. However, here we give cryptographic reductions (somewhat similarly to the standard model) that isolate the underlying communication complexity problems. These problems have diverse applications in other fields (such as circuit complexity, VLSI design, and combinatorial auctions), which motivate their study outside cryptographic contexts. Any improvement in lower bounds for them would also lead to improvements in the security/efficiency of cryptographic constructions. We discussed the benefits of proofs for arbitrary error above. As other examples, results in multi-party communication complexity would translate to the k-BRO model for $k > 2$ or those in quantum communication complexity can be used to build quantum-secure BRO combiners.

Combiner	**Strong OW**	**Weak PRG**	**Strong CR**
Concatenation	$s_1, s_2 = -(\sigma+1)\cdot n/2$ for $0 < \sigma < 1/3$	$s_1 = -n/2+1$, $s_2 = -n/2$	$s_1, s_2 \le -n/2-1$ (Conjectural)
Cascade	$s_1 = (1+\sigma)\cdot n, s_2 = -n$ for $-1/2 < \sigma < 0$	$s_1 = 2n$, $s_2 = -2n+1$	$s_1 = 2n$, $s_2 = -2n-1$ (Conjectural)
Xor	$s = \sigma \cdot n$ for $-0.42 < \sigma < 0$	Any s (Conjectural)	Any s (Conjectural)

Table 6.1: Overview of results for concatenation, cascade, and xor. Functions H_i have stretch s_i. For xor we assume $s_1 = s_2 = s$.

6.2 Random Preimage-Resistance and Oblivious PRGs

We define variants of the one-wayness and PRG games which will be helpful in our analyses. We formalize these games in Figure 6.2. The *random preimage-resistance* (rPre) game is defined similarly to everywhere preimage-resistance (ePre) [RS04] except that a random co-domain point (as opposed to any such point) must be inverted. This definition differs from one-way security in two aspects: the distribution of $\mathsf{C}(x)$ for a uniform x might not be uniform. Furthermore, some points in the co-domain might not have any preimages. We also define a decisional variant, called *oblivious PRG* (oPRG), where the adversary has to decide if a random co-domain point has a preimage. The advantage terms are defined as:

$$\mathsf{Adv}^{\mathrm{rpre}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{A}) := \Pr[\mathrm{rPre}^{\mathcal{A}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}] \qquad \mathsf{Adv}^{\mathrm{oprg}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{A}) := \Pr[\mathrm{oPRG}^{\mathcal{A}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}]$$

```
Game rPre$^{\mathcal{A}}_{\mathsf{C}}$
for $i = 1, 2$ do
  $\mathsf{H}_i \twoheadleftarrow \mathrm{Fun}[n_i, m_i]$
$y \twoheadleftarrow \{0,1\}^m$
$x' \leftarrow \mathcal{A}^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{BD}_1,\mathsf{BD}_2}(y)$
if $y \notin \mathsf{img}(\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2})$
    return $(x = \perp)$
return $(\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}(x) = y)$

Game oPRG$^{\mathcal{A}}_{\mathsf{C}}$
for $i = 1, 2$ do
  $\mathsf{H}_i \twoheadleftarrow \mathrm{Fun}[n_i, m_i]$
$y \twoheadleftarrow \{0,1\}^m$
$b' \leftarrow \mathcal{A}^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{BD}_1,\mathsf{BD}_2}(y)$
return $(b' = (y \in \mathsf{img}(\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2})))$
```

Figure 6.1: The random preimage-resistance (rPre) and oblivious PRG (oPRG) games for $\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2} \in \mathrm{Fun}[n,m]$. Let $\mathsf{img}(\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}) := \mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}(\{0,1\}^n)$.

Recall that weak analogues of security notions (for example weak rPre or weak oPRG) are defined by requiring the advantage to be bounded away from 1 (i.e., not to be overwhelming). These definitions can be formalized in the asymptotic language, but we use concrete parameters here.

We state two lemmas that relate OW and rPre as well as PRG and oPRG: for functions that have *uniform images*, as defined below, we show that OW security is implied by rPre security (Lemma 6.8) and PRG security is implied by oPRG security (Lemma 6.9).

6.2.1 Image Uniformity

In the image uniformity game IU for a construction $\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}$, defined in Figure 6.2, an adversary, given access to all backdoor oracles, must decide whether a given value is chosen uniformly at random from the image of $\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}$ or computed as the image of a value x chosen uniformly at random from the domain. The advantage term is

$$\mathsf{Adv}^{\mathrm{iu}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{A}) := 2 \cdot \Pr[\mathrm{IU}^{\mathcal{A}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}] - 1 \ .$$

```
Game $\mathrm{IU}^{\mathcal{A}}_{\mathsf{C}}$
for $i = 1, 2$ do
    $\mathsf{H}_i \twoheadleftarrow \mathrm{Fun}[n_i, m_i]$
$y_0 \twoheadleftarrow \mathsf{img}(\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2})$
$x \twoheadleftarrow \{0,1\}^n$
$y_1 \leftarrow \mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}(x)$
$b \twoheadleftarrow \{0,1\}$
$b' \leftarrow \mathcal{A}^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{BD}_1,\mathsf{BD}_2}(y_b)$
return $(b' = b)$
```

Figure 6.2: The image uniformity (IU) game for $\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2} \in \mathrm{Fun}[n, m]$.

The following lemma upper bounds the advantage of adversaries playing the image uniformity game for combiners with different stretch values. The results are then used to relate OW and rPre security as well as PRG and oPRG security notions.

Recall that we denote by $\mathcal{U}_S$ the uniform distribution over a set S. Let $f \in \mathrm{Fun}[n, m]$. By $\mathcal{U}^{\mathsf{p}}_f$ we denote the distribution over $\{0,1\}^m$, which for all $y \in \{0,1\}^m$ is defined by $\mathcal{U}^{\mathsf{p}}_f(y) = |f^{-1}(y)|/2^n$, i.e., the probability that an element of the co-domain of f is hit is proportional to the number of its preimages under f.

Lemma 6.1. *Image-uniformity of combiners Let $\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}_t \in \mathrm{Fun}[n, m]$ be a combiner for the type $t \in \{\|, \circ, \oplus\}$. Then it holds that*

$$\mathsf{Adv}^{\mathrm{iu}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}_t}(\mathcal{A}) \leq \mathbb{E}_{\mathsf{H} \in \mathrm{Fun}[n,m]} \left[\mathsf{SD}\big(\mathcal{U}_{\mathsf{img}(\mathsf{H})}, \mathcal{U}^{\mathsf{p}}_{\mathsf{H}}\big)\right] + 2 \cdot p_t \ ,$$

where $p_{\|} = p_{\oplus} = 0$ and $p_{\circ} \leq 2^{2n_1 - m_1}$ is the probability that $\mathsf{H}_1 \in \mathrm{Fun}[n_1, m_1]$ is not injective (i.e., it has at least one collision). Let $2^n = C \cdot 2^{m \cdot \gamma}$ for constants C and γ. Then the expected above statistical distance is negligible for $\gamma > 1$ and $0 < \gamma < 1$ when $C = 1$, while for $\gamma = 1$ and $C \leq 1$ it is less than $e^{-C} \cdot \big(C/(1 - e^{-C}) - 1\big)$ plus negligible terms.

Proof. Let $\mathsf{H}_1, \mathsf{H}_2$, and H be random variables denoting uniform functions in $\mathrm{Fun}[n_1, m_1]$, $\mathrm{Fun}[n_2, m_2]$, and $\mathrm{Fun}[n, m]$ respectively. The distinguishing advantage of an adversary $\mathcal{A}$ in the image-uniformity game is, by definition, upper bounded as

$$\mathsf{Adv}^{\mathrm{iu}}_{\mathsf{C}_t^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{A}) \le \mathsf{SD}\big((\mathsf{H}_1, \mathsf{H}_2, \mathsf{C}_t^{\mathsf{H}_1,\mathsf{H}_2}(X)), (\mathsf{H}_1, \mathsf{H}_2, Y)\big) \ ,$$

where X is uniform over $\{0,1\}^n$ and Y is uniform over the image of $\mathsf{C}_t^{\mathsf{H}_1,\mathsf{H}_2}$. Note that we can ignore the backdoor oracles in the statistical distance above, since the backdoor oracles can be fully determined by H_1 and H_2. We claim that

$$\begin{aligned} &\mathsf{SD}\big((\mathsf{H}_1, \mathsf{H}_2, \mathsf{C}_t^{\mathsf{H}_1,\mathsf{H}_2}(X)), (\mathcal{S}_t(\mathsf{H}), \mathsf{H}(X))\big) \le p_t \\ \text{and}\quad &\mathsf{SD}\big((\mathsf{H}_1, \mathsf{H}_2, Y), (\mathcal{S}_t(\mathsf{H}), Y)\big) \le p_t \ , \end{aligned}$$

where the simulators $\mathcal{S}_t(\mathsf{H})$ output simulated H_1 and H_2 tables as follows.

Concatenation: The simulator outputs the left m_1 bits and the right m_2 bits of the outputs of H as the two hash functions respectively. It is easily seen that such $(\mathsf{H}_1, \mathsf{H}_2)$ is identically distributed to $\mathcal{S}_{||}(\mathsf{H})$.

Cascade: The simulator samples uniformly at random H_1 from $\mathrm{Fun}[n_1, m_1]$ and H_2 from $\mathrm{Fun}[n_1, m_1]$. It then redefines H_2 to map all $\mathsf{H}_1(x)$ values to $\mathsf{H}(x)$. The latter results in a consistent random function unless $\mathsf{H}_1(x)$ is not injective. The probability of latter is, by definition, $p_\circ$.

Xor: The simulator samples a function H_1 uniformly at random from $\mathrm{Fun}[n_1, m_1]$ and then defines $\mathsf{H}_2(x) := \mathsf{H}(x) \oplus \mathsf{H}_1(x)$ for all x. It is easily seen that the pair $(\mathsf{H}_1, \mathsf{H}_2)$ is identically distributed to $\mathcal{S}_\oplus(\mathsf{H})$.

It thus remains to bound

$$\mathsf{SD}\big((\mathcal{S}_t(\mathsf{H}), \mathsf{H}(X)), (\mathcal{S}_t(\mathsf{H}), Y)\big) \ .$$

Since $\mathcal{S}_t(\mathsf{H})$ is a randomized transformation of H, we can simply bound $\mathsf{SD}\big((\mathsf{H}, \mathsf{H}(X)), (\mathsf{H}, Y)\big)$. Note that $\mathsf{H}(X)$ is identical to $\mathcal{U}^{\mathsf{p}}_{\mathsf{H}}$ and Y is identical to $\mathcal{U}_{\mathsf{img}(\mathsf{H})}$. We have

$$\begin{aligned} \mathsf{SD}\big((\mathsf{H}, \mathcal{U}_{\mathsf{img}(\mathsf{H})}), (\mathsf{H}, \mathcal{U}^{\mathsf{p}}_{\mathsf{H}})\big) &= \frac{1}{2} \sum_{(h,y)} \big|\Pr[(\mathsf{H}, \mathcal{U}_{\mathsf{img}(\mathsf{H})}) = (h,y)] - \Pr[(\mathsf{H}, \mathcal{U}^{\mathsf{p}}_{\mathsf{H}}) = (h,y)]\big| \\ &= \frac{1}{2} \sum_{(h,y)} \Pr[\mathsf{H} = h] \, \big|\Pr[h(x) = y] - \Pr[\mathcal{U}_{\mathsf{img}(h)} = y]\big| \\ &= \frac{1}{2} \sum_{h} \Pr[\mathsf{H} = h] \sum_{y} \big|\Pr[h(x) = y] - \Pr[\mathcal{U}_{\mathsf{img}(h)} = y]\big| \\ &= \frac{1}{2} \sum_{h} \Pr[\mathsf{H} = h] \mathsf{SD}(\mathsf{H}(x), \mathcal{U}_{\mathsf{img}(h)}) \\ &= \mathbb{E}_{h \twoheadleftarrow \mathrm{Fun}[n,m]} \big[\mathsf{SD}(\mathcal{U}_{\mathsf{img}(h)}, \mathcal{U}^{\mathsf{p}}_{h})\big] \ . \end{aligned}$$

For the upper bound on $p_\circ$, by Stirling's approximations and using $m_1 > 2n_1$ we have

$$\Pr_{\mathsf{H}_1 \twoheadleftarrow \mathrm{Fun}[n_1,m_1]}[\mathsf{H}_1 \text{ is injective}] = \frac{1}{(2^{m_1})^{2^{n_1}}} \cdot \frac{2^{m_1}!}{(2^{m_1} - 2^{n_1})!} > e^{-2^{2n_1 - m_1}} > 1 - 2^{2n_1 - m_1} \ .$$

Thus

$$p_\circ := \Pr_{\mathsf{H}_1 \twoheadleftarrow \mathrm{Fun}[n_1,m_1]}[\mathsf{H}_1 \text{ not injective}] \leq 2^{2n_1 - m_1} \ ,$$

which is negligible for $m_1 = (2+\epsilon) \cdot n_1$ with $\epsilon > 1$. We note that for $m_1 = 2n_1$ we asymptotically have that $p_\circ = 1 - 1/\sqrt{e}$.

The above expected statistical distance can be bounded by Corollaries 6.4, 6.5, and 6.7 as given below. □

Image Uniformity of Random Functions

In the remaining of this section we upper-bound the expected statistical distance $\mathbb{E}_{\mathsf{H}}\left[\mathsf{SD}\big(\mathcal{U}_{\mathsf{img(H)}}, \mathcal{U}_{\mathsf{H}}^{\mathsf{p}}\big)\right]$ from Lemma 6.1, i.e., for the random choice of $\mathsf{H} \twoheadleftarrow \mathrm{Fun}[n,m]$, we upper-bound the difference between choosing an image of H uniformly at random versus choosing an image proportional to the number of its preimages under H. For a more compact notation, in the following we use $[M]^N$ for the set of all functions $\mathsf{H} : [N] \to [M]$.

The expanding case. We start by treating expanding functions. The following lemma give us the expected image size for functions $\mathsf{H} \in [M]^N$.

Lemma 6.2. *Let $N, M \in \mathbb{N}$. Then the expected image size (i.e., the number of non-empty bins in the balls-in-bins problem) is*

$$\mathbb{E}_{\mathsf{H}}[|\mathsf{img}(\mathsf{H})|] = M - M \cdot (1 - \frac{1}{M})^N \ .$$

Proof. Below, all probabilities and expectations are taken over the random choice of $\mathsf{H} \twoheadleftarrow [M]^N$.

$$\begin{aligned}
\mathbb{E}_{\mathsf{H}}[|\mathsf{img}(\mathsf{H})|] &= \mathbb{E}[|\{y \in [M] : \exists x \in [N] : \mathsf{H}(x) = y\}|] \\
&= \sum_{y \in [M]} \Pr[\exists x \in [N] : \mathsf{H}(x) = y] \\
&= M \cdot (1 - \Pr[\forall x \in [N] : \mathsf{H}(x) \neq y]) \\
&= M \cdot (1 - (1 - 1/M)^N) \\
&= M - M \cdot (1 - 1/M)^N
\end{aligned}$$

In the second equality above we have used the linearity of expectation. □

We now bound the expected statistical distance for expanding functions in the following lemma.

Lemma 6.3. *Let N, M some integers, let $1 \leq \alpha_N < N$. Then*

$$\mathbb{E}_{\mathsf{H}}\left[\mathsf{SD}(\mathcal{U}_{\mathsf{img(H)}}, \mathcal{U}_{\mathsf{H}}^{\mathsf{p}})\right] \leq \Pr\left[N - |\mathsf{H}([N])| \geq \alpha_N\right] + \frac{\alpha_N}{N} \ .$$

Proof. Let $E_{\alpha_N} := \{\mathsf{H} \in [M]^N : N - |\mathsf{H}([N])| < \alpha_N\}$, i.e., the set of those functions with a support which has at most α_N elements less than the domain. We have

$$\mathbb{E}_{\mathsf{H}}\left[\mathsf{SD}(\mathcal{U}_{\mathsf{img}(\mathsf{H})},\mathcal{U}^{\mathsf{p}}_{\mathsf{H}})\right] \leq \Pr\left[N - |\mathsf{H}([N])| \geq \alpha_N\right] + \mathbb{E}_{\mathsf{H}\in E_{\alpha_N}}\left[\mathsf{SD}(\mathcal{U}_{\mathsf{img}(\mathsf{H})},\mathcal{U}^{\mathsf{p}}_{\mathsf{H}})\right] .$$

In order to upper bound $\mathsf{SD}(\mathcal{U}_{\mathsf{img}(\mathsf{H})},\mathcal{U}^{\mathsf{p}}_{\mathsf{H}})$ for each $\mathsf{H} \in E_{\alpha_N}$ we define

$$S^- := \{i \in [M] : \mathcal{U}_{\mathsf{img}(\mathsf{H})}(i) - \mathcal{U}^{\mathsf{p}}_{\mathsf{H}}(i) < 0\} .$$

Then, by the symmetry of statistical distance, and the fact that $|\mathsf{H}([N])| \leq N$ we have

$$\mathsf{SD}(\mathcal{U}_{\mathsf{img}(\mathsf{H})},\mathcal{U}^{\mathsf{p}}_{\mathsf{H}}) \leq \sum_{i\in S^-}\left(\frac{|\mathsf{H}^{-1}(i)|}{N} - \frac{1}{N}\right) \leq \sum_{i\in\mathsf{H}([N])}\left(\frac{|\mathsf{H}^{-1}(i)|}{N} - \frac{1}{N}\right) \leq \frac{N - |\mathsf{H}([N])|}{N} \leq \frac{\alpha_N}{N} .$$

We have also used the facts that the summands are positive. Note also that $\sum_{i\in\mathsf{H}([N])} |\mathsf{H}^{-1}(i)| = N$ and $\sum_{i\in\mathsf{H}([N])} 1 = |\mathsf{H}([N])|$. This concludes the proof of the claim. □

For large M, it holds that $\ln\left(1 - \frac{1}{M}\right) = -\frac{1}{M} - \frac{1}{2M^2} + \mathcal{O}\left(\frac{1}{M^3}\right)$. Hence, we can write $\left(1 - \frac{1}{M}\right)^N = \exp\left(N \cdot \ln\left(1 - \frac{1}{M}\right)\right) = \exp\left(-\frac{N}{M} - \frac{N}{2M^2} + \mathcal{O}\left(\frac{N}{M^3}\right)\right)$. Note that $\exp(x)$ is the natural exponential function, i.e., e^x.

Corollary 6.4. *Let $0 < \gamma < 1$ and C be a positive constant. Suppose $N = C \cdot M^\gamma$. Then $\mathbb{E}_{\mathsf{H}}(\mathsf{SD}(\mathcal{U}_{\mathsf{img}(\mathsf{H})},\mathcal{U}^{\mathsf{p}}_{\mathsf{H}}))$ is negligible in $\log N$.*

Proof. We set $\alpha_N := M^{(1-\gamma)/2}\left(N - M + M(1 - \frac{1}{M})^N\right)$. By construction, and using the expected number of collisions and Markov's inequality,[2] we have

$$\Pr\left[N - |\mathsf{H}([N])| \geq \alpha_N\right] \leq 1/M^{(1-\gamma)/2} .$$

Thus the first term in the bound shown in the claim above is negligible. It remains to show that α_N/N is also negligible in $\log N$. We have

$$\begin{aligned}\alpha_N &= M^{(1-\gamma)/2}\left(CM^\gamma - M + M \cdot \exp\left(-C \cdot M^{\gamma-1} - C/2 \cdot M^{\gamma-2} + \mathcal{O}\left(M^{\gamma-3}\right)\right)\right)\\ &= M^{(1-\gamma)/2}\left(C \cdot M^\gamma - M + M\left(1 - C \cdot M^{\gamma-1} + \mathcal{O}\left(M^{2\gamma-2}\right)\right)\right)\\ &= M^{(1-\gamma)/2}\mathcal{O}\left(M^{2\gamma-1}\right) = \mathcal{O}\left(M^{3/2\cdot\gamma-1/2}\right) .\end{aligned}$$

Thus $\alpha_N/N = \mathcal{O}(M^{(\gamma-1)/2})$, which is negligible for $\gamma < 1$. □

The length-preserving case. We now deal with the case where the function is either length preserving or only slightly expanding.

[2] Recall this states that $\Pr[X \geq t\mathbb{E}[X]] \leq 1/t$ for positive X.

Lemma 6.5. *Suppose $N = CM$ with $C \leq 1$, and $0 < \delta < 1$. Set $\mu_1 := M(1 - \frac{1}{e^C})$ and $\mu_2 := \frac{CM}{e^C}$. Then*

$$\begin{aligned}\mathbb{E}_{\mathsf{H}}[\mathsf{SD}(\mathcal{U}_{\mathsf{img}(\mathsf{H})}, \mathcal{U}^{\mathsf{p}}_{\mathsf{H}})] \leq & \Pr\left[\left|\mathsf{H}\left([N]\right)\right| < \left(1-\delta\right)\mu_1\right] \\ & + \Pr\left[\left|\left\{i \in [M] : \left|\mathsf{H}^{-1}\left(i\right)\right| = 1\right\}\right| > \left(1+\delta\right)\mu_2\right] \\ & + \frac{\left(1+\delta\right)\left(C - \left(1-\delta\right)\left(1-\frac{1}{e^C}\right)\right)}{e^C\left(1-\delta\right)\left(1-\frac{1}{e^C}\right)} \ ,\end{aligned}$$

where for negligible δ, the first two terms are negligible and the last term is $\frac{1}{e^C} \cdot \left(\frac{C}{\left(1-\frac{1}{e^C}\right)} - 1\right)$.

Proof. Let us define a set $F \subseteq [M]^N$ of approximately length-preserving hash functions as the intersection of two sets F_1 and F_2 (i.e., $F := F_1 \cap F_2$) as follows. Roughly speaking F_1 contains hash functions that are not too compressing, while F_2 contains hash functions that are not too expanding (or have a certain number of collisions). Later in the proof, we will set the variables δ, μ_1, and μ_2.

$$\begin{aligned}F_1 &:= \left\{\mathsf{H} \in [M]^N : \left|\mathsf{H}\left([N]\right)\right| \geq \left(1-\delta\right)\mu_1\right\} \\ F_2 &:= \left\{\mathsf{H} \in [M]^N : \left|\left\{i \in [M] : \left|\mathsf{H}^{-1}\left(i\right)\right| = 1\right\}\right| \leq \left(1+\delta\right)\mu_2\right\} \ .\end{aligned}$$

Then

$$\begin{aligned}\mathbb{E}_{\mathsf{H}}[\mathsf{SD}(\mathcal{U}_{\mathsf{img}(\mathsf{H})}, \mathcal{U}^{\mathsf{p}}_{\mathsf{H}})] &\leq \Pr\left[\mathsf{H} \notin F\right] + \mathbb{E}_{\mathsf{H} \in F}\left[\mathsf{SD}\left(\mathcal{U}_{\mathsf{img}(\mathsf{H})}, \mathcal{U}^{\mathsf{p}}_{\mathsf{H}}\right)\right] \\ &\leq \Pr\left[\,\left|\mathsf{H}\left([N]\right)\right| < \left(1-\delta\right)\mu_1\right] + \Pr\left[\,\left|\left\{i \in [M] : \left|\mathsf{H}^{-1}\left(i\right)\right| = 1\right\}\right| > \left(1+\delta\right)\mu_2\right] \\ &\quad + \mathbb{E}_{\mathsf{H} \in F}\left[\mathsf{SD}(\mathcal{U}_{\mathsf{img}(\mathsf{H})}, \mathcal{U}^{\mathsf{p}}_{\mathsf{H}})\right] \ .\end{aligned}$$

For each $\mathsf{H} \in F$ define $S^-_{\mathsf{H}} := \{i \in [M] : \mathcal{U}_{\mathsf{img}(\mathsf{H})}(i) - \mathcal{U}^{\mathsf{p}}_{\mathsf{H}}(i) < 0\}$. For $\mathsf{H} \in F$ we have $\left|\mathsf{H}\left([N]\right)\right| \geq \left(1-\delta\right)\mu_1$ and for $i \in S^-_{\mathsf{H}}$ we have $\mathcal{U}_{\mathsf{img}(\mathsf{H})}(i) < \mathcal{U}^{\mathsf{p}}_{\mathsf{H}}(i)$. Thus,

$$\frac{\left|\mathsf{H}^{-1}\left(i\right)\right|}{N} < \frac{1}{\left|\mathsf{H}\left([N]\right)\right|} \leq \frac{1}{\left(1-\delta\right)\mu_1} \ ,$$

$$\left|\mathsf{H}^{-1}\left(i\right)\right| \leq \frac{C}{\left(1-\frac{1}{e^C}\right)} \leq \frac{1}{\left(1-\frac{1}{e}\right)} < 2 \ .$$

The penultimate inequality is due to the fact that the map $C \mapsto \frac{C}{\left(1-\frac{1}{e^C}\right)}$ has a positive derivative and that we have assumed $C \leq 1$. Hence, $\left|\mathsf{H}^{-1}\left(i\right)\right| < 2$ and because $i \in \mathsf{H}\left([N]\right)$, we can deduce $\left|\mathsf{H}^{-1}\left(i\right)\right| = 1$ (since $\left|\mathsf{H}^{-1}\left(i\right)\right|$ is an integer). Hence, $S^-_{\mathsf{H}} \subseteq \{i \in [M] : \left|\mathsf{H}^{-1}\left(i\right)\right| = 1\}$, which allows us to upper bound the cardinality of S^-_{H} by $\left(1+\delta\right)\mu_2$ (since $\mathsf{H} \in F \subseteq F_2$). Now

$$\begin{aligned}\mathsf{SD}\left(\mathcal{U}_{\mathsf{img}(\mathsf{H})}, \mathcal{U}^{\mathsf{p}}_{\mathsf{H}}\right) &= \sum_{i \in S^-_{\mathsf{H}}}\left(\frac{1}{\left|\mathsf{H}(([N]))\right|} - \frac{\left|\mathsf{H}^{-1}\left(i\right)\right|}{N}\right) \leq \sum_{i \in S^-_{\mathsf{H}}}\left(\frac{1}{\left(1-\delta\right)M\left(1-\frac{1}{e^C}\right)} - \frac{1}{N}\right) \\ &\leq \frac{\left(1+\delta\right)CM}{e^C}\left(\frac{1}{\left(1-\delta\right)M\left(1-\frac{1}{e^C}\right)} - \frac{1}{CM}\right) = \frac{\left(1+\delta\right)\left(C - \left(1-\delta\right)\left(1-\frac{1}{e^C}\right)\right)}{e^C\left(1-\delta\right)\left(1-\frac{1}{e^C}\right)} \ .\end{aligned}$$

If we set $\delta := N^{-1/4}$ the (expectation of the) above term is asymptotically upper bounded by $\frac{1}{e^C} \cdot \left(\frac{C}{\left(1-\frac{1}{e^C}\right)} - 1 \right)$ as N grows. We can also use the Chernoff bounds to upper bound the two remaining probabilities. We note that

$$\mathbb{E}_{\mathsf{H}}\left[\left|\mathsf{H}\left([N]\right)\right|\right] = M - M\left(1 - \frac{1}{M}\right)^{CM} \geq M(1 - \frac{1}{e^C}) = \mu_1 \ ,$$

and that

$$\begin{aligned}\mathbb{E}_{\mathsf{H}}\left[\left|\left\{i \in [M] : \left|\mathsf{H}^{-1}\left(i\right)\right| = 1\right\}\right|\right] &= \sum_{i \in [M]} \Pr\left(\left|\mathsf{H}^{-1}\left(i\right)\right| = 1\right) = M \cdot \binom{CM}{1}\left(1 - \frac{1}{M}\right)^{CM-1} \frac{1}{M} \\ &\leq \frac{CM}{e^C} = \mu_2 \ .\end{aligned}$$

Thus μ_1 lower bounds the first expectation and μ_2 upper bounds the second expectation. Using these facts and Chernoff we obtain

$$\begin{aligned}&\Pr\left[\,\left|\mathsf{H}\left([N]\right)\right| < (1-\delta)\,\mu_1\right] \leq e^{-\frac{1}{2}\delta^2 \cdot M\left(1-\left(1-\frac{1}{M}\right)^{CM}\right)} \leq e^{-\frac{1}{2C}\cdot\sqrt{N}\left(1-\frac{1}{e^C}\right)} \\ &\Pr\left[\left|\left\{i \in [M] : \left|\mathsf{H}^{-1}\left(i\right)\right| = 1\right\}\right| > (1+\delta)\,\mu_2\right] \leq e^{-\frac{1}{3}\delta^2 \cdot CM\left(1-\frac{1}{M}\right)^{CM-1}} \leq e^{-\frac{1}{3e^C}\cdot\sqrt{N}} \ ,\end{aligned}$$

where we used the facts that $\left(1 - \frac{1}{M}\right)^{CM} \leq \frac{1}{e^C}$ and $\left(1 - \frac{1}{M}\right)^{CM-1} \geq \frac{1}{e^C}$. □

The compressing case. We now deal with the case of compressing functions.

Lemma 6.6. *Let N, M be integers and $\delta > 0$. Then*

$$\mathbb{E}_{\mathsf{H}}\left[\mathsf{SD}(\mathcal{U}_{\mathsf{img(H)}}, \mathcal{U}^{\mathsf{p}}_{\mathsf{H}})\right] \leq \Pr\left[\exists i \in [M] : \left|\left|\mathsf{H}^{-1}(i)\right| - \frac{N}{M}\right| \geq \delta\frac{N}{M}\right] + \Pr\left[\mathsf{H}\left([N]\right) \neq [M]\right] + \frac{\delta}{2} \ .$$

Proof. Let F be a set of compressing hash functions where the set of images covers the entire co-domain and where the number of preimages for each image is not far from N/M.

$$F := \left\{\mathsf{H} \in [M]^N : \mathsf{H}\left([N]\right) = [M] \wedge \forall i \in [M] \left|\left|\mathsf{H}^{-1}(i)\right| - \frac{N}{M}\right| < \delta\frac{N}{M}\right\} \ .$$

Then we can bound the desired statistical distance as follows.

$$\mathbb{E}_{\mathsf{H}}\left[\mathsf{SD}(\mathcal{U}_{\mathsf{img(H)}}, \mathcal{U}^{\mathsf{p}}_{\mathsf{H}})\right] \leq \Pr\left[\mathsf{H} \notin F\right] + \mathbb{E}_{\mathsf{H} \in F}\left[\mathsf{SD}(\mathcal{U}_{\mathsf{img(H)}}, \mathcal{U}^{\mathsf{p}}_{\mathsf{H}})\right] \ .$$

Let $\mathsf{H} \in F$. Then

$$\mathsf{SD}(\mathcal{U}_{\mathsf{img(H)}}, \mathcal{U}^{\mathsf{p}}_{\mathsf{H}}) \leq \frac{1}{2}\sum_{i \in [M]}\left|\frac{\left|\mathsf{H}^{-1}(i)\right|}{N} - \frac{1}{M}\right| \leq \frac{1}{2N}\sum_{i \in [M]}\left|\left|\mathsf{H}^{-1}(i)\right| - \frac{N}{M}\right| \leq \frac{1}{2N}\sum_{i \in [M]}\delta\frac{N}{M} \leq \frac{\delta}{2} \ .$$

□

Corollary 6.7. *Suppose $N = M^\gamma$ with $\gamma > 1$. Then $\mathbb{E}_{\mathsf{H}}\left[\mathsf{SD}(\mathcal{U}_{\mathsf{img(H)}}, \mathcal{U}^{\mathsf{p}}_{\mathsf{H}})\right]$ is negligible in $\log N$.*

Proof. Let $\delta := M^{(1-\gamma)/4}$. Note that δ is negligible in $\log M$. Furthermore,

$$\Pr\left[\mathsf{H}\left([N]\right) \neq [M]\right] \leq \Pr\left[\exists i \in [M] : f^{-1}(i) = \emptyset\right] \leq M \cdot \Pr\left[f^{-1}(0) = \emptyset\right]$$
$$\leq M\left(1 - \frac{1}{M}\right)^N \leq M e^{-M^{\gamma-1}} ,$$

which is negligible. We now upper bound the first term in the claim above.

$$\Pr\left[\exists i \in [M] : \left|\left|\mathsf{H}^{-1}(i)\right| - \frac{N}{M}\right| \geq \delta\frac{N}{M}\right] \leq \sum_{i=0}^{M-1} \Pr\left[\left|\left|\mathsf{H}^{-1}(i)\right| - \frac{N}{M}\right| \geq \delta\frac{N}{M}\right]$$
$$\leq M \cdot \Pr\left[\left|\left|\mathsf{H}^{-1}(0)\right| - \frac{N}{M}\right| \geq \delta\frac{N}{M}\right] .$$

By the Chernoff bounds we have

$$\Pr\left[\exists i \in [M] : \left|\left|\mathsf{H}^{-1}(i)\right| - \frac{N}{M}\right| \geq \delta\frac{N}{M}\right] \leq 2e^{-\frac{\delta^2 N}{3M}} = 2e^{-\frac{M^{(\gamma-1)/2}}{3}} ,$$

which when multiplied by M remains negligible. □

6.2.2 rPre and One-Way Security

Now we can relate our notions of rPre and oPRG to their classical variants, i.e., one-wayness and PRG security.

Lemma 6.8 (rPre + IU $\implies$ OW). *Let $\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2} \in \mathrm{Fun}[n,m]$ be a construction in the 2-BRO model. Then for any adversary $\mathcal{A}$ against the one-way security of $\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}$, there is an adversary $\mathcal{B}$ against the image uniformity and an adversary $\mathcal{C}$ against the* rPre *security of $\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}$, all in the 2-BRO model and using identical backdoor functionalities, such that*

$$\mathsf{Adv}^{\mathrm{ow}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathrm{iu}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{B}) + \frac{1}{\alpha} \cdot \mathsf{Adv}^{\mathrm{rpre}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{C}) - \frac{1-\alpha}{\alpha} ,$$

where $\alpha := \Pr[y \in \mathsf{img}(\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2})]$ over the random choice of $y \in \{0,1\}^m$, H_1, and H_2.

Proof. Let OW-RI (OW with random image) be a game which differs from the standard OW game in that in OW-RI the challenge y is chosen randomly from the set of possible images $\mathsf{img}(\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2})$ rather than indirectly as the image of a randomly chosen $x \in \{0,1\}^n$. By definition of these games, any difference in the advantage of an adversary $\mathcal{A}$ in these two games is bounded by the advantage of an adversary $\mathcal{B}$ in the IU game, i.e.,

$$\mathsf{Adv}^{\mathrm{ow}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{A}) - \mathsf{Adv}^{\mathrm{ow\text{-}ri}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathrm{iu}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{B}) .$$

We can now build an adversary $\mathcal{C}$ against rPre security of $\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}$ as follows. On input $y \in \{0,1\}^m$, algorithm $\mathcal{C}^{\mathsf{BD}_1,\mathsf{BD}_2}$ simply runs $\mathcal{A}^{\mathsf{BD}_1,\mathsf{BD}_2}(y)$, while using its own backdoor oracles to answer $\mathcal{A}$'s queries. After $\mathcal{A}$ terminates with some x, adversary $\mathcal{C}$ outputs x if $\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}(x) = y$ and $\perp$ otherwise.

Let a and c be the outputs of $\mathcal{A}^{\mathsf{BD}_1,\mathsf{BD}_2}(y)$ and $\mathcal{C}^{\mathsf{BD}_1,\mathsf{BD}_2}(y)$ respectively. Let $I := \mathsf{img}(\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2})$, i.e., the image of $\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}$. Below we analyze $\mathcal{C}$'s advantage in the game rPre where all probabilities are over $y \twoheadleftarrow \{0,1\}^m$ and random choices of H_1 and H_2.

$$\begin{aligned}
\mathsf{Adv}^{\mathrm{rpre}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{C}) &= \Pr[\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}(c) = y \wedge y \in I] + \Pr[c = \bot \wedge y \notin I] \\
&= \Pr[\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}(a) = y \wedge y \in I] + \Pr[\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}(a) \neq y \wedge y \notin I] \\
&= \Pr[\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}(a) = y \mid y \in I] \cdot \Pr[y \in I] + \Pr[\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}(a) \neq y \wedge y \notin I] \\
&= \mathsf{Adv}^{\mathrm{ow\text{-}ri}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{A}) \cdot \Pr[y \in I] + \Pr[\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}(a) \neq y \mid y \notin I] \cdot \Pr[y \notin I] \\
&= \mathsf{Adv}^{\mathrm{ow\text{-}ri}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{A}) \cdot \alpha + 1 \cdot (1-\alpha) \\
&\geq \mathsf{Adv}^{\mathrm{ow}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{A}) \cdot \alpha - \mathsf{Adv}^{\mathrm{iu}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{B}) \cdot \alpha + (1-\alpha)
\end{aligned}$$

Therefore, we get

$$\mathsf{Adv}^{\mathrm{ow}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathrm{iu}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{B}) + \frac{1}{\alpha} \cdot \mathsf{Adv}^{\mathrm{rpre}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{C}) - \frac{1-\alpha}{\alpha} \ .$$

□

As long as $\mathsf{Adv}^{\mathrm{rpre}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{C}) < 1 - \alpha \cdot \mathsf{Adv}^{\mathrm{iu}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{B})$ (and is not too close to 1), we get $\mathsf{Adv}^{\mathrm{ow}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{A}) < 1$, which means weak rPre with advantage smaller than $1 - \alpha \cdot \mathsf{Adv}^{\mathrm{iu}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{B})$ translates to weak one-way security. Note that a non-negligible bound on IU advantage would be sufficient for weak OW security as long as rPre advantage is sufficiently small. For overwhelming α and negligible image-uniformity advantage, strong rPre security implies strong OW security.

6.2.3 oPRG and PRG Security

An analogous result also holds for oPRG security.

Lemma 6.9 (oPRG + IU $\implies$ PRG)**.** *Let $\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2} \in \mathrm{Fun}[n,m]$ be a construction in the 2-BRO model which is expanding with $m - n \geq 0.53$. Then for any adversary $\mathcal{A}$ against the PRG security of $\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}$, there is an adversary $\mathcal{B}$ against the image uniformity and an adversary $\mathcal{C}$ against the* oPRG *security of $\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}$, both in the 2-BRO model and using identical backdoor functionalities, such that*

$$\mathsf{Adv}^{\mathrm{prg}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathrm{iu}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{B}) + \frac{1-\alpha}{\alpha} \cdot \mathsf{Adv}^{\mathrm{oprg}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{C}) - (1-\alpha) \ ,$$

where $\alpha := \Pr[y \in \mathsf{img}(\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2})]$ over the random choice of $y \in \{0,1\}^m$, H_1, and H_2.

Proof. Let PRG-RI (PRG with random image) be a game which differs from the standard PRG game in that in PRG-RI if a secret bit is 1 the challenge y is chosen randomly from the set of possible images $\mathsf{img}(\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2})$ and not indirectly as the image of a randomly chosen $x \in \{0,1\}^n$. By definition of these games, any difference in the advantage of an adversary $\mathcal{A}$ in these two games is bounded by the advantage of an adversary $\mathcal{B}$ in the IU game, i.e.,

$$\mathsf{Adv}^{\mathrm{prg}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{A}) - \mathsf{Adv}^{\mathrm{prg\text{-}ri}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathrm{iu}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{B}) \ .$$

Note that we use image uniformity only once above, since in case $b = 1$ the PRG and PRG-RI games are identical.

Next we build an adversary $\mathcal{C}$ against the oPRG security of $\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}$ as follows. On input $y \in \{0,1\}^m$, algorithm $\mathcal{C}$ simply runs $\mathcal{A}^{\mathsf{BD}_1,\mathsf{BD}_2}(y)$, while using its own backdoor oracles to answer $\mathcal{A}$'s queries. After $\mathcal{A}$ terminates with a guess bit, $\mathcal{C}$ outputs the same bit as its own guess.

Let a and c be shorthand for $\mathcal{A}^{\mathsf{BD}_1,\mathsf{BD}_2}(y) = 1$ and $\mathcal{C}^{\mathsf{BD}_1,\mathsf{BD}_2}(y) = 1$, respectively. Furthermore, let $I := \mathsf{img}(\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2})$. Below we analyze $\mathcal{C}$'s advantage in the game oPRG, where all probabilities are over $y \twoheadleftarrow \{0,1\}^m$ and random choices of functions H_1 and H_2.

$$\begin{aligned}\mathsf{Adv}^{\text{oprg}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{C}) &= \Pr[c \wedge y \in I] + \Pr[\neg c \wedge y \notin I] \\ &= \Pr[a \wedge y \in I] + \Pr[\neg a \wedge y \notin I] \\ &= \Pr[a \mid y \in I] \cdot \Pr[y \in I] + \Pr[\neg a \mid y \notin I] \cdot \Pr[y \notin I] \\ &= \Pr[a \mid y \in I] \cdot \alpha + \Pr[\neg a \mid y \notin I] \cdot (1-\alpha)\end{aligned}$$

Let $b_{\text{prg-ri}}$ be the challenge bit chosen in the PRG-RI game. We can now write the advantage of $\mathcal{A}$ in the intermediate PRG-RI game as below.

$$\begin{aligned}\mathsf{Adv}^{\text{prg-ri}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{A}) &= 2 \cdot \Pr[a = b_{\text{prg-ri}}] - 1 \\ &= \Pr[a \mid y \in I] + \Pr[\neg a] - 1 \\ &= \Pr[a \mid y \in I] + \Pr[\neg a \mid y \in I] \cdot \Pr[y \in I] + \Pr[\neg a \mid y \notin I] \cdot \Pr[y \notin I] - 1 \\ &= \Pr[a \mid y \in I] + \Pr[\neg a \mid y \in I] \cdot \alpha + \Pr[\neg a \mid y \notin I] \cdot (1-\alpha) - 1 \\ &= \Pr[a \mid y \in I] + (1 - \Pr[a \mid y \in I]) \cdot \alpha + \Pr[\neg a \mid y \notin I] \cdot (1-\alpha) - 1 \\ &= \Pr[a \mid y \in I] \cdot (1-\alpha) + \Pr[\neg a \mid y \notin I] \cdot (1-\alpha) - (1-\alpha)\end{aligned}$$

Putting the above together and assuming that $1-\alpha \geq 1/2$ (which follows from $s := m-n \geq 0.53$ and $\alpha = 1-(1-1/2^m)^{2^n} \approx 1-e^{-2^{-s}}$) we obtain

$$\begin{aligned}&\alpha \cdot \mathsf{Adv}^{\text{prg-ri}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{A}) - (1-\alpha) \cdot \mathsf{Adv}^{\text{oprg}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{C}) \\ &\qquad = ((1-\alpha) - 2(1-\alpha)^2) \cdot \Pr[\neg a \mid y \notin I] + (1-\alpha)^2 - (1-\alpha) \\ &\qquad \leq (1-\alpha)^2 - (1-\alpha)\ .\end{aligned}$$

The equality follows after some algebra and the inequality uses the fact that when $x^2 - 2x \leq 0$ for $x \geq 1/2$. Rearranging we get

$$\mathsf{Adv}^{\text{prg-ri}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{A}) \leq \frac{1-\alpha}{\alpha} \cdot \mathsf{Adv}^{\text{oprg}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{C}) - (1-\alpha)\ ,$$

and overall we conclude that

$$\mathsf{Adv}^{\text{prg}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{A}) \leq \mathsf{Adv}^{\text{iu}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{B}) + \frac{1-\alpha}{\alpha} \cdot \mathsf{Adv}^{\text{oprg}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{C}) - (1-\alpha)\ .$$

□

When $\mathsf{Adv}^{\mathrm{oprg}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{C}) < (2 - \alpha - \mathsf{Adv}^{\mathrm{iu}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{B})) \cdot \alpha/(1-\alpha)$ (and is not too close to 1), we get an overall bound that is less than 1 and, hence, weak PRG security.

6.3 Attacks in the 2-BRO Model

In this section we informally describe attacks against the concatenation and cascade combiners. Furthermore, we discuss why in building secure iterative hash functions in k-BRO, combiners should be used on the level of compression functions, rather than simply combining iterative hash functions. This section serves as a warm-up exercise to provide a better understanding of the potential issues when combining BROs before we move on to our positive results. The success of attacks on the concatenation and cascade combiners depends on the sizes of the domain and co-domain of the available hash functions. As we indicate in Section 7.3 the xor combiner seems to be secure independently of its parameter regime.

For constants $\alpha > 0$ and β we call a hash function $\mathsf{H} \in \mathrm{Fun}[n, \alpha \cdot n + \beta]$ compressing if $\alpha < 1$, approximately length preserving if $\alpha = 1$, and expanding if $\alpha > 1$. Furthermore, over a random choice of H, any point y in the co-domain of H will be in $\mathsf{img}(\mathsf{H})$ with probability

$$\Pr[\exists x : \mathsf{H}(x) = y] = 1 - \Pr[\forall x : \mathsf{H}(x) \neq y] = 1 - (1 - 1/2^{\alpha n + \beta})^{2^n} \ .$$

For $\alpha < 1$ this probability is overwhelming, for $\alpha = 1$ it is constant, and for $\alpha > 1$ it is negligible.

6.3.1 Concatenation

Consider two hash functions $\mathsf{H}_1 \in \mathrm{Fun}[n, \alpha_1 n + \beta_1]$ and $\mathsf{H}_2 \in \mathrm{Fun}[n, \alpha_2 n + \beta_2]$ used in the concatenation combiner. The combined function is in $\mathrm{Fun}[n, (\alpha_1 + \alpha_2)n + (\beta_1 + \beta_2)]$.

An adversary can attempt to invert $y_1^* \| y_2^* = \mathsf{H}_1(x) \| \mathsf{H}_2(x)$ by using the backdoor oracle BD_1 to compute preimages for y_1^* under H_1 and the backdoor oracle BD_2 to compute preimages for y_2^* under H_2. The two sets of preimages obtained must have a non-empty intersection in order to find a valid preimage for the combined function. When either of the hash functions, say H_1, is expanding or approximately length preserving, the corresponding set of preimages of y_1^* will be of polynomial size. Hence, a preimage under the concatenation combiner can be found with good probability after polynomially many inversion queries to figure out which one of y_1^*'s preimages is also a preimage for y_2^* under H_2. To attack PRG security note that whenever not even weak OW security is satisfied, weak PRG security is not, either. Since one can check if a value is pseudorandom or not by inverting it and checking whether it is a valid preimage. However, a slightly expanding function (say by 1 bit), built by concatenating two compressing functions, can be weakly OW and also weakly PRG secure. As for collision resistance, note that collisions for H_1 or H_2 do not necessarily lead to collisions for the combined function. Indeed, we are not able to give attacks for any stretch pattern here.

6.3.2 Cascade

Consider two hash functions $\mathsf{H}_1 \in \mathrm{Fun}[n, \alpha_1 n + \beta_1]$ and $\mathsf{H}_2 \in \mathrm{Fun}[\alpha_1 n + \beta_1, \alpha_2 n + \beta_2]$ used in the cascade combiner. The combined function is in $\mathrm{Fun}[n, \alpha_2 n + \beta_2]$.

When H_2 is somewhat expanding or approximately length preserving (that is, $\alpha_2 \geq \alpha_1$) an adversary can attempt to invert the challenge $y^* := \mathsf{H}_2(\mathsf{H}_1(x^*))$ under H_2 and with non-negligible probability obtain a point z such that $z = z^*$ where $z^* := \mathsf{H}_1(x^*)$. The adversary can then try to invert z^* under H_1 to obtain a preimage. Note that this last step will succeed if the adversary is lucky and gets $z = z^*$ in the first step. Otherwise, if H_1 is either approximately length preserving or somewhat compressing ($\alpha_1 \leq 1$) any obtained z will invert with good probability. In case H_1 is expanding, the adversary can proceed to the next z (polynomially many times, since $\alpha_2 \geq \alpha_1$) and try to invert it under H_1. This leaves the composition of a somewhat expanding H_1 with a somewhat compressing H_2 (the expand-then-compress construction) as the only option which can offer strong one-way security.

For PRG security, as seen above, if the outer function H_2 is not sufficiently compressing, a random point will fail to invert under H_2 with non-negligible probability, whereas an honestly computed PRG value always will.

For collision resistance, any collision for H_1 is also a collision for the combiner. Collisions can be easily found for H_1 in the BRO model and, hence, H_1 must be injective even for weak collision resistance. The probability that a random $\mathsf{H}_1 \in \mathrm{Fun}[n, m]$ with $m > n$ is injective is

$$\Pr[\mathsf{H}_1 \text{ is injective}] = \frac{1}{(2^m)^{2^n}} \frac{2^m!}{(2^m - 2^n)!} \approx e^{-2^{2n-m}} ,$$

using Stirling's approximations. If $\alpha_1 > 2$ the exponent $2n - m$ tends to $-\infty$ as n grows, and the above probability is overwhelming. Collisions for H_2, on the other hand, do not necessarily give rise to those for the combined function. One can thus set $\alpha_1 = 3$, $\beta_1 = 0$, $\alpha_2 = 1$ and $\beta_2 = -1$ to get a combined function that compresses by 1 bit and is conjecturally collision resistant.

6.3.3 Iterative Hash Functions

Here we argue that the approach of directly combining two iterative hash functions in the 2-BRO model, i.e., even if the underlying compression functions are independent BROs, suffers from serious security deficits. More precisely, we show at the example of collision resistance that the concatenation of two MD-based hash functions in 2-BRO is generally not secure. The attack is based on the well-known multi-collision attack by Joux [Jou04]. We conjecture that the issue can be generalized for other combiners, other security properties, and other domain-extenders using variants of multi-collision attacks [NS06]. Although more invasive, a safer choice is, hence, to immunize the underlying compression functions by a suitable choice of combiner and then iterating it.

Formally, for hash functions $\mathsf{H}_1 \in \mathrm{Fun}[n_1 + m_1, m_1]$ and $\mathsf{H}_2 \in \mathrm{Fun}[n_2 + m_2, m_2]$, the concatenation of their iterative MD-based hash functions is defined as $\mathsf{MD}^{\mathsf{H}_1,\mathsf{H}_2}_{\|,\mathrm{iv}_1,\mathrm{iv}_2}$ as:

$$\mathsf{MD}^{\mathsf{H}_1,\mathsf{H}_2}_{\|,\mathrm{iv}_1,\mathrm{iv}_2}(x) := \mathsf{MD}^{\mathsf{H}_1}_{\mathrm{iv}_1}(x) \| \mathsf{MD}^{\mathsf{H}_2}_{\mathrm{iv}_2}(x) .$$

For the sake of simplicity, let us assume that $m_1 = m_2$ and $n_1 = n_2$. Let $m := m_1$ and $n := n$ and suppose that $\mathsf{MD}^{\mathsf{H}_1,\mathsf{H}_2}_{\|,\mathrm{iv}_1,\mathrm{iv}_2} \in \mathrm{Fun}[\ell \cdot n, 2m]$ and no padding function is used. For more details on the MD-construction we refer to Section 4.2.1.

A 2^r-multi-collision for a hash function H is a set C of size 2^r, where for all $x, x' \in C$ it holds that $\mathsf{H}(x) = \mathsf{H}(x')$. In 2004 Joux presented a simple 2^r-multi-collision attack on iterated MD-based hash functions with a time complexity of $\Theta(r \cdot 2^{m/2})$, which is much smaller than the time complexity of the birthday attack, i.e., $\Theta(2^{m\cdot(2^r-1)/2^r})$ [Jou04]. In other words, finding 2^r-multi-collisions costs only about r times as much as ordinary 2-collisions. In the same paper, Joux used the multi-collision attack to show that the security of the concatenation combiner is far from being optimal. Suppose $\mathsf{H} \in \mathrm{Fun}[n+m, m]$ is the compression function in use and we have a collision finding oracle Col on hand, which given a chaining value $y_i \in \{0,1\}^m$ finds two values $x_i, x'_i \in \{0,1\}^n$ such that $x_i \neq x'_i$ and $\mathsf{H}(y_i \| x_i) = \mathsf{H}(y_i \| x'_i)$. The oracle can then be used on $y_{i+1} := \mathsf{H}(y_i \| x_i)$ and so on to build many collisions, as depicted below.

$$\begin{tikzcd}
y_0 := \mathrm{iv} \arrow[r, bend left, "x_0"] \arrow[r, bend right, "x'_0"'] & y_1 \arrow[r, bend left, "x_1"] \arrow[r, bend right, "x'_1"'] & y_2 \arrow[r, bend left, "x_2"] \arrow[r, bend right, "x'_2"'] & y_3 & \cdots & y_{r-1} \arrow[r, bend left, "x_{r-1}"] \arrow[r, bend right, "x'_{r-1}"'] & y_r
\end{tikzcd}$$

The above multi-collision algorithm finds 2^r messages of length $n \cdot r$ which all map to the same image. Interestingly, these messages collide after any number of blocks and can be represented by a combination of $2r$ many blocks. Hence, the output of the collision finding algorithm Col has a compact representation of only $2rn$ bits. Furthermore Col can easily be implemented by one call to each of the backdoor oracles BD_1 and BD_2. The adversary obtains two sets for the compact collisions under $\mathsf{MD}_{\mathrm{iv}_1}^{\mathsf{H}_1}$ and $\mathsf{MD}_{\mathrm{iv}_2}^{\mathsf{H}_2}$, builds all possible 2^r collisions for both functions. As a collision for the combined function, the adversary returns any (x, x') that is common in both sets, which can be made very likely by choosing a large enough r. In fact, for $r \geq m_2/2$ we can conclude by the birthday paradox that among the collisions under $\mathsf{MD}_{\mathrm{iv}_1}^{\mathsf{H}_1}$, there will be one that leads to a collision under $\mathsf{MD}_{\mathrm{iv}_2}^{\mathsf{H}_2}$ with good probability. Attacking one-wayness, i.e., inverting some $y_1^* \| y_2^*$ is very similar to finding collisions, except that instead of using the multi-collision attack to find two sets of 2^r many collisions to two random images, they have to map to y_1^* and y_2^*, which is again easy which access to backdoor oracles.

6.4 Distributional Communication Complexity of Set-Disjointness and Set-Intersection

Two central problems in communication complexity that have received substantial attention are the *set-disjointness* and the *set-intersection* problems. In set-disjointness two parties, holding sets S and T respectively, compute the binary function $\mathrm{DISJ}(S, T) := (S \cap T = \emptyset)$. In set-intersection, their goal is to compute the relation $\mathrm{INT}(S, T) := S \cap T$; that is, the last message of the protocol should be equal to some element in the intersection. Note that set-disjointness can be seen as a decisional version of set-intersection and is, therefore, easier. As mentioned, we are interested in average-case lower bounds for these tasks and moreover, we focus on *product* distributions, where the sets are chosen independently.

Two main results to this end have been proven in previous work.[3] A classical result of Babai, Frankl, and Simon [BFS86] establishes an $\Omega(\sqrt{N})$ lower bound for set-disjointness where the input

[3] We note that most of the work on distributional communication complexity is driven by Yao's min-max lemma, which lower bounds worst-case communication complexity using distributional communication complexity for *some* (often non-uniform) distribution.

sets S and T are independent random subsets of $[N]$ of size exactly $\sqrt{N}$. This result, however, is restrictive for us as it roughly translates to regular functions in the cryptographic setting. Moreover, its proof uses intricate combinatorial arguments, which are somewhat hard to work with.

A second result considers the following distribution. Each element $x \in [N]$ is thrown into S independently with probability p. (And similarly for T with probability q.) We can view S as an N-bit string X where its i-th bit x_i is 1 iff $i \in S$. Thus the distribution can be viewed as N independent and identically distributed Bernoulli random variables $x_i \twoheadleftarrow \mathrm{Ber}(p)$ where $p := \Pr[x_i = 1]$. Thus the elements of the sets form a binomial distribution, and accordingly we write $S \twoheadleftarrow \mathrm{Bin}(N, p)$ and $T \twoheadleftarrow \mathrm{Bin}(N, q)$. We define $\mu(p, q)$ as the *product* of these two distributions. When $p = q = 1/2$ we get the product uniform distribution over the subsets of $[N] \times [N]$, but typically we will be looking at much smaller values of p and q, e.g., of order $1/\sqrt{N}$.

Using information-theoretic techniques [BYJKS02], the following lower bound for set-disjointness can be established. Our proof, which follows those in [MB12, GC13], is applicable to the case of $pq = 1/(\delta N)$ for $\delta > 1$, which was originally only claimed for $p = q = 1/\sqrt{N}$. However, in the full version of the paper [BFM18b, Appendix C.3], a new refined proof is included which extends the theorem to $\delta \geq 0.8$ (instead of $\delta \geq 1$). Roughly speaking, our proof of Theorem 6.10 proceeds along the following lines. We can lower bound the communication complexity of any protocol by the total information leaked by its transcripts about each coordinate (x_i, y_i). The latter can be lower bounded based on the statistical distance in protocol transcripts when $x_i = 1 \wedge y_i = 0$ and $x_i = 0 \wedge y_i = 1$. This step uses a number of information-theoretic inequalities, which we have included with proofs in Section 2.6. Finally, we show that a highly correct protocol can be used as a distinguisher with constant advantage: When $x_i = 0 \wedge y_i = 0$, for a constant fraction of the inputs the sets will be disjoint with high probability. However, when $x_i = 1 \wedge y_i = 1$ they necessarily intersect, but this condition happens for a constant fraction of the inputs. We get a lower bound by averaging over the i's.

Theorem 6.10 (Set-disjointness lower bound). *Let $N \in \mathbb{N}$ and assume $p, q \in (0, 1/2]$ with $p \leq q$ and $pq = 1/(\delta N)$ for some $\delta > 1$. Let $\mu(p, q)$ be the product binomial distribution over subsets $S, T \subseteq [N]$. Assume $\varepsilon < \frac{(\delta-1)p_0}{(4+\delta)}$ and let $p_0 := \Pr[\mathrm{DISJ}(S, T) = 0]$. Then*

$$\mathsf{D}^{\varepsilon}_{\mu(p,q)}(\mathrm{DISJ}) \geq \frac{Np}{8} \cdot \big((\delta - 1)p_0 - (4 + \delta)\varepsilon\big)^2 \ .$$

Proof. Let π be a deterministic protocol with an error probability of at most ε. Hence

$$\Pr_{(S,T) \twoheadleftarrow \mu}[\pi(X, Y) = \mathrm{DISJ}(S, T)] \geq 1 - \varepsilon \ ,$$

where $X = (x_0, \ldots, x_{N-1})$ and $Y = (y_0, \ldots, y_{N-1})$ (with $x_i, y_i \in \{0, 1\}$) are bit string representations of S and T, as described above. Let $\Pi(X, Y)$ denote a random variable for the transcripts of protocol π on inputs (X, Y) such that the corresponding sets are drawn from μ, i.e., $(S, T) \twoheadleftarrow \mu$. We have

$$\begin{aligned}\mathsf{D}^{\varepsilon}_{\mu(p,q)}(\mathrm{DISJ}) &\geq \log|\mathrm{supp}(\Pi(X, Y))| \\ &\geq \mathbf{H}(\Pi(X, Y)) \\ &= \mathbf{H}(\Pi(X, Y)) + \mathbf{H}(X, Y) - \mathbf{H}(X, Y, \Pi(X, Y))\end{aligned}$$

$$
\begin{aligned}
&= \mathbf{I}(X, Y; \Pi(X,Y)) \\
&= \mathbf{I}(x_0, \ldots, x_{N-1}, y_0, \ldots, y_{N-1}; \Pi(X,Y)) \\
&\geq \sum_{i=0}^{N-1} \mathbf{I}(x_i, y_i; \Pi(X,Y)) \ ,
\end{aligned}
$$

where the last inequality holds by Lemma 2.3, since x_i's and y_i's are independent. Let $\Pi^i_{a,b}(X,Y)$ be $\Pi(X,Y)$ conditioned on the i-th coordinates of X and Y being fixed to a and b respectively:

$$
\Pi^i_{a,b}(X,Y) := \Pi(X,Y) \mid x_i = a \wedge y_i = b \ .
$$

By Lemma 2.4

$$
\mathbf{I}(x_i, y_i; \Pi(X,Y)) \geq \mathbb{E}_{(a,b)}[\Delta^2_{\mathsf{Hel}}(\Pi^i_{a,b}(X,Y), \Pi(X,Y))] \ ,
$$

where $(a,b) \twoheadleftarrow \mathrm{Ber}(p) \times \mathrm{Ber}(q)$. Since $q \geq p$ we have that $q(1-p) \geq p(1-q)$. Since $q \leq 1/2$, we also have that $p(1-q) \geq p/2$. Thus

$$
\begin{aligned}
\mathbf{I}(x_i, y_i; \Pi(X,Y)) &\geq p(1-q) \cdot \Delta^2_{\mathsf{Hel}}(\Pi^i_{1,0}(X,Y), \Pi(X,Y)) + q(1-p) \cdot \Delta^2_{\mathsf{Hel}}(\Pi^i_{0,1}(X,Y), \Pi(X,Y)) \\
&\geq p(1-q) \cdot \left(\Delta^2_{\mathsf{Hel}}(\Pi^i_{1,0}(X,Y), \Pi(X,Y)) + \Delta^2_{\mathsf{Hel}}(\Pi^i_{0,1}(X,Y), \Pi(X,Y))\right) \\
&\geq \frac{p}{2} \cdot \left(\Delta^2_{\mathsf{Hel}}(\Pi^i_{1,0}(X,Y), \Pi(X,Y)) + \Delta^2_{\mathsf{Hel}}(\Pi^i_{0,1}(X,Y), \Pi(X,Y))\right) \\
&\geq \frac{p}{4} \cdot \left(\Delta_{\mathsf{Hel}}(\Pi^i_{1,0}(X,Y), \Pi(X,Y)) + \Delta_{\mathsf{Hel}}(\Pi^i_{0,1}(X,Y), \Pi(X,Y))\right)^2 \\
&\geq \frac{p}{4} \cdot \Delta^2_{\mathsf{Hel}}(\Pi^i_{1,0}(X,Y), \Pi^i_{0,1}(X,Y)) \ .
\end{aligned}
$$

The last inequality is by the triangle inequality for the metric Δ_{Hel}, and the penultimate inequality uses $x^2 + y^2 \geq (x+y)^2/2$. Hence

$$
\begin{aligned}
\mathsf{D}^{\varepsilon}_{\mu(p,q)}(\mathrm{DISJ}) &\geq N \cdot \mathbb{E}_i[\mathbf{I}(x_i, y_i; \Pi(X,Y))] \\
&\geq \frac{Np}{4} \cdot \mathbb{E}_i[\Delta^2_{\mathsf{Hel}}(\Pi^i_{1,0}(X,Y), \Pi^i_{0,1}(X,Y))] \\
&= \frac{Np}{4} \cdot \mathbb{E}_i[\Delta^2_{\mathsf{Hel}}(\Pi^i_{0,0}(X,Y), \Pi^i_{1,1}(X,Y))] \\
&\geq \frac{Np}{8} \cdot \mathbb{E}_i[\mathsf{SD}^2(\Pi^i_{0,0}(X,Y), \Pi^i_{1,1}(X,Y))] \\
&\geq \frac{Np}{8} \cdot \left(\mathbb{E}_i[\mathsf{SD}(\Pi^i_{0,0}(X,Y), \Pi^i_{1,1}(X,Y))]\right)^2
\end{aligned}
$$

where the equality uses the cut-and-paste lemma (Lemma 2.2). The penultimate inequality uses $\mathsf{SD}(A,B) \leq \sqrt{2}\Delta_{\mathsf{Hel}}(A,B)$ (discussed in Section 2.6), which implies $\Delta^2_{\mathsf{Hel}}(\Pi^i_{0,0}(X,Y), \Pi^i_{1,1}(X,Y)) \geq (1/2)\mathsf{SD}^2(\Pi^i_{0,0}(X,Y), \Pi^i_{1,1}(X,Y))$, and the last inequality is by Jensen's inequality. Thus it remains to lower bound $\mathsf{SD}(\Pi^i_{0,0}(X,Y), \Pi^i_{1,1}(X,Y))$.

Let $E := (\mathrm{DISJ}(S,T) = 1 \wedge x_i = y_i = 0)$. By correctness,

$$
\begin{aligned}
1 - \varepsilon &\leq \Pr[\Pi(X,Y) = \mathrm{DISJ}(S,T)] \\
&= \Pr[\Pi(X,Y) = \mathrm{DISJ}(S,T) \wedge E] + \Pr[\Pi(X,Y) = \mathrm{DISJ}(S,T) \wedge \neg E]
\end{aligned}
$$

$$\begin{aligned}
&\leq \Pr[\Pi(X,Y) = \mathrm{DISJ}(S,T) \wedge E] + 1 - \Pr[E] \\
&= \Pr[\Pi(X,Y) = 1 \wedge E] + 1 - \Pr[E] \ .
\end{aligned}$$

Hence

$$\Pr[\Pi(X,Y) = 1 \wedge E] \geq \Pr[E] - \varepsilon \ .$$

For every i we have

$$\begin{aligned}
\Pr[\Pi^i_{0,0}(X,Y) = 1] &= \Pr[\Pi(X,Y) = 1 \wedge x_i = y_i = 0] / \Pr[x_i = y_i = 0] \\
&\geq \Pr[\Pi(X,Y) = 1 \wedge E] / \Pr[x_i = y_i = 0] \\
&\geq (\Pr[E] - \varepsilon) / \Pr[x_i = y_i = 0] \\
&= \Pr[\mathrm{DISJ}(S,T) = 1 | x_i = y_i = 0] - \varepsilon / \Pr[x_i = y_i = 0] \\
&\geq \Pr[\mathrm{DISJ}(S,T) = 1] - \varepsilon / \Pr[x_i = y_i = 0] \\
&\geq \Pr[\mathrm{DISJ}(S,T) = 1] - \varepsilon / ((1-p)(1-q))
\end{aligned}$$

Complementing the output bit and setting $p_0 := \Pr[\mathrm{DISJ}(S,T) = 0]$ we get

$$\Pr[\Pi^i_{0,0}(X,Y) = 0] \leq p_0 + \varepsilon / ((1-p)(1-q)) \leq p_0 + 4\varepsilon \ .$$

Now we look at the case of $x_i = y_i = 1$. Here, it holds that $\mathrm{DISJ}(S,T) = 0$. A natural strategy at this point would be to argue that π outputs 0 with good probability for every i (since it is highly correct). This, however, does not work as $\Pr[x_i = y_i = 1]$ can be small. Despite this, $\Pr[\exists i : x_i = y_i = 1]$ is high, and the protocol π over the *random* choice of i should output 0 with good probability. We proceed as follows. Note that

$$1 - \varepsilon \leq \Pr[\Pi(X,Y) = 0 \wedge \mathrm{DISJ}(S,T) = 0] + (1 - p_0) \ .$$

Thus $\Pr[\Pi(X,Y) = 0 \wedge \mathrm{DISJ}(S,T) = 0] \geq p_0 - \varepsilon$. Now

$$\begin{aligned}
\mathbb{E}_i[\Pr[\Pi^i_{1,1}(X,Y) = 0]] &= \frac{1}{N} \cdot \sum \Pr[\Pi(X,Y) = 0 | x_i = y_i = 1] \\
&= \frac{1}{N} \cdot \sum \Pr[\Pi(X,Y) = 0 \wedge x_i = y_i = 1] / \Pr[x_i = y_i = 1] \\
&= \frac{1}{Npq} \cdot \sum \Pr[\Pi(X,Y) = 0 \wedge x_i = y_i = 1] \\
&\geq \frac{1}{Npq} \cdot \Pr[\Pi(X,Y) = 0 \wedge \exists i : x_i = y_i = 1] \\
&= \frac{1}{Npq} \cdot \Pr[\Pi(X,Y) = 0 \wedge \mathrm{DISJ}(S,T) = 0] \\
&\geq (p_0 - \varepsilon)/(Npq)
\end{aligned}$$

Putting the above two bounds together we get

$$\mathbb{E}_i[\mathsf{SD}(\Pi^i_{0,0}(X,Y), \Pi^i_{1,1}(X,Y))] \geq (p_0 - \varepsilon)/(Npq) - p_0 - 4\varepsilon \ .$$

Since $pq = 1/(\delta N)$, we get a lower bound $(\delta - 1)p_0 - (4 + \delta)\varepsilon$, which is positive for

$$\varepsilon < \frac{(\delta - 1)p_0}{(4 + \delta)} \ .$$

Consequently, this proof does not extend to large error. Overall for ε satisfying the above we get

$$\mathsf{D}^{\varepsilon}_{\mu(p,q)}(\mathrm{DISJ}) \geq \frac{Np}{8} \cdot ((\delta - 1)p_0 - (4 + \delta)\varepsilon)^2 \ .$$

□

Next we also prove a communication complexity lower bound for the set-intersection problem over Bernoulli sets for which set-disjointness can be *easy.* Although the overall proof structure will be similar to that of set-disjointness, i.e., Theorem 6.10, our strategy differs in a number of places, when we turn to lower bounding $\mathsf{SD}(\Pi^i_{0,0}(X,Y), \Pi^i_{1,1}(X,Y))$. We use the fact that a candidate element can be checked to belong to the intersection (whereas a decision bit for disjointness cannot be checked for correctness). This ensures that the protocol error is one-sided, and allows us to remove the requirement of ε being sufficiently small. Additionally, we will bound the probability that the protocol outputs a *random element* in the intersection. This leads to a distinguisher that succeeds with smaller advantage, but it overall will lead to a non-trivial bound. We state and prove the formal result next. As above we leave the Bernoulli parameters free so as to be able to compute a feasible region where the lower bound will be non-trivial.

Theorem 6.11 (Set-intersection lower bound)**.** *Let $N \in \mathbb{N}$ and assume $p, q \in (0, 1/2]$ with $p \leq q$. Let $\mu(p,q)$ be the product binomial distribution over subsets $S, T \subseteq [N]$. Let ε be the protocol error and set $p_0 :=$ Pr$[\mathrm{DISJ}(S,T) = 0]$. If $\varepsilon \leq p_0$ then*

$$\mathsf{D}^{\varepsilon}_{\mu(p,q)}(\mathrm{INT}) \geq \frac{Np}{8} \cdot \left(\frac{p_0 - \varepsilon}{Npq}\right)^2 \ .$$

For sufficiently large N we have $p_0 = 1 - (1 - pq)^N \approx 1 - e^{-Npq}$. If $pq \gg 1/N$ we get that $p_0 \approx 1$ (the sets intersect with overwhelming probability) and for the theorem we would need that $\varepsilon \leq 1$.

Proof. Let π be a deterministic protocol with error at most ε, i.e.,

$$\Pr_{(S,T) \twoheadleftarrow \mu}[\pi(X,Y) \in \mathrm{INT}(S,T)] \geq 1 - \varepsilon \ ,$$

where $X = (x_0, \ldots, x_{N-1})$ and $Y = (y_0, \ldots, y_{N-1})$ are bit string representations of S and T as explained above. Let $\Pi(X,Y)$ denote a random variable for the transcripts of protocol π on inputs (X,Y) with corresponding sets $(S,T) \twoheadleftarrow \mu$. We have

$$\begin{aligned}
\mathsf{D}^{\varepsilon}_{\mu(p,q)}(\mathrm{INT}) &\geq \log|\mathrm{supp}(\Pi(X,Y))| \\
&\geq \mathbf{H}(\Pi(X,Y)) \\
&= \mathbf{H}(\Pi(X,Y)) + \mathbf{H}(X,Y) - \mathbf{H}(X,Y,\Pi(X,Y)) \\
&= \mathbf{I}((X,Y;\Pi(X,Y))
\end{aligned}$$

$$
\begin{aligned}
&= \mathbf{I}(x_0,\ldots,x_{N-1},y_0,\ldots,y_{N-1};\Pi(X,Y)) \\
&\geq \sum_{i=0}^{N-1} \mathbf{I}(x_i,y_i;\Pi(X,Y)) \ ,
\end{aligned}
$$

where the last inequality holds due to the independence of $x_0,\ldots,x_{N-1},y_0,\ldots,y_{N-1}$ (cf. Lemma 2.3 in Section 2.6). Let $\Pi^i_{a,b}(X,Y)$ be $\Pi(X,Y)$ conditioned on the i-th coordinates of X and Y being fixed to a and b respectively:

$$\Pi^i_{a,b}(X,Y) := \Pi(X,Y) \mid x_i = a \wedge y_i = b \ .$$

By Lemma 2.4 we know

$$\mathbf{I}(x_i,y_i;\Pi(X,Y)) \geq \mathbb{E}_{(a,b)}[\Delta^2_{\mathsf{Hel}}(\Pi^i_{a,b}(X,Y),\Pi(X,Y))] \ ,$$

where $(a,b) \twoheadleftarrow \mathrm{Ber}(p) \times \mathrm{Ber}(q)$ and Δ_{Hel} is the Hellinger distance. Since $q \geq p$ we have that $q(1-p) \geq p(1-q)$ and since $q \leq 1/2$, we also have that $p(1-q) \geq p/2$. Thus

$$
\begin{aligned}
\mathbf{I}(x_i,y_i;\Pi(X,Y)) &\geq p(1-q)\cdot\Delta^2_{\mathsf{Hel}}(\Pi^i_{1,0}(X,Y),\Pi(X,Y)) + q(1-p)\cdot\Delta^2_{\mathsf{Hel}}(\Pi^i_{0,1}(X,Y),\Pi(X,Y)) \\
&\geq p(1-q)\cdot\left(\Delta^2_{\mathsf{Hel}}(\Pi^i_{1,0}(X,Y),\Pi(X,Y)) + \Delta^2_{\mathsf{Hel}}(\Pi^i_{0,1}(X,Y),\Pi(X,Y))\right) \\
&\geq \frac{p}{2}\cdot\left(\Delta^2_{\mathsf{Hel}}(\Pi^i_{1,0}(X,Y),\Pi(X,Y)) + \Delta^2_{\mathsf{Hel}}(\Pi^i_{0,1}(X,Y),\Pi(X,Y))\right) \\
&\geq \frac{p}{4}\cdot\left(\Delta_{\mathsf{Hel}}(\Pi^i_{1,0}(X,Y),\Pi(X,Y)) + \Delta_{\mathsf{Hel}}(\Pi^i_{0,1}(X,Y),\Pi(X,Y))\right)^2 \\
&\geq \frac{p}{4}\cdot\Delta^2_{\mathsf{Hel}}(\Pi^i_{1,0}(X,Y),\Pi^i_{0,1}(X,Y)) \ .
\end{aligned}
$$

The last inequality is by the triangle inequality for the metric Δ_{Hel}, and the penultimate inequality uses $x^2+y^2 \geq (x+y)^2/2$. Hence

$$
\begin{aligned}
\mathsf{D}^{\varepsilon}_{\mu(p,q)}(\mathrm{INT}) &\geq N\cdot\mathbb{E}_i[\mathbf{I}(x_i,y_i;\Pi(X,Y))] \\
&\geq \frac{Np}{4}\cdot\mathbb{E}_i[\Delta^2_{\mathsf{Hel}}(\Pi^i_{1,0}(X,Y),\Pi^i_{0,1}(X,Y))] \\
&= \frac{Np}{4}\cdot\mathbb{E}_i[\Delta^2_{\mathsf{Hel}}(\Pi^i_{0,0}(X,Y),\Pi^i_{1,1}(X,Y))] \\
&\geq \frac{Np}{8}\cdot\mathbb{E}_i[\mathsf{SD}^2(\Pi^i_{0,0}(X,Y),\Pi^i_{1,1}(X,Y))] \\
&\geq \frac{Np}{8}\cdot\left(\mathbb{E}_i[\mathsf{SD}(\Pi^i_{0,0}(X,Y),\Pi^i_{1,1}(X,Y))]\right)^2 \ ,
\end{aligned}
$$

where the third inequality uses Lemma 2.2 which states that for any deterministic protocol π we have $\Delta^2_{\mathsf{Hel}}(\Pi^i_{1,0}(X,Y),\Pi^i_{0,1}(X,Y)) = \Delta^2_{\mathsf{Hel}}(\Pi^i_{0,0}(X,Y),\Pi^i_{1,1}(X,Y))$. The penultimate inequality uses $\Delta^2_{\mathsf{Hel}}(\Pi^i_{0,0}(X,Y),\Pi^i_{1,1}(X,Y)) \geq (1/2)\mathsf{SD}^2(\Pi^i_{0,0}(X,Y),\Pi^i_{1,1}(X,Y))$, and the last inequality is by Jensen's inequality. Thus it remains to lower bound $\mathsf{SD}(\Pi^i_{0,0}(X,Y),\Pi^i_{1,1}(X,Y))$. For every i we have

$$\Pr[\Pi^i_{0,0}(X,Y) = i] = 0 \ .$$

This is because we have conditioned on $x_i = y_i = 0$ and the two parties can actually check whether or not i belongs to their sets.

Now we look at $x_i = y_i = 1$. We show that the protocol over the *random* choice of i should output i with the expected probability, that is, $1/|S \cap T|$ since the common element can intuitively be any element in the intersection. Note that the expected size of the intersection is

$$\mathbb{E}[|S \cap T|] = \mathbb{E}[\sum_{i=0}^{N-1} x_i \cdot y_i] = \sum_{i=0}^{N-1} \mathbb{E}[x_i \cdot y_i] = Npq \ ,$$

where we have used the linearity of expectation and independence of x_i and y_i. We proceed as follows.

$$\begin{aligned}
\mathbb{E}_i[\Pr[\Pi^i_{1,1}(X,Y) = i]] &= \frac{1}{N} \sum_i \Pr[\Pi(X,Y) = i | x_i = y_i = 1] \\
&= \frac{1}{Npq} \sum_i \Pr[\Pi(X,Y) = i \wedge x_i = y_i = 1] \\
&= \frac{1}{Npq} \sum_i \sum_{(x,y):\ x_i = y_i = 1 \wedge \pi(x,y) = i} \Pr[(X,Y) = (x,y)] \\
&= \frac{1}{Npq} \sum_{(x,y):\ \pi(x,y) \text{ correct and } x \cap y \neq \emptyset} \Pr[(X,Y) = (x,y)] \\
&= \frac{1}{Npq} \Big(\sum_{(x,y)} \Pr[(x,y)] - \sum_{x \cap y = \emptyset} \Pr[(x,y)] - \sum_{\pi(x,y) \text{ fails}} \Pr[(x,y)] \Big) \\
&\geq \frac{1 - \Pr[\mathrm{DISJ}(S,T) = 1] - \varepsilon}{Npq} = \frac{p_0 - \varepsilon}{Npq} \ .
\end{aligned}$$

Thus we get that

$$\mathbb{E}_i[\mathsf{SD}(\Pi^i_{0,0}(X,Y), \Pi^i_{1,1}(X,Y))] \geq (p_0 - \varepsilon)/(Npq) \ ,$$

and overall we obtain the claimed bound of

$$\mathsf{D}^{\varepsilon}_{\mu(p,q)}(\mathrm{INT}) \geq \frac{Np}{8} \cdot \Big(\frac{p_0 - \varepsilon}{Npq}\Big)^2 \ .$$

□

Letting $p = 1/N^{\alpha}$ and $q = 1/N^{\beta}$ with $\alpha \geq \beta$ (since we assumed $p \leq q$), for a non-trivial lower bound—that is an exponentially large right-hand side in the displayed equation above—we would need to have that $\alpha + 2\beta > 1$. We also require that $1 - \alpha - \beta > 0$ so that the expected intersection size Npq is exponentially large, in which case $p_0 \approx 1$ and the set-disjointness problem is *easy.* These inequalities lead to the feasibility region shown in Figure 6.3. We have included the symmetric region for $\alpha \leq \beta$.

6.4.1 Multi-Set Variants

In this chapter, besides relying on set-disjointness and set-intersection problems, we also need the following *multi-set* extensions of them. These problems are additionally parameterized by the

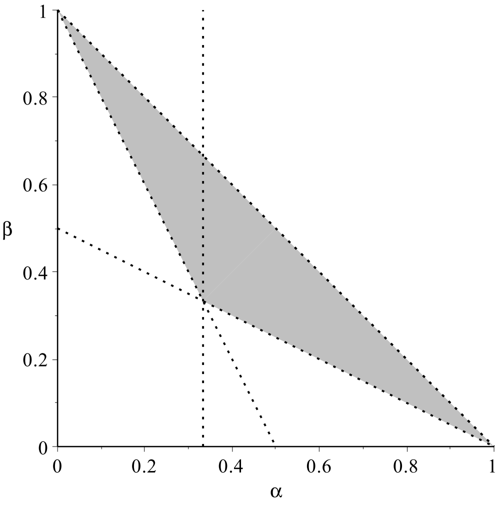


Figure 6.3: The gray region describes parameters where set-intersection is hard with $p = 1/N^{\alpha}$ and $q = 1/N^{\beta}$.

number of input sets. Here Alice holds M_1 sets $S_i \twoheadleftarrow \mathrm{Bin}(N, p)$ for $i \in [M_1]$ and Bob holds M_2 sets $T_j \twoheadleftarrow \mathrm{Bin}(N, q)$ for $j \in [M_2]$. Their goal is to solve the following problems.

- Find (i, x) such that $x \in S_i \cap T_i$, or return $\perp$ if all the intersections are empty. We call this the (M_1, M_2)-INT problem, a natural multi-set version of INT. A decisional variant would ask for an index i and a decision bit indicating if $S_i \cap T_i = \emptyset$. When $M_1 = M_2 = 1$, these problems are the usual INT and DISJ problems.

- Find (i, j, x, x') with $x \neq x'$ such that $x, x' \in S_i \cap T_j$, or return $\perp$ if no such tuple exists. We call this the (M_1, M_2)-2INT problem. When $M_1 = M_2 = 1$ this problem is at least as hard as the INT problem since finding two distinct elements in the intersection is harder than finding one element.

The INT problem is a harder task than the (M_1, M_2)-2INT problem (which we can also call the multi-set double-intersection problem). One can solve the (M_1, M_2)-2INT problem using a protocol for INT as follows. Alice chooses a random point x in one of its sets S_i and sends it to Bob. Bob will then search through his sets to find a set T_j such that $x \in T_j$. With high probability such a set exists if the number of sets and/or the probability parameters are large enough. Alice and Bob will then run the protocol for INT on sets S_i and T_j to find an $x' \in S_i \cap T_j$. This element will be different from x with good probability (again under appropriate choices of parameters). Indeed, this is simply the communication complexity way of saying "collision resistance implies one-wayness." However, we are interested in a reduction in the converse direction, as we already have proven lower bounds for INT. This seems hard as from a cryptographic point of view, as a classical impossibility result by Simon [Sim98] shows that collision resistance cannot be based on one-way functions (or even permutations) in a black-box way. Despite this, it is conceivable that direct information-theoretic analyses (similar to those for set-disjointness and set-intersection) can lead to non-trivial lower bounds. We leave proving hardness for this “collision-resistance” analogue of set-intersection as an interesting open problem for future work.

6.5 The Concatenation Combiner in the 2-BRO Model

In this section we study the security of the concatenation combiner $\mathsf{C}_{\|}^{\mathsf{H}_1,\mathsf{H}_2}(x) := \mathsf{H}_1(x)\|\mathsf{H}_2(x)$ in the 2-BRO model, where $\mathsf{H}_1 \in \text{Fun}[n, n+s_1]$ and $\mathsf{H}_2 \in \text{Fun}[n, n+s_2]$. We will prove one-way security, pseudorandomness, and collision resistance for this construction. Our results will rely on the hardness of set-intersection and set-disjointness for the first two properties, and the presumed hardness of finding two elements in the intersection given multiple instances.

6.5.1 One-Way Security

In Section 6.3.1 we showed that when H_1 or H_2 is approximately length preserving or somewhat expanding, the concatenation combiner is not (strongly) one-way in the 2-BRO model. In both cases, preimage sets will be only polynomially large and can be efficiently communicated. Accordingly, only when both hash functions are (somewhat) compressing we can achieve one-way security.

To this end we first give a reduction from random preimage resistance (rPre, as defined in Figure 6.2) to set-intersection. By Lemma 6.8 we know that any (weak) rPre-secure function is also a (weak) OW-secure function. In particular, for the highly compressing setting where $s_1, s_2 \leq -n/2-4$ we show *strong* one-way security. For settings where the parameters only enable weak security according to the set-intersection theorem, we can apply hardness amplification [Gol01] to get a strongly one-way function.

In our reductions to communication complexity protocols throughout this chapter, we make the following four simplifying assumptions without loosing of generality.

- The adversary is deterministic;
- It does not query H_i at all and instead computes hash values via the BD_i oracles;
- It queries BD_i oracles only on functions that have 1-bit outputs; and
- It starts with a query to BD_1.

Recall from Section 2.5 that we denote the protocol correctness by $\mathsf{Adv}_{\mu}^{\mathrm{f}}(\pi)$, with f being a placeholder for the name of the task f and μ being the distribution of the inputs. Also, we defined $\mathsf{Q}(\mathcal{A})$ in Section 3.3 as the total number of queries made by $\mathcal{A}$ to its oracles. We are now ready to prove our first cryptographic hardness result.

Theorem 6.12. *Let* $\mathsf{H}_1 \in \text{Fun}[n, n+s_1]$ *and* $\mathsf{H}_2 \in \text{Fun}[n, n+s_2]$ *and* $\mathsf{C}_{\|}^{\mathsf{H}_1,\mathsf{H}_2}(x) := \mathsf{H}_1(x)\|\mathsf{H}_2(x)$. *Then for any adversary* $\mathcal{A}$ *against the* rPre *security of* $\mathsf{C}_{\|}^{\mathsf{H}_1,\mathsf{H}_2}$ *in the* 2*-BRO model there is a* 2*-party protocol* π *against set-intersection with the distribution* $\mu := \mu(p, q)$ *where* $p := 1/2^{n+s_1}$ *and* $q := 1/2^{n+s_2}$ *and such that*

$$\mathsf{Adv}^{\mathrm{rpre}}_{\mathsf{C}_{\|}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathrm{int}}_{\mu}(\pi) \qquad \textit{and} \qquad \mathsf{D}_{\mu}(\pi) \leq \mathsf{Q}(\mathcal{A}) + 3n + s_1 + s_2 \ .$$

Proof. Let $\mathcal{A}$ be an adversary against the rPre security of $\mathsf{C}_{\|}^{\mathsf{H}_1,\mathsf{H}_2}$ in the 2-BRO model for H_1 and H_2. Adversary $\mathcal{A}$ is given a random point $y := y_1 \| y_2 \in \{0,1\}^{2n+s_1+s_2}$ and needs to either find an x such that $\mathsf{H}_1(x) \| \mathsf{H}_2(x) = y_1 \| y_2$ or say that no such x exists. Let us define two preimage sets

$$S_1 := \mathsf{H}_1^-(y_1) \qquad \text{and} \qquad S_2 := \mathsf{H}_2^-(y_2) \ .$$

Hence, $\mathcal{A}$ outputs an $x \in S_1 \cap S_2$ as long as $S_1 \cap S_2 \neq \emptyset$. We note that these sets are binomial. Indeed, for each x we have that $\Pr_{\mathsf{H}_1}[x \in S_1] = 1/2^{n+s_1}$ and $\Pr_{\mathsf{H}_2}[x \in S_2] = 1/2^{n+s_2}$, and these events are independent for different values of x.

We use $\mathcal{A}$ to build a 2-party protocol π for set-intersection over a product binomial distribution $\mu := \mu(p,q)$ with $p := 1/2^{n+s_1}$ and $q := 1/2^{n+s_2}$ as follows. Alice holds a set $S_1 \subseteq \{0,1\}^n$ and Bob holds a set $S_2 \subseteq \{0,1\}^n$ distributed according to μ. Alice (resp., Bob) samples hash function H_1 (resp., H_2) as follows. Alice picks a random $y_1 \in \{0,1\}^{n+s_1}$ and Bob picks a random $y_2 \in \{0,1\}^{n+s_2}$. Alice defines H_1 to map all points in S_1 to y_1. She maps remaining points $x \in \{0,1\}^n \setminus S_1$ to random points in $\{0,1\}^{n+s_1} \setminus \{y_1\}$. Similarly Bob defines H_2 to map all points $x \in S_2$ to y_2 and $x \in \{0,1\}^n \setminus S_1$ to random points in $\{0,1\}^{n+s_2} \setminus \{y_2\}$. As a result, Alice knows the full function table of H_1 and similarly Bob knows the full function table of H_2.

Alice and Bob now run two copies of $\mathcal{A}$ in tandem as follows, where the state values st_A and st_B are initially set to $y_1 \| y_2$ (with only $2n + s_1 + s_2$ bits of communication).

Alice: She resumes/starts $\mathcal{A}(st_A)$. She terminates if she receives a final guess x from Bob. She answers all pending BD_2 queries—there are none to start with—using the values just received from Bob. She answers all BD_1 queries using the function table of H_1 until $\mathcal{A}$ queries BD_2 or terminates. If $\mathcal{A}$ terminates with a final guess x, she forwards x to Bob and terminates. Else she saves the current state st_A of $\mathcal{A}$ locally and forwards all BD_1 answers that she has provided to $\mathcal{A}$ since the last resumption to Bob. She hands the execution over to Bob.

Bob: He resumes $\mathcal{A}(st_B)$. He terminates if he receives a final guess x from Alice. He answers all pending BD_1 queries using the values received from Alice. He answers all BD_2 queries using the function table of H_2 until $\mathcal{A}$ queries BD_1 or terminates. If $\mathcal{A}$ terminates with a final guess x, he forwards x to Alice and terminates. Else he saves the current state st_B of $\mathcal{A}$ locally and forwards all BD_2 answers that he has provided to $\mathcal{A}$ since the last resumption to Alice. He hands the execution over to Alice.

We claim that Alice and Bob run $\mathcal{A}$ in an environment that is identical to the rPre game in the 2-BRO model. First, we observe that the hash functions H_1 and H_2 sampled by Alice and Bob implement two independent random oracles. The fact that H_1 and H_2 are independent follows immediately from the simulation and the independence of S_1 and S_2. To see that H_1 is a random oracle we need to show that the probability that for any x and y we get $\mathsf{H}_1(x) = y$ is $1/2^{n+s_1}$, where the probability is over the coin tosses of Alice. Moreover, the probability should be the same even when conditioned on the rest of H_1 on points excluding x. To see this, let $R := \mathsf{H}_1\big(\{0,1\}^n \setminus \{x\}\big)$ be the rest of H_1. Then we can write

$$\begin{aligned}\Pr[\mathsf{H}_1(x) = y \mid R] = {} & \Pr[\mathsf{H}_1(x) = y \mid x \in S_1 \wedge y \neq y_1 \wedge R] \cdot \Pr[x \in S_1 \wedge y \neq y_1 \mid R] \\ & + \Pr[\mathsf{H}_1(x) = y \mid x \in S_1 \wedge y = y_1 \wedge R] \cdot \Pr[x \in S_1 \wedge y = y_1 \mid R]\end{aligned}$$

$$
\begin{aligned}
&+ \Pr\left[\mathsf{H}_1(x) = y \mid x \notin S_1 \wedge y \neq y_1 \wedge R\right] \cdot \Pr[x \notin S_1 \wedge y \neq y_1 \mid R] \\
&+ \Pr\left[\mathsf{H}_1(x) = y \mid x \notin S_1 \wedge y = y_1 \wedge R\right] \cdot \Pr[x \notin S_1 \wedge y = y_1 \mid R] \\
= {}& 0 \cdot \Pr[x \in S_1 \wedge y \neq y_1 \mid R] + 1 \cdot \Pr[x \in S_1 \wedge y = y_1 \mid R] \\
&+ \frac{1}{2^{n+s_1} - 1} \cdot \Pr[x \notin S_1 \wedge y \neq y_1 \mid R] + 0 \cdot \Pr[x \notin S_1 \wedge y = y_1 \mid R] \\
= {}& \frac{p}{2^{n+s_1}} + \frac{1}{2^{n+s_1} - 1}\left(1 - \frac{1}{2^{n+s_1}}\right)(1-p) \\
= {}& \frac{1}{2^{n+s_1}}
\end{aligned}
$$

The argument for H_2 is analogous. We also need to show that the challenge value $y_1 \| y_2$ as sampled above is uniformly chosen in the *co-domain* of the combiner. To see this note that y_1 is randomly chosen in the co-domain of H_1. Further the distribution of the preimage set S_1 induced by the Bernoulli distribution with parameter p is exactly that of preimages of a point in the co-domain (and, we emphasize, not in the image) of H_1. Similarly y_2 is chosen uniformly in the co-domain of H_2. Since the co-domain of the combiner is the concatenation of the co-domains of the two hash functions, we get that $y_1 \| y_2$ is uniformly distributed.

Thus Alice and Bob faithfully run $\mathcal{A}$ in the environment that it expects by answering its backdoor queries using their knowledge of the full tables of the two functions. Whenever $\mathcal{A}$ succeeds in breaking the rPre security of $\mathsf{C}_{\|}^{\mathsf{H}_1,\mathsf{H}_2}$, the protocol above computes an $x \in S_1 \cap S_2$ or says that no such x exists. In either case, the protocol solves the set-intersection problem. Thus the correctness of this protocol is at least the advantage of the adversary $\mathcal{A}$.

This execution of $\mathcal{A}$ by Alice and Bob ensures that oracle *queries* do not affect the communication cost of Alice and Bob. It is only their answers (plus the final x) that affects the communication cost, since the queried functions f are locally computed and only their answers are communicated. If $\mathcal{A}$ makes $\mathsf{Q}(\mathcal{A})$ queries to BD_1 and BD_2 in total and each query has a 1-bit output, the total communication complexity of the protocol is $\mathsf{Q}(\mathcal{A})$ plus those bits needed to communicate y_1 and y_2 and the final guess x. □

We now check that the parameters for hash functions can be set such that their concatenation is a one-way function.

Corollary 6.13. *For* $\mathsf{H}_1, \mathsf{H}_2 \in \mathrm{Fun}[n, (1-\sigma)n/2]$ *with* $0 < \sigma < 1/3$ *the concatenation combiner is a strongly one-way compressing function in* $\mathrm{Fun}[n, (1-\sigma)n]$.

Proof. The feasible region in Figure 6.3 for $\alpha = \beta$ consists of $1/3 < \alpha < 1/2$, i.e., where set-intersection is hard for protocols with arbitrary error. In our setting $\alpha = \beta = (1-\sigma)/2$, which means concatenation is strongly rPre secure when $0 < \sigma < 1/3$. Since the combined function is compressing (where $\gamma = 1/(1-\sigma) > 1$), the image-uniformity bound is negligible and also $\Pr[y \in \mathsf{img}(\mathsf{C}^{\mathsf{H}_i})]$ in Lemma 6.8 is overwhelming. Using these bounds and Lemma 6.8 we get that strong rPre security implies strong OW security. □

We conjecture that concatenation is strongly one-way even for $1/3 \leq \sigma < 1$. The intuition is that in the one-way game a point is “planted” in a large intersection, which seems hard to discover without essentially communicating the entire intersection. Tighter lower bounds for set-intersection can be used to establish this in the future.

6.5.2 PRG Security

We now consider the PRG security of the concatenation combiner. Our reduction in Theorem 6.12 from rPre to set-intersection can be easily adapted to the decisional setting. That is, we can show that a decisional variant of rPre can be reduced to the set-disjointness problem. The decisional variant of rPre asks the adversary to decide whether or not a random co-domain point $y_1 \| y_2$ has a preimage. This is exactly the oblivious PRG (oPRG) game that we defined in Figure 6.2. We get the following result.

Theorem 6.14. *Let* $\mathsf{H}_1 \in \mathrm{Fun}[n, n+s_1]$ *and* $\mathsf{H}_2 \in \mathrm{Fun}[n, n+s_2]$ *and* $\mathsf{C}_{\|}^{\mathsf{H}_1,\mathsf{H}_2}(x) := \mathsf{H}_1(x)\|\mathsf{H}_2(x)$*. Then for any adversary* $\mathcal{A}$ *against the oPRG security of* $\mathsf{C}_{\|}^{\mathsf{H}_1,\mathsf{H}_2}$ *in the* 2*-BRO model there is a* 2*-party protocol* π *against set-disjointness with the distribution* $\mu := \mu(p,q)$ *where* $p := 1/2^{n+s_1}$ *and* $q := 1/2^{n+s_2}$ *and such that*

$$\mathsf{Adv}^{\mathrm{oprg}}_{\mathsf{C}_{\|}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathrm{disj}}_{\mu}(\pi) \qquad \textit{and} \qquad \mathsf{D}_{\mu}(\pi) \leq \mathsf{Q}(\mathcal{A}) + 2n + s_1 + s_2 + 1 \ .$$

We next check if concrete parameters can be set to obtain an expanding PRG.

Corollary 6.15. *For* $\mathsf{H}_1 \in \mathrm{Fun}[n, n/2+1]$ *and* $\mathsf{H}_2 \in \mathrm{Fun}[n, n/2]$ *the concatenation combiner gives a weak* PRG *in* $\mathrm{Fun}[n, n+1]$.

Proof. The theorem gives a reduction to the set-disjointness problem with parameters $p = 1/2^{n/2+1}$ and $q = 1/2^{n/2}$. For large n we get, $\delta = 2$, $p_0 = 1 - e^{-1/2}$ and $(\delta-1)p_0/(4+\delta) < 0.0656$, which means we can set $\varepsilon = 0.065$. By set-disjointness lower bound, this means any adversary with advantage at least 0.935 must place at least $\mathcal{O}(2^{n/2})$ queries in total to its oracles.

By Lemma 6.1 we have that $\mathsf{Adv}^{\mathrm{iu}}_{\mathsf{C}^{\mathsf{H}_i}}(\mathcal{B}) \leq e^{-C} \cdot \big(C/(1-e^{-C}) - 1\big)$. In our case $C = 1/2 < 1$, and the right hand side above is upper bounded by 0.165. (We have removed the negligible terms and instead approximated the constants by slightly larger values.)

In Lemma 6.9 in order to meet the bound $\mathsf{Adv}^{\mathrm{oprg}}_{\mathsf{C}^{\mathsf{H}_i}}(\mathcal{C}) < (2 - \alpha - \mathsf{Adv}^{\mathrm{iu}}_{\mathsf{C}^{\mathsf{H}_i}}(\mathcal{B})) \cdot \alpha/(1-\alpha)$, we would need $0.935 \leq (2 - \alpha - 0.165) \cdot \alpha/(1-\alpha)$. After some algebra this gives $\alpha \geq 0.39343$. With $m = n + s$, we need to have $1 - e^{-2^{-s}} \geq 0.39343$, which means $s \leq 1.00018$. Thus we can set $s = 1$ (which also satisfies $s \geq 0.53$ as required in the lemma) and get a combiner that expands by one bit. □

We can obtain a strong PRG by amplification. However, we need an amplifier that woks on PRGs with (very) *small* stretch. Such a construction is given by Maurer and Tessaro [MT10]. In their so-called concatenate-and-extract (CaE) construction one sets

$$\mathsf{PRG}(r, x_1, \ldots, x_m) := r \| \mathsf{Ext}\big(r, \mathsf{C}_{\|}^{\mathsf{H}_1,\mathsf{H}_2}(x_1) \| \cdots \| \mathsf{C}_{\|}^{\mathsf{H}_1,\mathsf{H}_2}(x_m)\big) \ ,$$

where Ext is a sufficiently good randomness extractor, for instance a universal hash function. We refer to the original work for concrete parameters. It is safe to assume the extractor is backdoor-free, since it is an information-theoretic object and relatively easy to implement.

6.5.3 Collision-Resistance

The classical result of Simon [Sim98] shows that collision resistance relies on qualitatively stronger assumptions than one-way security. In the theorem below we prove collision resistance based on the hardness of the multi-set 2INT problem as defined in Section 6.4. As discussed in the final remark of that section, we do not expect that a reduction to the INT problem exists.

Theorem 6.16. *Let* $\mathsf{H}_1 \in \mathrm{Fun}[n, n+s_1]$ *and* $\mathsf{H}_2 \in \mathrm{Fun}[n, n+s_2]$ *and* $\mathsf{C}_{\|}^{\mathsf{H}_1,\mathsf{H}_2}(x) := \mathsf{H}_1(x)\|\mathsf{H}_2(x)$. *Then for any adversary* $\mathcal{A}$ *against the collision resistance of* $\mathsf{C}_{\|}^{\mathsf{H}_1,\mathsf{H}_2}$ *in the 2-BRO model there is a 2-party protocol* π' *against multi-set double-intersection problem* (M_1, M_2)-2INT *over* $\mu' := \mu'(p', q')$ *with* $p' := 2n \ln 2/2^{n+s_1}$ *and* $q' := 2n \ln 2/2^{n+s_2}$ *and where Alice holds* $M_1 := 2^{n+s_1}$ *sets and Bob holds* $M_2 := 2^{n+s_2}$ *sets such that*

$$\mathsf{Adv}^{\mathrm{cr}}_{\mathsf{C}_{\|}^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{A}) \le \mathsf{Adv}^{\text{mi-2int}}_{\mu'}(\pi') + 2 \cdot 2^{-n} \qquad \textit{and} \qquad \mathsf{D}_{\mu'}(\pi') \le \mathsf{Q}(\mathcal{A}) + 4n + s_1 + s_2 \ .$$

Proof. We follow an overall strategy that is similar to the one for the rPre reduction. For each $i \in \{0,1\}^{n+s_1}$, Alice sets $\mathsf{H}_1^-(i) := S_i$ and for each $j \in \{0,1\}^{n+s_2}$ Bob sets $\mathsf{H}_2^-(j) := T_j$ and they simulate the two hash functions. However, this leads to a problem: S_i's are not necessarily disjoint and, furthermore, their union does not necessarily cover the entire domain $\{0,1\}^n$. (The same is true for T_j's.) Hence, this way of simulation leads to some H_1 and H_2 that are not random oracles, even if defined on the whole domain. Put differently, the distributions of sets formed by hash preimages of co-domain points do not match independently chosen sets from a binomial distribution.

We handle this problem in two step. The first step is a direct reduction to a "partitioned" modification of the multi-set 2INT problem. In this partitioned problem Alice gets sets $S_i := \mathsf{H}_1^-(i)$ for $i \in \{0,1\}^{n+s_1}$ and a random oracle $\mathsf{H}_1 \in \mathrm{Fun}[n, n+s_1]$. Similarly, Bob gets sets $T_j := \mathsf{H}_2^-(j)$ for $j \in \{0,1\}^{n+s_2}$ and an independent random oracle $\mathsf{H}_2 \in \mathrm{Fun}[n, n+s_2]$. Their goal is to find a tuple (i, j, x, x') with $x \ne x'$ such that $x, x' \in S_i \cap T_j$. Thus these sets exactly correspond to hash preimages as needed in the reduction above, and a solution would translate to a collision for the combined hash function.

As the second and final step we show below (in Lemma 6.17) that the hardness of the (standard) multi-set double-intersection problem implies the hardness of the partitioned problem with an increase in the Bernoulli parameter with the same number of sets involved. □

Lemma 6.17 (Partitioned $\implies$ Independent)**.** *For any two-party protocol* π *against the partitioned multi-set* 2INT *problem there is a two-party protocol* π' *against multi-set* 2INT *problem such that*

$$\mathsf{Adv}^{\text{mi-2int}}_{\mu'}(\pi') \ge \mathsf{Adv}^{\text{part-2int}}_{\mu}(\pi) - 2 \cdot 2^{-n} \qquad \textit{and} \qquad \mathsf{D}_{\mu'}(\pi') \le \mathsf{D}_{\mu}(\pi) \ .$$

Here $\mu := \mu(p, q)$ *is the distribution induced by hash preimages in* $\{0,1\}^n$ *and* $\mu' := \mu'(p', q')$ *is a product Bernoulli with* $p' := 2n \ln 2 \cdot p$ *and* $q' := 2n \ln 2 \cdot q$.

Proof. To focus on the core ideas, we simplify and let $M_1 = M_2 = M = 2^{n+s}$ and $p = q = 1/2^{n+s}$. Suppose we have sets S_i and T_j for $i = 1, \ldots, M$ and $j = 1, \ldots, M$ as an instance for the multi-set 2INT. Let $p' = q' = 2n \ln 2 \cdot p$. Then

$$\Pr[\exists x \in \{0,1\}^n \ \forall i \in [M] : x \notin S_i] \le 2^n \Pr[\forall i \in [M] : x \notin S_i] \le 2^n (1 - p')^{1/p} \le 2^n e^{-2n \ln 2} = 2^{-n} \ ,$$

which is negligible. Thus with these parameters the sets S_i (and similarly T_j) will, with overwhelming probability, cover the full domain, i.e., $\bigcup_{i=1}^{M} S_i = \{0,1\}^n$.

```
ReDist($S_1, \ldots, S_M$)
for $x \in \{0,1\}^n$ do
    $A_x := \{i \in [M] : x \in S_i\}$
    $i_x \twoheadleftarrow A_x$
    for $j \in [M] \wedge j \neq i_x$ do
        $\widetilde{S}_j \leftarrow \widetilde{S}_j \setminus \{x\}$
return $(\widetilde{S}_1, \ldots, \widetilde{S}_M)$
```

Figure 6.4: Redistribution of elements to form a partition.

Our next step it to redistribute the elements among the sets so that they form partitions. We do this via the algorithm ReDist shown in Figure 6.4. ReDist iterates through elements x in the domain and leaves x in exactly one of the sets. Note that by the above covering property such a set always exits.

This procedure will be applied to S_i (resp., T_j) to produce non-overlapping sets $\tilde{S}_i$ (resp. $\tilde{T}_j$). Furthermore, we always have that $\tilde{S}_i \subseteq S_i$ and $\tilde{T}_i \subseteq T_i$, since elements are only deleted from the sets and never added to them. Thus $\tilde{S}_i \cap \tilde{T}_j \subseteq S_i \cap T_j$ as well, and this means that any solution with respect to the tweaked sets will also be a valid solution for the original (binomial) sets.

We still need to show that the distribution of the tweaked sets is identical to that given by hash preimages under a random oracle. Let $E_{x,i}$ be the event that $x \in \widetilde{S}_i$. Since the algorithm does not treat any of the i's in a special way, we claim that $\Pr[E_{x,i}]$ is independent of i. Indeed for any i, j we have

$$\Pr[E_{x,i}] = \Pr[x \in S_i] \Pr[i_x = i | x \in S_i] = \Pr[x \in S_j] \Pr[i_x = j | x \in S_j] = \Pr[E_{x,j}].$$

This is because $\Pr[x \in S_i] = \Pr[x \in S_j]$ and $\Pr[i_x = i | x \in S_i] = \Pr[i_x = j | x \in S_j]$, since S_i and S_j are independent sets with the same Bernoulli parameter. If we call the above common probability p_x, since x is guaranteed to belong to one of the M sets, we have that $\sum_{i \in [M]} p_x = 1$. Thus $p_x = 1/M = \Pr[\mathsf{H}_1(x) = i]$. Note that the algorithm assigns different values of x independently of all other values already assigned, we get that the event $\mathsf{H}_1(x) = i$ is independent for different x.

Finally, solutions with respect to the tweaked sets always exist when $s_1 + s_2 < 0$. This is because the problem is equivalent to finding collisions for a function $\mathsf{H}_1(x) \| \mathsf{H}_2(x)$ that is compressing, which necessarily exist. □

The birthday attack gives a $2^{\max(n+s_1, n+s_2)/2}$ upper bound on the security of the combined hash function. Balancing the digest lengths with $s_1 = s_2 = -n/2$, leads to a maximum collision security of at most $2^{n/4}$. Proving a lower bound, on the other hand, remains an interesting open problem. We formulate a conjecture towards proving this next.

Conjecture 6.1. *The multi-set double-intersection problem over binomial sets in a universe of size N with $p = q = 1/\sqrt{N}$ and $\sqrt{N}$ sets for each party has communication complexity*

$$\mathsf{D}^{\varepsilon}_{\mu(p,q)}((\sqrt{N}, \sqrt{N})\text{-2INT}) \geq \tilde{\Omega}(N^{1/4}),$$

for a sufficiently small protocol error ε and where $\tilde{\Omega}$ hides logarithmic factors.

We note that a lower bound for protocols with a sufficiently small error would be sufficient for feasibility results as collision resistance can also be amplified in a black-box way [CRS$^+$07].

6.6 The Cascade Combiner in the 2-BRO Model

We now look at the security of the cascade combiner $\mathsf{C}_\circ^{\mathsf{H}_1,\mathsf{H}_2}(x) := \mathsf{H}_2(\mathsf{H}_1(x))$ in the 2-BRO model, where $\mathsf{H}_1 \in \mathrm{Fun}[n, n+s_1]$ and $\mathsf{H}_2 \in \mathrm{Fun}[n+s_1, n+s_1+s_2]$. We will prove one-way security and pseudorandomness based on set-intersection and set-disjointness respectively, and collision resistance based on the hardness of the multi-set double-intersection problem given multiple instances for one party and a single set for the other.

6.6.1 One-Way Security

Here we give a reduction from the random preimage resistance (rPre) of the cascade combiner to hardness of set-intersection in an approach relatively similar to the one for concatenation combiner.

Theorem 6.18. *Let* $\mathsf{H}_1 \in \mathrm{Fun}[n, n+s_1]$ *and* $\mathsf{H}_2 \in \mathrm{Fun}[n+s_1, n+s_1+s_2]$ *and* $\mathsf{C}_\circ^{\mathsf{H}_1,\mathsf{H}_2}(x) := \mathsf{H}_2(\mathsf{H}_1(x))$. *Then for large enough n and any adversary $\mathcal{A}$ against the* rPre *security of* $\mathsf{C}_\circ^{\mathsf{H}_1,\mathsf{H}_2}$ *in the 2-BRO model there is a 2-party protocol π against set-intersection with $\mu := \mu(p,q)$ where $p := 1/2^{s_1}$ and $q := 1/2^{n+s_1+s_2}$ and such that*

$$\mathsf{Adv}^{\mathrm{rpre}}_{\mathsf{C}_\circ^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{A}) \leq \mathsf{Adv}^{\mathrm{int}}_{\mu}(\pi) + \sqrt{n}2^{-n/2}(1 + 2^{-s_2-s_1}) \qquad \textit{and} \qquad \mathsf{D}_\mu(\pi) \leq \mathsf{Q}(\mathcal{A}) + 3n + 2s_1 + s_2.$$

Proof. We follow a strategy similar to the reductions for the concatenation combiner in Section 6.5. Given a random $y^* \in \{0,1\}^{n+s_1+s_2}$ the task of the adversary $\mathcal{A}$ against the rPre security of $\mathsf{C}_\circ^{\mathsf{H}_1,\mathsf{H}_2}$ is to a find a z such that $\mathsf{C}_\circ^{\mathsf{H}_1,\mathsf{H}_2}(z) = y^*$. With such a z, one can then also compute $x := \mathsf{H}_1(z)$ and conclude that $x \in I \cap T$ where

$$I := \mathsf{H}_1(\{0,1\}^n) \qquad \text{and} \qquad T := \mathsf{H}_2^-(y^*)$$

with $I, T \subseteq \{0,1\}^{n+s_1}$. The set T is binomial with parameter $\Pr[y \in T] = 1/2^{n+s_1+s_2}$. Hence Bob can program a random oracle H_2 similar to the proof of Theorem 6.12 in a way that $\mathsf{H}_2^-(y^*) = T$ holds for a randomly chosen $y^* \in \{0,1\}^{n+s_1+s_2}$. However, although set I appears to be binomial,

$$\Pr[x \in I] = 1 - \Pr[\forall z : \mathsf{H}_1(z) \neq x] = 1 - (1 - 1/2^{n+s_1})^{2^n}$$

it is not, since these events are not independent for different values of x. In particular, observe that we always have $|I| \leq 2^n$ for a set of images I, whereas this is not always the case for a binomial set.

Our strategy to deal with this, and ultimately construct a protocol π for solving set-intersection, is to start with a binomial set S (Alice's input), and program H_1 on all $x \in \{0,1\}^n$ to values y that will be taken from S, but the values are also set to *collide* with the "right" probability. This will

```
HashSam($S$)
$X \leftarrow \emptyset$; $Y \leftarrow \emptyset$
for $i = 1, \ldots, 2^n$ do
    $x \twoheadleftarrow \{0,1\}^n \setminus X$; $X \leftarrow X \cup \{x\}$
    $b \twoheadleftarrow \mathrm{Ber}(|Y|/2^m)$
    if $b = 1$ then $y \twoheadleftarrow Y$
    if $b = 0 \wedge S = \emptyset$ then
        $y \twoheadleftarrow \{0,1\}^m \setminus Y$; $Y \twoheadleftarrow Y \cup \{y\}$
    if $b = 0 \wedge S \neq \emptyset$ then
        $y \twoheadleftarrow S$; $Y \leftarrow Y \cup \{y\}$; $S \leftarrow S \setminus \{y\}$
    $\mathsf{H}_1 \leftarrow \mathsf{H}_1 : [x \mapsto y]$
return $\mathsf{H}_1$
```

Figure 6.5: Hash sampler centered around a binomial set S.

ensure that the image of H_1 contains most of S and is also distributed as the image of a random oracle.

We proceed as follows. Initially the set of assigned domain points X and assigned co-domain points Y are empty. We then iterate through $x \in \{0,1\}^n$ in a random order. A bit b at each iteration decides if the hash value y for x should collide with a previously assigned value or not. If so, we sample y from the set of already assigned values Y. Otherwise, y should be a non-colliding value and we sample it from S if S is non-empty (and remove y from S), or otherwise we sample it outside the already assigned points Y. The pseudo-code for this algorithm, which we call HashSam, is shown in Figure 6.5.

Setting $m := n + s_1$, we now need to check that (1) the returned hash function H_1 is distributed as a random oracle in $\mathrm{Fun}[n, m]$ when the set S is binomial with parameter $p = 1/2^{s_1}$, and (2) if $x \in \mathsf{H}_1(\{0,1\}^n) \cap \mathsf{H}_2^-(y^*)$, then we also have that $x \in S \cap T$ with good probability.

We first prove (1). The intuition is that the algorithm treats all inputs and outputs in a uniform way, and hence no particular values are special. Formally, let x^* and y^* be any fixed values. We show that $\Pr[\mathsf{H}_1(x^*) = y^*] = 1/2^m$, even given the previously assigned values. We use a subscript i to denote the values of various variables in the i-th iteration. Looking at different execution branches of the algorithm we can calculate $\Pr[y_i = y^* | x_i = x^*, Y_i, X_i]$ as

$$\Pr[b_i = 1] \Pr[y^* \in Y_i] \frac{1}{|Y_i|} + \Pr[b_i = 0] \Big(\Pr[S_i = \emptyset] \Pr[y^* \notin Y_i] \frac{1}{2^m - |Y_i|} \\ + \Pr[S_i \neq \emptyset] \Pr[y^* \in S_i] \frac{1}{|S_i|} \Big) .$$

Letting $\theta_i := \Pr[S_i = \emptyset]$ we can simplify to

$$\frac{|Y_i|}{2^m} \frac{|Y_i|}{2^m} \frac{1}{|Y_i|} + \Big(1 - \frac{|Y_i|}{2^m}\Big) \Big(\theta_i \big(1 - \frac{|Y_i|}{2^m}\big) \frac{1}{2^m - |Y_i|} + (1 - \theta_i) \frac{|S_i|}{2^m} \frac{1}{|S_i|} \Big) = \frac{1}{2^m} .$$

Note we have used the fact that S_i is a binomial set in $\Pr[y^* \in S_i] = |S_i|/2^m$. Hence

$$\Pr[\mathsf{H}_1(x^*) = y^*|Y, X] = \sum_{i=1}^{2^n} \Pr[y_i = y^*|x_i = x^*, Y_i, X_i] \Pr[x_i = x^*] = \frac{1}{2^m} \ .$$

Therefore, the probability of sampling any given hash function is $(1/2^m)^{2^n}$, as required.

Let us now consider (2). When $I \subseteq S$, which is the case where images are never picked from outside of S (i.e., the second if-statement has not ever been true), any solution with respect to I is also one with respect to S, in other words, solutions are not lost. Hence we only look at the case $S \subseteq I$ and bound $|I \setminus S| = |I| - |S|$. This corresponds to the setting, where all elements of S are in I, and once S has become empty, potentially new elements are also added to I. Since $|I| \leq 2^n$ and $\mathbb{E}[|S|] = 2^{n+s_1}/2^{s_1} = 2^n$, we get that for any t

$$\Pr[|I| - |S| > t] \leq \Pr[2^n - |S| > t] = \Pr[\mathbb{E}[|S|] - |S| > t] \ .$$

Applying the Chernoff bounds we obtain

$$\Pr\left[\mathbb{E}[|S|] - |S| > t\mathbb{E}[|S|]\right] \leq e^{-t^2/(2+t)\mathbb{E}[|S|]} \ .$$

Setting $t := \sqrt{n/2^n}$, we get with overwhelming probability (of at least $1 - \sqrt{n} \cdot 2^{-n/2}$) that $|I \setminus S| \leq \sqrt{n}2^{n/2}$. Hence $T \cap (I \setminus S)$ will be non-empty with negligible probability $\sqrt{n}2^{-n/2-s_1-s_2}$, in which case if $x \in I \cap T \implies x \in S \cap T$.

For the communication complexity of π note that it is upper bounded by $Q(\mathcal{A})$ (similar to all reductions) plus $n + s_1 + s_2$ bits for communicating y^*, n bits as the output of $\mathcal{A}$ which potentially needs to reach Alice, such that she can compute and output its image, and $n + s_1$ bits to send (as the final message) the element in the intersection of two sets. □

For hash functions $\mathsf{H}_1 \in \text{Fun}[n, (2+\sigma)n]$ and $\mathsf{H}_2 \in \text{Fun}[(2+\sigma)n, (1+\sigma)n]$) we have a reduction to set-intersection with parameters $N = 2^{(2+\sigma)n}$, $p = 1/2^{(1+\sigma)n}$, and $q = 1/2^{(1+\sigma)n}$. Thus with notation as in the description of the feasible region in Figure 6.3 we have that $\alpha = \beta = (1+\sigma)/(2+\sigma)$. As in Corollary 6.13 we would need that the point (α, β) lies in the feasible region for $1/3 < (1+\sigma)/(2+\sigma) < 1/2$, which means $-1/2 < \sigma < 0$. Since the combined function is compressing (with $\gamma = 1/(1+\sigma) > 1$) and $p_\circ \approx 1 - e^{-2^{-\sigma \cdot n}}$ is negligible, the image uniformity bound is negligible and also $\Pr[y \in \mathsf{img}(\mathsf{C}^{\mathsf{H}_i})]$ in Lemma 6.8 is overwhelming. Hence, similarly to Corollary 6.13 we get strong OW security.

6.6.2 PRG and CR Security

Here we outline how to treat the PRG security and collision resistance of the cascade combiner. We omit the proofs as the techniques and proof structures are similar to our other results above.

In proving oblivious PRG security of the cascade construction, the reduction is identical to the one given for rPre security in Theorem 6.18, except that the underlying assumption is set-disjointness. Setting $s_1 = 2n$ (H_1 is length doubling) and $s_2 = -2n + 1$ (H_2 compresses by almost a factor of 3) leads to a reduction to an instance of set-intersection with parameters $N = 2^{3n}$, $p = 1/2^{2n}$, and

$q = 1/2^{n+1}$. In this case $\delta = 2$ and $p_0 = 1 - e^{-1/2}$. With these parameters we can carry out an analysis similar to Corollary 6.15: We set the error $\varepsilon = 0.065$ which is smaller than $(\delta - 1)p_0/(4 + \delta) < 0.0656$ as required in Theorem 6.10 for an exponential number of queries. The combined hash function maps n bits to $n + 1$ bits and hence $C = 1/2$. Furthermore, $p_\circ$ is negligible as a function from n bits to $3n$ bits is injective with overwhelming probability. Thus we can apply Lemma 6.9 with $s = 1$ as in Corollary 6.15 to get a weak PRG.

We can treat the collision resistance of cascade similarly. The difference is that in the reduction Alice will use the $\mathsf{HashSam}$ algorithm in Figure 6.5 to adapt a (single) binomial set S that she holds to a hash image set I. On the other hand, Bob uses the ReDist algorithm in Figure 6.4 to redistribute elements in multiple binomial sets that he holds so that they form a partition of the entire domain of H_2. The rest of the proof proceeds similarly to Lemma 6.17. For setting parameters, observe that any collision for H_1 is necessarily a collision for $\mathsf{H}_2(\mathsf{H}_1(\cdot))$. Since collisions for H_1 can be easily found using BD_1, we need H_1 to be injective. For example, $s_1 = 2n$ (co-domain points are $3n$ bits) would lead to an injective H_1 with overwhelming probability.

6.7 The XOR Combiner in the 2-BRO Model

In this section we study the security of the xor combiner $\mathsf{C}_\oplus^{\mathsf{H}_1,\mathsf{H}_2}(x) := \mathsf{H}_1(x) \oplus \mathsf{H}_2(x)$ in the 2-BRO model, where both $\mathsf{H}_1, \mathsf{H}_2 \in \mathrm{Fun}[n, n+s]$. We will prove one-way security based on the hardness of the set-intersection problem and briefly discuss PRG security and collision resistance. In the next chapter, specifically Section 7.3, we show that security of this combiner may go well beyond the three properties discussed in this section: there we prove indifferentiability against adversaries with bounded adaptivity with respect to the backdoor oracle queries.

6.7.1 One-Way Security

We prove rPre-security of the xor combiner similarly to the previous combiners. Although the communication complexity problem directly underlying our reduction is multi-set, we can still relate it to a standard single-instance INT problem, for which lower bounds are known.

Theorem 6.19. *Let* $\mathsf{H}_1, \mathsf{H}_2 \in \mathrm{Fun}[n, n+s]$ *and* $\mathsf{C}_\oplus^{\mathsf{H}_1,\mathsf{H}_2}(x) := \mathsf{H}_1(x) \oplus \mathsf{H}_2(x)$*. Then for any adversary* $\mathcal{A}$ *against the* rPre *security of* $\mathsf{C}_\oplus^{\mathsf{H}_1,\mathsf{H}_2}$ *in the* 2*-BRO model there is a* 2*-party protocol* π *against set-intersection with the distribution* $\mu = \mu(p', q')$ *where* $p' = q' = 2n \ln 2/2^{n+s}$ *and such that*

$$\mathsf{Adv}^{\mathrm{rpre}}_{\mathsf{C}_\oplus^{\mathsf{H}_1,\mathsf{H}_2}}(\mathcal{A}) \le \mathsf{Adv}^{\mathrm{int}}_\mu(\pi) + 2 \cdot 2^{-n} \qquad \text{and} \qquad \mathsf{D}_\mu(\pi) \le \mathsf{Q}(\mathcal{A}) + 4n + 2s\ .$$

Proof. Similarly to Theorem 6.16 we present the proof in two steps: we identify the underlying communication complexity problem and then relate it to a more standard one, for which we have proven lower bounds.

In the rPre game for the xor combiner an adversary $\mathcal{A}$ is given a random point $y^* \in \{0,1\}^{n+s}$ and its task is to find a point $x^* \in \{0,1\}^n$ such that $\mathsf{H}_1(x^*) \oplus \mathsf{H}_2(x^*) = y^*$. Such an adversary exists iff there is an adversary that can output a pair (i, x^*) such that $\mathsf{H}_1(x^*) = i$ and $\mathsf{H}_2(x^*) = i \oplus y^*$. That is, this adversary finds an $x^* \in S_i \cap T_{i\oplus y^*}$, where $S_i := \mathsf{H}_1^-(i)$ and $T_{i\oplus y^*} := \mathsf{H}_2^-(i \oplus y^*)$.

We simplify further. Given y^*, let $\tilde{\mathsf{H}}_2(x^*) := \mathsf{H}_2(x^*) \oplus y^*$. Then the problem becomes equivalent to finding an $x^* \in S_i \cap \tilde{T}_i$ where S_i is as before and $\tilde{T}_i := \tilde{\mathsf{H}}_2^-(i)$. Thus the problem at hand is $(2^{n+s}, 2^{n+s})$-INT on a product binomial distribution with parameters $p = q = 1/2^{n+s}$.

We can relate this problem to the single-instance set-intersection as follows. Suppose we have an instance (S^*, T^*) for set-intersection with parameters $p' := q' := 2n \ln 2 \cdot p$ over subsets of the universe $\{0,1\}^{(n+s)+n}$. We consider elements $x = x_l \| x_r \in S^*$ where $x_l \in \{0,1\}^{n+s}$ and $x_r \in \{0,1\}^n$, and compute the probability that the x_r's do not cover $\{0,1\}^n$ as follows.

$$\begin{aligned}\Pr[\exists x_r \in \{0,1\}^n \ \forall x_l \in \{0,1\}^{n+s} : (x_l \| x_r) \notin S^*] &\le 2^n \Pr[\forall x_l \in \{0,1\}^{n+s} : (x_l \| x_r) \notin S^*] \\ &\le 2^n (1-p')^{2^{n+s}} \le 2^n e^{-2n \ln 2} = 2^{-n} \ ,\end{aligned}$$

which is negligible. Thus with parameter p', the x_r's in S^* will cover $\{0,1\}^n$ with overwhelming probability. A similar result holds for T^*. Thus if we define $S_{x_l} := \{x_r\}$ and $\tilde{T}_{x_l} := \{x_r\}$ we will have with overwhelming probability that

$$\bigcup_{x_l \in \{0,1\}^{n+s}} S_{x_l} = \{0,1\}^n \qquad \text{and} \qquad \bigcup_{x_l \in \{0,1\}^{n+s}} \tilde{T}_{x_l} = \{0,1\}^n \ .$$

Our next step it to redistribute the elements among the sets so that they form partitions. As in Lemma 6.17, we can do this using the ReDist algorithm of Figure 6.4. As shown in the proof of Lemma 6.17, the sets output by ReDist will be correctly distributed as preimages under a random oracle. Therefore, the two simulated functions H_1 (by Alice) and H_2 (by Bob) are random oracles and they can run $\mathcal{A}$ in tandem (similar to the proof of Theorem 6.12). Finally, after $\mathcal{A}$ terminates with a preimage $x_r \in \{0,1\}^n$ of y^* under the combiner, Alice computes $x_l := \mathsf{H}_1(x_r)$ and outputs $x_l \| x_r$ as an element in the intersection of S^* and T^*. (Since ReDist only removes duplicates, an intersection will be found).

The communication complexity of such a protocol π run by Alice and Bob is bounded by the query complexity of $\mathcal{A}$, plus $n + s$ bits for communicating y^*, n bits for potentially sending x_r to Alice, and $2n + s$ for the final message being the element $x_l \| x_r$. □

If $\mathsf{H}_1, \mathsf{H}_2 \in \text{Fun}[n, (1+\sigma)n]$, we have a reduction to INT with $p' = q' = 2n \ln 2 / 2^{(1+\sigma)n}$ and a universe of size $N = 2^{n+(1+\sigma)n}$. We can write p' (and q') as $1/N^\alpha$ with $\alpha = \frac{(1+\sigma)n - \log(2n \ln 2)}{(2+\sigma)n}$. The feasible region in Figure 6.3 for $\alpha = \beta$ consists of $1/3 < \alpha < 1/2$. These inequalities for $n \ge 128$ translate to $-\frac{13}{32} + \frac{3}{256} \log(\ln 2) \approx -0.42 < \sigma < 0$. For such parameters we thus get strong rPre security. Furthermore, since $\sigma < 0$ the functions are (highly) compressing, which means the image uniformity bound is negligible. With an analysis similar to Corollary 6.13 we get strong one-way security.

6.7.2 PRG and CR Security

For oPRG security, a reduction to set-disjointness can be given following the structure of Theorem 6.19. With $\mathsf{H}_1, \mathsf{H}_2 \in \text{Fun}[n, n+s]$, we have $N = 2^{2n+s}$ and $p' = q' = 2n \ln 2 / 2^{n+s}$ in the reduction and hence $\delta = 1/(Npq) = 2^s/(2n \ln 2)^2$. We set $s = 2 \log n + 2$, in which case we have that $\delta = 1/(\ln 2)^2 > 1$, and the set-disjointness lower bound is non-trivial (and is at least $2^{n - \mathcal{O}(\log n)}$) when the error $\varepsilon < 0.067$. This establishes the oPRG security of xor for $s = 2 \log n + 2$. However, an attempt to show weak PRG security (as for the other combiners) fails. The term corresponding to image uniformity bound in Lemma 6.1 when $s = 2\log n + 2$ (i.e., when $C := 1/2^s = 1/(4n^2)$) is less than 0.001 for $n \geq 12$ (and hence small enough). However, in order to upper bound the PRG advantage via Lemma 6.9, we would have to use $\Pr[y \in I] \approx 1 - e^{-1/(4n^2)} \approx \frac{1}{4n^2}$ and thus would need the overall bound $0.001 + (4n^2 - 1) \cdot (1 - \varepsilon) - (1 - \frac{1}{4n^2}) < 1$. This means that ε should be $1 - \mathcal{O}(1/n^2)$, whereas the DISJ bound is only established for sufficiently small values of protocol error. We conjecture, however, that the xor combiner is a PRG in 2-BRO for arbitrary stretch and the issues with setting the parameters can be overcome with tighter proofs and lower bounds for problems with distributions that match xor better.

Similarly, collision resistance of xor in 2-BRO is also conjectural. We, however, note that its analysis can be based on a multi-set 2INT problem.

In the next chapter (cf. Section 7.3) , we continue exploring the security of the xor combiner in more generality. There, we take a different approach and use different proof techniques to study the power of xor in the context of indifferentiability.

Chapter 7

Indifferentiability-Combiners for Backdoored Random Oracles

In this chapter we examine the possibility of building a hash function which is indifferentiable from a backdoor-free random oracle by combining multiple backdoored random oracles. We make considerable progress in this direction by showing that the xor combiner in the 2-BRO model goes well beyond security against preprocessing attacks. Indeed it provides indifferentiability as long as the adversary's backdoor queries do not switch back and forth between the backdoor oracles more than a logarithmic number of times. We also study a 2-out-of-3-source extractor in the 3-BRO model and show that it achieves indifferentiability even with respect to a linear number of switches. To prove these results we refine a technique by Göös et al. [GLM^{+}15] for decomposing distributions with high min-entropy into convex combinations of high min-entropy distributions with more structure and show how this technique can be applied in more involved settings, where adaptive backdoor queries are allowed. Finally, we define a notion of indifferentiability with auxiliary input, which is a natural restriction of indifferentiability in the BRO model and provide two secure constructions.

My Scientific Contribution in this Chapter

The material in this chapter was published in [DFMT20a] and its full version [DFMT20b], which are joint work with Yevgeniy Dodis, Pooya Farshim, and Stefano Tessaro. Recall that in Chapter 6 we conjectured that the xor combiner in the 2-BRO model retains OW, PRG, and CR-security for all stretch values. Pooya raised the question of whether this combiner in the 2-BRO model is indifferentiable from a random oracle. Pooya and I then discussed potential indifferentiability simulators that took advantage of the decomposition technique. I developed the refined decomposition technique of Section 7.2 with the help of Pooya, Stefano, and Yevgeniy. The indifferentiability proof of xor in Section 7.3 and of the 2-out-of-3-source extractor in Section 7.4 are the result of many iterations and discussions with all authors. Stefano and Yevgeniy proposed the idea of using an extractor-based combiner to achieve better bounds. Yevgeniy suggested pairwise inner-product as a potential instantiation and the proof of Lemma 7.9, which we wrote together. The parameter estimations for xor (at the end of Section 7.3) and for pairwise inner-product (at the end of Section 7.4) were conducted jointly by all authors. The auxiliary-input indifferentiability notions and achieving them via combiners and salting, included in Section 7.5, were mainly conducted by Pooya, with my help.

7.1 Introduction

In the previous chapter, we studied the security of concatenation, cascade, and xor combiners in the 2-BRO model. Using new types of reductions to problems with high communication complexity, we proved that central cryptographic security properties, such as one-way security, pseudorandom generator security and collision resistance are indeed achievable by these combiners. However, reductions to communication complexity problems are at times tedious and very specific to the combiner. Furthermore, hardness of the communication complexity problem underlying collision resistance, i.e., multi-set double-intersection, is conjectural and remains to be proven. Moreover, a number of deployed protocols have only been shown to be secure in the random-oracle model, and thus may rely on security guarantees beyond one-wayness, pseudorandomness, or collision resistance. This raises the question whether or not other cryptographic properties expected from a good hash function are also met by these combiners. We formalize and study this question in the *indifferentiability framework* (cf. Section 2.4.1), which has been immensely successful in justifying the soundness of hash-function designs.

7.1.1 Indifferentiability in the k-BRO Model

The central question tackled in this chapter is whether combiners that are indifferentiable from a conventional (backdoor-free) random oracle exist, when the underlying primitives are two (or more) backdoored random oracles. For a more general definition of indifferentiability we refer to Section 2.4.1 in the preliminaries. Recall that in the k-BRO model the underlying honest interfaces are k random oracles H_i and the adversarial interfaces are the respective k backdoor oracles BD_i. Also let us recall the indifferentiability advantage of an adversary $\mathcal{D}$ with respect to a combiner $\mathsf{C}^{\mathsf{H}_1,\ldots,\mathsf{H}_k}$ in the k-BRO model and a simulator $\mathsf{Sim} := (\mathsf{SimH}_1, \ldots, \mathsf{SimH}_k, \mathsf{SimBD}_1, \ldots, \mathsf{SimBD}_k)$, which we defined in Section 3.3.1 as

$$\mathsf{Adv}^{\mathrm{indiff}}_{\mathsf{C}^{\mathsf{H}_1,\ldots,\mathsf{H}_k},\mathsf{Sim}}(\mathcal{D}) := \left| \Pr\left[\mathcal{D}^{\mathsf{C}^{\mathsf{H}_1,\ldots,\mathsf{H}_k},\mathsf{H}_1,\ldots,\mathsf{H}_k,\mathsf{BD}_1,\ldots,\mathsf{BD}_k}\right] - \Pr\left[\mathcal{D}^{\mathsf{RO},\mathsf{SimH}_1^{\mathsf{RO}},\ldots,\mathsf{SimH}_k^{\mathsf{RO}},\mathsf{SimBD}_1^{\mathsf{RO}},\ldots,\mathsf{SimBD}_k^{\mathsf{RO}}}\right] \right| .$$

Here RO is a random oracle whose domain and co-domain match those of C. We emphasize that the simulators do not get access to any backdoor oracles and have to simulate these on their own. This ensures that any attack against a construction with backdoors translates to one against the random oracles without any backdoors.

Let us consider the concatenation combiner $\mathsf{H}_1(x)\|\mathsf{H}_2(x)$, where H_1 and H_2 are both backdoored. This construction was shown in Section 6.5 to be one-way, PRG secure, and (conjecturally) collision resistant if both underlying functions are highly compressing. Despite this, the concatenation combiner cannot be indifferentiable from a random oracle: using the backdoor oracle for H_1, an attacker can compute two inputs x and x' such that $\mathsf{H}_1(x) = \mathsf{H}_1(x')$, query them to the construction and return 1 iff the left sides of the outputs collide. However, any simulator attempting to find such a pair with respect to a backdoor-free random oracle must place an exponentially large number of queries. Note that attacks on the cascade combiner $\mathsf{H}_2(\mathsf{H}_1(x))$ were given in Section 6.3.2 for a wide range of parameter regimes, leaving only the expand-then-compress case as potentially indifferentiable. On the other hand, the xor combiner $\mathsf{H}_1(x) \oplus \mathsf{H}_2(x)$, which is simpler, more efficient and one of the most common ways to combine hash functions, resists these. Furthermore, an indifferentiability proof of the expand-then-compress cascade combiner seem to closely follow that of the xor combiner and thus we focus on the latter here.

Decomposition of distributions. When proving results in the presence of auxiliary input, Uhruh [Unr07] observed that pre-computation (or leakage) on a random oracle can leak a significant amount of information only on restricted parts of its support. The problem of dealing with auxiliary input was later revised in a number of works [DGK17, CDGS18, CDG18]. In particular, Coretti et al. [CDGS18], building on work in communication complexity, employed a *pre-sampling technique* to prove a number of positive results in the RO model with auxiliary input (AI-RO) in a more generic way. At a high level, this method permits writing a high min-entropy distribution (here, over a set of functions) as the convex combination of a (large) number of distributions which are fixed on a certain number of points and highly unpredictable on the rest, the so-called $(p, 1-\delta)$-dense distributions. Roughly speaking, such distributions are fixed on p points, while the min-entropy on the remaining points is high and correlated with $1 - \delta$. This decomposition technique was originally introduced in the work of Göös et al. [GLM$^+$15].

The simulator. Our indifferentiability simulator for the xor combiner builds on the technique to decompose distributions into a convex combination of $(p, 1 - \delta)$-dense distributions. Simulation of backdoor oracles is arguably quite natural and proceeds as follows. Starting with uniform random oracles H_1 and H_2, on each backdoor query f for H_1 the simulator computes $z = f(\mathsf{H}_1)$ and updates the distribution of the random oracle H_1 to be the uniform distribution over all functions of the same signature, conditioned on the output of f being equal to z. This distribution is then decomposed into a convex combination of some $(p, 1 - \delta)$-dense distributions, from which one function H_1 is sampled by the simulator in order to respond to future queries. For all the fixed points, the simulator sets the value of H_2 consistently with the random oracle (and H_1) and the distribution of H_2 is updated accordingly. An analogous procedure is implemented as the simulator for the second backdoored random oracle.

Technical analysis. The first technical contribution of our work is a refinement of the decomposition technique which can be used to *adaptively* decompose distributions after backdoor queries. We show that this refinement is sufficiently powerful to allow proving indifferentiability up to a logarithmic (in the input size) number of query switches between the backdoor oracles. We prove this via a sequence of games which are carefully designed so as to be compatible with the decomposition technique. A key observation is that in contrast to previous works in the AI-RO model, we do not replace the dense (intuitively, unpredictable) part of the distribution of random oracles with uniform: queried backdoor functions “see” the entire table of the random oracle and this replacement would result in a noticeable change. Second, we modify the number of fixed points in the (partially) dense distributions so that progressively smaller sets of points are fixed. Even though each leakage corresponds to fixing a large number of points, it is proportionally smaller than the previous number of fixed points. Thus the overall bound that we obtain remains small. In order to overcome the bounded switch restriction and prove full indifferentiability, one would require an improved decomposition technique which fixes considerably less points after each leakage. We discuss this open question briefly in Chapter 9.

Simulator efficiency. Our simulator runs in doubly exponential time in the bit-length of the random oracle and thus is of use in information-theoretic settings. These include the vast majority of symmetric constructions. Protocols based on computational assumptions (such as public-key encryption) escape this treatment: the overall adversary obtained via the composition theorem (cf. Theorem 2.1) would run the decomposition algorithm and, hence, will not be poly-time. This observation, however, also applies in more generality to the BRO model as the backdoor oracles also allow for non-polynomial time computation, trivially breaking any computational assumption if unrestricted. We leave exploring solutions around this to future work, but we will include some discussion in Chapter 9.

An extractor-based combiner with improved security. We apply the above proof technique to the analysis of an alternative combiner for three independent backdoored random oracles, which relies on 2-out-of-3-source extractors that output good randomness as long as two out of the three of the inputs have sufficient min-entropy. Given such an extractor Ext, our combiner is

$$\mathsf{C}_{3\mathrm{ext}}^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_3}(x) := \mathsf{Ext}\big(\mathsf{H}_1(x), \mathsf{H}_2(x), \mathsf{H}_3(x)\big) \ .$$

As mentioned above, our simulator for the xor combiner programs H_2 on the fixed points for H_1 (and vice versa) using the random oracle. This results in a loss in the security level since dense values are replaced with uniform values. In contrast, here the extractor ensures that image values are closer to uniform and thus the overall loss is lower. We show that a 2-out-of-3-source extractor can tolerate even a number of switches between the backdoor oracles which is linear in the size of the BRO inputs. This gives us more hope for unbounded adaptivity, in case improved decomposition techniques are found.

Composition. Let c denote the number of times the adversary switches between one backdoor oracle to the other. Regarding the query complexities of our simulators, each query to a backdoor oracle translates to roughly $N^{1-2^{-c}}$ queries to the random oracle for the xor combiner and roughly $N^{1-3/(c+3)}$ queries to the random oracle for the extractor combiner. This in particular means that, for a wide range of parameters, composition is only meaningful with respect to security notions whereby the random oracle can tolerate a large number of queries. This, for example, would be the case for one-way, PRG, and PRF security notions where the security bounds are of the form $\mathcal{O}(q/N)$. However, with respect to a smaller number of switches (as well as in the auxiliary-input setting with no adaptivity), collision resistance can still be achieved.

7.1.2 Indifferentiability with Auxiliary Input

When our definition of indifferentiability is restricted so that only a single backdoor query to each function at the onset is allowed, we obtain a notion that formalizes *indifferentiability with auxiliary input.* This definition is interesting as it is sufficiently strong to allow for the generic replacement of random oracles with iterative constructions even in the presence of preprocessing attacks, where the adversary has potentially invested resources during an off-line phase to build-up data structures that considerably speed up the actual on-line phase of the attack. Accordingly our positive results in the BRO model when considered with no adaptivity translate to indifferentiability with independent preprocessing attacks. To complement this picture, we also discuss the case of auxiliary-input indifferentiability with a single BRO and show, as expected, that a salted indifferentiable construction is also indifferentiable in presence of an auxiliary input.

7.2 Refined Decomposition of High Min-Entropy Distributions

Any distribution with high min-entropy can be written as a convex combination of distributions that are fixed on a number of points and have high min-entropy on the rest. Such distributions are referred to as $(p, 1-\delta)$-dense distributions, for some number of points $p \in \mathbb{N}$ and some min-entropy rate $\delta > 0$. Formally, we have the following definitions (adopted from [GLM$^+$15]) of (partially) dense distributions, resp. probability density functions. Intuitively, bit strings from a dense distribution are unpredictable not only as a whole but also in any of their substrings (or blocks) and any combination of those substrings.

Definition 7.1 (Dense distributions). *Let μ be a probability density function over the domain $[M]^N$. Recall that for a random variable $F \sim \mu$ and a set $I \subseteq [N]$, we denote by F_I a random variable over the domain $[M]^{|I|}$ which corresponds to F projected on blocks determined by I. Then μ is called*

- $(1-\delta)$-dense *if for the random variable $F \sim \mu$, it holds that for every subset $I \subseteq [N]$ we have $\mathbf{H}_\infty(F_I) \geq (1-\delta) \cdot |I| \cdot \log M$.*
- $(p, 1-\delta)$-dense *if for the random variable $F \sim \mu$ there exists a set $I \subseteq [N]$ of size $|I| \leq p$ such that $\mathbf{H}_\infty(F_I) = 0$, while for every subset $J \subseteq [N] \setminus I$ we have $\mathbf{H}_\infty(F_J) \geq (1-\delta) \cdot |J| \cdot \log M$. That is, μ is fixed on at most p coordinates and is $(1-\delta)$-dense on the rest.*

We call a distribution dense, if the corresponding probability density function is dense.

The decomposition technique introduced by Göös et al. [GLM$^+$15] has its origins in communication complexity theory. We generalize this technique, with a terminology closer to that of Kothari et al. [KMR17], in order to allow for adaptive leakage. The original lemma, also used by Coretti et al. [CDGS18], can be easily derived as a special case of our lemma. For this, one assumes that the starting distribution before the leakage was uniform, in other words $(0, 1)$-dense.

When proving results in the AI-RO model, Uhruh [Unr07] observed that pre-computation (or leakage) on a random oracle can cause a significant decrease of its min-entropy only on restricted parts of its support (i.e., at most on p points), causing that part to become practically fixed, while the rest remains indistinguishable from random to a *bounded-query* distinguisher. This means that for proofs in the AI-RO model, after fixing p coordinates of the random oracle, the rest can be lazily sampled from a uniform distribution. Coretti et al. [CDGS18] recently gave a different and tighter proof of this technique consisting of two main steps. First, the decomposition technique is used to show that the distribution of a random oracle given some leakage is *statistically* close to a $(p, 1-\delta)$-dense distribution. Second, they prove that no *bounded-query* algorithm can distinguish a $(p, 1-\delta)$-dense distribution from one that is fixed on the same p points and is otherwise uniform (a so-called *p-bit-fixing* distribution), as suggested by Unruh [Unr07].

Since in the BRO model adaptive queries to the backdoor oracles are allowed, a function queried to a backdoor oracle is able to "see" the entire random oracle, rather than a restricted part of it. Hence, when analyzing the distribution of a random oracle after adaptive leakage, it is crucial that we keep the distributions *statistically close.* In other words we use $(p, 1-\delta)$-dense distributions instead of p-bit-fixing.

In the k-BRO model, we are concerned with multiple queries to the backdoor oracles, i.e., continuous and adaptive leakage that can depend on previously leaked information about both hash functions. Intuitively, since the leakage function can be arbitrary, it can in particular depend on the previously leaked values. We still need to argue that the distribution obtained after leakage about a starting $(p_{\text{strt}}, 1-\delta_{\text{strt}})$ distribution, which is not necessarily uniform, is also close to a convex combination of $(p, 1-\delta)$ distributions. Naturally, we have $\delta \geq \delta_{\text{strt}}$, since min-entropy decreases after new leakage, and $p \geq p_{\text{strt}}$, since additional points are fixed.

Looking ahead, in the indifferentiability proofs, this refined decomposition lemma allows us to simply fix a new portion p_{frsh} of the simulated hash function after each leakage (i.e., backdoor query) and not to worry about the rest, which still has high entropy and can be lazily sampled (from a dense distribution) upon receiving the next query.

Lemma 7.1 (Refined decomposition after leakage)**.** *Let μ be a $(p_{\text{strt}}, 1-\delta_{\text{strt}})$-dense density function over $[M]^N$ for some $p_{\text{strt}}, \delta_{\text{strt}} \geq 0$. Let $f : [M]^N \to \{0,1\}^\ell$ be an arbitrary function and $z \in \{0,1\}^\ell$ be a bit string. Then for any $p_{\text{frsh}}, \gamma > 0$, the density function conditioned on the leakage $\mu|_{f(\cdot)=z}$ is γ-close to a convex combination of finitely many $(p, 1-\delta)$-dense density functions for some p and δ such that*

$$p_{\text{strt}} \leq p \leq p_{\text{strt}} + p_{\text{frsh}} \quad \textit{and} \quad \delta_{\text{strt}} \leq \delta \leq \frac{\delta_{\text{strt}} \cdot \log M \cdot (N - p_{\text{strt}}) + \ell_z + \log \gamma^{-1}}{p_{\text{frsh}} \cdot \log M} \ ,$$

where $\ell_z := \mathbf{H}_\infty(G) - \mathbf{H}_\infty(F)$ is the min-entropy deficiency of $F \sim \mu|_{f(\cdot)=z}$ compared to $G \sim \mu$.

Proof. This refined decomposition lemma differs from the original lemma in that the starting density function μ is $(p_{\text{strt}}, 1-\delta_{\text{strt}})$-dense. As a first step, we modify the original decomposition algorithm from [GLM+15, KMR17] so that it additionally gets the set of p_{strt} indices $I_{\text{strt}} \subseteq [N]$ that are already fixed in μ from the start.

Our refined decomposition algorithm $\mathsf{RefinedDecomp}$, given below, recursively decomposes the domain $[M]^N$, according to the density function after leakage $\mu_z := \mu|_{f(\cdot)=z}$, into $d+1$ partitions $D_1, \ldots, D_d, D_{\mathsf{err}} \subseteq [M]^N$ such that $\left(\bigcup_{i=1}^d D_i\right) \cup D_{\mathsf{err}} = [M]^N$, where err stands for erroneous. For all i with $1 \leq i \leq d$ the partition D_i defines a $(p, 1-\delta)$-dense density function $\mu_z|_{D_i}$.

Each recursive call on a domain D to $\mathsf{RefinedDecomp}$ (other than the call leading to D_{err}, which we will discuss shortly) returns a pair (D_i, I_i), where D_i represents a subset of $[M]^N$, where the images of all points in the set $I_i \subset [N]$ are fixed to the same values under all functions $\mathsf{H} \in D_i$. In other words, we have $\mathsf{H}_{I_i} = \alpha_i$ for some $\alpha_i \in [M]^{|I_i|}$. The algorithm finds such a pair (D_i, I_i) by considering the biggest set I_i (excluding those points fixed from the start, i.e., I_{strt}) such that the min-entropy of F_{I_i} (for $F \sim \mu_z|_D$) is too small (as determined by the rate δ) and then finding some α_i which is a very likely value of F_{I_i}. Then I_i is returned with some D_i as the partition that contains all H with $\mathsf{H}_{I_i} = \alpha_i$. The next recursive call will exclude D_i from the considered domain.

Decomposition halts either if the probability of a sample falling into the current domain is smaller than γ (i.e., $\mu_z(D) \leq \gamma$) or the current distribution is already $(p_{\text{strt}}, 1-\delta)$-dense. In both cases the algorithm returns the current domain D together with an empty set. In the former case the returned domain is marked as an erroneous domain $D_{\mathsf{err}} := D$, since it may not define a $(p, 1-\delta)$-dense distribution. Let us without loss of generality assume that μ_z is not $(p_{\text{strt}} + p_{\text{frsh}}, 1-\delta)$-dense, as otherwise the claim holds trivially.

The formal definition of RefinedDecomp is given below. Before calling RefinedDecomp, we initialize the desired min-entropy rate as $\delta := \frac{\delta_{\text{strt}} \cdot \log M \cdot (N - p_{\text{strt}}) + \ell_z + \log \gamma^{-1}}{p_{\text{frsh}} \cdot \log M}$.

```
RefinedDecomp$[\mu_z, \delta, \gamma, I_{\text{strt}}](D)$
if $\mu_z(D) \leq \gamma$ then return $(D_{\mathsf{err}} \leftarrow D, \emptyset)$
if $\mu_z|_D$ is $(|I_{\text{strt}}|, 1-\delta)$-dense then return $(D, \emptyset)$
for the random variable $F \sim \mu_z|_D$ let $I \subseteq [N]$ be a maximal set such that
        $\mathbf{H}_\infty(F_I) < (1-\delta) \cdot |I| \cdot \log M$   and   $I \cap I_{\text{strt}} = \emptyset$.
let $\alpha \in [M]^{|I|}$ be such that $\Pr[F_I = \alpha] > 2^{-(1-\delta) \cdot |I| \cdot \log M}$.
$D_\alpha \leftarrow D \cap \{\mathsf{H} \in [M]^N \mid \mathsf{H}_I = \alpha\}$
$D_{\neq\alpha} \leftarrow D \cap \{\mathsf{H} \in [M]^N \mid \mathsf{H}_I \neq \alpha\}$
return $((D_\alpha, I), \mathsf{RefinedDecomp}[\mu_z, \delta, \gamma, I_{\text{strt}}](D_{\neq\alpha}))$
```

Now we turn our attention to proving that every partition D_i (other than D_{err}) returned by the above decomposition algorithm defines a density function $\mu_z|_{D_i}$ which is $(p, 1-\delta)$-dense.

Claim. *For all i with $1 \leq i \leq d$ we have that $\mu_z|_{D_i}$ is $(p, 1 - \frac{\delta_{\text{strt}} \cdot \log M \cdot (N - p_{\text{strt}}) + \ell_z + \log \gamma^{-1}}{p_{\text{frsh}} \cdot \log M})$-dense, where $p_{\text{strt}} \leq p \leq p_{\text{strt}} + p_{\text{frsh}}$.*

Proof. Let $\delta := \frac{\delta_{\text{strt}} \cdot \log M \cdot (N - p_{\text{strt}}) + \ell_z + \log \gamma^{-1}}{p_{\text{frsh}} \cdot \log M}$. Let I be the set of freshly fixed points in $\mu_z|_{D_i}$ and $\overline{I \cup I_{\text{strt}}} := [N] \backslash (I \cup I_{\text{strt}})$. Let $\alpha_\cup \in [N]^{|I \cup I_{\text{strt}}|}$ be such that $\mathsf{H}_{I \cup I_{\text{strt}}} = \alpha_\cup$ for all functions $\mathsf{H} \twoheadleftarrow \mu_z|_{D_i}$. We show the claim in two steps. First we argue for the $(1-\delta)$-density of $\mu_z|_{D_i}$ on values projected to $\overline{I \cup I_{\text{strt}}}$ and in a second step we upper bound the size of I.

1. Suppose to the contrary that $\mu_z|_{D_i}$ is not $(1-\delta)$-dense on $\overline{I \cup I_{\text{strt}}}$. Then there must exist a non-empty set which violates the density property. That is, there exists a non-empty set $J \subseteq \overline{I \cup I_{\text{strt}}}$ and some $\beta \in [M]^{|J|}$ such that

$$\Pr[F_J = \beta] = \Pr[F_J = \beta \mid F_{I \cup I_{\text{strt}}} = \alpha_\cup] > 2^{-(1-\delta) \cdot |J| \cdot \log M} \ ,$$

   with probabilities taken over the random choice of functions according to the random variable F corresponding to the density function $\mu_z|_{D_i}$, i.e., $F \sim \mu_z|_{D_i}$. Now the union of the three sets $I^* := I \cup I_{\text{strt}} \cup J$ forms a new set such that for some value $\beta^* \in [M]^{|I \cup I_{\text{strt}} \cup J|}$ we have

$$\begin{aligned}\Pr[F_{I^*} = \beta^*] &= \Pr[F_{I \cup I_{\text{strt}}} = \alpha_\cup \wedge F_J = \beta] \\ &= \Pr[F_{I \cup I_{\text{strt}}} = \alpha_\cup] \cdot \Pr[F_J = \beta | F_{I \cup I_{\text{strt}}} = \alpha_\cup] \\ &> 2^{-(1-\delta) \cdot |I \cup I_{\text{strt}}| \cdot \log M} \cdot 2^{-(1-\delta) \cdot |J| \cdot \log M}\end{aligned}$$

$$= 2^{-(1-\delta)\cdot|I\cup I_{\mathrm{strt}}\cup J|\cdot\log M} \ .$$

Since J was assumed to be non-empty and disjoint from $I \cup I_{\mathrm{strt}}$ (and in particular with I), its existence violates the maximality of I. Therefore, we conclude that $F_{\overline{I\cup I_{\mathrm{strt}}}}$ is $(1-\delta)$ dense.

2. We now bound the size of I, given that $\delta = \frac{\delta_{\mathrm{strt}}\cdot\log M\cdot(N-p_{\mathrm{strt}})+\ell_z+\log\gamma^{-1}}{p_{\mathrm{frsh}}\cdot\log M}$. Let $F \sim \mu_z$ and $G \sim \mu$. We have $\mathbf{H}_\infty(F) = \mathbf{H}_\infty(G) - \ell_z \geq (1-\delta_{\mathrm{strt}})\cdot(N-p_{\mathrm{strt}}).\log M - \ell_z$, where the inequality holds, since μ is $(1-\delta_{\mathrm{strt}})$-dense when ignoring p_{strt} many fixed values. For an arbitrary $\beta \in [M]^{|I|}$ we can write

$$\begin{aligned}
\Pr_{\mu_z|_{D_i}}[F_I = \beta] &\leq \Pr_{\mu_z}[F_I = \beta]/\mu_z(D_i)\\
&\leq \Pr_{\mu_z}[F_I = \beta]/\gamma\\
&= \sum_{\beta'\in[M]^{N-|I|-|I_{\mathrm{strt}}|}} \Pr_{\mu_z}[F_I = \beta \wedge F_{[N]\setminus(I\cup I_{\mathrm{strt}})} = \beta']/\gamma\\
&\leq 2^{(N-|I|-p_{\mathrm{strt}})\cdot\log M}\cdot 2^{-\mathbf{H}_\infty(F)}/\gamma \qquad (7.1)\\
&\leq 2^{(N-|I|-p_{\mathrm{strt}})\cdot\log M}\cdot 2^{-((1-\delta_{\mathrm{strt}})\cdot(N-p_{\mathrm{strt}})\cdot\log M-\ell_z)}/\gamma\\
&= 2^{\delta_{\mathrm{strt}}\cdot N\cdot\log M-\delta_{\mathrm{strt}}\cdot p_{\mathrm{strt}}\cdot\log M-|I|\cdot\log M+\ell_z}/\gamma\\
&= 2^{\delta_{\mathrm{strt}}\cdot\log M\cdot(N-p_{\mathrm{strt}})-|I|\cdot\log M+\ell_z+\log\gamma^{-1}} \ .
\end{aligned}$$

By definition of the decomposition there exists an $\alpha \in [M]^I$ such that $\Pr_{\mu_z|_{D_i}}[F_I = \alpha] > 2^{-(1-\delta)\cdot|I|\cdot\log M}$. Therefore, we obtain

$$|I| \leq \frac{\delta_{\mathrm{strt}}\cdot\log M\cdot(N-p_{\mathrm{strt}})+\ell_z+\log\gamma^{-1}}{\delta\cdot\log M} \ .$$

Substituting δ by $\frac{\delta_{\mathrm{strt}}\cdot\log M\cdot(N-p_{\mathrm{strt}})+\ell_z+\log\gamma^{-1}}{p_{\mathrm{frsh}}\cdot\log M}$, we obtain $|I| \leq p_{\mathrm{frsh}}$ and, therefore, for the total number of fixed points $p := |I \cup I_{\mathrm{strt}}|$ we get

$$p_{\mathrm{strt}} \leq p \leq p_{\mathrm{strt}} + p_{\mathrm{frsh}} \ ,$$

as stated in the claim.

□

Overall μ_z can be written as a convex combination of $\mu_z|_{D_1}, \ldots, \mu_z|_{D_d}$ and $\mu_z|_{D_{\mathsf{err}}}$, i.e.,

$$\mu_z = \sum_{i=1}^{d} \mu_z(D_i)\cdot\mu_z|_{D_i} + \mu_z(D_{\mathsf{err}})\cdot\mu_z|_{D_{\mathsf{err}}} \ .$$

Since $\mu_z(D_{\mathsf{err}}) \leq \gamma$ when the algorithm $\mathsf{RefinedDecomp}$ terminates, we can say that μ_z is γ-close to a convex combination of $(p, 1-\delta)$-distributions. □

A special case of the above lemma for a uniform (i.e., $(0,1)$-dense) starting distribution μ, where $p_{\text{strt}} = 0$ and $\delta_{\text{strt}} = 0$, implies the bound $\delta \leq (\ell_z + \log \gamma^{-1})/(p_{\text{frsh}} \cdot \log M)$ used by Coretti et al. [CDGS18].

On achieving better bounds. Note that the coefficient of δ_{strt} in the right hand side of the inequality established in the above lemma is of the order $\mathcal{O}(N/p_{\text{frsh}})$. Looking ahead (see discussions on parameter estimation for indifferentiability proofs) this results in a large increase in the number of points that the simulator needs to fix in order to make sure that the advantage of differentiators remains small. Fixing these points by the simulator needs to happen every time before a differentiator switches from querying one backdoor oracle to the other. Thus any improvement in the bound established in this lemma would translate to tolerating a higher level of adaptivity with respect to the number of switches between backdoor oracles and/or obtaining an improved bound on the indifferentiability. In our current proof, when estimating the number of fixed points, the step to Line 7.1 above only uses the min-entropy of F. There may be a more fine-grained proof which uses the fact that the distribution before leakage was actually $(1-\delta)$-dense, which is more desirable than a high min-entropy distribution. We leave exploring possible improvements for future.

When ℓ bits of information is leaked about a random variable, it is hard to make a generic statement about how much the min-entropy of that random variable actually falls. Below we show, however, that the *expected* min-entropy deficiency after leaking ℓ bits of information can be upper-bounded by ℓ bits. We will make use of this lemma later in our indifferentiability proofs.

Lemma 7.2. *Let F be a random variable over $[M]^N$ and $f : [M]^N \to \{0,1\}^\ell$ be an arbitrary function. Let $\ell_z := \mathbf{H}_\infty(F) - \mathbf{H}_\infty(F|f(F)=z)$ be the min-entropy deficiency of $F|f(F)=z$. Then, we have*

$$\mathbb{E}_{z \in f(\mathsf{supp}(F))}[\ell_z] \leq \ell \ .$$

Proof. Recall that $\widetilde{\mathbf{H}}_\infty(A|B) := -\log\left(\mathbb{E}_b\left[\max_a \Pr[A=a|B=b]\right]\right)$ defines the average min-entropy of A, given B.

$$\begin{aligned}
\mathbb{E}_{z \in f(\mathsf{supp}(\mathsf{F}))}[\ell_z] &= \mathbf{H}_\infty(\mathsf{F}) - \mathbb{E}_{z \in f(\mathsf{supp}(\mathsf{F}))}[\mathbf{H}_\infty(\mathsf{F}|f(\mathsf{F}) = z)] \\
&\leq \mathbf{H}_\infty(\mathsf{F}) - \widetilde{\mathbf{H}}_\infty(\mathsf{F}|f(\mathsf{F}) = z) && (7.2) \\
&\leq \mathbf{H}_\infty(\mathsf{F}) - \mathbf{H}_\infty(\mathsf{F}) + \log|f(\mathsf{supp}(\mathsf{F}))| && (7.3) \\
&\leq \ell \ ,
\end{aligned}$$

where for Line 7.2 we use Jensen's inequality and for Line 7.3 we use [DRS04, Lemma 2.2.b].[1] □

[1] The lemma is as follows. Let A, B be random variables. Then we have $\widetilde{\mathbf{H}}_\infty(A|B) \geq \mathbf{H}_\infty(A,B) - n \geq \mathbf{H}_\infty(A) - n$, where B has at most 2^n possible values.

7.3 Indifferentiability of the XOR Combiner in the 2-BRO Model

In this section we study the indifferentiability of the xor combiner $\mathsf{C}_{\oplus}^{\mathsf{H}_1,\mathsf{H}_2}(x) := \mathsf{H}_1(x) \oplus \mathsf{H}_2(x)$ in the 2-BRO model from a random oracle RO. We show indifferentiability against adversaries that switch between the two backdoor oracles BD_1 and BD_2 only a logarithmic number of times in the input size of the hash functions, while making arbitrary queries to the underlying BROs H_1 and H_2, as well as to the random oracle RO. To prove indifferentiability we need to show the existence of a simulator $\mathsf{Sim} := (\mathsf{SimH}_1^{\mathsf{RO}}, \mathsf{SimH}_2^{\mathsf{RO}}, \mathsf{SimBD}_1^{\mathsf{RO}}, \mathsf{SimBD}_2^{\mathsf{RO}})$ such that no distinguisher placing a "reasonable" number of queries can distinguish

$$(\mathsf{C}_{\oplus}^{\mathsf{H}_1,\mathsf{H}_2}, \mathsf{H}_1, \mathsf{H}_2, \mathsf{BD}_1, \mathsf{BD}_2) \quad \text{and} \quad (\mathsf{RO}, \mathsf{SimH}_1^{\mathsf{RO}}, \mathsf{SimH}_2^{\mathsf{RO}}, \mathsf{SimBD}_1^{\mathsf{RO}}, \mathsf{SimBD}_2^{\mathsf{RO}})\ .$$

Such a simulator is described in Figure 7.1. As discussed in the introduction to this chapter, our simulator is not efficient, since it needs to sample entire BROs and decompose distributions, which require exponential memory and time. However, the result is still useful in information-theoretic settings, including the vast majority of symmetric constructions.

Simulating the evaluation queries to H_1 and H_2 is quite straightforward. In simulating the backdoor queries, we take advantage of the refined decomposition technique (discussed in Section 7.2) for transforming high min-entropy distributions on functions into convex combinations of distributions on functions that have a number of fixed points and are dense otherwise. The backdoor simulator SimBD_1 (resp. SimBD_2) computes the queried function f on the truth table of H_1 (resp. H_2), where H_1 and H_2 are initialized by picking two functions uniformly at random. For the sake of simplicity, we consider an adversary that makes Q consecutive queries, ignoring evaluation and RO-queries in between, to one backdoor oracle before moving to the other. After the i-th sequence of Q queries to one of the backdoor oracles, the leaked backdoor information is translated into fixing p_i rows of the hash function such that the rest is dense and the resulting distribution is statistically close to the true one. In other words, the distribution conditioned on the leakage is γ-close (for some $\gamma > 0$) to a convex combination of $(p, 1-\delta)$-dense distributions obtained after decomposition.

Regarding the min-entropy rate δ_i's, we use odd values of i for the distributions obtained after backdoor queries on H_1 and even values of i for distributions of H_2. Note that is crucial for the statistical distance of these two distributions on the entire table to remain small, since the distinguisher can adaptively query a backdoor oracle which sees and can depend on the *entire* hash function table, as opposed to a limited number of coordinates.

Finding a distribution, which is partly fixed and partly dense, is performed by the $\mathsf{FixRows}$ algorithm also described in Figure 7.1. On input of a distribution μ_z, integer $p \in \mathbb{N}$, and a set $I_{\mathrm{strt}} \in [N]$, the algorithm $\mathsf{FixRows}$ returns a new distribution which is fixed on points in a set I of size at most $p + |I_{\mathrm{strt}}|$ and is for some δ, $(1-\delta)$-dense on the rest, together with a set of assignments A for elements in I according to the output distribution. The $\mathsf{FixRows}$ algorithm internally calls the refined decomposition algorithm $\mathsf{RefinedDecomp}$, which is described in the proof of Lemma 7.1. The distribution returned by $\mathsf{FixRows}$ is indeed one of the distributions in the convex combination built by $\mathsf{RefinedDecomp}$.

```
$\mathsf{RO}(x)$
**if** $\exists y \in [M]$ s.t. $(x, y) \in \mathsf{hst}_{\mathsf{RO}}$ **then return** $y$
$y \twoheadleftarrow [M]$
$\mathsf{hst}_{\mathsf{RO}} \leftarrow \mathsf{hst}_{\mathsf{RO}} \cup \{(x, y)\}$
**return** $y$
```

```
$\mathsf{SimH}_1^{\mathsf{RO}}(x)$
$y_1 \leftarrow \mathsf{H}_1(x)$
$\mathsf{hst}_1 \leftarrow \mathsf{hst}_1 \cup \{(x, y_1)\}$
$\mathsf{hst}_2 \leftarrow \mathsf{hst}_2 \cup \{(x, \mathsf{RO}(x) \oplus y_1)\}$
$\mu_1 \leftarrow \mu_1|_{\mathsf{hst}_1}$
$\mu_2 \leftarrow \mu_2|_{\mathsf{hst}_2}$
$\mathsf{H}_2 \twoheadleftarrow \mu_2$
**return** $y_1$
```

```
$\mathsf{SimH}_2^{\mathsf{RO}}(x)$
$y_2 \leftarrow \mathsf{H}_2(x)$
$\mathsf{hst}_2 \leftarrow \mathsf{hst}_2 \cup \{(x, y_2)\}$
$\mathsf{hst}_1 \leftarrow \mathsf{hst}_1 \cup \{(x, \mathsf{RO}(x) \oplus y_2)\}$
$\mu_2 \leftarrow \mu_2|_{\mathsf{hst}_2}$
$\mu_1 \leftarrow \mu_1|_{\mathsf{hst}_1}$
$\mathsf{H}_1 \twoheadleftarrow \mu_1$
**return** $y_2$
```

```
$\mathsf{SimBD}_1^{\mathsf{RO}}[\bar{p}, \gamma](f)$
$q \leftarrow q + 1$
$z \leftarrow f(\mathsf{H}_1)$
$\mu_1 \leftarrow \mu_1|_{f(\cdot)=z}$
**if** $q = Q$ **do**
  $(\mu_1, A_1) \twoheadleftarrow \mathsf{FixRows}[\gamma](\mu_1, p_{2s+1}, \mathsf{hst}_{1.1})$
  $\mathsf{H}_1 \twoheadleftarrow \mu_1$
  $\mathsf{hst}_1 \leftarrow \mathsf{hst}_1 \cup A_1$
  **for** $(x, y_1) \in A_1$ **do**
    $\mathsf{hst}_2 \leftarrow \mathsf{hst}_2 \cup \{(x, \mathsf{RO}(x) \oplus y_1)\}$
  $\mu_2 \leftarrow \mu_2|_{\mathsf{hst}_2}$
  $\mathsf{H}_2 \twoheadleftarrow \mu_2$
  $q \leftarrow 0$
**return** $z$
```

```
$\mathsf{SimBD}_2^{\mathsf{RO}}[\bar{p}, \gamma](f)$
$q \leftarrow q + 1$
$z \leftarrow f(\mathsf{H}_2)$
$\mu_2 \leftarrow \mu_2|_{f(\cdot)=z}$
**if** $q = Q$ **then**
  $(\mu_2, A_2) \twoheadleftarrow \mathsf{FixRows}[\gamma](\mu_2, p_{2s+2}, \mathsf{hst}_{2.1})$
  $\mathsf{H}_2 \twoheadleftarrow \mu_2$
  $\mathsf{hst}_2 \leftarrow \mathsf{hst}_2 \cup A_2$
  **for** $(x, y_2) \in A_2$ **do**
    $\mathsf{hst}_1 \leftarrow \mathsf{hst}_1 \cup \{(x, \mathsf{RO}(x) \oplus y_2)\}$
  $\mu_1 \leftarrow \mu_1|_{\mathsf{hst}_1}$
  $\mathsf{H}_1 \twoheadleftarrow \mu_1$
  $q \leftarrow 0$
  $s \leftarrow s + 1$
**return** $z$
```

```
$\mathsf{FixRows}[\gamma](\mu_z, p, I_{\mathrm{strt}})$
$((D_1, I_1), \ldots, (D_d, I_d), (D_{\mathsf{err}}, I_{\mathsf{err}})) \twoheadleftarrow \mathsf{RefinedDecomp}[\mu_z, p, \gamma, I_{\mathrm{strt}}]([M]^N)$
$D_{\mathsf{err}} \leftarrow [M]^N$
$(D_i, I_i) \twoheadleftarrow \{(D_1, I_1), \ldots, (D_d, I_d), (D_{\mathsf{err}}, I_{\mathsf{err}})\}$ with probability $\mu_z(D_i)$, where $i \in \{1, \ldots, d, \mathsf{err}\}$
$A \leftarrow \emptyset$; $\mathsf{H} \twoheadleftarrow D_i$
**for** $x \in I_i$ **do** $A \leftarrow A \cup \{(x, \mathsf{H}(x)\}$
**return** $(\mu|_{D_i}, A)$
```

Figure 7.1: Indifferentiability simulator for the xor combiner. We assume initial values $\mathsf{hst}_1 = \mathsf{hst}_2 = \mathsf{hst}_{\mathsf{RO}} := \emptyset$, $\mu_1 = \mu_2 := \mathcal{U}_{[M]^N}$, $\mathsf{H}_1, \mathsf{H}_2 \twoheadleftarrow \mathcal{U}_{[M]^N}$, $q := 0$, and $s := 0$.

Upon fixing $\mathsf{H}_1(x)$ for any row x, the simulator SimBD_1 has to immediately set $\mathsf{H}_2(x) := \mathsf{RO}(x) \oplus \mathsf{H}_1(x)$ (and, analogously, so does SimBD_2) to assure consistency with RO. The simulator specifies a priori the number of points that it can afford to fix (since every such query requires a call to the random oracle RO) and the statistical distance γ that it wants to keep. This strategy of assuring consistency with RO is also followed by evaluation simulators SimH_1 and SimH_2, whereby at most one coordinate of each BRO is fixed upon every query.

Note that SimBD_1 programs values of H_2, which were supposed to be dense (after a first SimBD_2 query), to values that are uniform instead. Therefore, we need to argue later that the statistical distance between a uniform and a dense distribution is small for the number of points that are being treated this way. This is formalized in Lemma 7.3, below. Looking ahead, the need to keep the advantage of the differentiator small is the reason why our simulator adapts the number of fixed points with a differentiator's switch to the other backdoor oracle. Finally, via a hybrid argument we can upper bound the total number of random oracle queries by the simulator and the advantage of the differentiator.

Lemma 7.3. *Let $\mathcal{U}$ be the uniform distribution and $\mathcal{V}$ be a $(1-\delta)$-dense distribution, both over the domain $[M]^t$. Then we have*

$$\mathsf{SD}(\mathcal{U}, \mathcal{V}) \leq t \cdot \delta \cdot \log M \ .$$

Proof. This proof follows that of [CDGS18, Claim 3]. Let V_+ be the set of all values $z \in [M]^t$ for which $\Pr[\mathcal{V} = z] > 0$ holds. We can write the statistical distance between $\mathcal{U}$ and $\mathcal{V}$ as:

$$\begin{aligned} \mathsf{SD}(\mathcal{U}, \mathcal{V}) &= \sum_{z \in [M]^t} \max\left\{0, \Pr[\mathcal{V} = z] - \Pr[\mathcal{U} = z]\right\} \\ &= \sum_{z \in V_+} \max\left\{0, \Pr[\mathcal{V} = z] - \Pr[\mathcal{U} = z]\right\} \\ &= \sum_{z \in V_+} \Pr[\mathcal{V} = z] \cdot \max\left\{0, 1 - \frac{\Pr[\mathcal{U} = z]}{\Pr[\mathcal{V} = z]}\right\} \ . \end{aligned}$$

Now, observe that for any value $z \in [M]^t$, we have $\Pr[\mathcal{V} = z] \leq M^{-(1-\delta)\cdot t}$ and $\Pr[\mathcal{U} = z] = M^{-t}$. Hence, we have:

$$\mathsf{SD}(\mathcal{U}, \mathcal{V}) \leq 1 - M^{-\delta \cdot t} \leq t \cdot \delta \cdot \log M \ ,$$

where the last inequality uses the fact that for all $x \geq 0$, it holds that $2^{-x} \geq 1 - x$ (and, therefore, $x \geq 1 - 2^{-x}$). □

The following theorem states the indifferentiability of the xor combiner.

Theorem 7.4 (Indifferentiability of xor in 2-BRO with bounded adaptivity)**.** *Consider the xor combiner $\mathsf{C}_{\oplus}^{\mathsf{H}_1,\mathsf{H}_2}(x) := \mathsf{H}_1(x) \oplus \mathsf{H}_2(x)$ in the 2-BRO model with backdoored hash functions $\mathsf{H}_1, \mathsf{H}_2 \in [M]^N$. It holds that for any $\bar{p} := (p_1, \ldots, p_{c+1}) \in \mathbb{N}^{c+1}$, $0 < \gamma < 1$, and an integer $c \geq 0$, there exists a simulator $\mathsf{Sim}[\bar{p}, \gamma] := (\mathsf{SimH}_1^{\mathsf{RO}}, \mathsf{SimH}_2^{\mathsf{RO}}, \mathsf{SimBD}_1^{\mathsf{RO}}[\bar{p}, \gamma], \mathsf{SimBD}_2^{\mathsf{RO}}[\bar{p}, \gamma])$ such that for any differentiator $\mathcal{D}$ that always makes Q consecutive queries to a backdoor oracle (starting from BD_1 and always receiving an ℓ-bit response) before switching to the other, with a total number of c switches,* *while being allowed to make up to* q_{H} *primitive queries as well as* q_{C} *construction queries, we have*

$$\begin{aligned}\mathsf{Adv}^{\text{indiff}}_{\mathsf{C}_{\oplus}^{\mathsf{H}_1,\mathsf{H}_2},\mathsf{Sim}[\bar{p},\gamma]}(\mathcal{D}) &\leq (c+1)\cdot\gamma \\ &\qquad + \log M \cdot \Big(\sum_{i=1}^{c} p_i \cdot \delta_{i-1} + q_{\mathsf{H}} \cdot \delta_{c+1} + q_{\mathsf{C}} \cdot (\delta_c + \delta_{c+1})\Big) \ ,\end{aligned}$$

where $\delta_{-1} := \delta_0 := 0$ *and the density rate after the i-th sequence of Q-many backdoor queries is* $\delta_i := \big(\delta_{i-2} \cdot (N - \sum_{j=1}^{i-2} p_j) \cdot \log M + Q \cdot \ell + \log \gamma^{-1}\big) / \big(p_i \cdot \log M\big)$*. The simulator places at most* $q_{\mathsf{Sim}} \leq q_{\mathsf{H}} + \sum_{i=1}^{c+1} p_i$ *queries to the random oracle* RO*.*

Proof. We prove indifferentiability by (1) defining a simulator, (2) upper bounding the advantage of any differentiator in distinguishing the real and the simulated worlds, and (3) upper bounding the number of queries that the simulator makes to the random oracle.

Simulator. All four sub-algorithms of the simulator are described in Figure 7.1. They are parts of the same simulator and, therefore, share state. In particular they share variables to keep track of the fixed history and the current distributions. Two sets $\mathsf{hst}_1, \mathsf{hst}_2$ are used to keep track of the fixed coordinates of the simulated BROs H_1 and H_2, respectively. The density functions, from which the simulated BROs will be sampled, are denoted by μ_1 and μ_2. Furthermore, the simulator uses a counter s to recognize switches from one backdoor oracle to the other in order to use the appropriate number of points to fix from the list $\bar{p}$. It also maintains a counter q for counting the number of consecutive queries to a backdoor oracle in order to decompose, i.e., substitute the current distribution with a partially fixed and partially dense distribution, only when necessary which is the case after each set of Q backdoor queries. We assume the initial values $\mu_1 = \mu_2 := \mathcal{U}_{[M]^N}$, $\mathsf{H}_1, \mathsf{H}_2 \twoheadleftarrow \mathcal{U}_{[M]^N}$, $\mathsf{hst}_1 = \mathsf{hst}_2 = \mathsf{hst}_{\mathsf{RO}} := \emptyset$, $q := 0$, and $s := 0$.

Security analysis. Next we analyze the indifferentiability of the xor combiner with respect to our simulator. For this purpose we use a sequence of eight cryptographic games $\mathsf{Game}_0, \ldots, \mathsf{Game}_7$. The starting game Game_0 is the *real* game, where the adversary has access to the real oracles $\mathsf{C}_{\oplus}^{\mathsf{H}_1,\mathsf{H}_2}$, H_1, H_2, BD_1, and BD_2. The final game is the *ideal* game, where the adversary has access to the oracles RO, $\mathsf{SimH}_1^{\mathsf{RO}}$, $\mathsf{SimH}_2^{\mathsf{RO}}$, $\mathsf{SimBD}_1^{\mathsf{RO}}$, and $\mathsf{SimBD}_2^{\mathsf{RO}}$.

We define the intermediate games Game_1 through Game_6 by gradually modifying the oracles and highlighting the changes in each step. Unchanged oracles are omitted in the description of these games and correspond to those from their direct predecessor. In each step, we bound the advantage of differentiators in distinguishing every two consecutive games. In what follows we use the shorthand notation $\Pr[\mathcal{D}^{\mathsf{Game}_i}] := \Pr[\mathcal{D}^{\mathsf{Game}_i} = 1]$, where $\mathcal{D}^{\mathsf{Game}_i}$ denotes the interaction of an adversary $\mathcal{D}$ (respecting the assumptions put by the theorem) with a game Game_i.

```
$\mathsf{Game}_0 : \mathsf{C}_{\oplus}^{\mathsf{H}_1,\mathsf{H}_2}(x)$
$y_1 \leftarrow \mathsf{H}_1(x);\ y_2 \leftarrow \mathsf{H}_2(x)$
$y \leftarrow y_1 \oplus y_2$
return $y$

$\mathsf{Game}_0 : \mathsf{H}_1(x)$
$y_1 \leftarrow \mathsf{H}_1(x)$
return $y_1$

$\mathsf{Game}_0 : \mathsf{H}_2(x)$
$y_2 \leftarrow \mathsf{H}_2(x)$
return $y_2$

$\mathsf{Game}_0 : \mathsf{BD}_1(f)$
$z \leftarrow f(\mathsf{H}_1)$
return $z$

$\mathsf{Game}_0 : \mathsf{BD}_2(f)$
$z \leftarrow f(\mathsf{H}_2)$
return $z$
```

Game_1. We next update the distributions of hash functions based on past evaluation queries, backdoor queries, and the history of coordinates that have been fixed through construction queries. The distributions μ_i are conditioned on these updates but are never actually used (i.e., sampled from) anywhere in the game. Thus, it is easy to see that these two games are actually identical, i.e., we have

$$\mathsf{SD}(\mathsf{Game}_0, \mathsf{Game}_1) = 0\ .$$

```
$\mathsf{Game}_1 : \mathsf{C}_{\oplus}^{\mathsf{H}_1,\mathsf{H}_2}(x)$
$y_1 \leftarrow \mathsf{H}_1(x);\ y_2 \leftarrow \mathsf{H}_2(x)$
$\boxed{\mathsf{hst}_1 \leftarrow \mathsf{hst}_1 \cup \{(x, y_1)\}}$
$\boxed{\mathsf{hst}_2 \leftarrow \mathsf{hst}_2 \cup \{(x, y_2)\}}$
$\boxed{\mu_1 \leftarrow \mu_1|_{\mathsf{hst}_1}};\ \boxed{\mu_2 \leftarrow \mu_2|_{\mathsf{hst}_2}}$
$y \leftarrow y_1 \oplus y_2$
return $y$

$\mathsf{Game}_1 : \mathsf{H}_1(x)$
$y_1 \leftarrow \mathsf{H}_1(x)$
$\boxed{\mathsf{hst}_1 \leftarrow \mathsf{hst}_1 \cup \{(x, y_1)\}}$
$\boxed{\mu_1 \leftarrow \mu_1|_{\mathsf{hst}_1}}$
return $y_1$

$\mathsf{Game}_1 : \mathsf{H}_2(x)$
$y_2 \leftarrow \mathsf{H}_2(x)$
$\boxed{\mathsf{hst}_2 \leftarrow \mathsf{hst}_2 \cup \{(x, y_2)\}}$
$\boxed{\mu_2 \leftarrow \mu_2|_{\mathsf{hst}_2}}$
return $y_2$

$\mathsf{Game}_1 : \mathsf{BD}_1(f)$
$z \leftarrow f(\mathsf{H}_1)$
$\boxed{\mu_1 \leftarrow \mu_1|_{f(\cdot)=z}}$
return $z$

$\mathsf{Game}_1 : \mathsf{BD}_2(f)$
$z \leftarrow f(\mathsf{H}_2)$
$\boxed{\mu_2 \leftarrow \mu_2|_{f(\cdot)=z}}$
return $z$
```

Game_2. Here, after each sequence of Q queries to a backdoor oracle, i.e., right before a switch, a $(p, 1-\delta)$-dense distribution μ_i' is obtained using the algorithm $\mathsf{FixRows}$ by decomposing the distribution of the corresponding hash function after responding to the last query in the sequence (i.e., $\mu_i|_{f(\cdot)=z}$). However, since the new distributions μ_i' are never actually used elsewhere, Game_2 remains identical to Game_1, i.e., it holds that

$$\mathsf{SD}(\mathsf{Game}_1, \mathsf{Game}_2) = 0\ .$$

```
$\mathsf{Game}_2 : \mathsf{BD}_1(f)$
$\boxed{q \leftarrow q+1}$
$z \leftarrow f(\mathsf{H}_1);\ \mu_1 \leftarrow \mu_1|_{f(\cdot)=z}$
$\boxed{\textbf{if } q = Q \textbf{ then}}$
    $\boxed{(\mu'_1, A_1) \twoheadleftarrow \mathsf{FixRows}[\gamma](\mu_1, p_{2s+1}, \mathsf{hst}_{1.1})}$
    $\boxed{q \leftarrow 0}$
$\textbf{return } z$
```

```
$\mathsf{Game}_2 : \mathsf{BD}_2(f)$
$\boxed{q \leftarrow q+1}$
$z \leftarrow f(\mathsf{H}_2);\ \mu_2 \leftarrow \mu_2|_{f(\cdot)=z}$
$\boxed{\textbf{if } q = Q \textbf{ then}}$
    $\boxed{(\mu'_2, A_2) \twoheadleftarrow \mathsf{FixRows}[\gamma](\mu_2, p_{2s+2}, \mathsf{hst}_{2.1})}$
    $\boxed{q \leftarrow 0}$
    $\boxed{s \leftarrow s+1}$
$\textbf{return } z$
```

Game_3**.** In this game evaluation queries on a value x, fix its images under both functions, i.e., to $\mathsf{H}_1(x)$ and $\mathsf{H}_2(x)$. Similarly, in backdoor simulation, for each x-coordinate in the set of assignments A_1 (resp. A_2), images of the same x are fixed under the other hash function H_2 (resp. H_1) according to its current distribution. In both games, the oracles' responses are at all times consistent with their past responses (and with the construction). Moreover, we still do not sample from the updated distributions. Overall it does not matter if more or less of the hash function tables are fixed in each query and, therefore, the two games are identical, i.e., we obtain

$$\mathsf{SD}(\mathsf{Game}_2, \mathsf{Game}_3) = 0 .$$

```
$\mathsf{Game}_3 : \mathsf{H}_1(x)$
$y_1 \leftarrow \mathsf{H}_1(x)$
$\mathsf{hst}_1 \leftarrow \mathsf{hst}_1 \cup \{(x, y_1)\}$
$\boxed{\mathsf{hst}_2 \leftarrow \mathsf{hst}_2 \cup \{(x, \mathsf{H}_2(x))\}}$
$\mu_1 \leftarrow \mu_1|_{\mathsf{hst}_1}$
$\boxed{\mu_2 \leftarrow \mu_2|_{\mathsf{hst}_2}}$
$\textbf{return } y_1$
```

```
$\mathsf{Game}_3 : \mathsf{H}_2(x)$
$y_2 \leftarrow \mathsf{H}_2(x)$
$\mathsf{hst}_2 \leftarrow \mathsf{hst}_2 \cup \{(x, y_2)\}$
$\boxed{\mathsf{hst}_1 \leftarrow \mathsf{hst}_1 \cup \{(x, \mathsf{H}_1(x))\}}$
$\mu_2 \leftarrow \mu_2|_{\mathsf{hst}_2}$
$\boxed{\mu_1 \leftarrow \mu_1|_{\mathsf{hst}_1}}$
$\textbf{return } y_2$
```

```
$\mathsf{Game}_3 : \mathsf{BD}_1(f)$
$q \leftarrow q+1$
$z \leftarrow f(\mathsf{H}_1);\ \mu_1 \leftarrow \mu_1|_{f(\cdot)=z}$
$\textbf{if } q = Q \textbf{ then}$
    $(\mu'_1, A_1) \twoheadleftarrow \mathsf{FixRows}[\gamma](\mu_1, p_{2s+1}, \mathsf{hst}_{1.1})$
    $\boxed{\textbf{for } x \in A_{1.1} \textbf{ do}}$
        $\boxed{\mathsf{hst}_2 \leftarrow \mathsf{hst}_2 \cup \{(x, \mathsf{H}_2(x))\}}$
    $\boxed{\mu_2 \leftarrow \mu_2|_{\mathsf{hst}_2}}$
    $q \leftarrow 0$
$\textbf{return } z$
```

```
$\mathsf{Game}_3 : \mathsf{BD}_2(f)$
$q \leftarrow q+1$
$z \leftarrow f(\mathsf{H}_2);\ \mu_2 \leftarrow \mu_2|_{f(\cdot)=z}$
$\textbf{if } q = Q \textbf{ then}$
    $(\mu'_2, A_2) \twoheadleftarrow \mathsf{FixRows}[\gamma](\mu_2, p_{2s+2}, \mathsf{hst}_{2.1})$
    $\boxed{\textbf{for } x \in A_{2.1} \textbf{ do}}$
        $\boxed{\mathsf{hst}_1 \leftarrow \mathsf{hst}_1 \cup \{(x, \mathsf{H}_1(x))\}}$
    $\boxed{\mu_1 \leftarrow \mu_1|_{\mathsf{hst}_1}}$
    $q \leftarrow 0$
    $s \leftarrow s+1$
$\textbf{return } z$
```

Game_4. In this game the distributions obtained by decomposition finally replace the distributions conditioned on leakage. Hence, the histories are also updated and a new H_i is later sampled for potential usage in the construction. According to Lemma 7.1, there is a convex combination of $(p, 1-\delta)$-dense distributions which is γ-close to the real distribution, one of such distributions being the one returned by FixRows as defined in Figure 7.1. Thus, the distinguishing advantage can increase by γ for every Q sequence of backdoor queries, leading to

$$\left|\Pr[\mathcal{D}^{\mathsf{Game}_3}] - \Pr[\mathcal{D}^{\mathsf{Game}_4}]\right| \leq (c+1) \cdot \gamma \ .$$

```
$\mathsf{Game}_4 : \mathsf{BD}_1(f)$
$q \leftarrow q+1$
$z \leftarrow f(\mathsf{H}_1);\ \mu_1 \leftarrow \mu_1|_{f(\cdot)=z}$
if $q = Q$ then
    $\boxed{(\mu_1, A_1) \twoheadleftarrow \mathsf{FixRows}[\gamma](\mu_1, p_{2s+1}, \mathsf{hst}_{1.1})}$
    $\boxed{\mathsf{hst}_1 \leftarrow \mathsf{hst}_1 \cup A_1}$
    $\boxed{\mathsf{H}_1 \twoheadleftarrow \mu_1}$
    for $x \in A_{1.1}$ do
        $\mathsf{hst}_2 \leftarrow \mathsf{hst}_2 \cup \{(x, \mathsf{H}_2(x))\}$
    $\mu_2 \leftarrow \mu_2|_{\mathsf{hst}_2}$
    $q \leftarrow 0$
return $z$
```

```
$\mathsf{Game}_4 : \mathsf{BD}_2(f)$
$q \leftarrow q+1$
$z \leftarrow f(\mathsf{H}_2);\ \mu_2 \leftarrow \mu_2|_{f(\cdot)=z}$
if $q = Q$ then
    $\boxed{(\mu_2, A_2) \twoheadleftarrow \mathsf{FixRows}[\gamma](\mu_2, p_{2s+2}, \mathsf{hst}_{2.1})}$
    $\boxed{\mathsf{hst}_2 \leftarrow \mathsf{hst}_2 \cup A_2}$
    $\boxed{\mathsf{H}_2 \twoheadleftarrow \mu_2}$
    for $x \in A_{2.1}$ do
        $\mathsf{hst}_1 \leftarrow \mathsf{hst}_1 \cup \{(x, \mathsf{H}_1(x))\}$
    $\mu_1 \leftarrow \mu_1|_{\mathsf{hst}_1}$
    $q \leftarrow 0$
    $s \leftarrow s+1$
return $z$
```

Game_5. This game behaves exactly as Game_4 except when fixing values for the distribution of the other BRO. The game fixes those points by calling $\mathsf{C}_\oplus$ (rather than doing so directly using H_1 resp. H_2) and then it redundantly updates the history of fixed points, e.g., with some $(x, \mathsf{H}_1(x) \oplus \mathsf{C}_\oplus(x))$. Finally, the game samples a new BRO from the updated distribution. However, since the construction $\mathsf{C}_\oplus$ itself calls the BROs, Game_5 is only taking a detour and the two games are perfectly indistinguishable, i.e., we have

$$\mathsf{SD}(\mathsf{Game}_4, \mathsf{Game}_5) = 0 \ .$$

```
$\mathsf{Game}_5 : \mathsf{H}_1(x)$
$y_1 \leftarrow \mathsf{H}_1(x)$
$\mathsf{hst}_1 \leftarrow \mathsf{hst}_1 \cup \{(x, y_1)\}$
$\boxed{\mathsf{hst}_2 \leftarrow \mathsf{hst}_2 \cup \{(x, \mathsf{C}_\oplus(x) \oplus y_1\}}$
$\mu_1 \leftarrow \mu_1|_{\mathsf{hst}_1};\ \mu_2 \leftarrow \mu_2|_{\mathsf{hst}_2}$
$\boxed{\mathsf{H}_2 \twoheadleftarrow \mu_2}$
return $y_1$

$\mathsf{Game}_5 : \mathsf{H}_2(x)$
$y_2 \leftarrow \mathsf{H}_2(x)$
$\mathsf{hst}_2 \leftarrow \mathsf{hst}_2 \cup \{(x, y_2)\}$
$\boxed{\mathsf{hst}_1 \leftarrow \mathsf{hst}_1 \cup \{(x, \mathsf{C}_\oplus(x) \oplus y_2)\}}$
$\mu_2 \leftarrow \mu_2|_{\mathsf{hst}_2};\ \mu_1 \leftarrow \mu_1|_{\mathsf{hst}_1}$
$\boxed{\mathsf{H}_1 \twoheadleftarrow \mu_1}$
return $y_2$

$\mathsf{Game}_5 : \mathsf{BD}_1(f)$
$q \leftarrow q + 1$
$z \leftarrow f(\mathsf{H}_1);\ \mu_1 \leftarrow \mu_1|_{f(\cdot)=z}$
if $q = Q$ then
    $(\mu_1, A_1) \twoheadleftarrow \mathsf{FixRows}[\gamma](\mu_1, p_{2s+1}, \mathsf{hst}_{1.1})$
    $\mathsf{hst}_1 \leftarrow \mathsf{hst}_1 \cup A_1$
    $\mathsf{H}_1 \twoheadleftarrow \mu_1$
    $\boxed{\textbf{for } (x, y_1) \in A_1 \textbf{ do}}$
        $\boxed{\mathsf{hst}_2 \leftarrow \mathsf{hst}_2 \cup \{(x, \mathsf{C}_\oplus(x) \oplus y_1)\}}$
    $\mu_2 \leftarrow \mu_2|_{\mathsf{hst}_2}$
    $\boxed{\mathsf{H}_2 \twoheadleftarrow \mu_2}$
    $q \leftarrow 0$
return $z$

$\mathsf{Game}_5 : \mathsf{BD}_2(f)$
$q \leftarrow q + 1$
$z \leftarrow f(\mathsf{H}_2);\ \mu_2 \leftarrow \mu_2|_{f(\cdot)=z}$
if $q = Q$ then
    $(\mu_2, A_2) \twoheadleftarrow \mathsf{FixRows}[\gamma](\mu_2, p_{2s+2}, \mathsf{hst}_{2.1})$
    $\mathsf{hst}_2 \leftarrow \mathsf{hst}_2 \cup A_2$
    $\mathsf{H}_2 \twoheadleftarrow \mu_2$
    $\boxed{\textbf{for } (x, y_2) \in A_2 \textbf{ do}}$
        $\boxed{\mathsf{hst}_1 \leftarrow \mathsf{hst}_1 \cup \{(x, \mathsf{C}_\oplus(x) \oplus y_2)\}}$
    $\mu_1 \leftarrow \mu_1|_{\mathsf{hst}_1}$
    $\boxed{\mathsf{H}_1 \twoheadleftarrow \mu_1}$
    $q \leftarrow 0$
    $s \leftarrow s + 1$
return $z$
```

Game_6**.** This is our last intermediate game before reach the ideal world. Here we modify $\mathsf{C}_\oplus$ to start to resemble a lazily sampled random oracle. In the new construction oracle, a query is stored together with its image in the history $\mathsf{hst}_{\mathsf{RO}}$. In case a query is repeated, its stored image is simply returned. Otherwise, there are three cases to consider: an image for the current query x is fixed in both hash functions, in one of them, or in neither one. In the first case the output of the construction is computed by xoring the individual images stored in hst_1 and hst_2. In the second case, a uniformly random value is chosen (and stored in $\mathsf{hst}_{\mathsf{RO}}$). In the final case, Game_6 behaves exactly as Game_5. So, the distinguishing advantage is bounded by distinguishing uniform points (set to uniform when xoring with the returned uniform value of $\mathsf{C}_\oplus$) from dense points. In fact, according to Lemma 7.3, for each evaluation query it adds at most $\delta_{c+1} \cdot \log M$, since δ_{c+1} is the largest δ_i. Moreover, for all points that are fixed upon a backdoor query this adds $p_i \cdot \delta_{i-1} \cdot \log M$, except for the last one, since there will be no backdoor query after that which can see all p_{c+1} points. Overall we obtain

$$\big|\Pr[\mathcal{D}^{\mathsf{Game}_5}] - \Pr[\mathcal{D}^{\mathsf{Game}_6}]\big| \leq \log M \cdot \Big(\sum_{i=1}^{c} p_i \cdot \delta_{i-1} + q_{\mathsf{H}} \cdot \delta_{c+1}\Big) \ .$$

```
$\mathsf{Game}_6 : \mathsf{C}_{\oplus}^{\mathsf{H}_1,\mathsf{H}_2}(x)$
if $\exists y \in [M]$ s.t. $(x, y) \in \mathsf{hst}_{\mathsf{RO}}$ then return $y$
if $\exists y_1, y_2 \in [M]$ s.t. $(x, y_1) \in \mathsf{hst}_1 \wedge (x, y_2) \in \mathsf{hst}_2$ then return $y_1 \oplus y_2$
if $\exists y' \in [M]$ s.t. $(x, y') \in \mathsf{hst}_1 \vee (x, y') \in \mathsf{hst}_2$ then
    $y \twoheadleftarrow [M]$
else
    $y_1 \leftarrow \mathsf{H}_1(x)$; $y_2 \leftarrow \mathsf{H}_2(x)$
    $\mathsf{hst}_1 \leftarrow \mathsf{hst}_1 \cup \{(x, y_1)\}$; $\mathsf{hst}_2 \leftarrow \mathsf{hst}_2 \cup \{(x, y_2)\}$
    $\mu_1 \leftarrow \mu_1|_{\mathsf{hst}_1}$; $\mu_2 \leftarrow \mu_2|_{\mathsf{hst}_2}$
    $y \leftarrow y_1 \oplus y_2$
$\mathsf{hst}_{\mathsf{RO}} \leftarrow \mathsf{hst}_{\mathsf{RO}} \cup \{(x, y)\}$
return $y$
```

Game_7. The construction oracle in our final game differs from Game_6 in that it never evaluates the individual hash functions anymore (in particular, not even if x is in neither history). We can safely remove the second case distinction, where x is in both hst_1 and hst_2, since this case is covered by the first case where x has been queried to the construction itself. With these modifications the construction oracle becomes a lazily sampled random oracle. It remains to bound the adversary's advantage in distinguishing the two games while making fresh queries x to the construction oracle that are prior to the query fixed for neither hash function.

Lemma 7.5. *Let X and Y be two independent $(1-\delta)$ and $(1-\delta')$-dense distributions over a domain $[M]^N$. Then the xor distribution $X \oplus Y$ is $(1-(\delta+\delta'))$-dense over the same domain $[M]^N$.*

Proof. Let $I \subseteq [N]$ and $z \in [M]^{|I|}$ be arbitrary. Then we have:

$$\begin{aligned}\Pr[X_I \oplus Y_I = z] &= \sum_x \Pr[X_I = x \wedge Y_I = x \oplus z] = \sum_x \Pr[X_I = x] \cdot \Pr[Y_I = x \oplus z] \\ &\leq 2^{|I| \cdot \log M} \cdot 2^{-(1-\delta)\cdot|I|\cdot\log M} \cdot 2^{-(1-\delta')\cdot|I|\cdot\log M} = 2^{-(1-(\delta+\delta'))\cdot|I|\cdot\log M} \ .\end{aligned}$$

□

We can now bound the distinguisher's advantage by computing the distance between the sum of two dense distributions from uniform, given that only q_{C} queries to $\mathsf{C}_{\oplus}$ are allowed. Below, in the second line, we use the fact that according to Lemma 7.1, δ's should increase.

$$\begin{aligned}\left|\Pr[\mathcal{D}^{\mathsf{Game}_6}] - \Pr[\mathcal{D}^{\mathsf{Game}_7}]\right| &\leq q_{\mathsf{C}} \cdot \log M \cdot \max_{0 \leq i \leq c} \{\delta_i + \delta_{i+1}\} \\ &= q_{\mathsf{C}} \cdot \log M \cdot (\delta_c + \delta_{c+1}) \ .\end{aligned}$$

```
$\mathsf{Game}_7 : \mathsf{C}_{\oplus}^{\mathsf{H}_1,\mathsf{H}_2}(x)$
if $\exists y \in [M]$ s.t. $(x, y) \in \mathsf{hst}_{\mathsf{RO}}$ then return $y$
~~if $\exists y_1, y_2 \in [M]$ s.t. $(x, y_1) \in \mathsf{hst}_1 \wedge (x, y_2) \in \mathsf{hst}_2$ then return $y_1 \oplus y_2$~~
~~if $\exists y' \in [M]$ s.t. $(x, y') \in \mathsf{hst}_1 \vee (x, y') \in \mathsf{hst}_2$ then~~
  $y \twoheadleftarrow [M]$
~~else~~
  ~~$y_1 \leftarrow \mathsf{H}_1(x);\ y_2 \leftarrow \mathsf{H}_2(x)$~~
  ~~$\mathsf{hst}_1 \leftarrow \mathsf{hst}_1 \cup \{(x, y_1)\};\ \mathsf{hst}_2 \leftarrow \mathsf{hst}_2 \cup \{(x, y_2)\}$~~
  ~~$\mu_1 \leftarrow \mu_1|_{\mathsf{hst}_1};\ \mu_2 \leftarrow \mu_2|_{\mathsf{hst}_2}$~~
  ~~$y \leftarrow y_1 \oplus y_2$~~
$\mathsf{hst}_{\mathsf{RO}} \leftarrow \mathsf{hst}_{\mathsf{RO}} \cup \{(x, y)\}$
return $y$
```

The last game Game_7 is identical to the simulated world. Therefore, the overall advantage of $\mathcal{D}$ is as stated in the theorem.

Query complexity. The queries made by the simulator to RO consist of those made when simulating evaluation oracles and those made when simulating backdoor oracles. Responding to each evaluation query requires exactly one query to RO, which makes a total of q_{H} queries. Right after the Q-th consecutive backdoor query (i.e., right before a switch), the simulator fixes some points of the other BRO, where for each fixed point one query to the random oracle RO is made. The maximum number of points that should be fixed after each sequence of Q queries to BD_1 or BD_2 is predetermined by the simulator's parameter $\bar{p} := (p_1, \ldots, p_{c+1})$. Hence, we obtain the following bound on the query complexity of the simulator.

$$q_{\mathsf{Sim}} \leq q_{\mathsf{H}} + \sum_{i=1}^{c+1} p_i \ .$$

□

We now provide estimates for the involved parameters.

Corollary 7.6. *Let the number of switches be $c \geq 1$. Then for any $\alpha_1 > 1 - 1/F_{c+1}$, where F_i are the Fibonacci numbers, there is an indifferentiability simulator Sim for the $\mathsf{C}_{\oplus}$ construction in the 2-BRO model which has query complexity $q_{\mathsf{H}} + (c+1) \cdot N^{\alpha_1}$ for any distinguisher with q_{H} queries to the underlying BROs. Furthermore, any such distinguisher which places q_{C} construction queries and Q consecutive queries to the same backdoor oracle before switching has advantage at most*

$$(c+1) \cdot \gamma + \log M \cdot (c^2 B + 2q_{\mathsf{H}} + 2q_{\mathsf{C}}) \cdot N^{(1-\alpha_1) \cdot F_{c+1}/F_{c+2} - 1/F_{c+2}} \ ,$$

against the simulator, where $B := (Q\ell + \log \gamma^{-1})/\log M$. Asymptotically the query complexity is $q_{\mathsf{H}} + \mathcal{O}(N^{1-1/F_{c+2}})$ and the advantage $\mathcal{O}((q_{\mathsf{H}} + q_{\mathsf{C}}) \cdot Q \cdot \ell / N^{0.38/F_{c+2}})$.

Proof. From Lemma 7.1 we have that

$$\delta_i \leq (\delta_{i-2} \cdot A + B)/p_i \ ,$$

where $A := N$ and $B := (Q\ell + \log \gamma^{-1})/\log M$. Recursively applying the equation we get for odd i

$$\delta_i \leq \frac{B}{p_i} + \frac{AB}{p_i p_{i-2}} + \cdots + \frac{A^{(i-1)/2}B}{p_i p_{i-2} \cdots p_1}$$

Using $p_i < A$, the terms progressively get larger. Thus, in general

$$\delta_i \leq \frac{c \cdot N^{(i-2+i \bmod 2)/2}B}{p_i p_{i-2} \cdots p_{1+(i+1) \bmod 2}} \ .$$

For the indifferentiability advantage to be small, we would need to minimize

$$\sum_{i=1}^{c} p_i \cdot \delta_{i-1} + (q_{\mathsf{H}} + q_{\mathsf{C}})(\delta_c + \delta_{c+1}).$$

Let's assume $p_i = N^{\alpha_i}$ for some $\alpha_i \in [0, 1)$. Then the i-th summand for $i > 1$ is

$$c \cdot B \cdot N^{\alpha_i - \alpha_{i-1} - \alpha_{i-3} - \cdots - \alpha_{1+i \bmod 2} + (i-3+(i-1) \bmod 2)/2} \ .$$

To minimize, we set all terms equal to a common value $c \cdot B \cdot N^{\theta}$. We obtain

$$\alpha_i - \alpha_{i-1} - \ldots - \alpha_{1+i \bmod 2} + (i - 3 + (i-1) \bmod 2)/2 = \theta \ ,$$

Solving this system of linear equations gives

$$\alpha_i = F_i \cdot \theta + F_{i-1} \cdot (\alpha_1 - 1) + 1 \ ,$$

where F_i are the Fibonacci numbers with $F_0 = 0$ and $F_1 = 1$.

We may arrange the terms so that $(\delta_c + \delta_{c+1}) = 2 \cdot N^{\theta}$ (not including the $(q_{\mathsf{H}} + q_{\mathsf{C}})$ factor). To this end, we set $\alpha_{c+2} = 0$ so that $\delta_{c+1} = N^{\theta}/p_{c+2} = N^{\theta}$ and $\delta_c = N^{\theta}/p_{c+1} \leq N^{\theta}/p_{c+2} = N^{\theta}$. Thus we set $\alpha_{c+2} = 0$. This gives $\theta = (1 - \alpha_1) \cdot F_{c+1}/F_{c+2} - 1/F_{c+2}$. Now for $\theta < 0$ we would need that $\alpha_1 > 1 - 1/F_{c+1}$. This means that the query complexity of the simulator is $q_{\mathsf{H}} + (c+1) \cdot N^{\alpha_1}$ and its advantage is

$$(c+1) \cdot \gamma + \log M \cdot (c^2 B + 2q_{\mathsf{H}} + 2q_{\mathsf{C}}) \cdot N^{(1-\alpha_1) \cdot F_{c+1}/F_{c+2} - 1/F_{c+2}} \ .$$

We obtain the bound stated in the asymptotic part of the corollary by setting $\alpha_1 := 1 - 1/F_{c+2} > 1 - 1/F_{c+1}$. □

We note that in the special case where $c = 1$, we must have that $\alpha_1 > 1 - 1/F_2 = 0$. In particular we can set $\alpha_1 := 1/4$ to obtain a simulator that places $N^{\alpha_1} = N^{1/4} \leq \sqrt{N}$ queries. Thus in this case we obtain collision resistance. Note, however, that as soon as $c \geq 2$ we would need to have that

$\alpha_1 > 1 - 1/F_3 = 1/2$, which means the simulator places at least $\sqrt{N}$ queries, and we do not get collision resistance.

The above corollary shows that the xor combiner can only tolerate a logarithmic number of switches in $\log N$, which we think of as the security parameter. This is due to the fact that the simulator complexity needs to be less than $N/2$ for it to be non-trivial. Although our bounds are arguably weak, they are still meaningful, and we conjecture that much better bounds in reality hold.

7.4 Indifferentiability of 2-out-of-3-Source Extractors in the 3-BRO Model

In this section we study the indifferentiability of extractor-based combiners and show that they can give better security parameters compared to the xor combiner of Section 7.3. Recall that the k-BRO model considers adversaries that have access to all k backdoor oracles. A query to the backdoor oracle BD_i reveals some information about the underlying BRO H_i. The resulting distribution of H_i conditioned on the leakage can, using the refined decomposition technique of Section 7.2, be translated into a distribution on functions that have a number of fixed coordinates, while the distribution of the other coordinates remains dense. An indifferentiability simulator then fixes some rows in the function table of the other BRO(s) in a way that consistency with the random oracle RO (which is to be indistinguishable from the construction) is ensured.

We demonstrated this idea in the previous section for the xor combiner, where before a switch to the other backdoor oracle, the simulator always substituted p images of that BRO by uniformly random values, more precisely, the RO images xored with the images just fixed. According to Lemma 7.3, this replacement causes a security loss of $p \cdot \delta \cdot \log M$ per switch, for some δ computed according to Lemma 7.1. This loss corresponds to the advantage of an adversary distinguishing p uniform values from p $(1-\delta)$-dense ones.

Multi-source extractors as combiners. Now consider a multi-source $(k_1, \ldots, k_t, \varepsilon)$-extractor as a combiner in t-BRO. The hope in using such an extractor is that as long as the images of the BROs have high min-entropy, the output of the extractor is ε-close to uniform. This makes it possible for us to express the loss described above, caused by adjusting images in BROs in order to assure consistency with RO, in terms of a negligible value ε and forgo the requirement on δ to be negligible. Overall, such a combiner requires fixing considerably less points, since δ's (recall that they have some p in their denominator) do not need to be negligible. Indeed we can show that a multi-source extractor can tolerate up to a linear number of switched between the backdoor queries, giving hope to achieve indifferentiability with unbounded number of switches in the future.

We focus on 2-out-of-3-source extractors, i.e., extractors that only require a minimal amount of min-entropy from two of the sources. More formally, let $\mathsf{Ext} : [M]^3 \to [2]$ be a 2-out-of-3-source $(k_1, k_2, k_3, \varepsilon)$-extractor. For three functions $\mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3 : [N] \to [M]$, the combiner $\mathsf{C}_{2/3\mathrm{ext}}^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_3} : [N] \to [2]$ is defined as:

$$\mathsf{C}_{2/3\mathrm{ext}}^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_3}(x) := \mathsf{Ext}\big(\mathsf{H}_1(x), \mathsf{H}_2(x), \mathsf{H}_3(x)\big) \ .$$

We show in this section that $\mathsf{C}_{2/3\mathrm{ext}}^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_3}$ in the 3-BRO model is indifferentiable from a random oracle. For some background on randomness extractors we refer the reader to Section 2.3.2.

Why not a two-source extractor? Note that we cannot guarantee that images which are being fixed by the simulator in some H_i as a result of a BD_i-query have *any* min-entropy whatsoever. To understand why, simply consider an adversary that makes a backdoor query to BD_1 requesting a preimage of the zero-string $y^* := 0^{\log M}$ under the function H_1. Suppose BD_1 responds to this query with $x^* \in [N]$. In this case $\mathsf{H}_1(x^*)$ has no min-entropy, since $y^* = \mathsf{H}_1(x^*)$ was chosen by the adversary and is, therefore, completely predictable. Hence, $\mathsf{H}_1(x^*)$ cannot be used in a (k_1, k_2, ε)-two-source extractor, i.e., $\mathsf{Ext}(\mathsf{H}_1(x^*), \mathsf{H}_2(x^*))$, which relies on min-entropy from both sources in order to output a string that is ε-close to uniform. In other words, the output of a combiner based on a two-source extractor is not ε-close to the output of a random oracle RO. Overall, using a two-source extractor does not seem to have any advantage over the xor combiner in 2-BRO. On the contrary, when using a 2-out-of-3-source extractor, assuming that the rows under consideration are not already fixed in the function tables of all three BROs due to some previous query, there will be two images with high min-entropy, from which we can extract a value ε-close to uniform.

Theorem 7.7 (Indifferentiability of 2-out-of-3-source extractors in 3-BRO with bounded adaptivity)**.** *Let* $\mathsf{Ext} : [M]^3 \to [2]$ *be a* $(k_1, k_2, k_3, \varepsilon)$*-2-out-of-3-source randomness extractor, where* ε *is a function of* k_1, k_2, k_3*. Consider the combiner* $\mathsf{C}_{3\mathrm{ext}}^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_3}(x) := \mathsf{Ext}(\mathsf{H}_1(x), \mathsf{H}_2(x), \mathsf{H}_3(x))$ *in the 3-BRO model with backdoored hash functions* $\mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3 \in [M]^N$*. It holds that for all values of* $\bar{p} := (p_1, \dots, p_{c+1}) \in \mathbb{N}^{c+1}$*,* $0 < \gamma < 1$*, and an integer* $c \geq 0$*, there exists a simulator* $\mathsf{Sim}[\bar{p}, \gamma] := (\mathsf{SimH}_1^{\mathsf{RO}}, \mathsf{SimH}_2^{\mathsf{RO}}, \mathsf{SimH}_3^{\mathsf{RO}}, \mathsf{SimBD}_1^{\mathsf{RO}}[\bar{p}, \gamma], \mathsf{SimBD}_2^{\mathsf{RO}}[\bar{p}, \gamma], \mathsf{SimBD}_3^{\mathsf{RO}}[\bar{p}, \gamma])$ *such that for any differentiator* $\mathcal{D}$ *that always makes* Q *consecutive queries to one backdoor oracle (always receiving an* ℓ*-bit response) before switching to the next, with a total number of* c *switches, while making up to* q_{H} *primitive queries and* q_{C} *construction queries, we have*

$$\begin{aligned}\mathsf{Adv}^{\mathrm{indiff}}_{\mathsf{C}_{3\mathrm{ext}}^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_3}, \mathsf{Sim}[\bar{p},\gamma]}(\mathcal{D}) \leq (c+1) \cdot \gamma \\ + \sum_{i=1}^{c} \mathsf{SD}\big(E_1 | \cdots | E_{p_i}, \mathcal{U}_{[2]^{p_i}}\big) + q_{\mathsf{H}} \cdot \mathsf{SD}\big(E_1, \mathcal{U}_{[2]}\big) \\ + q_{\mathsf{C}} \cdot \varepsilon\big((1-\delta_{c-1}) \cdot \log M, (1-\delta_c) \cdot \log M, (1-\delta_{c+1}) \cdot \log M\big) \ ,\end{aligned}$$

where for all $n \in \mathbb{N}$*, we define* $E_n := \mathsf{Ext}(X, Y, Z)$ *for some random variables* X, Y, Z *over* $[M]$ *such that at least 2 of them have min-entropy* $(1 - \delta_c) \cdot \log M$*. Furthermore, we let* $\delta_{-2} := \delta_{-1} := \delta_0 := 0$ *and for other values of* $i \leq c+1$ *let* $\delta_i := \big(\delta_{i-3} \cdot (N - \sum_{j=1}^{i-3} p_j) \cdot \log M + Q \cdot \ell + \log \gamma^{-1}\big) / \big(p_i \cdot \log M\big)$ *be the density rate after the* i*-th sequence of* Q*-many backdoor queries. The simulator places at most* $q_{\mathsf{Sim}} \leq q_{\mathsf{H}} + \sum_{i=1}^{c+1} p_i$ *queries to the random oracle* RO*.*

Proof. The proof structure closely follows the proof of Theorem 7.4. We show indifferentiability by (1) defining a simulator, (2) upper bounding the advantage of any differentiator in distinguishing the real world from the simulated world, and (3) upper-bounding the number RO-queries made by the given simulator.

```
$\mathsf{SimH}_i^{\mathsf{RO}}(x)$
$y_i \leftarrow \mathsf{H}_i(x);\ \mathsf{hst}_i \leftarrow \mathsf{hst}_i \cup \{(x, y_i)\}$
$\mu_i \leftarrow \mu_i|_{\mathsf{hst}_i}$
$j \leftarrow (i \bmod 3) + 1;\ k \leftarrow (j \bmod 3) + 1$
$y \leftarrow \mathsf{RO}(x)$
if $i = 1$ then $\mathsf{H}_k \twoheadleftarrow \mu_k|_{\mathsf{Ext}(y_i, \mathsf{H}_j(x), \mathsf{H}_k(x)) = y}$
elseif $i = 2$ then $\mathsf{H}_k \twoheadleftarrow \mu_k|_{\mathsf{Ext}(\mathsf{H}_k(x), y_i, \mathsf{H}_j(x)) = y}$
else $\mathsf{H}_k \twoheadleftarrow \mu_k|_{\mathsf{Ext}(\mathsf{H}_j(x), \mathsf{H}_k(x), y_i) = y}$
$\mathsf{hst}_j \leftarrow \mathsf{hst}_j \cup \{(x, \mathsf{H}_j(x))\}$
$\mathsf{hst}_k \leftarrow \mathsf{hst}_k \cup \{(x, \mathsf{H}_k(x))\}$
$\mu_j \leftarrow \mu_j|_{\mathsf{hst}_j}$
$\mu_k \leftarrow \mu_k|_{\mathsf{hst}_k}$
$\mathsf{H}_k \twoheadleftarrow \mu_k$
return $y_i$
```

```
$\mathsf{RO}(x)$
if $\exists y \in [2]$ s.t. $(x, y) \in \mathsf{hst}_{\mathsf{RO}}$ then
  return $y$
$y \twoheadleftarrow [2]$
$\mathsf{hst}_{\mathsf{RO}} \leftarrow \mathsf{hst}_{\mathsf{RO}} \cup \{(x, y)\}$
return $y$
```

```
$\mathsf{SimBD}_i^{\mathsf{RO}}[\bar{p}, \gamma](f)$
$q \leftarrow q + 1$
$z \leftarrow f(\mathsf{H}_i)$
$\mu_i \leftarrow \mu_i|_{f(\cdot) = z}$
$j \leftarrow (i \bmod 3) + 1$
$k \leftarrow (j \bmod 3) + 1$
if $q = Q$ then
  $(\mu_i, A_i) \twoheadleftarrow \mathsf{FixRows}[\gamma](\mu_i, p_{3s+i}, \mathsf{hst}_{i.1})$
  $\mathsf{H}_i \twoheadleftarrow \mu_i$
  $\mathsf{hst}_i \leftarrow \mathsf{hst}_i \cup A_i$
  for $x \in A_{i.1}$ do $r_x \leftarrow \mathsf{RO}(x)$
  if $i = 1$ then $\mathsf{H}_k \twoheadleftarrow \mu_k|_{\forall (x, y_i) \in A_i.\, \mathsf{Ext}(y_i, \mathsf{H}_j(x), \mathsf{H}_k(x)) = r_x}$
  elseif $i = 2$ then $\mathsf{H}_k \twoheadleftarrow \mu_k|_{\forall (x, y_i) \in A_i.\, \mathsf{Ext}(\mathsf{H}_k(x), y_i, \mathsf{H}_j(x)) = r_x}$
  else $\mathsf{H}_k \twoheadleftarrow \mu_k|_{\forall (x, y_i) \in A_i.\, \mathsf{Ext}(\mathsf{H}_j(x), \mathsf{H}_k(x), y_i) = r_x}$
  for $x \in A_{i.1}$ do
    $\mathsf{hst}_j \leftarrow \mathsf{hst}_j \cup \{(x, \mathsf{H}_j(x))\}$
    $\mathsf{hst}_k \leftarrow \mathsf{hst}_k \cup \{(x, \mathsf{H}_k(x))\}$
  $\mu_j \leftarrow \mu_j|_{\mathsf{hst}_j}$
  $\mu_k \leftarrow \mu_k|_{\mathsf{hst}_k}$
  $\mathsf{H}_k \twoheadleftarrow \mu_k$
  $q \leftarrow 0$
  if $i = 3$ then $s \leftarrow s + 1$
return $z$
```

Figure 7.2: Indifferentiability simulator for the 2-out-of-3-source extractor. We assume for $i = 1..3$ initialization values $\mathsf{hst}_i = \mathsf{hst}_{\mathsf{RO}} := \emptyset$, $\mu_i := \mathcal{U}_{[M]^N}$, $\mathsf{H}_i \twoheadleftarrow \mathcal{U}_{[M]^N}$, $q := 0$, and $s := 0$. The $\mathsf{FixRows}$ algorithm is identical to that of Figure 7.1.

Simulator. The simulator is described in Figure 7.2 by algorithms SimH_i and SimBD_i, where $i \in \{1, 2, 3\}$. The simulator sub-algorithms share state and keep track of the current distribution of the backdoored hash functions. The histories $\mathsf{hst}_1, \mathsf{hst}_2, \mathsf{hst}_3$, initialized as empty sets, are used to keep track of the fixed coordinates of the simulated BROs. The distributions, according to which the simulated backdoored hash functions are sampled, are denoted by μ_1, μ_2, and μ_3 and initialized as $\mathcal{U}_{[M]^N}$, since the hash functions without the backdoors are supposed to behave like random oracles. The corresponding hash functions are initialized as uniform random functions $\mathsf{H}_i \twoheadleftarrow \mathcal{U}_{[M]^N}$. Furthermore, the simulator uses a counter q to keep track of the number of consecutive queries to a backdoor oracle and use this information to decompose the current distribution, only when necessary (i.e., when $q = Q$), as opposed to doing so upon every backdoor query.

Each time images of a simulated H_i are fixed by the simulator BD_i, images of the same rows must be fixed for H_j and H_k (i.e., the other two functions) to provide consistency with the random oracle RO. For this, images of H_j are fixed truthfully according to the currently sampled function, while H_k is tweaked in a way that the extracted values match images of RO. Note that the simulators need to re-sample H_i and H_k if their distribution is modified in a non-trivial way, i.e., not just fixing more values but either through $\mathsf{FixRows}$ or when forcing consistency with RO.

Security Analysis. We analyze indifferentiability of the 3ext-combiner using a sequence of eight games $\mathsf{Game}_0, \ldots, \mathsf{Game}_7$. The starting game Game_0 is the *real* game, where the adversary has access to the real oracles $\mathsf{C}_{2/3\mathrm{ext}}^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_3}$, H_1, H_2, H_3, BD_1, BD_2, and BD_3. The final game Game_7 is our *ideal* game, where the adversary has access to the oracles RO, $\mathsf{SimH}_1^{\mathsf{RO}}$, $\mathsf{SimH}_2^{\mathsf{RO}}$, $\mathsf{SimH}_3^{\mathsf{RO}}$,$\mathsf{SimBD}_1^{\mathsf{RO}}$, $\mathsf{SimBD}_2^{\mathsf{RO}}$, and $\mathsf{SimBD}_3^{\mathsf{RO}}$. We define the intermediate games Game_1 through Game_6 by gradually modifying the oracles. The modified lines in each game are highlighted. Oracles are omitted in some games if they have not changed since the previous game. In each game hop, we bound the adversary's advantage in distinguishing two consecutive games from one another.

```
$\mathsf{Game}_0 : \mathsf{C}_{2/3\mathrm{ext}}^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_3}(x)$
for $i = 1..3$ do
    $y_i \leftarrow \mathsf{H}_i(x)$
$y \leftarrow \mathsf{Ext}(y_1, y_2, y_3)$
return $y$
```

```
$\mathsf{Game}_0 : \mathsf{H}_i(x)$
$y_i \leftarrow \mathsf{H}_i(x)$
return $y_i$
```

```
$\mathsf{Game}_0 : \mathsf{BD}_i(f)$
$z \leftarrow f(\mathsf{H}_i)$
return $z$
```

Game_1. In the first intermediate game, distributions of the simulated hash functions are updated based on evaluation queries, backdoor queries, and the history of coordinates that are fixed through construction queries. The distributions μ_i are conditioned on these values but are never actually sampled from anywhere in the game. Therefore, the two games are identical to any distinguisher, i.e., we have

$$\mathsf{SD}(\mathsf{Game}_0, \mathsf{Game}_1) = 0 \ .$$

```
$\mathsf{Game}_1 : \mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_3}_{2/3\text{ext}}(x)$

for $i = 1..3$ do
    $y_i \leftarrow \mathsf{H}_i(x)$
    $\boxed{\mathsf{hst}_i \leftarrow \mathsf{hst}_i \cup \{(x, y_i)\}}$
    $\boxed{\mu_i \leftarrow \mu_i|_{\mathsf{hst}_i}}$
$y \leftarrow \mathsf{Ext}(y_1, y_2, y_3)$
return $y$
```

```
$\mathsf{Game}_1 : \mathsf{H}_i(x)$

$y_i \leftarrow \mathsf{H}_i(x)$
$\boxed{\mathsf{hst}_i \leftarrow \mathsf{hst}_i \cup \{(x, y_i)\}}$
$\boxed{\mu_i \leftarrow \mu_i|_{\mathsf{hst}_i}}$
return $y_i$
```

```
$\mathsf{Game}_1 : \mathsf{BD}_i(f)$

$z \leftarrow f(\mathsf{H}_i)$
$\boxed{\mu_i \leftarrow \mu_i|_{f(\cdot)=z}}$
return $z$
```

Game_2. In game Game_2, after each sequence of Q queries to a backdoor oracle BD_i, i.e., right before switching to a different one, a $(p, 1-\delta)$-dense distribution μ'_i is obtained from the real distribution using the algorithm $\mathsf{FixRows}$ by decomposing the distribution of the corresponding hash function after responding to the last query in the sequence (i.e., $\mu_i|_{f(\cdot)=z}$). The number of fixed points p is a parameter determined by the simulator and the min-entropy rate δ can be obtained by applying Lemma 7.1. However, since the new distributions μ'_i are never used elsewhere, Game_2 remains identical to the previous Game_1, i.e., we still have

$$\mathsf{SD}(\mathsf{Game}_1, \mathsf{Game}_2) = 0\ .$$

```
$\mathsf{Game}_2 : \mathsf{BD}_i(f)$

$\boxed{q \leftarrow q + 1}$
$z \leftarrow f(\mathsf{H}_i)$
$\mu_i \leftarrow \mu_i|_{f(\cdot)=z}$
$\boxed{\textbf{if } q = Q \textbf{ then}}$
    $\boxed{(\mu'_i, A_i) \twoheadleftarrow \mathsf{FixRows}[\gamma](\mu_i, p_{3s+i}, \mathsf{hst}_{i.1})}$
    $\boxed{q \leftarrow 0}$
    $\boxed{\textbf{if } i = 3 \textbf{ then } s \leftarrow s + 1}$
return $z$
```

Game_3. In this game the fixed rows in one simulated BRO are also fixed for the other two BROs. For instance, in backdoor simulation, the rows in the set of assignment A_i are fixed for H_j and H_k. In both games Game_2 and Game_3, the oracles' behaviors are at all times consistent with their past responses as well as the construction. Hence, it does not matter if more or less of the hash function tables are fixed in each query. The two games are again perfectly indistinguishable, i.e., it holds that

$$\mathsf{SD}(\mathsf{Game}_2, \mathsf{Game}_3) = 0\ .$$

```
$\mathsf{Game}_3 : \mathsf{H}_i(x)$
$y_i \leftarrow \mathsf{H}_i(x)$
$\mathsf{hst}_i \leftarrow \mathsf{hst}_i \cup \{(x, y_i)\}$
$\mu_i \leftarrow \mu_i|_{\mathsf{hst}_i}$
$\boxed{j \leftarrow (i \bmod 3) + 1}$
$\boxed{k \leftarrow (j \bmod 3) + 1}$
$\boxed{\mathsf{hst}_j \leftarrow \mathsf{hst}_j \cup \{(x, \mathsf{H}_j(x))\}}$
$\boxed{\mathsf{hst}_k \leftarrow \mathsf{hst}_k \cup \{(x, \mathsf{H}_k(x))\}}$
$\boxed{\mu_j \leftarrow \mu_j|_{\mathsf{hst}_j}}$
$\boxed{\mu_k \leftarrow \mu_k|_{\mathsf{hst}_k}}$
return $y_i$
```

```
$\mathsf{Game}_3 : \mathsf{BD}_i(f)$
$q \leftarrow q + 1$
$z \leftarrow f(\mathsf{H}_i);\ \mu_i \leftarrow \mu_i|_{f(\cdot)=z}$
$\boxed{j \leftarrow (i \bmod 3) + 1}$
$\boxed{k \leftarrow (j \bmod 3) + 1}$
if $q = Q$ then
    $(\mu'_i, A_i) \twoheadleftarrow \mathsf{FixRows}[\gamma](\mu_i, p_{3s+i}, \mathsf{hst}_{i.1})$
    $\boxed{\textbf{for } x \in A_{i.1} \textbf{ do}}$
        $\boxed{\mathsf{hst}_j \leftarrow \mathsf{hst}_j \cup \{(x, \mathsf{H}_j(x))\}}$
        $\boxed{\mathsf{hst}_k \leftarrow \mathsf{hst}_k \cup \{(x, \mathsf{H}_k(x))\}}$
    $\boxed{\mu_j \leftarrow \mu_j|_{\mathsf{hst}_j}};\ \boxed{\mu_k \leftarrow \mu_k|_{\mathsf{hst}_k}}$
    $q \leftarrow 0$
    if $i = 3$ then $s \leftarrow s + 1$
return $z$
```

Game_4. In Game_4 the distribution obtained by $\mathsf{FixRows}$ finally replaces the true distribution, i.e., the one conditioned on the recent backdoor responses. Therefore, the history is updated. Notably, a new function H_i must be sampled for future references, since its distribution has changed in a non-trivial way. According to Lemma 7.1, there is a convex combination of $(p, 1-\delta)$-dense distributions which is γ-close to the real distribution, one of such distributions being the one returned by $\mathsf{FixRows}$. Thus the distinguishing advantage increases by γ after each sequence of backdoor queries.

$$\left| \Pr[\mathcal{D}^{\mathsf{Game}_3}] - \Pr[\mathcal{D}^{\mathsf{Game}_4}] \right| \leq (c+1) \cdot \gamma \ .$$

```
$\mathsf{Game}_4 : \mathsf{BD}_i(f)$
$q \leftarrow q + 1$
$z \leftarrow f(\mathsf{H}_i);\ \mu_i \leftarrow \mu_i|_{f(\cdot)=z}$
$j \leftarrow (i \bmod 3) + 1;\ k \leftarrow (j \bmod 3) + 1$
if $q = Q$ then
    $\boxed{(\mu_i, A_i) \twoheadleftarrow \mathsf{FixRows}[\gamma](\mu_i, p_{3s+i}, \mathsf{hst}_{i.1})}$
    $\boxed{\mathsf{H}_i \twoheadleftarrow \mu_i}$
    $\boxed{\mathsf{hst}_i \leftarrow \mathsf{hst}_i \cup A_i}$
    for $x \in A_{i.1}$ do
        $\mathsf{hst}_j \leftarrow \mathsf{hst}_j \cup \{(x, \mathsf{H}_j(x))\}$
        $\mathsf{hst}_k \leftarrow \mathsf{hst}_k \cup \{(x, \mathsf{H}_k(x))\}$
    $\mu_j \leftarrow \mu_j|_{\mathsf{hst}_j};\ \mu_k \leftarrow \mu_k|_{\mathsf{hst}_k}$
    $q \leftarrow 0$
    if $i = 3$ then $s \leftarrow s + 1$
return $z$
```

Game_5. Contrary to Game_4, the next game Game_5 somewhat indirectly fixes images of rows x in the set of fresh assignments A_i (and x queries to SimH_i) for the other functions H_j and H_k. More precisely, the simulator calls the construction $\mathsf{C}_{2/3\mathrm{ext}}$ on freshly fixed rows according to A_i and samples a H_k in such a way that it is consistent with those construction images and aligned H_i and H_k images. Notice that a query to the construction already fixes the images for the underlying BROs and, hence, sampling H_k in a consistent way and fixing coordinates of H_j and H_k in the simulator is simply redundant. Therefore, we have

$$\mathsf{SD}(\mathsf{Game}_4, \mathsf{Game}_5) = 0 \ .$$

```
$\mathsf{Game}_5 : \mathsf{H}_i(x)$
$y_i \leftarrow \mathsf{H}_i(x)$
$\mathsf{hst}_i \leftarrow \mathsf{hst}_i \cup \{(x, y_i)\}$
$\mu_i \leftarrow \mu_i|_{\mathsf{hst}_i}$
$j \leftarrow (i \bmod 3) + 1;\ k \leftarrow (j \bmod 3) + 1$
$\boxed{y \leftarrow \mathsf{RO}(x)}$
$\boxed{\textbf{if } i = 1 \textbf{ then } \mathsf{H}_k \twoheadleftarrow \mu_k|_{\mathsf{Ext}(y_i, \mathsf{H}_j(x), \mathsf{H}_k(x)) = y}}$
$\boxed{\textbf{elseif } i = 2 \textbf{ then } \mathsf{H}_k \twoheadleftarrow \mu_k|_{\mathsf{Ext}(\mathsf{H}_k(x), y_i, \mathsf{H}_j(x)) = y}}$
$\boxed{\textbf{else } \mathsf{H}_k \twoheadleftarrow \mu_k|_{\mathsf{Ext}(\mathsf{H}_j(x), \mathsf{H}_k(x), y_i) = y}}$
$\mathsf{hst}_j \leftarrow \mathsf{hst}_j \cup \{(x, \mathsf{H}_j(x))\}$
$\mathsf{hst}_k \leftarrow \mathsf{hst}_k \cup \{(x, \mathsf{H}_k(x))\}$
$\mu_j \leftarrow \mu_j|_{\mathsf{hst}_j};\ \mu_k \leftarrow \mu_k|_{\mathsf{hst}_k}$
$\mathsf{H}_k \twoheadleftarrow \mu_k$
return $y_i$
```

```
$\mathsf{Game}_5 : \mathsf{BD}_i(f)$
$q \leftarrow q + 1$
$z \leftarrow f(\mathsf{H}_i);\ \mu_i \leftarrow \mu_i|_{f(\cdot) = z}$
$j \leftarrow (i \bmod 3) + 1;\ k \leftarrow (j \bmod 3) + 1$
if $q = Q$ then
  $(\mu_i, A_i) \twoheadleftarrow \mathsf{FixRows}[\gamma](\mu_i, p_{3s+i}, \mathsf{hst}_{i.1})$
  $\mathsf{H}_i \twoheadleftarrow \mu_i$
  $\mathsf{hst}_i \leftarrow \mathsf{hst}_i \cup A_i$
  $\boxed{\textbf{for } x \in A_{i.1} \textbf{ do } r_x \leftarrow \mathsf{C}_{2/3\mathrm{ext}}(x)}$
  $\boxed{\textbf{if } i = 1 \textbf{ then}}$
    $\boxed{\mathsf{H}_k \twoheadleftarrow \mu_k|_{\forall (x, y_i) \in A_i .\, \mathsf{Ext}(y_i, \mathsf{H}_j(x), \mathsf{H}_k(x)) = r_x}}$
  $\boxed{\textbf{elseif } i = 2 \textbf{ then}}$
    $\boxed{\mathsf{H}_k \twoheadleftarrow \mu_k|_{\forall (x, y_i) \in A_i .\, \mathsf{Ext}(\mathsf{H}_k(x), y_i, \mathsf{H}_j(x)) = r_x}}$
  $\boxed{\textbf{else } \mathsf{H}_k \twoheadleftarrow \mu_k|_{\forall (x, y_i) \in A_i .\, \mathsf{Ext}(\mathsf{H}_j(x), \mathsf{H}_k(x), y_i) = r_x}}$
  for $x \in A_{i.1}$ do
    $\mathsf{hst}_j \leftarrow \mathsf{hst}_j \cup \{(x, \mathsf{H}_j(x))\}$
    $\mathsf{hst}_k \leftarrow \mathsf{hst}_k \cup \{(x, \mathsf{H}_k(x))\}$
  $\mu_j \leftarrow \mu_j|_{\mathsf{hst}_j};\ \mu_k \leftarrow \mu_k|_{\mathsf{hst}_k}$
  $\boxed{\mathsf{H}_k \twoheadleftarrow \mu_k}$
  $q \leftarrow 0$
  if $i = 3$ then $s \leftarrow s + 1$
return $z$
```

Game_6. In this game we modify $\mathsf{C}_{2/3\mathrm{ext}}$ so that it starts to resemble a lazily sampled random oracle. Query-response pairs of the construction are kept in a set $\mathsf{hst}_{\mathsf{RO}}$ and in case a query is repeated, the stored image is simply returned. Otherwise, we distinguish three cases: (a) the corresponding row to the current query x is fixed in all hash functions, (b) in one of them, or (c) in none of them. In case (a), Game_6 computes the output of the construction by extracting from the individual images stored in histories of the BROs. Note, however, that this case is never reached, since if the current x is in all individual histories, then the construction must have already been called on x in some previous evaluation or backdoor query. Hence, x must also be in $\mathsf{hst}_{\mathsf{RO}}$. In case (b), a uniformly random value is chosen (and stored in $\mathsf{hst}_{\mathsf{RO}}$). In the final case (c), Game_6 behaves exactly as Game_5.

Overall, the distinguishing advantage is bounded by distinguishing p uniform (chosen by the construction) points each time a backdoor query fixes p points from values that were supposed to be extracted from three sources, from which one is not guaranteed to have any min-entropy, as well as q_{H} many times distinguishing a single extracted value from random. Let $E_n := \mathsf{Ext}\big(\mathsf{H}_1(x_n), \mathsf{H}_2(x_n), \mathsf{H}_3(x_n)\big)$, where $x_n \in A_{i.1}$ is a row being fixed. Note that we can assume that two of the input images have a min-entropy of at least $(1 - \delta_c) \cdot \log M$ as stated in the theorem, since δ_i's increase with i. We obtain:

$$\big|\Pr[\mathcal{D}^{\mathsf{Game}_5}] - \Pr[\mathcal{D}^{\mathsf{Game}_6}]\big| \leq \sum_{i=1}^{c} \mathsf{SD}\big(E_1| \cdots |E_{p_i}, \mathcal{U}_{[2]^{p_i}}\big) + q_{\mathsf{H}} \cdot \mathsf{SD}\big(E_1, \mathcal{U}_{[2]}\big) \ ,$$

```
$\mathsf{Game}_6 : \mathsf{C}_{2/3\mathrm{ext}}^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_3}(x)$
if $\exists y \in [M]$ s.t. $(x, y) \in \mathsf{hst}_{\mathsf{RO}}$ then
    return $y$
if $\exists y_1, y_2, y_3 \in [M]$ s.t. $(x, y_1) \in \mathsf{hst}_1 \wedge (x, y_2) \in \mathsf{hst}_2 \wedge (x, y_3) \in \mathsf{hst}_3$ then
    return $\mathsf{Ext}(y_1, y_2, y_3)$
if $\exists y' \in [M]$ s.t. $(x, y') \in \mathsf{hst}_1 \vee (x, y') \in \mathsf{hst}_2 \vee (x, y') \in \mathsf{hst}_3$ then
    $y \twoheadleftarrow [M]$
else
    for $i = 1..3$ do
        $y_i \leftarrow \mathsf{H}_i(x)$
        $\mathsf{hst}_i \leftarrow \mathsf{hst}_i \cup \{(x, y_i)\}$
        $\mu_i \leftarrow \mu_i|_{\mathsf{hst}_i}$
    $y \leftarrow \mathsf{Ext}(y_1, y_2, y_3)$
$\mathsf{hst}_{\mathsf{RO}} \leftarrow \mathsf{hst}_{\mathsf{RO}} \cup \{(x, y)\}$
return $y$
```

Game_7. The $\mathsf{C}_{2/3\mathrm{ext}}$ oracle in Game_7 differs from Game_6 in that it never evaluates the underlying BROs and rather acts as a lazily sampled random oracle. We can safely remove the case distinction (a), where x is included in all histories hst_1, hst_2, and hst_3, since this x would also be in $\mathsf{hst}_{\mathsf{RO}}$.

It remains to bound the adversary's advantage in distinguishing the two games while making up to q_{C} fresh queries x to the construction $\mathsf{C}_{2/3\mathrm{ext}}$ that are not fixed for any of the BROs. While the outputs of the construction are uniformly random in Game_7, they are extracted from three dense images in Game_6. The distinguisher can only try to maximize the distance between q_{C} uniform values vs. values extracted from three dense images of BROs by querying the construction on values and at times which it can choose freely.

$$\big|\Pr[\mathcal{D}^{\mathsf{Game}_6}] - \Pr[\mathcal{D}^{\mathsf{Game}_7}]\big| \leq \sum_{t=1}^{q_{\mathsf{C}}} \max_{x_t,\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_3} \Big(\mathsf{SD}\big(\mathsf{Ext}(\mathsf{H}_1(x_t), \mathsf{H}_2(x_t), \mathsf{H}_3(x_t)), \mathcal{U}_{[2]}\big)\Big)$$

$$\leq q_{\mathsf{C}} \cdot \max_{x,\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_3} \Big(\mathsf{SD}\big(\mathsf{Ext}(\mathsf{H}_1(x),\mathsf{H}_2(x),\mathsf{H}_3(x)),\mathcal{U}_{[2]}\big)\Big)$$
$$\leq q_{\mathsf{C}} \cdot \varepsilon\big((1-\delta_{c-1})\cdot\log M,(1-\delta_{c})\cdot\log M,(1-\delta_{c+1})\cdot\log M\big)\ ,$$

where according to Lemma 7.1 we have δ_i as defined in the theorem statement with ℓ_i being the min-entropy deficiency after the i-th sequence of Q-many backdoor queries. Note that the maximum statistical distance corresponds to minimum entropy of the BRO-images, which is in turn given for the last three $(c-1,c,c+1)$ values of the min-entropy rate.

```
$\mathsf{Game}_7 : \mathsf{C}_{2/3\text{ext}}^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_3}(x)$
if $\exists y \in [M]$ s.t. $(x,y) \in \mathsf{hst}_{\mathsf{RO}}$ then
    return $y$
~~if $\exists y_1,y_2,y_3 \in [M]$ s.t. $(x,y_1) \in \mathsf{hst}_1 \wedge (x,y_2) \in \mathsf{hst}_2 \wedge (x,y_3) \in \mathsf{hst}_3$ then~~
    ~~return $\mathsf{Ext}(y_1,y_2,y_3)$~~
~~if $\exists y' \in [M]$ s.t. $(x,y') \in \mathsf{hst}_1 \vee (x,y') \in \mathsf{hst}_2 \vee (x,y') \in \mathsf{hst}_3$ then~~
    $y \twoheadleftarrow [M]$
~~else~~
    ~~for $i = 1..3$ do~~
        ~~$y_i \leftarrow \mathsf{H}_i(x)$~~
        ~~$\mathsf{hst}_i \leftarrow \mathsf{hst}_i \cup \{(x,y_i)\}$~~
        ~~$\mu_i \leftarrow \mu_i|_{\mathsf{hst}_i}$~~
    ~~$y \leftarrow \mathsf{Ext}(y_1,y_2,y_3)$~~
$\mathsf{hst}_{\mathsf{RO}} \leftarrow \mathsf{hst}_{\mathsf{RO}} \cup \{(x,y)\}$
return $y$
```

The last game Game_7 is identical to the simulated world. Therefore, the overall advantage of $\mathcal{D}$ is as stated in the theorem.

Query complexity. The simulator makes queries to the random oracle RO in order to set images of the other BROs each time one point of some BRO is fixed, which is either caused by evaluation queries or by backdoor queries right after the Q-th consecutive backdoor query (i.e., before a switch). We obtain the following upper bound on the number of queries that the simulator makes to RO.

$$q_{\mathsf{Sim}} \leq q_{\mathsf{H}} + \sum_{i=1}^{c+1} p_i\ .$$

□

7.4.1 Instantiation with the Pairwise Inner-Product Extractor

Next we investigate a concrete instantiation of such a 2-out-of-3-source extractor. Most multi-source extractors such as those from [BKS$^+$05, Raz05, Li15] require a minimal amount of min-entropy from *every* source and are, therefore, inapplicable in our setting. We can, however, use the pairwise inner-product extractor as introduced by Lee et al. [LLTT05], which roughly speaking needs the sum of min-entropies to be sufficient. Recall the definition of the pairwise inner-product extractor $\mathsf{Ext}_{\text{pip}} : [M]^t \to [2]$, which we gave in Section 2.3.1 as:

$$\mathsf{Ext}_{\text{pip}}(x_1, \ldots, x_t) := \sum_{1 \leq i < j \leq t} x_i \cdot x_j \ .$$

The above extractor is proven ([LLTT05], Corollary 1) to be a $(k_1, \ldots, k_t, \varepsilon)$-extractor with an indistinguishability of $\varepsilon = 2^{-(k+k'-\log M+1)/2}$ from random, where k and k' are the two largest values among $k_1, \ldots, k_t$. Thus, $\mathsf{Ext}_{\text{pip}}$ is also a 2-out-of-t extractor.

Corollary 7.8. *Let* $\mathsf{Ext}_{\text{pip}} : [M]^t \to [2]$ *be a pairwise inner-product extractor. Then the construction* $\mathsf{C}_{\text{pip}}^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_3}(x) := \mathsf{Ext}_{\text{pip}}(\mathsf{H}_1(x), \mathsf{H}_2(x), \mathsf{H}_3(x))$ *in the* 3*-BRO model is indifferentiable from a random oracle, where*

$$\mathsf{Adv}^{\text{indiff}}_{\mathsf{C}_{2/3\text{ext}}^{\mathsf{H}_1,\mathsf{H}_2,\mathsf{H}_3},\mathsf{Sim}[p,\gamma]}(\mathcal{D}) \leq (c+1) \cdot \gamma + c \cdot \sqrt{(e^{p \cdot M^{-(1-2\delta_c)}} - 1)/2} + \big(q_{\mathsf{H}} + q_{\mathsf{C}}\big) \cdot 2^{-((1-2\delta_{c+1}) \cdot \log M + 1)/2} \ ,$$

while the simulator makes up to $q_{\mathsf{Sim}} \leq q_{\mathsf{H}} + (c+1) \cdot p$ *queries to* RO.

Proof. The differentiator's advantage stated in the corollary is easily obtained by upper bounding the term $\mathsf{SD}\big(E_1, \mathcal{U}_{[2]}\big)$ by $2^{-((1-2\delta_{c+1}) \cdot \log M+1)/2}$ and upper bounding the term $\mathsf{SD}\big(E_1| \cdots |E_p, \mathcal{U}_{[2]^p}\big)$ from the advantage in Theorem 7.7, using the following claim.

Lemma 7.9. *Let* $x_i \in [N]$ *and* $E_i := \mathsf{Ext}_{\text{pip}}\big(\mathsf{H}_1(x_i), \mathsf{H}_2(x_i), \mathsf{H}_3(x_i)\big)$ *for* $i = 1..n$*. Suppose that for all* x_i*, at least two of the (distributions of the) functions* H_1*,* H_2*,* H_3 *are* $(1-\delta)$*-dense. Then for all* $n \in \mathbb{N}$ *we have:*

$$\mathsf{SD}(E_1| \cdots |E_n, \mathcal{U}_{[2]^n}) \leq \sqrt{(e^{n \cdot M^{-(1-2\delta)}} - 1)/2} \ .$$

Proof. In the proof below we use the parity lemma [2] (1) and the fact that the pairwise inner product (in $[L]$) is linear, i.e., $\sum_{n \in I} E_n = \sum_{n \in I} \mathsf{Ext}_{\text{pip}}(\mathsf{H}_1(x_n), \mathsf{H}_2(x_n), \mathsf{H}_3(x_n)) = \mathsf{Ext}_{\text{pip}}(\mathsf{H}_1(x_1)| \cdots |\mathsf{H}_1(x_{|I|}),$ $\mathsf{H}_2(x_1)| \cdots |\mathsf{H}_2(x_{|I|}), \mathsf{H}_3(x_1)| \cdots |\mathsf{H}_3(x_{|I|})) = E_I$ (2).

$$\begin{aligned}
\mathsf{SD}(E_1| \cdots |E_n, \mathcal{U}_{[2]^n}) &\leq \sqrt{\sum_{0^{n \log 2} \neq a \in [2]^n} \big(\mathsf{SD}(E_1| \cdots |E_n \cdot a, \mathcal{U}_{[2]})\big)^2} && (1) \\
&= \sqrt{\sum_{\emptyset \neq I \subseteq \{1,\ldots,n\}} \Big(\mathsf{SD}\big(\sum_{i \in I} E_i, \mathcal{U}_{[2]}\big)\Big)^2} \\
&= \sqrt{\sum_{\emptyset \neq I \subseteq \{1,\ldots,n\}} \big(\mathsf{SD}(E_I, \mathcal{U}_{[2]})\big)^2} && (2) \\
&\leq \sqrt{\sum_{\emptyset \neq I \subseteq \{1,\ldots,n\}} 2^{-(|I| \cdot \log M \cdot (1-2\delta)+2-\log 2)}}
\end{aligned}$$

[2] Let X be a random variable over $[2^\ell]$. Then we have $\mathsf{SD}(X, \mathcal{U}_{[2^\ell]}) \leq \sqrt{\sum_{0^\ell \neq a \in [2^\ell]} \big(\mathsf{SD}(X \cdot a, \mathcal{U}_{[2]})\big)^2}$.

$$
\begin{aligned}
&= \sqrt{2^{-2+\log 2} \cdot \sum_{\emptyset \neq I \subseteq \{1,\ldots,n\}} 2^{-|I| \cdot \log M \cdot (1-2\delta)}} \\
&= \sqrt{2^{-1} \cdot \sum_{\emptyset \neq I \subseteq \{1,\ldots,n\}} \left(M^{-(1-2\delta)}\right)^{|I|}} \\
&= \sqrt{\left(\left(1 + M^{-(1-2\delta)}\right)^{n} - 1\right)/2} \\
&\leq \sqrt{(e^{n \cdot M^{-(1-2\delta)}} - 1)/2}
\end{aligned}
$$

□

Hence, the claim about the advantage holds. The query complexity of the simulator is bounded by the sum of q_{H} and $(c+1) \cdot p$. □

We now provide estimates for the involved parameters.

Corollary 7.10. *Let the number of switches be $c \geq 1$ and assume the range size of the three random oracles are $M \geq N^9$. Then there is an indifferentiability simulator Sim for the $\mathsf{C}_{\mathrm{pip}}$ construction in the 3-BRO model that places at most*

$$
q_{\mathsf{H}} + (c+1) \cdot \left(\frac{6Q\ell}{\log M}\right)^{1/\alpha(c)} \cdot N^{1-1/\alpha(c)}
$$

queries to RO, where $\alpha(c) := \left\lfloor \frac{c}{3} \right\rfloor + 1$, against any distinguisher with q_{H} queries to the underlying BROs. Further, any such distinguisher with q_{C} construction queries and Q consecutive queries to the same backdoor oracle before switching, has advantage at most $(c+1) \cdot \gamma + (c + q_{\mathsf{H}} + q_{\mathsf{C}})/N$ against this simulator.

Proof. The recurrence relations for δ_i in the statement of Theorem 7.7 can be written as

$$
\delta_i \leq A \cdot \delta_{i-3} + B \ ,
$$

where $A := N/p$ and $B := (Q\ell + \log \gamma^{-1})/p \log M$. Solving this recurrence relation we get

$$
\delta_i \leq \frac{A^{\left\lfloor \frac{i-1}{3} \right\rfloor + 1} - 1}{A - 1} \cdot B \ .
$$

We set $\delta_{c+1} \leq 1/3$ so that the term $1 - 2\delta_{c+1}$ is positive. To this end, it is sufficient to have that

$$
\frac{A^{\left\lfloor \frac{c}{3} \right\rfloor + 1} - 1}{A - 1} \cdot B \leq \frac{1}{3} \ .
$$

Substituting A and B and removing the -1 in the numerator we need to have that

$$
\left(\frac{N}{p}\right)^{\left\lfloor \frac{c}{3} \right\rfloor + 1} \leq \frac{A-1}{3B} = \frac{(N/p - 1) p \log M}{3Q\ell} = \frac{N \log M - p \log M}{3Q\ell} \leq \frac{N \log M}{6Q\ell} \ ,
$$

where for the last inequality we have assumed that $p \leq N/2$. Thus,

$$p \geq \left(\frac{6Q\ell}{\log M}\right)^{1/\alpha(c)} \cdot N^{1-1/\alpha(c)} \ ,$$

where $\alpha(c) := \left\lfloor \frac{c}{3} \right\rfloor + 1$. For sufficiently large c, the factor above is at most 2.

The advantage stated in Corollary 7.8 is

$$(c+1) \cdot \gamma + c \cdot \sqrt{p/M^{1-2\delta_c}} + (q_{\mathsf{H}} + q_{\mathsf{C}}) \cdot \sqrt{1/M^{1-2\delta_{c+1}}} \ .$$

Since $1 - 2\delta_{c+1} \leq 1 - 2/3 = 1/3$, $\delta_c \leq \delta_{c+1}$, $p \leq N$ and $M \geq N^9$, the advantage is upper-bounded by $(c+1) \cdot \gamma + (c + q_{\mathsf{H}} + q_{\mathsf{C}})/N$. □

Note that for $c = 1, 2$ the query complexity of the simulator does not involve the $N^{1-1/\alpha(c)}$ factor, and hence we obtain collision resistance. For $c \geq 3$, however there is a factor of at least $N^{1/2}$.

The above corollary shows that the extractor combiner can tolerate a *linear* number of switches in $\log N$ (which can be thought of as the security parameter) for the simulator query complexity to be less than $N/2$. As for the xor combiner we conjecture that (much) better bounds for the extractor combiner are possible.

7.5 Indifferentiability with Auxiliary Input

In this section we discuss indifferentiability in a setting where there is no adaptivity and the backdoor oracles are called only once at the onset. Although this may seem overly restrictive, the resulting definition is sufficiently strong to model indifferentiability in the presence of auxiliary input, i.e., against non-uniform adversaries, whereby we would like to securely replace random oracles in generic applications even in the presence of auxiliary input. Generally speaking, security against non-uniform adversaries is more desirable, since it takes into consideration that the adversary may have done some off-line preprocessing to prepare useful information or data structures (e.g., rainbow tables) that help speed up the actual on-line attack, when additional information becomes known to the adversary.

In this setting we can view an indifferentiability simulator as operating in two stages: An off-line stage which responds to the single backdoor queries for each BRO, and an on-line stage which simulates direct evaluation calls to the underlying BROs. The off-line phase of the simulator can pass an arbitrary state to its on-line phase. Further, both stages have access to the reference object oracles (although the query complexities of both stages need to be small). More precisely, this definition in the 2-RO model with auxiliary input requires that for any $(\mathcal{D}_{0,1}, \mathcal{D}_{0,2}, \mathcal{D}_1)$ in the real world with two random oracles H_1 and H_2 with

$$z_1 \twoheadleftarrow \mathcal{D}_{0,1}(\mathsf{H}_1); \ \ z_2 \twoheadleftarrow \mathcal{D}_{0,2}(\mathsf{H}_2, z_1); \ \ b \twoheadleftarrow \mathcal{D}_1^{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}, \mathsf{H}_1, \mathsf{H}_2}(z_1, z_2) \ ,$$

there exists some $(\mathsf{Sim}_{0,1}^{\mathsf{RO}}, \mathsf{Sim}_{0,2}^{\mathsf{RO}}, \mathsf{Sim}_{1,1}^{\mathsf{RO}}, \mathsf{Sim}_{1,2}^{\mathsf{RO}})$ in the ideal (simulated) world

$$(z_1, st) \twoheadleftarrow \mathsf{Sim}_{0,1}^{\mathsf{RO}}();\;\; (z_2, st) \twoheadleftarrow \mathsf{Sim}_{0,2}^{\mathsf{RO}}(st);\;\; b \twoheadleftarrow \mathcal{D}_1^{\mathsf{RO}, \mathsf{Sim}_{1,1}^{\mathsf{RO}}[st], \mathsf{Sim}_{1,2}^{\mathsf{RO}}[st]}(z_1, z_2)\ ,$$

with indistinguishable outputs b. The on-line simulators can also share state.

Let us now take a step back and define indifferentiability with auxiliary input driven by a composition theorem: for any game $\mathcal{G}$ and any attacker $\mathcal{A}_1$ in this game against $\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}$ which receives auxiliary input on H_1 and H_2, there is an attacker $\mathcal{B}_1$ on RO in the same game $\mathcal{G}$ but now *without* auxiliary input. More explicitly, the real world

$$z \twoheadleftarrow \mathcal{A}_0(\mathsf{H}_1, \mathsf{H}_2);\;\; b \twoheadleftarrow \mathcal{G}^{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}, \mathcal{A}_1^{\mathsf{H}_1,\mathsf{H}_2}(z)}$$

and the ideal world

$$(z, st) \twoheadleftarrow \mathcal{B}_0^{\mathsf{RO}}();\;\; b \twoheadleftarrow \mathcal{G}^{\mathsf{RO}, \mathcal{B}_1^{\mathsf{RO}}(z, st)}$$

are indistinguishable. Once again the query complexity of $\mathcal{B}_0$ should be small (or even zero) to obtain a definition which meaningfully formalizes indifferentiability from random oracles without auxiliary input. This definition, however, turns out to be unachievable: $\mathcal{A}_0$ can simply encode a pair of collisions for the construction, which $\mathcal{B}_0$ will not be able to match (with respect to RO) without an exponentially large number of queries to RO.[3] This, however, does not come as a surprise, since a random oracle in presence of auxiliary input is not collision resistant and, hence, it should not be indifferentiable for any reasonable definition of indifferentiability.

There are two natural ways to overcome this: (1) restrict the interface of the construction; or (2) restrict the form of preprocessing. The former is motivated by use of salting as a means to defeat preprocessing, and the latter by independence of preprocessing for BROs.

A final question arises here: is it possible to simplify this definition further by removing the quantification over $\mathcal{A}_1$ (as done for standard indifferentiability)? This could be done in the standard way by absorbing $\mathcal{A}_1$ into $\mathcal{G}$ to form a differentiator $\mathcal{D}_1$. However, this means that $\mathcal{D}_1$ must receive the auxiliary information z. The resulting notion is stronger and models composition with respect to *games* that also depend on preprocessing. Thus, due to its simplicity, strength, and the fact that we can establish positive results for it, we focus on this definitional approach. Below we make the two definitions arising from restriction (1) and (2) explicit.

Salted AI-indifferentiability. We call a construction C^{H} *salted* if it takes a salt $\mathrm{hk} \in \{0,1\}^k$ as input and prepends all calls to H with hk. We define salted AI-indifferentiability from a random oracle by requiring that for any $(\mathcal{D}_0, \mathcal{D}_1)$ in the real world

$$z \twoheadleftarrow \mathcal{D}_0(\mathsf{H});\;\; \mathrm{hk} \twoheadleftarrow \{0,1\}^k;\;\; b \twoheadleftarrow \mathcal{D}_1^{\mathsf{C}^{\mathsf{H}(\mathrm{hk},\cdot)}(\mathrm{hk},\cdot), \mathsf{H}}(\mathrm{hk}, z)$$

[3] One can formulate an intermediate notion of indifferentiability from random oracle *with* auxiliary input. Without salting, this notion would not be of great help, either. Consider, for example, the case of domain extension via an iterative hashing mode. Due to Joux's multi-collision attack [Jou04] one can encode exponentially many collisions for the construction in a small auxiliary input, whereas this would not be possible for the random oracle.

there is a simulator $(\mathsf{Sim}_0^{\mathsf{RO}}, \mathsf{Sim}_1^{\mathsf{RO}})$ in the ideal world

$$(z, st) \twoheadleftarrow \mathsf{Sim}_0^{\mathsf{RO}}(); \;\; \mathrm{hk} \twoheadleftarrow \{0,1\}^k; \;\; b \twoheadleftarrow \mathcal{D}_1^{\mathsf{RO}(\mathrm{hk},\cdot), \mathsf{Sim}_1^{\mathsf{RO}}[st]}(\mathrm{hk}, z)$$

resulting in indistinguishable outputs b. We denote the advantage of an adversary $\mathcal{D}$ in the salted AI-indifferentiability game with simulator Sim for a construction C^{H} by $\mathsf{Adv}^{\text{s-ai-indiff}}_{\mathsf{C}^{\mathsf{H}},\mathsf{Sim}}(\mathcal{D})$. Notice that in the above definition, the distinguisher gets access to a *salted* RO. A different definition arises when the distinguisher gets access to an unsalted RO instead. However, since the simulated auxiliary information is computed given access to an unsalted RO (which can be interpreted as having implicit access to the salt), such a definition calls for the existence of a more powerful simulator. In particular, such Sim_0 and $\mathcal{D}_1$ can easily call RO on common points. The practical implications of such a definition are unclear to us, and moreover, it is strictly weaker than our definition.

AI-indifferentiability with independent preprocessing. We define AI-indifferentiability with independent preprocessing by requiring that for any adversary $(\mathcal{D}_{0,1}, \mathcal{D}_{0,2}, \mathcal{D}_1)$ in the real world

$$z_1 \twoheadleftarrow \mathcal{D}_{0,1}(\mathsf{H}_1); \;\; z_2 \twoheadleftarrow \mathcal{D}_{0,2}(\mathsf{H}_2); \;\; b \twoheadleftarrow \mathcal{D}_1^{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}, \mathsf{H}_1, \mathsf{H}_2}(z_1, z_2)$$

there is a simulator $(\mathsf{Sim}_{0,1}^{\mathsf{RO}}, \mathsf{Sim}_{0,2}^{\mathsf{RO}}, \mathsf{Sim}_{1,1}^{\mathsf{RO}}, \mathsf{Sim}_{1,2}^{\mathsf{RO}})$ in the ideal world

$$(z_1, st) \twoheadleftarrow \mathsf{Sim}_{0,1}^{\mathsf{RO}}(); \;\; (z_2, st) \twoheadleftarrow \mathsf{Sim}_{0,2}^{\mathsf{RO}}(st); \;\; b \twoheadleftarrow \mathcal{D}_1^{\mathsf{RO}, \mathsf{Sim}_{1,1}^{\mathsf{RO}}[st], \mathsf{Sim}_{1,2}^{\mathsf{RO}}[st]}(z_1, z_2)$$

resulting in indistinguishable outputs b. Note that this is slightly weaker than the definition of indifferentiability in 2-BRO since z_2 is fully independent of z_1, whereas BRO indifferentiability allows for a limited amount of dependence. We denote by $\mathsf{Adv}^{\text{ai-indiff}}_{\mathsf{C}^{\mathsf{H}},\mathsf{Sim}}(\mathcal{D})$ the advantage of $\mathcal{D}$ in the AI-indifferentiability game with independent preprocessing with respect to a simulator Sim and a construction $\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}$ in the 2-BRO model.

We are now ready to prove our feasibility results for AI-indifferentiability.

Theorem 7.11 (AI-Indifferentiability)**.** *Any construction $\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2}$ that is indifferentiable with backdoors from a random oracle with no backdoor adaptive queries is also AI-indifferentiable from a random oracle with respect to independent preprocessing attacks. More precisely, for any auxiliary-input differentiator $\mathcal{D} := (\mathcal{D}_{0,1}, \mathcal{D}_{0,2}, \mathcal{D}_1)$ with independent preprocessing for two random oracles there is a 2-BRO differentiator $\tilde{\mathcal{D}} := (\tilde{\mathcal{D}}_{0,1}, \tilde{\mathcal{D}}_{0,2}, \tilde{\mathcal{D}}_1)$ with one-time non-adaptive access to each backdoor oracle such that for any 2-BRO indifferentiability simulator $\tilde{\mathsf{Sim}} := (\tilde{\mathsf{Sim}}_{0,1}, \tilde{\mathsf{Sim}}_{0,2}, \tilde{\mathsf{Sim}}_{1,1}, \tilde{\mathsf{Sim}}_{1,2})$ there is an auxiliary-input simulator $\mathsf{Sim} := (\mathsf{Sim}_{0,1}, \mathsf{Sim}_{0,2}, \mathsf{Sim}_{1,1}, \mathsf{Sim}_{1,2})$ such that*

$$\mathsf{Adv}^{\text{ai-indiff}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2},\mathsf{Sim}}(\mathcal{D}) = \mathsf{Adv}^{\text{indiff}}_{\mathsf{C}^{\mathsf{H}_1,\mathsf{H}_2},\tilde{\mathsf{Sim}}}(\tilde{\mathcal{D}}) \; .$$

Further, any salted construction C^{H} (with k bits of salt) that is indifferentiable (in the standard sense) from a random oracle is also salted AI-indifferentiable from a random oracle. More precisely, for any auxiliary-input differentiator $\mathcal{D} := (\mathcal{D}_0, \mathcal{D}_1)$, with an auxiliary input of size ℓ, there is a (standard) differentiator $\tilde{\mathcal{D}} := (\tilde{\mathcal{D}}_0, \tilde{\mathcal{D}}_1)$ such that for any indifferentiability simulator $\tilde{\mathsf{Sim}} := (\tilde{\mathsf{Sim}}_0, \tilde{\mathsf{Sim}}_1)$ there is an auxiliary-input simulator $\mathsf{Sim} := (\mathsf{Sim}_0, \mathsf{Sim}_1)$ such that for any $p \in \mathbb{N}$ and *any* $\gamma > 0$

$$\mathsf{Adv}^{\text{s-ai-indiff}}_{\mathsf{C}^{\mathsf{H}},\mathsf{Sim}}(\mathcal{D}) \leq \mathsf{Adv}^{\text{indiff}}_{\mathsf{C}^{\mathsf{H}},\tilde{\mathsf{Sim}}}(\tilde{\mathcal{D}}) + \frac{\ell + \log \gamma^{-1}}{p} + \frac{p}{2^k} + \gamma \ .$$

Proof. The first part of the theorem follows directly from the discussion above that indifferentiability with backdoors and no adaptivity is stronger than indifferentiability with auxiliary input for independent preprocessing.

We now prove the second part of the theorem.

Game_0**:** We start with the real game in the salted AI-indifferentiability game:

$$z \twoheadleftarrow \mathcal{D}_0(\mathsf{H});\ \ \mathrm{hk} \twoheadleftarrow \{0,1\}^k;\ \ b \twoheadleftarrow \mathcal{D}_1^{\mathsf{C}^{\mathsf{H}(\mathrm{hk},\cdot)}(\mathrm{hk},\cdot),\mathsf{H}}(\mathrm{hk}, z)\ .$$

Game_1**:** We now move to the bit-fixing RO model

$$(z, A) \twoheadleftarrow \tilde{\mathcal{D}}_0();\ \ \mathrm{hk} \twoheadleftarrow \{0,1\}^k;\ \ b \twoheadleftarrow \mathcal{D}_1^{\mathsf{C}^{\mathsf{H}[A](\mathrm{hk},\cdot)}(\mathrm{hk},\cdot),\mathsf{H}[A]}(\mathrm{hk}, z)\ .$$

Here $\tilde{\mathcal{D}}_0$ runs $\mathcal{D}_0$ by simulating an H for it and then runs the decomposition algorithm to get a set of assignments A for p fixed points (for any $p \in \mathbb{N}$) according to the auxiliary input z output by $\mathcal{D}_0$. We may now apply [CDGS18, Theorem 5] to deduce that for any $\gamma > 0$,

$$\Pr[\mathsf{Game}_1] - \Pr[\mathsf{Game}_0] \leq \frac{\ell + \log \gamma^{-1}}{p} + \gamma\ ,$$

where ℓ is the size of auxiliary information.

Game_2**:** We now move to a setting where C uses H rather than $\mathsf{H}[A]$, i.e.,

$$(z, A) \twoheadleftarrow \tilde{\mathcal{D}}_0();\ \ \mathrm{hk} \twoheadleftarrow \{0,1\}^k;\ \ b \twoheadleftarrow \mathcal{D}_1^{\mathsf{C}^{\mathsf{H}(\mathrm{hk},\cdot)}(\mathrm{hk},\cdot),\mathsf{H}[A]}(\mathrm{hk}, z)\ .$$

This modification is justified by the fact that the probability that a uniform hk is (the prefix of the first component of some point) in A is at most $p/2^k$. Hence, $\Pr[\mathsf{Game}_2] - \Pr[\mathsf{Game}_1] \leq p/2^k$.

Game_3**:** We now move to a world where $\mathcal{D}_1$ is replaced by a differentiator $\tilde{\mathcal{D}}_1$ that gets the list A and does not query H on points in A:

$$(z, A) \twoheadleftarrow \tilde{\mathcal{D}}_0();\ \ \mathrm{hk} \twoheadleftarrow \{0,1\}^k;\ \ b \twoheadleftarrow \tilde{\mathcal{D}}_1^{\mathsf{C}^{\mathsf{H}(\mathrm{hk},\cdot)}(\mathrm{hk},\cdot),\mathsf{H}}(\mathrm{hk}, z, A)\ .$$

Here $\tilde{\mathcal{D}}_1(\mathrm{hk}, z, A)$ runs $\mathcal{D}_1(\mathrm{hk}, z)$ relaying its queries to the first oracle to its own first oracle and the second oracle queries to its own second oracle except when a queried point appears as a prefix of the first component of an entry in A in which case $\tilde{\mathcal{D}}_1$ uses A to answer the query. We have that $\Pr[\mathsf{Game}_3] - \Pr[\mathsf{Game}_2] = 0$.

Game_4**:** We now absorb $\tilde{\mathcal{D}}_0$ and $\tilde{\mathcal{D}}_1$ into a single differentiator $\tilde{\mathcal{D}}$:

$$b \twoheadleftarrow \tilde{\mathcal{D}}^{\mathsf{C}^{\mathsf{H}(\mathrm{hk},\cdot)}(\mathrm{hk},\cdot),\mathsf{H}}\ .$$

Here $\tilde{\mathcal{D}}$ simply runs $\tilde{\mathcal{D}}_0$, followed by picking $\mathrm{hk} \twoheadleftarrow \{0,1\}^k$, and then running $\tilde{\mathcal{D}}_1$. We have that $\Pr[\mathsf{Game}_4] - \Pr[\mathsf{Game}_3] = 0$.

Game_5**:** We now use the standard indifferentiability of the construction to move to the world

$$b \twoheadleftarrow \tilde{\mathcal{D}}^{\mathsf{RO}(\mathrm{hk},\cdot),\widetilde{\mathsf{Sim}}^{\mathsf{RO}}} ,$$

where $\widetilde{\mathsf{Sim}}$ is an indifferentiability simulator. We get $\Pr[\mathsf{Game}_3] - \Pr[\mathsf{Game}_2] \leq \mathsf{Adv}^{\mathrm{indiff}}_{\mathsf{C}^{\mathsf{H}},\widetilde{\mathsf{Sim}}}(\tilde{\mathcal{D}})$.

Game_6**:** We now syntactically unroll $\tilde{\mathcal{D}}$ into $(\tilde{\mathcal{D}}_0, \tilde{\mathcal{D}}_1)$:

$$(z, A) \twoheadleftarrow \tilde{\mathcal{D}}_0(); \;\; \mathrm{hk} \twoheadleftarrow \{0,1\}^k; \;\; b \twoheadleftarrow \tilde{\mathcal{D}}_1^{\mathsf{RO}(\mathrm{hk},\cdot),\widetilde{\mathsf{Sim}}^{\mathsf{RO}}}(\mathrm{hk}, z, A) \ .$$

We have that $\Pr[\mathsf{Game}_6] - \Pr[\mathsf{Game}_5] = 0$.

Game_7**:** We further unroll $\tilde{\mathcal{D}}_1$ into $\mathcal{D}_1$ and define $\mathsf{Sim}_1[A]$ to be $\widetilde{\mathsf{Sim}}$ except that it uses A to answers queries in A:

$$(z, A) \twoheadleftarrow \tilde{\mathcal{D}}_0(); \;\; \mathrm{hk} \twoheadleftarrow \{0,1\}^k; \;\; b \twoheadleftarrow \mathcal{D}_1^{\mathsf{RO}(\mathrm{hk},\cdot),\mathsf{Sim}_1^{\mathsf{RO}}[A]}(\mathrm{hk}, z) \ .$$

We have that $\Pr[\mathsf{Game}_7] - \Pr[\mathsf{Game}_6] = 0$.

Game_8**:** Finally, we define $\mathsf{Sim}_0 := \tilde{\mathcal{D}}_0$ and arrive at the simulated world

$$(z, A) \twoheadleftarrow \mathsf{Sim}_0(); \;\; \mathrm{hk} \twoheadleftarrow \{0,1\}^k; \;\; b \twoheadleftarrow \mathcal{D}_1^{\mathsf{RO}(\mathrm{hk},\cdot),\mathsf{Sim}_1^{\mathsf{RO}}[A]}(\mathrm{hk}, z) \ .$$

We have that $\Pr[\mathsf{Game}_8] - \Pr[\mathsf{Game}_7] = 0$.

Hence, the second part of theorem follows by summing the (in)equalities established above; that is for any $p \in \mathbb{N}$ and any $\gamma > 0$ we get that

$$\begin{aligned}\mathsf{Adv}^{\text{s-ai-indiff}}_{\mathsf{C}^{\mathsf{H}},(\mathsf{Sim}_0,\mathsf{Sim}_1)}(\mathcal{D}_0, \mathcal{D}_1) &= \Pr[\mathsf{Game}_0] - \Pr[\mathsf{Game}_8] \\ &\leq \mathsf{Adv}^{\mathrm{indiff}}_{\mathsf{C}^{\mathsf{H}},\widetilde{\mathsf{Sim}}}(\tilde{\mathcal{D}}) + \frac{\ell + \log \gamma^{-1}}{p} + \frac{p}{2^k} + \gamma \ .\end{aligned}$$

□

We may instantiate the first part of the theorem with the xor combiner and an indifferentiability simulator for it given in Section 7.3. Additionally, the extractor combiner from Section 7.4 provides similar guarantees in the 3-BRO model. Let us discuss the xor combiner in more detail. In this case the off-line phase of the simulator makes no queries to the RO (and outputs simulated auxiliary inputs by picking hash functions for the queried backdoor functions to BD_1 and BD_2). This off-line phase also outputs two sets of p_1 and p_2 preset points as its state, which will be shared with the on-line phase of simulation. The second phase of the simulator is a simple xor indifferentiability simulator which ensures consistency with the preset points. Here our simulator fixes p_1 points for H_1 and p_2 points for H_2. This results in a simulator query complexity of $q_{\mathsf{H}} + p_1 + p_2$. The corresponding advantage bound is at most $2\gamma + q_{\mathsf{H}} \cdot \log M \cdot \delta_2 + q_{\mathsf{C}} \cdot \log M(\delta_1 + \delta_2)$ which is of order $\mathcal{O}(q_{\mathsf{H}}\ell/p_2 + q_{\mathsf{C}}(\ell/p_1 + \ell/p_2))$.

Setting $p_1 = p_2 = p$ we get a simulator with $\mathcal{O}(q_{\mathsf{H}} + p)$ queries for an advantage $\mathcal{O}((q_{\mathsf{H}} + 2q_{\mathsf{C}})\ell/p)$. For $p = o(\sqrt{N})$ we get a bound that is meaningful for collision resistance.

As a result, we get that the xor combiner is collision resistant in the presence of independent auxiliary input (with no-salting). We note that the xor construction comes with added advantage that its security goes beyond AI-indifferentiability, and is also more domain efficient than the salting approach. Strictly speaking, however, the two settings are incomparable as the form of auxiliary information changes, requiring independent preprocessing in the 2-BRO model.

PART II

Subverted Implementations

Chapter 8

Self-Guarding Primitives against Subverted Implementations

In this chapter we introduce the notion of self-guarding cryptographic primitives to combat a type of algorithm substitution attacks. We investigate how self-guarding primitives can resist substitution attacks that are conducted or activated after the primitive has been working properly and securely during an initial phase. We present constructions of basic primitives for public-key and private-key encryption as well as digital signatures. We also show how to self-guard a PUF-based key exchange protocol. Our constructions put great value on the simplicity and efficiency of the operations employed in making the primitive under attack self-guarding, so as to keep the trusted core as small as possible. On the downside, self-guarding schemes can usually only guarantee a limited number of secure executions.

My Scientific Contribution in this Chapter

The material in this chapter is a joint work with Marc Fischlin and appears in [FM18]. Marc and I jointly devised the idea of making use of secure initial states of certain primitives to re-establish security in their future executions. I developed the model in Section 8.2 with Marc's help. Regarding the constructions, my focus lay mainly on the homomorphic and the symmetric encryption schemes presented in Sections 8.3 and 8.4, respectively. Marc focused on the signature scheme of Section 8.5 and the PUF-based key exchange protocol of Section 8.6.

8.1 Introduction

The formal study of algorithm substitution attacks (ASAs) as an instrument of mass surveillance was initiated by Bellare, Paterson, and Rogaway [BPR14] on the example of symmetric encryption. ASAs constitute a class of attacks, where the adversary subverts the implementation of a (well-designed, non-backdoored) cryptosystem by a malicious one in order to compromise its security. The adversary's goal in this setting is formalized in [BPR14] roughly as violating the cryptosystem's security while remaining undetectable to users. Note that an attack that is *easy* to detect is highly unlikely to be carried out by a cautious adversary. For instance, a NOBUS subversion of an encryption algorithm would not simply output messages in clear nor would it induce a high decryption failure. Prior work has been quite successful in raising awareness of the danger of ASAs in terms of describing various attacks that can be mounted while risking detection as little as possible [DFP15, BJK15, AP19b, AP19a]. However, the list of countermeasures to ASAs, which we will discuss shorty, is unfortunately still relatively sparse and the practicality of some of them is questionable. A first observation that we made to understand how positive results can look like was the following: to prevent ASAs, a detection-based definition is not a particularly suitable one, since it implies that resistance against ASAs either means that the subverted system is still secure, or that the attack is detectable. Detectability, however, does not say much about how feasible it is that the attack is actually detected in the real world. In other words, such a definition would declare many potential real-world subversions as secure for the mere fact that they are theoretically detectable.

8.1.1 Detecting Substitution Attacks

Detecting algorithm substitution attacks can be hard, sometimes even impossible. It was shown that randomized symmetric encryption is prone to ASAs that can leak the secret key, while not only avoiding detection by any efficient detector (a.k.a. *watchdog* as called in [RTYZ16]) with black-box access to the algorithm, but can even produce ciphertexts that are perfectly decryptable [BPR14, BJK15]. Attacks strategies that slightly relax the correctness assumptions can still be practically undetectable. Degabriele, Farshim, and Poettering illustrated this threat for subversions with input-triggered misbehavior [DFP15]. Armour and Poettering also showed how minor but strategic decryption or verification error can lead to exfiltration of secret keys in authenticated encryption schemes and message authentication codes [AP19b, AP19a]. Taking into account the strong black-box impossibility results, some works have suggested to use deterministic encryption schemes with unique ciphertexts such that one can compare against the expected values [BPR14, BH15, BJK15, DFP15].

Even in situations where detection is theoretically possible, it is arguably very difficult to design proper detectors in practice. A detector gets access to an implementation which, due to the nature of the attack, may be arbitrarily subverted, and the detector has to decide if any efficient adversary is able to violate its security. Overall, it is unclear which kinds of irregularities the detector should look for and which detection capabilities are reasonable to assume. For instance, deterministic schemes are considered detectable by *on-line* detectors, since they can compare the output of the possibly subverted algorithm with the expected output at runtime [BPR14, DFP15, BJK15]. Aside from being rather inefficient, this assumes that the detector has a good implementation of the same algorithm at hand and that scans are performed while the system is active.

Another tricky situation arises when implementations behave honestly only as long as they are under scrutiny, say, through an off-line detector. However, malicious behavior can be triggered to wake up at a later point in time or in some future state. Degabriele et al. [DFP15] describe an undetectable ASA on randomized symmetric encryption that is triggered to leak the secret key upon being called on a special input. Such attacks have also been discussed in the context of software and also in the domain of cryptographic hardware backdoors and trojans, e.g., [WS11, DFS16]. For protection against similar attacks in hardware tokens that implement a deterministic function, Dziembowski, Faust, and Standaert [DFS16] make use of a semi-on-line detector that regularly tests the tokens against the specification that they are supposed to implement, while malicious input-triggers are prevented via masking, a technique which can also be useful in our setting.

Conceptually, a malicious or simply insecure software update also fits into the category of ASAs that become active by a trigger after running securely for some time. Two prominent testimonies are the heartbleed vulnerability in the open source library OpenSSL and the Juniper Dual_EC incident. The heartblead bug was introduced with version 1.0.1 in 2011 and went unnoticed for over two years (see `heartbleed.com`). In 2015, Juniper Networks announced that the source code of ScreenOS, the operating system of their VPN routers, was maliciously modified in 2012 [CMG$^+$16]. Although one can speculate about whether these and similar attacks were inadvertent or not, they showcase the possibility of substitution attacks being performed in the real world as a mean of mass surveillance.

The problem of detecting ASAs by comparing outputs with trusted variants can even be acute if hardware components are involved. For example, in case of physical uncloneable functions (PUFs) this seems to be impossible: a good PUF ideally implements a random and uncloneable function (unlike hardware tokens considered by Dziembowski et al. [DFS16] which implement a specified function), but the internal computations are not assessable. It is unclear what the detector should check for, maybe except for basic properties such as the absence of collisions. Furthermore, the detector may not be able to check output values later if it does not have access to the PUF anymore. The infeasiblility of verifying security of hardware tokens also motivated Camenisch, Drijvers, and Lehmann [CDL17] to design an anonymous attestation protocol which achieves privacy even with subverted trusted platform modules (TPM).

Another issue with detectors is that they need to be trustworthy entities and their implementations also need to be reliable. A detector that colludes with a mass surveillance agency or is subverted itself is clearly unreliable. Even worse, in some scenarios the detection algorithm requires access to the secret key [BPR14, DFP15], introducing other potential security risks. The latter may make detection also hard in case of hardware components, if keys are not allowed to leave the devices that they are stored on.

8.1.2 Preventing Substitution Attacks

Considering the difficulty of detecting algorithm substitution attacks, a question that comes to mind is if it is possible to neutralize attacks without the requirement to detect them first. Although neutralizing algorithm substitution attacks is a highly challenging task, it can be more promising than detection-based approaches both in terms of security and efficiency. Furthermore the approach of (off-line) detection is often reactive and cannot obscure loss of crucial information when the system is being used. A proactive solution, however, can prevent leakage in the first place.

Indeed cryptographic *reverse firewalls*, introduced by Mironov and Stephens-Davidowitz [MS15], follow the approach of prevention [MS15, DMS16]. The idea of reverse firewalls is to distribute the trust between the user and a firewall. The outgoing communication, say, a signature, is first routed through the firewall which may take further cryptographic steps, such as verifying the signature and re-randomizing it, in order to prevent subliminal channels. The targeted security guarantee is: as long as one of the two components (user or firewall) is trustworthy and has a proper implementation, no information can be transferred through the firewall. Reverse firewalls may not be readily applicable to every existing protocol. In fact, a goal of [MS15, DMS16] is to design protocols that can be used with reverse firewalls. How to design amenable protocols for symmetric-key primitives, or when using hardware tokens, without complex detection mechanisms remains open.

In two recent works by Russel et al. [RTYZ16, RTYZ17] the detector model is combined with some prevention mechanism. Their approach is based on a split-program methodology, where an algorithm is split into deterministic and probabilistic parts that can be individually tested by the detector. This allows prevention of for instance rejection-sampling attacks by not letting the deterministic part of the algorithm request more randomness that it needs, and prevention of input-triggered attacks by adding a random value to the input, i.e., masking it. Despite remarkable improvements in the power that a detector in the split-program model gains, some of our criticisms, e.g., requiring a good implementation and failing to detect state-dependent attacks, remain.

8.1.3 The Concept of Self-Guarding Security

Our contribution is providing an alternative defense mechanism to reverse firewalls which, too, proactively thwarts ASAs but does not depend on external parties. We focus on a setting where the party at some point holds a genuine version of the algorithm, before the algorithm gets substituted by e.g., a malicious software update, or before an input or a state triggers the malicious behavior of the algorithm. In other words, our "security anchor" is the assumption of having a secure initial phase. We call a cryptographic scheme that is secure despite making black-box use of possibly subverted underlying schemes *self-guarding*. Such a scheme uses information (from now on called *samples*) gathered from its underlying primitives during their good initial phase in addition to basic operations to prevent leakage later on, or to implement a new protocol securely without implementing the required primitives securely from scratch. We emphasize that subverting or exfiltrating samples (e.g., pairs of message and ciphertexts, which depend on users' keys) would require targeted attacks, which is a considerably more difficult attack to carry out in a large scale than algorithm substitution attacks. The downside of this approach is, however, that especially in case the underlying primitives are subverted with stateful algorithms, it is tricky to reuse samples for self-guarding and the number of secure executions after a possible ASA becomes effectively limited.

Related Approaches

Our idea of a trusted initialization phase resembles several other methods in the literature. In the area of program self-correction [BLR90] an algorithm can take advantage of a program which computes correctly on an overwhelming fraction of inputs to build one which always outputs correct answers with high probability. This bootstrapping is similar to our idea here, only that we use temporary correctness and security of the program. In self-correction, as well as program checking [BK89], it is important to not trivialize the problem by implementing a trusted program oneself. Instead, one should only use basic operations on top. Another related setting is the one of non-interactive verifiable computing introduced by Gennaro, Gentry, and Parno [GGP10], whereby a powerful but untrusted server performs some computation for a client, which the client can verify using simple operations. Similar austerity principle applies in our setting, with the main difference that we are interested in reviving the security and not necessarily the correctness of the computations.

The concept also appears in the context of digital certificates. A technique called HTTP Public Key Pinning (HPKP) [EPS15], albeit argued about, is a trust-on-first-use technique for checking the validity of certificates. On first usage certificates are declared as trustworthy ("pinned") such that substitution of certificates in subsequent executions becomes infeasible.

Finally, in interactive protocols involving physical uncloneable functions (PUFs) or other hardware tokens, the question of security in the presence of malicious tokens has been brought up (e.g., in [OSVW13, Rüh16, BKOV17]). Here, the sender of the token typically first holds a genuine version of the token. The adversary may subvert the token later, when in transmission. This corresponds to our setting with a trusted initialization phase and subversion afterwards, only that the protocol involves two parties and hardware tokens.

Comparison to Detectors and Reverse Firewalls

As mentioned before, self-guarding schemes, as well as reverse firewalls, prevent leakage by construction. The difference is how the security anchor is provided: In reverse firewalls it is ensured by trust distribution, in self-guarding schemes it is based on a temporary trusted phase. Self-guarding applies more smoothly to symmetric primitives and hardware tokens, but for other primitives it currently comes with a performance inferior to today's reverse firewall solutions.

At first glance one might think that the initialization phase of self-guarding schemes could simply be executed by a detector with the specified program, such that we immediately get a detecting solution. However, our self-guarding schemes will pass state between the phases, whereas detectors typically do not forward data to individual users. Furthermore, although one could in principle combine our approach with some detection mechanism, self-guarding does not aim at spotting malicious behavior innately. Another noteworthy difference between self-guarding and the detector model is that self-guarding schemes do not need the subverted algorithm in the beginning.

8.1.4 Self-Guarding Constructions

We show how to build self-guarding solutions for multiple basic primitives: public-key encryption, symmetric encryption, and signatures. To show that our model applies to hardware tokens, too, we also show how to build a self-guarding PUF-based key exchange protocol if the adversary can substitute tokens in transmission. While the general idea of passing samples of the primitive in question from the initialization phase to the execution is shared by all solutions, the techniques differ in details and also in terms of security guarantees and efficiency.

We first give a simple and efficient construction for a self-guarding IND-CPA-secure public-key encryption scheme from any homomorphic encryption scheme, e.g., ElGamal encryption. Our scheme is self-guarding even against stateful subversions of the underlying scheme. Yet, the disadvantage is that, if the subverted scheme is stateful, we are limited to only encrypting as many messages securely as we have sample ciphertexts from the first phase. Our construction is inspired by an elegant idea by Russel et al. [RTYZ17] to prohibit input-triggered attacks in encryption schemes. This is achieved by xoring a random message to the input of an encryption algorithm and sending the random message along with the ciphertext.

The second construction provides an IND-CPA-secure self-guarding symmetric-key encryption scheme, starting with any regular IND-CPA-secure scheme. Here the number of encryptions is again limited by the number of available samples, and the message space is also bounded. Despite these limitations, we find this construction quite intriguing, since the only other proposal for subversion-resisting randomized symmetric encryption is in a split-program detection-based model [RTYZ17].

Our third construction is a self-guarding signature scheme. It is built upon any deterministic EUF-CMA-unforgeable signature scheme. This time, however, the overhead is bigger than in case of encryption, and it only self-guards against stateless subversion of the underlying scheme. In contrast, it can be securely used to sign arbitrarily often after the substitution took place. Moreover, contrary to re-randomizing reverse firewalls for signatures, as proposed in [AMV15], it does not rely on an honest implementation of the verification algorithm for signing. We do, however, assume that the verification algorithm is not subverted so that we can exclude such trivial attacks where the verification algorithm accepts invalid signatures.

Finally, we give a PUF-based key exchange protocol that is self-guarding against subversion of the PUF with malicious, stateful, and encapsulated PUFs. This is noteworthy as for more complex tasks such as oblivious transfer there are negative results concerning such malicious PUFs [vR12, DFK$^+$14, Rüh16]. Our key exchange protocol has four rounds and uses only a single genuine sample from the initialization phase for deriving each key. It thus matches the non-self-guarding PUF-based protocols in terms of the number of samples.

8.2 Modeling Self-Guarding Primitives

In this section we first define the syntax of self-guarding schemes. We then give a generic model of cryptographic games and finally turn to defining a generic security notion for self-guarding.

To distinguish genuine from potentially malicious implementations, we use a notation similar to [RTYZ16, RTYZ17], i.e, we use indices GNN for a trusted and genuine implementation, and SBV for a subverted implementation. For simplicity, we sometimes omit the index GNN for the original algorithms, if this is clear from the context. We are interested in schemes Π which make use of a possibly subverted primitive Σ. We require Π to obey a specific interface. In particular, it should provide means for generating parameters for the scheme and sampling their algorithm interfaces. More formally, given a cryptographic scheme Σ, we define $\Pi^\Sigma := (\Pi.\mathsf{Gen}^\Sigma, \Pi.\mathsf{Sample}^\Sigma, \Pi.\mathsf{X}_1^\Sigma, \ldots, \Pi.\mathsf{X}_n^\Sigma)$ for some $n \in \mathbb{N}$, where

- $\Pi.\mathsf{Gen}^\Sigma(1^\lambda) \twoheadrightarrow P = (P_s, P_p)$. On input of a security parameter 1^λ, this PPT algorithm outputs secret parameters P_s and public parameters P_p.
- $\Pi.\mathsf{Sample}^\Sigma(P) \twoheadrightarrow S = (S_1, \ldots, S_n)$. On input of parameters $P = (P_s, P_p)$, this PPT algorithm outputs collections S_i of input-output samples of $\Pi.\mathsf{X}_i^\Sigma$ for $1 \leq i \leq n$. The size of each S_i is determined by the protocol Π and may depend on the security parameter.
- $\Pi.\mathsf{X}_i^\Sigma$ are PPT or DPT algorithms that are placeholders for other functionalities of Π, for all i with $1 \leq i \leq n$.

We remark here that Π can basically take two different roles. It can either provide a different, possibly more complex functionality, while remaining secure despite using a subverted algorithm Σ_{SBV}, or it can simply neutralize a possible substitution attack by Σ_{SBV}. For the former role, Π can for instance be a key exchange protocol using some cryptographic primitive Σ (e.g., a signature scheme), and the security game may capture the indistinguishability of the derived keys. Intuitively, the adversary's goal is now to distinguish keys from random by subverting Σ. In the latter case, Π and Σ can have a similar functionality, for example providing the usual interfaces for encryption and decryption. In this case Π's task is to neutralize a potential subversion attack on Σ.

In principle, preventing an attack is trivial to achieve in real executions, simply by having Π deploy its own secure implementation of Σ_{GNN}, ignoring the potentially substituted implementation Σ_{SBV}. To avoid such issues we assume that Π makes only black-box use of Σ and only implements very basic extra steps for the immunization, which are easy to implement and hard to subvert. In other words, in a practical construction, the internal part of Π concerning the immunization, i.e., excluding the queries to Σ and possibly an own functionality, must be as simple as possible. This assumption is important in order to keep the trusted component, i.e., Π, as small as possible.

For a meaningful definition we require the genuine implementations, i.e., $\Pi^{\Sigma_{\mathsf{GNN}}}$, to be correct. Since our main objective here is preventing ASAs, we generally do not expect a correct functionality from $\Pi^{\Sigma_{\mathsf{SBV}}}$, i.e., in the event of subversion. In particular, if Π detects subversion of Σ_{SBV}, it may simply output an error message $\perp$. However, for Π to be able to meaningfully use Σ_{SBV} as a black box, we do assume that Σ_{SBV} respects the interface of the corresponding genuine scheme Σ_{GNN}, i.e., it has the same set of algorithms with the same number and space of input and outputs.

We follow [HH09] and [RTYZ16] in giving a definition for security of standard cryptographic schemes. Our definition is generic enough to capture regular security games, but it is adapted in a way that parameters P and samples S of the scheme can be provided as inputs to the security game. Looking ahead, this is done to model the parameter and sample generation as genuine, non-subverted computations. One may think of parameters P_p and P_s as for instance being the public and secret key, respectively. For an ordinary (non-self-guarding) scheme, S is empty.

Definition 8.1 (Cryptographic Game)**.** *A game-based security property for a scheme* Π *is defined by a probabilistic algorithm* Game *and associated constants* $\alpha, \delta \in \mathbb{N}$*. On input of scheme parameters P and potentially a set of samples S, the algorithm* $\mathrm{Game}_{\Pi,P,S}$ *interacts with an adversary $\mathcal{A}$ on input of a security parameter 1^λ and outputs a bit b. We denote this interaction by $b \twoheadleftarrow \mathrm{Game}^{\mathcal{A}}_{\Pi,P,S}(1^\lambda)$. The advantage of an adversary $\mathcal{A}$ in the game* Game *is defined as:*

$$\mathsf{Adv}^{\mathrm{game}}_{\Pi,P,S}(\mathcal{A},\lambda) := \alpha \cdot \Pr\left[\mathrm{Game}^{\mathcal{A}}_{\Pi,P,S}(1^\lambda)\right] - \delta \ ,$$

where the probability is over the internal coins of the game and the adversary. We say that the scheme Π *is* Game*-secure if for any* PPT *adversary $\mathcal{A}$ and a randomized choice of parameters $P \twoheadleftarrow \Pi.\mathsf{Gen}(1^\lambda)$ and samples $S \twoheadleftarrow \Pi.\mathsf{Sample}(P)$ the advantage $\mathsf{Adv}^{\mathrm{game}}_{\Pi,P,S}(\mathcal{A},\lambda)$ is negligible.*

As a rule of thumb, for unpredictability games (e.g., one-wayness, collision resistance, unforgeability) we usually have $\alpha = 1$ and $\delta = 0$, whereas for indistinguishability games (e.g., PRG and PRF security, key-indistinguishability) we have $\alpha = 2$ and $\delta = 1$.

8.2.1 Self-Guarding Security

In the self-guarding game SGuard we consider two phases. During the first phase the challenger has access to a genuine version of the underlying scheme, i.e., Σ_{GNN}. There, the challenger can set up $\Pi^{\Sigma_{\mathsf{GNN}}}$ by generating parameters and creating n samples from Σ_{GNN} and storing them in a collection S (which are meant to be chosen and used by the user later in real executions).

Afterwards, the challenger starts the second phase of the self-guarding game by calling the adversary $\mathcal{A}(\mathtt{subv}, P_p)$ on a command $\mathtt{subv}$ and public parameters P_p, giving $\mathcal{A}$ the opportunity to provide an arbitrary (while still respecting the original interface) implementation Σ_{SBV} for Σ and a state $st \in \{0,1\}^*$. In the subsequent steps, the challenger will either use the original algorithm Σ_{GNN} or the subverted version Σ_{SBV}. The choice is made based on a fixed value $\beta \in \{\mathsf{GNN}, \mathsf{SBV}\}$. In either case the challenger runs the adversary $\mathcal{A}(\mathtt{break}, P_p, st)$ on a command $\mathtt{break}$, the public parameters and the state obtained from the subverting adversary to play the security game Game for the scheme Π^{Σ_β} with parameters P and secure samples S.

The self-guarding ability of Π under subversion of Σ now states that the adversary's success probability in winning the game Game should not increase significantly when Σ is subverted. In other words, winning should not depend significantly on the value of β, which indicates if the original or the subverted algorithm is used.

```
Game $\text{SGuard}(\text{Game})^{\mathcal{A},\beta}_{\Pi^\Sigma}(1^\lambda)$ with $\beta \in \{\mathsf{GNN}, \mathsf{SBV}\}$

                — trusted setup phase —
$(P_s, P_p) \twoheadleftarrow \Pi.\mathsf{Gen}^{\Sigma_{\mathsf{GNN}}}(1^\lambda)$
$S \twoheadleftarrow \Pi.\mathsf{Sample}^{\Sigma_{\mathsf{GNN}}}(P_s, P_p)$

                — subversion phase —
$\Sigma_{\mathsf{SBV}}, st \twoheadleftarrow \mathcal{A}(\mathtt{subv}, P_p)$
$b \twoheadleftarrow \text{Game}^{\mathcal{A}(\mathtt{break}, P_p, st)}_{\Pi^{\Sigma_\beta}, P_s, P_p, S}(1^\lambda)$
return $b$
```

Figure 8.1: Game for self-guarding of Π against subversion of Σ.

Definition 8.2 (Self-guarding against Subversion)**.** *Let* Σ *and* Π *be cryptographic schemes, and let* Game *be a security game for* Π^Σ*. The advantage of an adversary* $\mathcal{A}$ *in the self-guarding game of Figure 8.1 is defined by:*

$$\mathsf{Adv}^{\text{sguard(game)}}_{\Pi^\Sigma}(\mathcal{A}, \lambda) := \Pr\left[\text{SGuard}(\text{Game})^{\mathcal{A},\mathsf{SBV}}_{\Pi^\Sigma}(1^\lambda)\right] - \Pr\left[\text{SGuard}(\text{Game})^{\mathcal{A},\mathsf{GNN}}_{\Pi^\Sigma}(1^\lambda)\right].$$

We say that Π^Σ *is* self-guarding *with respect to* Game *against subversion of* Σ *if for all* PPT *adversaries* $\mathcal{A}$*, the advantage* $\mathsf{Adv}^{\text{sguard(game)}}_{\Pi^\Sigma}(\mathcal{A}, \lambda)$ *is negligible.*

Intuitively, the above definition requires that the security of a self-guarding scheme Π^Σ should not decrease noticeably if an adversary subverts the underlying primitive Σ. As discussed before, the simplicity of the guarding performed by Π is crucial in practical applications. In particular, we assume that Π does not implement its own secure version of Σ, nor does it implement "heavy" detection procedures. This also implies that Π is usually not able to verify correctness of Σ itself, still allowing the adversary to modify Σ at will.

Our definition is (almost) non-adaptive in the sense that the subverted algorithm is chosen before the actual security game starts, but it may depend on the public parameters. This complies with some efforts in the literature, such as the subversion-resistant signature scheme in [RTYZ16] where the subverted algorithm may not even depend on the signer's public key. The subversion attacks on symmetric encryption schemes in [BPR14, DFP15] are also non-adaptive in nature. Such notions provide a basic level of robustness in the setting of mass surveillance where targeted attacks may be too cumbersome to mount. At the same time, targeted attacks may still be an important aspect, e.g., when the signer is a certification authority such that forging signatures allows to create arbitrary certificates. We stress that there are adaptive notions in the literature, for example for the subversion-resistant signature schemes by Ateniese, Magri, and Venturi [AMV15], but in general the distinction is not explicit.

Our notion is able to capture a type of misbehavior that happens after a certain point in time. While we do not have a notion of time in our model, the adversary can in principle provide an algorithm which uses the original algorithm as a subroutine and only enters a malicious mode after some calls, if the algorithm can be stateful and, say, maintain a counter. Similarly input-triggers can also be implemented directly in the subverted algorithm.

8.3 Self-Guarding Public-Key Encryption

In this section we show how to build a self-guarding encryption scheme, with respect to IND-CPA security, from any homomorphic encryption scheme. The guarding mechanism is simple, efficient, and for typical instantiations, such as under ElGamal encryption, it does not need to perform any modular exponentiation, nor to change anything on the decryptor's side. Another gain is that the solution enjoys security even in case the subverted encryption algorithm keeps state, a property which is usually hard to achieve. If the subverted scheme is stateful, we can, however, only encrypt as long as a fresh sample is available, since a stateful algorithm can store the used samples.

The idea of our generic construction of a self-guarding scheme $\mathsf{HE}^{\mathrm{sg}}$, described in Figure 8.2, is as follows. The sample generator encrypts multiple ciphertexts of random messages $m_{\$,i}$ and outputs them at once. At this point, the encryption algorithm still complies with the specification, such that the samples are valid encryptions. Since we do not need any specific order of these samples, we will store them in a queue structure, which we can access with the usual enq and deq commands, and check if the queue is empty via $\mathsf{is\text{-}empty}$.

When encrypting a given message m, we call the (now potentially subverted) encryption algorithm to encrypt the message $m \circ m_{\$,i}$ for a fresh message $m_{\$,i}$ from the sample queue. The idea is that the subverted algorithm then only gets to "see" a random (i.e., blinded) message when producing the ciphertext. To decrypt a ciphertext, one needs to unblind the message using the homomorphic property, dividing out the ciphertext for $m_{\$,i}$. For instance, when we use ElGamal encryption this corresponds to two modular multiplications and an inversion in the group. Remarkably, we do not need to be able to distinguish valid and invalid ciphertexts returned by the encryption algorithm, which saves us for example from performing an exponentiation for such a check for ElGamal encryption. Note that although re-randomizing samples allows for an unlimited number of secure encryptions, such involved techniques, which quasi means to implement one's own encryption algorithm, should be avoided.

```
$\mathsf{HE}^{\mathrm{sg}}.\mathsf{Gen}(1^\lambda)$
$(\mathrm{sk},\mathrm{pk}) \twoheadleftarrow \mathsf{HE}.\mathsf{Gen}(1^\lambda)$
return $(\mathrm{sk},\mathrm{pk})$
```

```
$\mathsf{HE}^{\mathrm{sg}}.\mathsf{Sample}(\mathrm{pk})$
$S \leftarrow \square$
for $i = 1..n$ do
    $m_{\$,i} \twoheadleftarrow M$
    $c_{\$,i} \twoheadleftarrow \mathsf{HE}.\mathsf{Enc}(\mathrm{pk}, m_{\$,i})$
    $\mathsf{enq}(S, (m_{\$,i}, c_{\$,i}))$
return $S$
```

```
$\mathsf{HE}^{\mathrm{sg}}.\mathsf{Enc}(\mathrm{pk}, S, m)$
if $\mathsf{is\text{-}empty}(S)$ then
    return $\perp$
$(m_\$, c_\$) \leftarrow \mathsf{deq}(S)$
$c \twoheadleftarrow \mathsf{HE}.\mathsf{Enc}(\mathrm{pk}, m \circ m_\$)$
$c^{\mathrm{sg}} \leftarrow c \diamond c_\$^{-1}$
return $c^{\mathrm{sg}}$
```

```
$\mathsf{HE}^{\mathrm{sg}}.\mathsf{Dec}(\mathrm{sk}, c)$
$m \leftarrow \mathsf{HE}.\mathsf{Dec}(\mathrm{sk}, c)$
return $m$
```

Figure 8.2: Self-guarding encryption scheme $\mathsf{HE}^{\mathrm{sg}}$ from homomorphic encryption scheme HE.

```
Game $\text{IND-CPA}^{\mathcal{A}}_{\mathsf{E},P,S}(1^\lambda)$
$b \twoheadleftarrow \{0,1\}$
$b' \twoheadleftarrow \mathcal{A}^{\mathrm{LoR}(P,S,b,\cdot,\cdot)}(P_p)$
return $(b = b')$
```

```
$\mathrm{LoR}(P,S,b,m_0,m_1)$
if $|m_0| \neq |m_1|$ then return $\perp$
$c \twoheadleftarrow \mathsf{E.Enc}(P,S,m_b)$
return $c$
```

Figure 8.3: IND-CPA game for encryption schemes E.

Correctness. As long as fresh samples are available and the underlying scheme HE is not under a subversion attack, correctness of our encryption scheme $\mathsf{HE}^{\mathrm{sg}}$ follows from correctness and the homomorphic property (which also implies that $\big(\mathsf{Enc}(\mathrm{pk}, m)\big)^{-1}$ and $\mathsf{Enc}(\mathrm{pk}, m^{-1})$ have the same distribution for all pk and m) of the underlying $\mathsf{HE}_{\mathsf{GNN}}$. Considering that the encryption algorithm $\mathsf{HE}^{\mathrm{sg}}.\mathsf{Enc}$ practically stops working afterwards, correctness beyond that point is clearly not provided.

Before proving security of our construction, we formally define the notion of IND-CPA follows the common left-or-right security game and is given in Figure 8.3 in our terminology. We capture security for both public-key and private-key encryption simultaneously, by setting $P_p = \mathrm{pk}$ and $P_s = \mathrm{sk}$ resp. $P_p = \perp$ and $P_s = \mathrm{k}$. Looking ahead, the notion for private-key encryption will be used in Section 8.4. Recall once more that the game basically describes the second phase of substitution attacks. Also, in the self-guarding game of Figure 8.1, if the adversary always chooses $\mathsf{E}_{\mathsf{SBV}} = \mathsf{E}_{\mathsf{GNN}}$ and S is empty, we obtain the standard security notions without a substitution attack.

Theorem 8.1. *The encryption scheme* $\mathsf{HE}^{\mathrm{sg}}$ *from Figure 8.2 is self-guarding with respect to* IND-CPA*-security against subversion of the underlying scheme* HE*, if* HE *is an* IND-CPA*-secure homomorphic encryption scheme.*

Proof. Assume that adversary $\mathcal{A}$ plays the self-guarding game $\mathrm{SGuard}(\text{IND-CPA})^{\mathcal{A},\beta}_{\mathsf{HE}^{\mathrm{sg}}}$. We argue in the following that $\mathcal{A}$'s probabilities for predicting the secret bit b in the two settings, $\beta = \mathsf{GNN}$ and $\beta = \mathsf{SBV}$, are almost equal.

The case $\beta = \mathsf{GNN}$. Consider first the case that the security game uses a genuine scheme $\mathsf{HE}_{\mathsf{GNN}}$. Then $\mathcal{A}$'s probability of predicting b is negligibly close to $\frac{1}{2}$. To see this, note that $\mathcal{A}$, upon receiving pk, provides the subverted algorithm $\mathsf{HE}_{\mathsf{SBV}}$. This encryption algorithm is then ignored. It follows that in each challenge query m_0, m_1 of $\mathcal{A}$, where the remaining number of samples is not exhausted yet, the adversary receives an encryption of m_0 or of m_1, only computed as the product of genuine ciphertexts, derived via $\mathsf{Enc}_{\mathsf{GNN}}$. Since the homomorphic property implies that this has the same distribution as a fresh encryption of the message, it follows immediately from the IND-CPA-security of HE that the probability of predicting b is negligibly close to $\frac{1}{2}$.

The case $\beta = \mathsf{SBV}$. Next, consider the case that the security game now uses the subverted algorithm $\mathsf{HE}_{\mathsf{SBV}}$ in the second phase. In each such query for a pair of messages m_0, m_1 we use a message $m_\$$ to mask the challenge message and encrypt the masked message under the subverted algorithm. The message $m_\$$ has been chosen at random and encrypted under the genuine algorithm $\mathsf{Enc}_{\mathsf{GNN}}(\mathrm{pk}, m_\$)$ in the sampling phase. The final ciphertext c^{sg} is derived by multiplying the first ciphertext with the inverse of the second one.

Suppose now that, instead of using $c_\$$, which is the encryption of $m_\$$ under $\mathsf{Enc}_{\mathsf{GNN}}$, in each challenge query, we use the encryption of an independent random message $m'_\$$ and compute the final ciphertext from that. Then the adversary cannot possibly learn anything more than the random message $m_\$ \circ m_0$ or $m_\$ \circ m_1$, possibly leaked through the subverted algorithm $\mathsf{Enc}_{\mathsf{SBV}}$, and the encryption of an independent random message $m'_\$$. This also covers the case that the ciphertext in the challenge phase is malformed, e.g., does not belong to the correct subgroup. In other words, each challenge query yields an answer which is independently distributed from the bit b. The adversary's success probability for predicting b is then at most the guessing probability of $\frac{1}{2}$.

It remains to argue that encrypting $m'_\$$ instead of $m_\$$ does not significantly add to $\mathcal{A}$'s success probability. This follows directly from the IND-CPA-security of HE by using $\mathcal{A}$ to build an adversary $\mathcal{B}$ against HE. The adversary $\mathcal{B}$ initially receives a public key pk and forwards it to $\mathcal{A}$. Adversary $\mathcal{A}$ then provides the subverted algorithm $\mathsf{HE}_{\mathsf{SBV}}$. Then $\mathcal{B}$ simulates the rest of the self-guarding game, picking the bit $b \twoheadleftarrow \{0,1\}$ itself but trying to predict an external secret bit b' in its IND-CPA-game.

Adversary $\mathcal{B}$ answers each challenge query m_0, m_1 of $\mathcal{A}$ for a message with a ciphertext c^{sg}, computed as follows. Adversary $\mathcal{B}$ picks a random message $m_\$ \twoheadleftarrow M$, computes $m_\$ \circ m_0$ (for $b = 0$) resp. $m_\$ \circ m_1$ (for $b = 1$). It encrypts this message under $\mathsf{Enc}_{\mathsf{SBV}}$ and pk to get a ciphertext c. It picks another random message $m'_\$$ and forwards $m_\$$ and $m'_\$$ to its own challenge oracle to get a ciphertext $c_\$$. It returns $c^{\mathrm{sg}} \leftarrow c \diamond c_\$^{-1}$ to adversary $\mathcal{A}$. When $\mathcal{A}$ eventually outputs a guess for bit b, adversary $\mathcal{B}$ checks if the guess is correct. If so, it outputs the prediction 0 for bit b', else it outputs 1.

For the analysis note that, if $\mathcal{A}$'s success probability drops significantly from the case that one correctly encrypts $m_\$$ to the case where one encrypts $m'_\$$, then this would immediate a contradiction to the IND-CPA-security of HE. That is, letting $\mathcal{B} = 0$ and $\mathcal{B} = 1$ denote the events that $\mathcal{B}$ outputs 0 and 1, respectively, and $b_\mathcal{A}$ denote the event that $\mathcal{A}$ predicts b correctly, we have:

$$\begin{aligned}
\Pr[\mathcal{B} = b'] &= \frac{1}{2} \cdot \Pr[\mathcal{B} = 0 \mid b' = 0] + \frac{1}{2} \cdot \Pr[\mathcal{B} = 1 \mid b' = 1] \\
&= \frac{1}{2} \cdot \Pr[\mathcal{B} = 0 \mid b' = 0] + \frac{1}{2} \cdot (1 - \Pr[\mathcal{B} = 0 \mid b' = 1]) \\
&= \frac{1}{2} + \frac{1}{2} \cdot (\Pr[\mathcal{B} = 0 \mid b' = 0] - \Pr[\mathcal{B} = 0 \mid b' = 1]) \\
&= \frac{1}{2} + \frac{1}{2} \cdot (\Pr[b_\mathcal{A} \mid b' = 0] - \Pr[b_\mathcal{A} \mid b' = 1]) \,.
\end{aligned}$$

The difference in the parentheses is non-negligible, by assumption, such that our algorithm $\mathcal{B}$ has a non-negligibly larger prediction probability than $\frac{1}{2}$. □

8.4 Self-Guarding Symmetric Encryption

In this section we present a self-guarding mechanism for randomized symmetric encryption without restricting the subversion strategy. In particular, our construction is self-guarding against biased-ciphertext attack (cf. [BPR14]) and stateful subversions. Unlike public-key encryption, a subverted symmetric encryption algorithm has access to the secret key and can potentially leak it. Therefore simply masking the plaintext cannot fully neutralize the attack. Interestingly, we can thwart leakage

```
$\mathsf{E}^{\mathrm{sg}}.\mathsf{Gen}(1^\lambda)$
    $\mathrm{k} \twoheadleftarrow \mathsf{E}.\mathsf{Gen}(1^\lambda)$
    return $\mathrm{k}$

$\mathsf{E}^{\mathrm{sg}}.\mathsf{Sample}(\mathrm{k})$
    $S \leftarrow \square$
    for $i = 1..n$ do
        $m_{\$,i} \twoheadleftarrow \{0,1\}^\ell$
        $c_{\$,i} \twoheadleftarrow \mathsf{E}.\mathsf{Enc}(\mathrm{k}, m_{\$,i})$
        $\mathsf{enq}(S, (m_{\$,i}, c_{\$,i}))$
    return $S$

$\mathsf{E}^{\mathrm{sg}}.\mathsf{Enc}(\mathrm{k}, S, m)$
    $c \twoheadleftarrow \mathsf{E}.\mathsf{Enc}(\mathrm{k}, m)$
    if $\mathsf{is\text{-}empty}(S)$
        or $|m| > \ell - x - 1$
        or $|c| > \ell - 1$ then
       return $\perp$
    $(m_\$, c_\$) \leftarrow \mathsf{deq}(S)$
    $c^{\mathrm{sg}} \leftarrow [c \| 10^{\ell - |c| - 1}] \oplus m_\$$
    return $(c^{\mathrm{sg}}, c_\$)$

$\mathsf{E}^{\mathrm{sg}}.\mathsf{Dec}(\mathrm{k}, (c^{\mathrm{sg}}, c_\$))$
    $m_\$ \leftarrow \mathsf{E}.\mathsf{Dec}(\mathrm{k}, c_\$)$
    $c \| 10\ldots0 \leftarrow c^{\mathrm{sg}} \oplus m_\$$
    $m \leftarrow \mathsf{E}.\mathsf{Dec}(\mathrm{k}, c)$
    return $m$
```

Figure 8.4: Self-guarding symmetric encryption scheme E^{sg} built upon a symmetric encryption scheme E with maximum ciphertext expansion of e bits for each message.

by using a random message to mask the output. We then send along the encryption of this random message (computed in the trusted sampling phase) so that the resulting ciphertext can be decrypted.

The computational overhead of the proposed scheme is small. For encryption we basically need a reliable $\oplus$-operation, and for decrypting we need two calls to the decryption of the underlying scheme, and again a trustworthy implementation of $\oplus$. On the downside, we can only encrypt securely as long as a fresh sample is available. Moreover, the self-guarding decryption algorithm differs from the underlying decryption algorithm, and also we can only encrypt messages that are shorter than the sampled messages.

Our construction E^{sg} from Figure 8.4 is built upon an arbitrary IND-CPA-secure encryption scheme E that has a maximum ciphertext expansion x. Here, the ciphertext expansion describes the maximum number of extra bits in the ciphertext to encrypt a message, e.g., to store a random IV.

In the sampling phase we generate multiple ciphertexts of random messages $m_{\$,i}$ of bit length ℓ. At this point, the encryption algorithm still complies with the specification, such that the samples are valid and secure encryptions $c_{\$,i}$. Similar to the previous section, we store the samples in a queue structure S, such that we can access the queue with the usual enq and deq commands, and check if it is empty via $\mathsf{is\text{-}empty}$. The sample messages are used as a one-time-pad key to hide the ciphertext produced by a potentially malicious implementation. This is intuitively the reason why we need sample messages that are at least as long as the ciphertexts produced by E. To deal with potentially shorter ciphertexts we use the common padding $10\ldots0$ to expand all ciphertexts to equal length. To make sure that the receiver is able to decrypt, the honest encryption of the sample message is sent along with the actual encryption.

In theory we are able to lift the restriction on the new message space by using a pseudorandom function to expand the sample messages. However, we chose to keep the construction simple and the number of trusted components to a minimum.

Correctness. As long as fresh samples are available and the construction is not under a subversion attack, correctness of our symmetric encryption scheme E^{sg} for messages with maximum length of $\ell - x - 1$, follows trivially from correctness of the underlying symmetric encryption scheme $\mathsf{E}_{\mathsf{GNN}}$. Since the encryption algorithm $\mathsf{E}^{\mathrm{sg}}.\mathsf{Enc}$ essentially aborts if no more samples are left, correctness is not given beyond that point.

Next we prove that our construction is self-guarding with respect to the IND-CPA game, given that the underlying encryption scheme is IND-CPA during the initial trusted phase.

Theorem 8.2. *The symmetric encryption scheme* E^{sg} *from Figure 8.4 is self-guarding with respect to* IND-CPA*-security against subversion of the underlying scheme* E*, if* E *is an* IND-CPA*-secure symmetric encryption scheme with a maximum ciphertext expansion of* $x \in \mathbb{N}$*.*

Proof. Consider an adversary $\mathcal{A}$ playing the self-guarding game $\text{SGuard(IND-CPA)}_{\mathsf{E}^{\mathrm{sg}}}^{\mathcal{A},\beta}$. We show that $\mathcal{A}$'s probability of predicting the secret bit b in the two settings, $\beta = \mathsf{GNN}$ and $\beta = \mathsf{SBV}$, cannot differ significantly.

The case $\beta = \mathsf{GNN}$**.** Consider first the case that the security game uses the actual scheme $\mathsf{E}_{\mathsf{GNN}}$. Then $\mathcal{A}$'s probability of predicting b is negligibly close to $\frac{1}{2}$, because the subverted algorithm $\mathsf{E}_{\mathsf{SBV}}$ is ignored, such that on each query m_0, m_1 (of at most $\ell - x - 1$ bits each) the adversary receives an encryption of m_0 or of m_1, where the (padded) ciphertext is xored with a random message $m_\$$, which is at least as long as the ciphertext. The adversary also receives the genuine encryption of $m_\$$ (i.e., $c_\$$) from S, previously computed using $\mathsf{Enc}_{\mathsf{GNN}}$.

For the genuine encryption algorithm $\mathsf{Enc}_{\mathsf{GNN}}$ we actually get a "twofold" secure encryption. First, and this suffices for the formal argument, the encryption of the challenge message under the IND-CPA-secure scheme $\mathsf{Enc}_{\mathsf{GNN}}$ already hides the secret bit b. This can be straightforwardly formalized by simulating the extra layer of the encryption with $m_\$$ and creating the ciphertext $c_\$$ with the help of the encryption oracle in the IND-CPA game, also keeping track of the number of available samples. At the same time, we could also argue along the security of $c_\$$, hiding $m_\$$, which in turn is then used to mask the ciphertext. Hence, we can conclude that the probability of predicting b in the self-guarding game for $\mathsf{Enc}_{\mathsf{GNN}}$ is negligibly close to $\frac{1}{2}$ by the IND-CPA-security of $\mathsf{E}_{\mathsf{GNN}}$.

The case $\beta = \mathsf{SBV}$**.** Next, consider the case that the security game uses the subverted algorithm $\mathsf{E}_{\mathsf{SBV}}$ in the challenge queries. In each such query for a pair of messages m_0, m_1 we, hence, encrypt the message under the subverted algorithm and get a possibly malicious ciphertext c. We check the validity of the length of the ciphertext, and then add the message $m_\$$ to the (padded) ciphertext to obtain c^{sg}. The result, together with the genuine encryption $c_\$ \twoheadleftarrow \mathsf{Enc}_{\mathsf{GNN}}(\mathrm{k}, m_\$)$, which was computed during the sampling phase, is output as the final ciphertext $(c^{\mathrm{sg}}, c_\$)$.

Suppose now that, instead of using the encryption $c_\$$ of $m_\$$ under $\mathsf{Enc}_{\mathsf{GNN}}$, we use an independent random message $m'_\$ \twoheadleftarrow \{0,1\}^\ell$ for masking and send its encryption as the second part of the ciphertext. Then, the adversary would learn no more than the encryption of an independent random message $m'_\$$ under a secure encryption algorithm, since c^{sg} is hidden by another random message $m_\$$. In other words, each challenge query yields an answer which is independently distributed from the bit b. The adversary's success probability for predicting b is then at most the guessing probability of $\frac{1}{2}$.

We are once more left to argue that encrypting $m'_\$$ instead of $m_\$$ does not significantly contribute to $\mathcal{A}$'s success probability. This follows once more from the IND-CPA-security of E. From adversary $\mathcal{A}$ we build an adversary $\mathcal{B}$ against E. Adversary $\mathcal{A}$ first provides the subverted algorithm $\mathsf{E}_{\mathsf{SBV}}$. Then $\mathcal{B}$ simulates the rest of the self-guarding game, picking the bit $b \twoheadleftarrow \{0,1\}$ itself but playing against an external secret bit b' in its IND-CPA-game.

Adversary $\mathcal{B}$ answers each challenge query m_0, m_1 of $\mathcal{A}$ of length at most $\ell - x - 1$ as follows. It first checks the length restrictions and makes sure the number of samples is not yet exceeded. If so, $\mathcal{B}$ encrypts the message under $\mathsf{Enc}_{\mathsf{SBV}}$ and xors the padded result with a randomly chosen message $m_\$ \twoheadleftarrow \{0,1\}^\ell$ to get a ciphertext c^{sg}. It picks another random message $m'_\$$ of the same length and forwards $m_\$$ and $m'_\$$ to its own challenge oracle and receives $c_\$$. It returns $(c^{\mathrm{sg}}, c_\$)$ to $\mathcal{A}$.

When $\mathcal{A}$ eventually outputs a guess for bit b, adversary $\mathcal{B}$ checks if the prediction is correct. If so, it outputs 0 for bit b', else it outputs 1. The analysis is now identical to the case of the public-key scheme (see proof of Theorem 8.1) and omitted here. By assumption, our algorithm $\mathcal{B}$, therefore, has a non-negligibly larger winning probability than $\frac{1}{2}$. □

8.5 Self-Guarding Signatures

A subverted signing algorithm may try to leak information about the secret key, or a different signature. A subverted verification algorithm can enable attacks in a trivial way, e.g., by accepting invalid signatures. Here, we focus on the former scenario, which is arguably both more dangerous and more realistic. We assume that the verification algorithm is not subverted and (even subverted) signatures will ultimately be verified using genuine algorithms.

In the domain of reverse firewalls, the idea of Ateniese et al. [AMV15] is to have any signature created by the signer verified by the (trusted) reverse firewall with respect to the public key and, if correct, re-randomized before it is sent out. [1] The combination of re-randomization and verification of signatures prevents against subliminal leakage even when triggered by malicious inputs. For unique signatures, which have only one valid signature for each message under the public key, the re-randomization step is trivial and can be omitted.

It is possible to apply the same idea in our self-guarding setting, *if the verification step and the re-randomization step can be implemented robustly.* In this case the signer generates a signature, verifies it with the trustworthy verification step, and re-randomizes it securely. This approach may be viable in some settings, e.g., when verifying FDH-RSA signatures with low exponents such as $e = 2^{16} + 1$, where only a few modular multiplications and, what is more critical, a hash evaluation would need to be carried out safely. Still, in other scenarios implementing the full verification procedure securely may just be beyond the signer's capabilities, whereas storing a number of message and signature pairs reliably is usually a much easier task than correctly implementing cryptographic code. We, therefore, propose an alternative solution below.

Our self-guarding signature scheme is built upon a deterministic signature scheme. Here we consider only stateless subversions. In the initialization phase random messages $m_\$$ and corresponding signatures $\sigma_\$$ are computed and stores in S. Later, when signing a given message m, we hand over the two messages $m_\$$ and $m_\$ \oplus [m \| \sigma_\$]$ in random order to the potentially subverted signing algorithm such that if the algorithm deviates for one of the two signatures, we will detect this with probability $\frac{1}{2}$ and abort forever. Including the signature $\sigma_\$$ in the second message prevents combination attacks against unforgeability, and requires that $m_\$$ is long enough to range over $m \| \sigma_\$$. Note that deterministic signatures can produce shorter signatures, e.g., if first hashing the message.

[1] Interestingly, Ateniese et al. [AMV15] define re-randomization with respect to the original signature algorithm, but the solution presumably requires re-randomization of maliciously generated signatures under the subverted algorithm.

Since we assume that the substituted algorithm is stateless such that both messages look equally random to it, even if we re-use the random message across multiple signature creations. To increase the detection probability to overwhelming we will repeat the above λ times with independent key pairs. The independence of the keys ensures that, even if the adversary manages to leak information about some signing keys, the other keys are still fresh. Although the trusted samples are used here to enforce security via detection, the detection strategy is very simple and avoids the problems with usual detectors discussed in the introduction to this chapter 8.1.

More formally, our self-guarding signature scheme $\mathsf{S}^{\mathrm{sg}} = (\mathsf{KGen}^{\mathrm{sg}}, \mathsf{Sig}^{\mathrm{sg}}, \mathsf{Vf}^{\mathrm{sg}})$ is based on a deterministic signature scheme $\mathsf{S} = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$ and works as follows. The key generation algorithm $\mathsf{KGen}^{\mathrm{sg}}(1^\lambda)$ creates λ key pairs $(\mathrm{sk}_i, \mathrm{pk}_i) \twoheadleftarrow \mathsf{S}.\mathsf{KGen}(1^\lambda)$ of the underlying signature scheme. It sets $\mathrm{sk}^{\mathrm{sg}} \leftarrow (\mathrm{sk}_1, \ldots, \mathrm{sk}_\lambda)$ and $\mathrm{pk}^{\mathrm{sg}} \leftarrow (\mathrm{pk}_1, \ldots, \mathrm{pk}_\lambda)$ and outputs them together with a flag *err* initially set to 0, indicating that no invalid signature was detected.

In the initialization phase we also pick λ random messages $m_{\$,1}, \ldots, m_{\$,\lambda} \leftarrow \{0,1\}^\ell$ and create the signatures $\sigma_{\$,1}, \ldots, \sigma_{\$,\lambda}$ for all messages. We store the pairs $(m_{\$,i}, \sigma_{\$,i})$ in the sample queue S. The common bit length ℓ of the messages $m_{\$,i}$ determines an upper bound on the messages m which can later be signed. Namely, any message m can be at most the length of $m_{\$,i}$, minus the bit length for signatures, where we assume without loss of generality that all signatures are of equal length s. Longer messages m may be hashed first, outside of the signing algorithm. For sake of a cleaner presentation we assume below that all input messages are tightly of length $\ell - s$. With padding for shorter messages, the proof can be transferred to a more general setting easily.

When signing a message m under the possibly subverted algorithm Sig, with the key $\mathrm{sk}^{\mathrm{sg}}$ and the samples S, the self-guarding signing algorithm does the following. If no invalid signatures were detected so far, i.e., *err* is still 0, for each $i = 1..\lambda$ pick a random bit $b_i \twoheadleftarrow \{0,1\}$ and call the signing algorithm twice, one time for $\mathrm{sk}_i, m_{\$,i}$ and the other time for $\mathrm{sk}_i, m_{\$,i} \oplus [m \| \sigma_{\$,i}]$. Use this order if $b_i = 0$ and the reverse order if $b_i = 1$. Let σ_i be the returned signature for the message $m_{\$,i} \oplus [m \| \sigma_{\$,i}]$. For each i check that the provided signature for $m_{\$,i}$ equals $\sigma_{\$,i}$. If not, abort after setting *err* to 1. Else output $\sigma^{\mathrm{sg}} \leftarrow (m_{\$,1}, \sigma_{\$,1}, \sigma_1, \ldots, m_{\$,\lambda}, \sigma_{\$,\lambda}, \sigma_\lambda)$ as the signature.

Verification is straightforward. For each i build the message $m_{\$,i} \oplus [m \| \sigma_{\$,i}]$ from the given message m and the data in the signature, and verify the signature σ_i with respect to pk_i, as well as the signature $\sigma_{\$,i}$ for $m_{\$,i}$. Accept iff all verification steps succeed.

Regarding security, we aim at showing self-guarding of our scheme with respect to unforgeability against chosen-message attacks, i.e., EUF-CMA. We define EUF-CMA in our terminology in Figure 8.6 (with $\alpha = 1$ and $\delta := 0$). In the self-guarding game of Figure 8.1, if $S = \square$ and the adversary always chooses $\mathsf{S}_{\mathsf{SBV}} = \mathsf{S}_{\mathsf{GNN}}$, we obtain the standard security notion without a substitution attack.

For the security proof we need another property of the underlying signature scheme, namely, that the (equal length) signature strings are never zero-bitstrings. This can be easily achieved by prepending or appending a bit ‘1’ to any signature and verifying that this bit really appears in the signature. We call such signature schemes *zero-evading.*

Theorem 8.3. *The signature scheme* S^{sg} *from Figure 8.5 is self-guarding with respect to EUF-CMA security against stateless subversion of the signing algorithm* $\mathsf{S}.\mathsf{Sig}$*, if* S *is a deterministic, EUF-CMA-secure, and zero-evading signature scheme.*

Proof. Consider an adversary $\mathcal{A}$ playing the self-guarding game $\mathrm{SGuard(EUF\text{-}CMA)}_{\mathsf{S}^{\mathrm{sg}}}^{\mathcal{A},\beta}$. We can

```
$\mathsf{S}^{\text{sg}}.\mathsf{Gen}(1^\lambda)$
for $i = 1..\lambda$ do
    $(\text{sk}_i, \text{pk}_i) \twoheadleftarrow \mathsf{S.KGen}(1^\lambda)$
$(\text{sk}^{\text{sg}}, \text{pk}^{\text{sg}}) \leftarrow ((\text{sk}_1, \ldots, \text{sk}_\lambda), (\text{pk}_1, \ldots, \text{pk}_\lambda))$
$err \leftarrow 0$
return $(\text{sk}^{\text{sg}}, \text{pk}^{\text{sg}}, err)$

$\mathsf{S}^{\text{sg}}.\mathsf{Sample}(\text{sk}^{\text{sg}})$
$S \leftarrow \Box$
for $i = 1..\lambda$ do
    $m_{\$,i} \twoheadleftarrow \{0,1\}^\ell$
    $\sigma_{\$,i} \leftarrow \mathsf{S.Sig}(\text{sk}_i, m_{\$,i})$
    $\mathsf{enq}(S, (m_{\$,i}, \sigma_{\$,i}))$
return $S$

$\mathsf{S}^{\text{sg}}.\mathsf{Vf}(\text{pk}^{\text{sg}}, m, \sigma)$
$\sigma = (m_{\$,1}, \sigma_{\$,1}, \sigma_1, \ldots, m_{\$,\lambda}, \sigma_{\$,\lambda}, \sigma_\lambda)$
$d \leftarrow [|m| = \ell - s]$
for $i = 1..\lambda$ do
    $d \leftarrow d \wedge |m_{\$,i}| = \ell \wedge |\sigma_{\$,i}| = s$
    $d \leftarrow d \wedge \mathsf{S.Vf}(\text{pk}_i, m_{\$,i}, \sigma_{\$,i})$
    $d \leftarrow d \wedge \mathsf{S.Vf}(\text{pk}_i, m_{\$,i} \oplus [m \| \sigma_{\$,i}], \sigma_i)$
return $d$
```

```
$\mathsf{S}^{\text{sg}}.\mathsf{Sig}(\text{sk}^{\text{sg}}, m, S, err)$
if $err = 1$ or $|m| \neq \ell - s$ then
    return $\perp$
$S' \leftarrow S$
for $i = 1..\lambda$ do
    $(m_{\$,i}, \sigma_{\$,i}) \leftarrow \mathsf{deq}(S')$
    $b_i \twoheadleftarrow \{0,1\}$
    if $b_i = 0$ then
        $(m^0, m^1) \leftarrow (m_{\$,i}, m_{\$,i} \oplus [m \| \sigma_{\$,i}])$
    else
        $(m^0, m^1) \leftarrow (m_{\$,i} \oplus [m \| \sigma_{\$,i}], m_{\$,i})$
    $\sigma^0 \leftarrow \mathsf{S.Sig}(\text{sk}_i, m^0)$
    $\sigma^1 \leftarrow \mathsf{S.Sig}(\text{sk}_i, m^1)$
    if $\sigma^{b_i} \neq \sigma_{\$,i}$ then
        $err \leftarrow 1$
        return $\perp$
    $\sigma_i \leftarrow \sigma^{1-b_i}$
    if $|\sigma_i| \neq s$ then return $\perp$
$\sigma \leftarrow (m_{\$,1}, \sigma_{\$,1}, \sigma_1, \ldots, m_{\$,\lambda}, \sigma_{\$,\lambda}, \sigma_\lambda)$
return $\sigma$
```

Figure 8.5: Self-guarding signature scheme S^{sg} with message space $\{0,1\}^{\ell-s}$ built from signature scheme S producing signatures of length s.

```
Game $\text{EUF-CMA}^{\mathcal{A}}_{\mathsf{S},P_s,P_p,S}(1^\lambda)$
$M \leftarrow \emptyset$
$(m^*, \sigma^*) \twoheadleftarrow \mathcal{A}^{\mathsf{Sig}(P_s,S,\cdot)}(P_p)$
if $m^* \notin M$ and $\mathsf{S.Vf}(P_p, m^*, \sigma^*)$ then
    return 1
return 0

$\mathsf{Sig}(P_s, S, m)$
$M \leftarrow M \cup \{m\}$
$\sigma \twoheadleftarrow \mathsf{S.Sig}(P_s, S, m)$
return $\sigma$
```

Figure 8.6: EUF-CMA game for signature schemes S. We have $\alpha := 1$ and $\delta := 0$

show that the advantage of $\mathcal{A}$ is negligible by the security of the underlying signature scheme S, implying that $\mathcal{A}$ cannot increase its success probability noticeably by subverting S.

In the attack, as well as in the reduction below, we denote the message in the j-th signature query by m_j. The i-th signature component in the j-th query for message $m_{\$,i} \oplus [m_j \| \sigma_{\$,i}]$ is denoted as $\sigma_{i,j}$. We assume that $\mathcal{A}$ makes q signature queries. The forgery attempt is denoted by m^* and $(m^*_{\$,1}, \sigma^*_{\$,1}, \sigma^*_1, \ldots, m^*_{\$,\lambda}, \sigma^*_{\$,\lambda}, \sigma^*_\lambda)$.

Reduction to signature scheme. We construct an adversary $\mathcal{B}$ against the unforgeability of the underlying signature scheme S via a black-box reduction. Upon receiving as input a verification key pk, the adversary $\mathcal{B}$ first picks $k \twoheadleftarrow \{1, 2, \ldots, \lambda\}$ at random and sets $\mathrm{pk}_k \leftarrow \mathrm{pk}$. It generates all the other key pairs $(\mathrm{sk}_i, \mathrm{pk}_i) \twoheadleftarrow \mathsf{S.KGen}(1^\lambda)$ for $i \neq k$ itself. Then $\mathcal{B}$ picks the messages $m_{\$,i}$ and creates the signatures $\sigma_{\$,i}$, for $i \neq k$ with the help of the signing key sk_i, and for $i = k$ by calling its signature oracle. It starts the attack of $\mathcal{A}$.

Whenever $\mathcal{B}$ is supposed to create a signature, it executes the same steps as the self-guarding algorithm for any index $i \neq k$. In particular, it checks if the returned signature components for $m_{\$,i}$ match the previously sampled value. For the k-th index it uses the previously obtained oracle value $\sigma_{\$,k}$ and it now calls the external oracle to get a signature for $m_{\$,k} \oplus [m \| \sigma_{\$,k}]$; it does not need to check these answers. Algorithm $\mathcal{B}$ uses all these data to assemble the signature in the same way our signing algorithm does.

If the adversary eventually outputs a forgery for message m^* and signature $(m^*_{\$,1}, \sigma^*_{\$,1}, \sigma^*_1, \ldots, m^*_{\$,\lambda}, \sigma^*_{\$,\lambda}, \sigma^*_\lambda)$, then $\mathcal{B}$ does the following:

- If $m^*_{\$,k} = m_{\$,k}$ then output the message $m^*_{\$,k} \oplus [m^* \| \sigma^*_{\$,k}]$ together with σ^*_k as the signature.
- Else, if $m^*_{\$,k} \neq m_{\$,k} \oplus [m_j \| \sigma_{\$,k}]$ for all $j = 1, 2, \ldots, q$, then output the message $m^*_{\$,k}$ with signature $\sigma^*_{\$,k}$.
- Else, if $m^*_{\$,k} = m_{\$,k} \oplus [m_j \| \sigma_{\$,k}]$ for some j, then output the message $m^*_{\$,k} \oplus [m^* \| \sigma^*_{\$,k}]$ with the signature σ^*_k.

Analysis. We first argue that the subverted signing algorithm, with overwhelming probability, must output only valid signatures for one of the keys pk_i in the actual attack. Let us call the i-th entries $\sigma_{\$,i,j}$ and $\sigma_{i,j}$ in the j-th signature reply valid if they correspond to the signature for the messages under the specified signature algorithm. Then we claim that, with overwhelming probability, there must be some index i such that the signature entries in all queries are valid. In case of an abort, this refers to all signatures created up to the aborting query (exclusively), and else this refers to all queries.

Suppose that for each i the subverted algorithm outputs an invalid signature in some query j. This query may vary with the index i. Since the algorithm is stateless, the input messages $m_{\$,i,j}$ and message $m_{\$,i,j} \oplus [m_j \| \sigma_{\$,i,j}]$ both look random to the algorithm. Furthermore, the order of the signing request is determined by a random bit b_i, such that the algorithm creates an invalid signature for the sampled message $m_{\$,i,j}$ with probability $\frac{1}{2}$. Hence our self-guarding signature algorithm will detect this with probability $\frac{1}{2}$ and abort after setting *err* to 1.

Any detected invalid signature will lead to an immediate abort and prohibits computing future signatures, and for each key pk_i the detection probability is independent of the other case. Hence, if the adversary tries to output an invalid signature in some query for any key pk_i, our algorithm will detect this, except with probability $2^{-\lambda}$. We can, therefore, from now on condition on the event that for some index i all signature entries in all queries are valid, losing only a probability $2^{-\lambda}$ in $\mathcal{A}$'s success probability.

If there is a good index i for which the subverted algorithm never outputs a wrong signature, then $\mathcal{B}$ picks this index $k = i$ with probability $\frac{1}{\lambda}$. Given this, all signatures created via the external oracle perfectly mimic the values returned by the subverted algorithm. From now on assume that this is the case. Since any values of invalid length will lead to an abort, we assume that the signature values are of correct length.

It remains to analyze the probability that, in a good simulation, adversary $\mathcal{B}$ creates a valid signature for a fresh message. Note that a valid forgery of $\mathcal{A}$ must be for a new message m^* and must consist of a vector of valid signatures, such that each component carries a valid signature. We distinguish the three cases for $\mathcal{A}$'s output as in the output generation of $\mathcal{B}$:

- If $m^*_{\$,k} = m_{\$,k}$ (and, by determinism, therefore, $\sigma^*_{\$,k} = \sigma_{\$,k}$ for a valid signature), then we must have that $\mathcal{B}$'s output message satisfies

  $$m^*_{\$,k} \oplus [m^* \| \sigma^*_{\$,k}] = m_{\$,k} \oplus [m^* \| \sigma_{\$,k}] \neq m_{\$,k} \oplus [m_j \| \sigma_{\$,k}]$$

  for all j, since $m^* \neq m_1, \ldots, m_q$ for a successful forgery of $\mathcal{A}$. Furthermore, since $\sigma^*_{\$,k} = \sigma_{\$,k} \neq 0$ by assumption about the zero-evasion of the signature scheme, $\mathcal{B}$'s output message cannot match $m_{\$,k}$ either. We conclude that $\mathcal{B}$ has never queried its signing oracle about this message, neither in the sampling phase, nor in a signing step. But since this message is checked against σ^*_k under pk_k, adversary $\mathcal{B}$ would also win if $\mathcal{A}$ does.

- Else, if $m^*_{\$,k} \neq m_{\$,k} \oplus [m_j \| \sigma_{\$,k}]$ for all $j = 1, 2, \ldots, q$, then the message $m^*_{\$,k}$ is new; it is distinct from all queries in the signing step and also different from the query $m_{\$,k}$ in the sampling step, by the first case.

- Else, if $m^*_{\$,k} = m_{\$,k} \oplus [m_j \| \sigma_{\$,k}]$ for some j, the adversary $\mathcal{A}$ has swapped this message part from the j-th query to the other signature position for the k-th entry. But in the signature verification one checks in the other component that the message

  $$m^*_{\$,k} \oplus [m^* \| \sigma^*_{\$,k}] = m_{\$,k} \oplus [(m^* \oplus m_j) \| (\sigma_{\$,k} \oplus \sigma^*_{\$,k})]$$

  is valid. Since $m^* \neq m_j$ this message cannot match $m_{\$,k}$ for which $\mathcal{B}$ has called the oracle for the sampling step. Moreover, zero evasion implies that $\sigma^*_{\$,k}$ is not zero and, therefore, $\sigma_{\$,k} \oplus \sigma^*_{\$,k} \neq \sigma_{\$,k}$. It follows that $\mathcal{B}$ has not called its oracle about the output message in any of the signing requests either.

In summary, we have

$$\Pr\left[\mathrm{SGuard}(\text{EUF-CMA})^{\mathcal{A},\mathsf{SBV}}_{\mathsf{S}^{\mathrm{sg}\mathsf{S}}}(1^\lambda)\right] \leq \lambda \cdot \mathsf{Adv}^{\text{euf-cma}}_{\mathsf{S}}(\mathcal{B}, \lambda) + \lambda \cdot 2^{-\lambda}.$$

This is negligible if we presume unforgeability of S. □

8.6 Self-Guarding PUF-based Key Exchange

The interesting aspect of using PUFs in key exchange protocols is that one can achieve information-theoretic security, when the PUF is ideal and the time during which the adversary has access to the PUF is limited. There exist already several proposals for PUF-based key exchange protocols in this line [vD10, Rüh11, BFSK11, vR12]. However, these protocols do not withstand substitution attacks as we briefly exemplify for the case of [vR12] below. We then design a new PUF-based key exchange protocol and prove that it indeed resists subversion of the PUF. For some basics on PUFs, we refer the reader to Section 2.3.3.

At the core of any PUF-based key exchange protocol, a party, Alice, measures the PUF at a random challenge point and then sends her PUF to the other party, Bob. After making sure that Bob has received the PUF, Alice sends him the challenge through an authenticated channel.[2] Both parties use the PUF's response on this challenge to derive a shared secret key. An adversary, not yet knowing the challenge when getting access to the PUF during transmission, may evaluate it on at most polynomially many challenges. Then the adversary relays the PUF to Bob, therefore, loses access, and only afterwards learns the actual challenge used by the parties. With high probability this challenge will not be among the ones queried by the adversary before, implying that the derived key looks random to the adversary. Instead of sending a fresh PUF for each key derivation, the PUF may also be used multiple times.

Example attack on a PUF-based key exchange. Before introducing our protocol, we briefly argue that the common technique of checking validity of a PUF by verifying a challenge-response pair is vulnerable to substitution attacks. The derived PUF is stateless and encapsulates the original PUF. We describe the attack on the protocol suggested by van Dijk and Rührmair [vR12] (see Figure 8.7 below), which is secure against malicious PUFs that produce responses that are simulatable by software. Here we consider an attacker that can build a subverted (stateless) PUF by encapsulating the original PUF. When stimulated, this malicious PUF flips a coin and either returns the original response, or the all-zero string. Then, with probability $\frac{1}{4}$ it will pass the check *and* make Bob output the all-zero key, thus breaking the protocol. This process can even be derandomized by using a 2-wise independent hash function H outputting a single bit, returning the original response on challenge c if $\mathsf{H}(c) = 1$, and the zero string if $\mathsf{H}(c) = 0$.

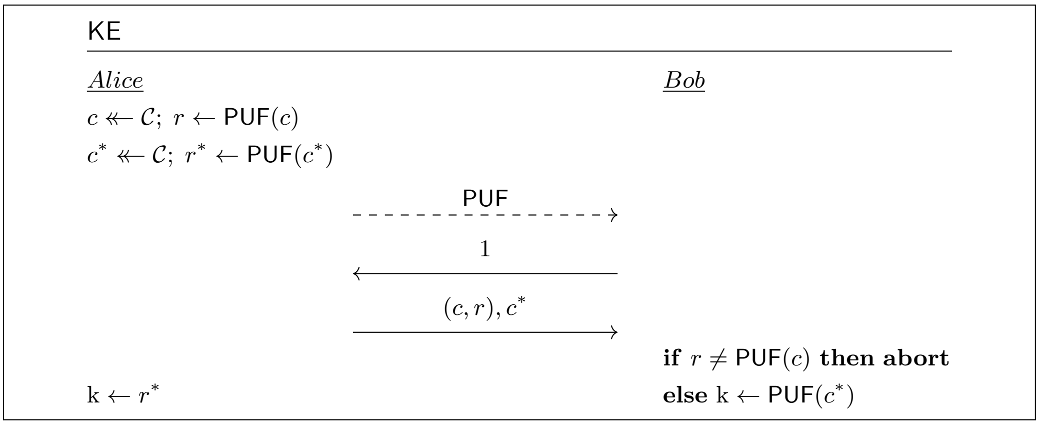

KE
Alice
Bob
$c \twoheadleftarrow \mathcal{C};\ r \leftarrow \mathsf{PUF}(c)$
$c^* \twoheadleftarrow \mathcal{C};\ r^* \leftarrow \mathsf{PUF}(c^*)$
PUF
1
$(c, r), c^*$
if $r \neq \mathsf{PUF}(c)$ **then abort**
$\mathrm{k} \leftarrow r^*$
else $\mathrm{k} \leftarrow \mathsf{PUF}(c^*)$

Figure 8.7: PUF-based key exchange protocol from [vR12], vulnerable to PUF-substitution attack.

[2] See [BFSK11] for a discussion that an authenticated digital channel is necessary for reasonable protocols.

Our self-guarding protocol. Motivated by the above attack, we draw connections to algorithm substitution attacks, which in this scenario can be more accurately described as token substitution attacks. In Figure 8.8 we propose a PUF-based key exchange protocol that self-guards against subversion of the PUF. It intuitively does so by splitting the initially derived key y into a test part and an evaluation part. This splitting is done via universal hash functions h_{univ}, h'_{univ}, where $h_{\text{univ}}(y)$ and $h'_{\text{univ}}(y)$ act as authentication codes of the key (towards Bob resp. towards Alice), and an extractor h_{extr} which is used to extract sufficiently many random bits $h_{\text{extr}}(y)$ from the remaining bits. Alice transmits $h_{\text{univ}}, h'_{\text{univ}}, h_{\text{extr}}$, and $h_{\text{univ}}(y)$ over the authenticated channel, and Bob checks that the authentication part matches its initially derived key. The receiver replies with its authentication tag.

For our protocol, we consider a PUF that has super-logarithmic input bit size and output size 5λ. If we have a PUF with only λ output bits, we can expand the output size via domain separation, and evaluate the PUF at points $000\|x$, ..., $100\|x$, and concatenates the responses. Furthermore, we model the manufacturing process as a function $\mathsf{create}()$, which creates a new PUF and a unique PUF identifier $pid \in \mathbb{N}$. We denote the concrete PUF then as PUF_{pid}, or following our self-guarding notations, as $\mathsf{PUF}_{\mathsf{GNN}}$. Similarly, we denote a subverted PUF by $\mathsf{PUF}_{\mathsf{SBV}}$. In order to capture the possibility of encapsulated PUF attacks, we also allow create to be called with a malicious algorithm A in which case the PUF evaluates A on the input, or with A and previously created PUF identifiers $pid_1, \dots, pid_n$ in which case A may also call the PUFs with these identifiers as subroutine. We say that a PUF pid' encapsulates a PUF pid if pid' has been created by including pid.

We further denote by $\mathsf{H}[5\lambda, \lambda, p]$ a family of hash functions with input bit size 5λ and output bit size λ, having some property p. Here, p is either being $2^{-\lambda}$-universal, saying that for fixed $r \neq r' \in \{0,1\}^{5\lambda}$ we have $h_{\text{univ}}(r) = h_{\text{univ}}(r')$ with probability at most $2^{-\lambda}$ over the random choice of h_{univ} from the family. Similarly, for property p being a $(3\lambda, 2^{-\lambda})$-extractor we have that $(h_{\text{extr}}, h_{\text{extr}}(y))$ has statistical distance $2^{-\lambda}$ from (h_{extr}, z) for uniform $z \in \{0,1\}^{\lambda}$, as long as y has min-entropy at least 3λ. After losing at most 2λ bits through the authentication tags $h_{\text{univ}}(y)$ and $h'_{\text{univ}}(y)$, we still have this min-entropy left in the uniform value y.

Our result holds with respect to malicious, stateful, and encapsulated PUFs. This is not subsumed by any of the previous results, nor does it contradict any of the impossibility results so far. As for positive results note that the oblivious transfer protocol of Brzuska et al. [BFSK11], in the version of Dachman-Soled et al. [DFK$^+$14], from which one could build a key exchange protocol, does not withstand encapsulating and stateful PUFs. Considering that the physical channel used for transmitting the PUF may not be authenticated, the adversary is now not only able to measure the PUF but can also subvert it, potentially even encapsulating the original PUF, e.g., send PUF_{pid^*} for $pid^* \leftarrow \mathsf{create}(A, pid)$. Even with an authenticated physical channel, a more powerful adversary may be able to gain physical access to the PUF while it is in control of one of the parties for a short time, just enough to subvert it. The impossibility result to build key exchange protocols by van Dijk and Rührmair [vR12] only applies to PUFs which are accessible by the adversary after the execution, a property which we do not consider here.

In some works PUFs are also treated as random functions per se, but we prove the result to hold more generally also for (computationally) pseudorandom functions. In Figure 8.9, we give a simplified game for pseudorandomness in our terminology that suffices for our purposes. Note that unclonability is basically ensured by allowing the adversary to internally create further PUFs. A PUF

```
$\mathsf{KE}^{\mathrm{sg}}.\mathsf{Gen}(1^\lambda)$
$pid \leftarrow \mathsf{create}()$ such that $\mathsf{PUF}_{pid} :\in \mathrm{Fun}[\lambda, 5\lambda]$
return $pid$
```

```
$\mathsf{KE}^{\mathrm{sg}}.\mathsf{Sample}(pid)$
$S \leftarrow \square$
for $i = 1..n$ do
    $x_{\$,i} \twoheadleftarrow \{0,1\}^\lambda$
    $y_{\$,i} \leftarrow \mathsf{PUF}_{pid}(x_{\$,i})$
    $\mathsf{enq}(S, (x_{\$,i}, y_{\$,i}))$
return $S$
```

```
$\mathsf{KE}^{\mathrm{sg}}.\mathsf{Exch}(pid, S)$
Alice$(pid, S)$                                                         Bob
                               $\xrightarrow{\mathsf{PUF}_{pid}}$ (dashed)
                               $\xleftarrow{1}$
                          for $i = 1..n$ do
    $(x, y) \leftarrow \mathsf{deq}(S)$
    $h_{\mathrm{univ}} \twoheadleftarrow \mathsf{H}[5\lambda, \lambda, 2^{-\lambda}\text{-universal}]$
    $h'_{\mathrm{univ}} \twoheadleftarrow \mathsf{H}[5\lambda, \lambda, 2^{-\lambda}\text{-universal}]$
    $h_{\mathrm{extr}} \twoheadleftarrow \mathsf{H}[5\lambda, \lambda, (3\lambda, 2^{-\lambda})\text{-extractor}]$
    $a \leftarrow h_{\mathrm{univ}}(y)$
                               $\xrightarrow{x, h_{\mathrm{univ}}, h'_{\mathrm{univ}}, a, h_{\mathrm{extr}}}$
                                                                            $y \leftarrow \mathsf{PUF}_{pid}(x)$
                                                                            if $a \neq h_{\mathrm{univ}}(y)$ then abort
                                                                            $a' \leftarrow h'_{\mathrm{univ}}(y)$
                               $\xleftarrow{a'}$
    if $a' \neq h'_{\mathrm{univ}}(y)$ then abort
    $\mathrm{k}_i \leftarrow h_{\mathrm{extr}}(y)$                                         $\mathrm{k}_i \leftarrow h_{\mathrm{extr}}(y)$
                          endfor
$\mathrm{k} \leftarrow (\mathrm{k}_1, \ldots, \mathrm{k}_n)$                                        $\mathrm{k} \leftarrow (\mathrm{k}_1, \ldots, \mathrm{k}_n)$
```

Figure 8.8: Self-guarding PUF-based key exchange protocol $\mathsf{KE}^{\mathrm{sg}}$, where PUF has challenge and response space $\{0,1\}^\lambda$ and $\{0,1\}^{5\lambda}$, respectively. Solid arrows denote authenticated digital transmissions, while the dashed arrow denotes a physical transmission.

is pseudorandom if the probability of predicting the challenge bit b, given one challenge-response pair, is negligibly close to $\frac{1}{2}$ in the game. A hybrid argument implies that the same is true if we use n challenge-response values instead of only one, where the advantage over $\frac{1}{2}$ grows by the factor n. Note that the second-stage adversary that gets to see a challenge and the real or a random response, does not anymore have access to the PUF. Otherwise it could trivially win the game by simply evaluating the PUF.

```
Game $\mathrm{IND}^{\mathcal{A}}_{\mathsf{PUF},pid}(1^\lambda)$
$(\mathsf{done}, st) \leftarrow \mathcal{A}^{\mathsf{PUF}_{pid},\mathsf{create}}(1^\lambda)$
$c \twoheadleftarrow \mathcal{C}$
$r_0 \leftarrow \mathsf{PUF}_{pid}(c)$
$r_1 \twoheadleftarrow \{0,1\}^{|r_0|}$
$b \twoheadleftarrow \{0,1\}$
$b' \twoheadleftarrow \mathcal{A}^{\mathsf{create}}(st, c, r_b)$
return $(b = b')$
```

Figure 8.9: Game for pseudorandomness of PUF responses with the constants $\alpha := 2$ and $\delta := 1$ for the advantage.

We now need to define the targeted security property of a key exchange protocol for self-guarding. Since we assume authenticated digital transmissions, the adversary may read but cannot tamper with the transmissions (beyond replacing the PUF). We are interested in a strong type of key confidentiality, namely, that the adversary cannot even distinguish keys from random, as well as robustness in the sense that, if both parties compute their keys, then they also hold the same key. In the security game we, therefore, give the adversary a transcript of a run of the key exchange protocol (where the adversary may have replaced the PUF before, however), and hand over the n keys derived by one party, or random values instead. The choice is made according to some secret bit b. We declare the adversary to win if it either manages to predict b, or to make both parties accept with different keys (in which case we hand over b, unifying the threshold to the guessing probability of $\frac{1}{2}$ for both cases).

The game is formalized in Figure 8.10. With a slight abuse of notation, this game assumes that a key exchange protocol returns a transcript transc of the communication together with the generated keys. Note that IND-KEY is needed in the second phase of the self-guarding game. In the first phase, one creates the PUF (and a unique identifier), then samples challenge and responses, while in the second phase the adversary may subvert the PUF and then plays the robust key indistinguishability game for a key exchange protocol using either the original or the subverted PUF.

Theorem 8.4. *Our key exchange protocol* $\mathsf{KE}^{\mathsf{sg}}$ *from Figure 8.8 is self-guarding with respect to the robust key indistinguishability game* IND-KEY*, against subversion of* PUF*, if the initial* PUF *is pseudorandom.*

Proof. Consider an adversary $\mathcal{A}$ playing the self-guarding game $\mathrm{SGuard}(\text{IND-KEY})^{\mathcal{A},\beta}_{\mathsf{KE}^{\mathsf{sg}}\mathsf{PUF}}$. We argue that $\mathcal{A}$'s success probability in distinguishing keys, established by the protocol, from random strings is negligible, regardless of which value β takes. We let q denote the number of queries which $\mathcal{A}$ makes to the original $\mathsf{PUF}_{\mathsf{GNN}}$ itself, before it reaches Bob.

```
$\text{IND-KEY}^{\mathcal{A}}_{\mathsf{KE},P_s,P_p,S}(1^\lambda)$
$(\mathrm{k}_1^0,\ldots,\mathrm{k}_n^0,\mathsf{transc}) \twoheadleftarrow \mathsf{KE.Exch}(P_s,P_p,S)$
$b \twoheadleftarrow \{0,1\}$
if $\mathrm{k}^0_{i,\text{Alice}} \neq \mathrm{k}^0_{i,\text{Bob}}$ and $\mathrm{k}^0_{i,\text{Alice}},\mathrm{k}^0_{i,\text{Bob}} \neq \bot$ for some $i$ then $a \leftarrow b$ else $a \leftarrow \bot$ fi
$(\mathrm{k}_1^1,\ldots,\mathrm{k}_n^1) \twoheadleftarrow K^n$
$b' \leftarrow \mathcal{A}(P_p,\mathrm{k}_1^b,\ldots,\mathrm{k}_n^b,\mathsf{transc},a)$
return $(b = b')$
```

Figure 8.10: The robust key-indistinguishability game IND-KEY with associated constants $\alpha := 2$ and $\delta := 1$ for a key exchange protocol KE. Here $\mathrm{k}_i^0 = \mathrm{k}^0_{i,\text{Alice}}$ and $\mathrm{k}^0_{i,\text{Bob}}$ denote the key output in the i-th execution by Alice and Bob, respectively. We assume that keys are set to $\bot$ for non-accepting executions; transc denotes the communication between by all parties in all executions; K denotes the key space.

We first note that, instead of using a pseudorandom $\mathsf{PUF}_{\mathsf{GNN}}$, we may equally well use a truly random PUF. If this would decrease $\mathcal{A}$'s success probability significantly, then we would immediately derive a contradiction to the pseudorandomness of the PUF.

Next, we argue that, if the adversary does not encapsulate $\mathsf{PUF}_{\mathsf{GNN}}$ in $\mathsf{PUF}_{\mathsf{SBV}}$ then, except with negligible probability, neither party will accept in any of the n runs. To see this note that the probability that $\mathcal{A}$ queries $\mathsf{PUF}_{\mathsf{GNN}}$ on any of the n challenge values x, before *some* $\mathsf{PUF}_{\mathsf{SBV}}$ is delivered to Bob, is at most $nq \cdot 2^{-\lambda}$. Condition now on the event that such a query has not happened.

In the moment when $\mathsf{PUF}_{\mathsf{SBV}}$ is handed over, and by the authenticated 1-acknowledgement sent by Bob this happens before the adversary gets to learn the challenges, each value $y \leftarrow \mathsf{PUF}_{\mathsf{GNN}}(x)$ is distributed independently of the function in $\mathsf{PUF}_{\mathsf{SBV}}$. Here we use that $\mathsf{PUF}_{\mathsf{SBV}}$ does not encapsulate $\mathsf{PUF}_{\mathsf{GNN}}$. Hence, for each challenge, except with probability of at most $2^{-5\lambda}$, the random response y is different from the response y' computed by $\mathsf{PUF}_{\mathsf{SBV}}$. In this case, with probability at most $2 \cdot 2^{-\lambda}$ by the property of the universal hash functions $h_{\text{univ}}, h'_{\text{univ}}$ (also chosen independently of y, y'), either of the authentication tags a or a' complies with the expected answer. Summing over all n challenges implies that only with negligible probability $\mathcal{A}$ can afford to not encapsulate $\mathsf{PUF}_{\mathsf{GNN}}$ and still make either party accept in any execution.

If, on the other hand, $\mathsf{PUF}_{\mathsf{SBV}}$ encapsulates $\mathsf{PUF}_{\mathsf{GNN}}$, then the adversary cannot determine any of the random value $y \leftarrow \mathsf{PUF}_{\mathsf{GNN}}(x)$, since we have already ruled out that it has queried x before. Since the value contains 5λ bits of min-entropy, and we lose at most 2λ bits through the two hash values a, a', the extractor ensures that k is $2^{-\lambda}$ close to uniform, given h_{extr}. The statistical distance of all n independent samples is then given by $n \cdot 2^{-\lambda}$.

The same line of reasoning for the case that the substituted PUF does not encapsulate the original one, shows that the adversary cannot make the parties accept but for different keys, except with negligible probability. For this note that they can only derive distinct keys if they end up with different values $y \neq y'$. Here, the universality of h_{univ} and h'_{univ} and the fact that the responses are determined independently of the choice of the hash function again imply that the probability of such a collision with the expected value a or a' is at most $2 \cdot 2^{-\lambda}$.

In conclusion, we obtain that the success probability of any adversary $\mathcal{A}$ in winning IND-KEY against KE^{sg} is negligibly close to $\frac{1}{2}$. □

Note that the hashing steps are crucial for security. If one would, say, simply divide y into strings

$a\|a'\|\mathrm{k}$ of lengths λ, λ and 3λ, respectively, then the adversary could send an encapsulated PUF which agrees upon the first 2λ bits but returns a different part k. In this case both parties would accept but with distinct keys. Even worse, Bob's key part k may be easy to predict for the adversary.

Chapter 9

Conclusion and Open Problems

Reviving cryptography at times of surveillance has been receiving increased attention in recent years. I hope that this thesis has made at least a small step towards this goal and away from a future in 1984. There are still numerous open questions and unsolved problems in this area. Backdoor and subversion-resilient constructions of many primitives remain to be built and further realistic assumptions for achieving security remain to be explored. Below we describe a few open problems that are closely related to the topics studied in this thesis, categorized roughly by the three different models that we considered.

Standard-Model Backdoored Hash Functions

A natural open question is, whether immunizing hash functions is possible in the standard model without using secret keys and without relying on strong assumptions. Since adversaries can use malicious inputs to trigger misbehavior in backdoored hash functions, a generic solution that applies a public transformation to the inputs to destroy their potentially malicious structure (similar to the solutions we discussed for HMAC and HKDF in Chapter 5) does not seem to exist for hash functions without secret keys or further assumptions. Furthermore, the security proofs in the 2-BRO and 3-BRO models (i.e., our positive results in Chapters 6 and 7) build on results in communication complexity theory which are somewhat non-trivial and unlikely to be useful for proofs in the standard model. We also showed, in Section 3.4, an impossibility result for $(0, k)$-combiners in the standard model. Overall, although building backdoor-resilient standard-model hash functions seems to be a challenging task, impossibility results can often be overcome. In particular, there may be immunization strategies that are specific to certain applications of hash functions or with respect to less powerful backdoors. We leave the question of finding secure constructions outside idealized models of computation open.

In order to facilitate detection of backdoored functions in practice and design hash functions that inherently resist backdoors, it is helpful to understand the various ways backdoors can be embedded in hash functions. Our constructions in Chapter 4 show that it is mathematically feasible to embed a very powerful backdoor in a hash function, while not affecting its efficiency. A backdoored version of SHA-1, studied by Albertini et al. [AAE$^+$14], somewhat shows the other extreme, where the backdoor gives only a single collision. However, their modification of SHA-1 is relatively unsuspicious, except that its internal constants look arbitrary and are not NUMS-numbers. An important direction here is to study further backdooring strategies with different variations in such a trade-off between the adversaries' exploitation power and their deniability.

Moreover, it is unclear whether constructions of powerfully backdoored hash functions are possible which do not expose their backdoor key in adversarial inputs, unlike our constructions in Chapter 4, and do not rely on public-key primitives directly or the usage of obfuscation to hide a secret key in the hash function and use it to internally decrypt malicious triggers. In other words, it is interesting to investigate the exposure of the backdoor key (when used by the adversary) and to what extend such exposure is inherent to non-public-key-based and efficient backdoored hash functions.

Backdoored Random Oracles

We introduced the BRO model in Section 3.3 as an extension to the well-known random-oracle model, where a backdoor oracle exists that can compute functions of the random oracle for adversaries. We observed that backdoor oracles also allow for non-polynomial-time computation and if unrestricted, they can be, somewhat counterintuitively, used to solve computationally hard problems, such as factoring or finding discrete logarithms. Despite this, in a setting where the computational assumption holds relative to the backdoor oracles, positive results may be provable. To achieve this, we can for example look for meaningful restrictions of the backdoor capability class. Another promising avenue is to rely on an independent idealized model such as the generic-group model (GGM) and for instance, prove IND-CCA security of Hashed ElGamal in the k-BRO and (backdoor-free) GGM models.

In order to overcome the bounded-switch restriction in our indifferentiability proofs of Chapter 7 and prove full indifferentiability, one would require an improved decomposition technique which fixes considerably less points after each backdoor query. This seems to be a challenging task. In particular, such a result is likely to simultaneously prove known communication complexity lower bounds for a host of problems, such as set-disjointness and set-intersection, and potentially provide a first lower bound for the conjecturally hard problem of multi-set double-intersection, which we defined in Section 6.4. Indeed the xor combiner and the extractor combiner may achieve security well beyond what was established in this thesis. Furthermore, as the extractor combiner suggests, the form of the combiner and the number of available BROs can also affect the overall bounds and it is worth taking such ideas into account.

Some of the other open problems that are closer to work in communication complexity involve improving or finding lower bounds for the problems that we use in Chapter 6. For instance, lower bounds for the set-disjointness problem that do not assume a small error would improve the security and/or efficiency of our PRG constructions. Moreover, we do not currently have a lower bound for the multi-set double-intersection problem that we need for proving collision resistance of combiners.

As discussed before, achieving security in the 1-BRO model with respect to an unstricted backdoor capability class is impossible. A natural question here is, hence, what are the maximal backdoor capabilities in the 1-BRO model, under which hardness can be bootstrapped. Furthermore, one can additionally rely on hardness assumptions, as done by Golovnev et al. [GGH$^+$19] who rely on the hardness of a preprocessing version of the 3SUM problem to build from a random oracle a one-way function that is secure even when a massive amount of auxiliary information is available to the adversary.

Finally, we leave defining backdoored variants of other ideal objects such as ideal ciphers and random permutations, as well as studying combiners for them, for future work.

Self-Guarding Primitives

Currently the biggest concern for our self-guarding constructions of Chapter 8 is to improve their efficiency. For the self-guarding public-key and symmetric encryption schemes it is less the computational overhead, but rather that one can encrypt securely only as long as fresh samples are available. More generally, in face of stateful ASAs, it seems difficult to fully revive the security of cryptographic schemes using a small set of secure samples. Recall that involved techniques such as re-randomization of samples, which quasi means to implement one's own encryption algorithm, should be avoided. In this context, one can instead study the possibility of reusing samples after each system reboot to protect against stateful subversions that can only use a volatile memory to store their states.

When considering weaker ASAs, such as stateless ones, better solutions may exist. Indeed our self-guarding signature scheme can be applied an unbounded number of times for stateless subverted algorithms. However, it requires many calls to the signature algorithm and produces large signatures. Here, using specific signature schemes may be helpful in overcoming these limitations.

In terms of efficiency, our self-guarding PUF-based key exchange protocol is reasonably fast. It remains an interesting open question if other PUF-based protocols, e.g., for oblivious transfer (OT), can be self-guarded. As for negative results, Rührmair [Rüh16] argues that the strategy of interleaving test and evaluation challenges fails for the OT protocol of Dachman-Soled et al. [DFK$^+$14]. But this attack is based on the specific OT protocol where the adversary has some control over the input to the PUFs. An option may be to use a different OT protocol where the adversary has less influence on the inputs fed into the PUF.

A different, yet also promising, approach to combat ASAs is to combine multiple implementations of the same scheme or alternatively, different schemes. How security can be achieved in a setting where all schemes are subverted and to what extent this is possible is an interesting question to study. Nonetheless, in case at least one secure implementation is available (even when not knowing which one), existing $(1, k)$-combiners, e.g., those studied by Herzberg in [Her02] or the ones that we studied for BROs, can also be useful against ASAs.

Bibliography

[AAB$^+$97] Hal Abelson, Ross J. Anderson, Steven M. Bellovin, Josh Benaloh, Matt Blaze, Whitfield Diffie, John Gilmore, Peter G. Neumann, Ronald L. Rivest, Jeffrey I. Schiller, and Bruce Schneier. The risks of key recovery, key escrow, and trusted third-party encryption. *World Wide Web Journal*, 2:241–257, 1997.

[ABD$^+$15] David Adrian, Karthikeyan Bhargavan, Zakir Durumeric, Pierrick Gaudry, Matthew Green, J. Alex Halderman, Nadia Heninger, Drew Springall, Emmanuel Thomé, Luke Valenta, Benjamin VanderSloot, Eric Wustrow, Santiago Zanella-Béguelin, and Paul Zimmermann. Imperfect forward secrecy: How Diffie-Hellman fails in practice. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015: 22nd Conference on Computer and Communications Security*, pages 5–17, Denver, CO, USA, October 12–16, 2015. ACM Press.

[AAE$^+$14] Ange Albertini, Jean-Philippe Aumasson, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Malicious hashing: Eve's variant of SHA-1. In Antoine Joux and Amr M. Youssef, editors, *SAC 2014: 21st Annual International Workshop on Selected Areas in Cryptography*, volume 8781 of *Lecture Notes in Computer Science*, pages 1–19, Montreal, QC, Canada, August 14–15, 2014. Springer, Heidelberg, Germany.

[AY15] Riham AlTawy and Amr M. Youssef. Watch your constants: malicious streebog. *IET Information Security*, 9(6):328–333, 2015.

[ABD$^+$13] Elena Andreeva, Andrey Bogdanov, Yevgeniy Dodis, Bart Mennink, and John P. Steinberger. On the indifferentiability of key-alternating ciphers. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 531–550, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

[Arc13] Snowden Surveillance Archive. `https://snowdenarchive.cjfe.org/greenstone/cgi-bin/library.cgi`, 2013. Archive of documents leaked by Edward Snowden, maintained by Canadian Journalists for Free Expression (CJFE) and the Politics of Surveillance Project at the Faculty of Information at the University of Toronto.

[AMSY16] Frederik Armknecht, Daisuke Moriyama, Ahmad-Reza Sadeghi, and Moti Yung. Towards a unified security model for physically unclonable functions. In Kazue Sako, editor, *Topics in Cryptology – CT-RSA 2016*, volume 9610 of *Lecture Notes in Computer*

Science, pages 271–287, San Francisco, CA, USA, February 29 – March 4, 2016. Springer, Heidelberg, Germany.

[AP19a] Marcel Armour and Bertram Poettering. Substitution attacks against message authentication. *IACR Transactions on Symmetric Cryptology*, 2019(3):152–168, 2019.

[AP19b] Marcel Armour and Bertram Poettering. Subverting decryption in AEAD. In Martin Albrecht, editor, *17th IMA International Conference on Cryptography and Coding*, volume 11929 of *Lecture Notes in Computer Science*, pages 22–41, Oxford, UK, December 16–18, 2019. Springer, Heidelberg, Germany.

[AFMV19] Giuseppe Ateniese, Danilo Francati, Bernardo Magri, and Daniele Venturi. Public immunization against complete subversion without random oracles. In Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung, editors, *ACNS 19: 17th International Conference on Applied Cryptography and Network Security*, volume 11464 of *Lecture Notes in Computer Science*, pages 465–485, Bogota, Colombia, June 5–7, 2019. Springer, Heidelberg, Germany.

[AMV15] Giuseppe Ateniese, Bernardo Magri, and Daniele Venturi. Subversion-resilient signature schemes. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015: 22nd Conference on Computer and Communications Security*, pages 364–375, Denver, CO, USA, October 12–16, 2015. ACM Press.

[ABK18] Benedikt Auerbach, Mihir Bellare, and Eike Kiltz. Public-key encryption resistant to parameter subversion and its realization from efficiently-embeddable groups. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018: 21st International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 10769 of *Lecture Notes in Computer Science*, pages 348–377, Rio de Janeiro, Brazil, March 25–29, 2018. Springer, Heidelberg, Germany.

[AGK20] Benedikt Auerbach, Federico Giacon, and Eike Kiltz. Everybody's a target: Scalability in public-key encryption. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part III*, volume 12107 of *Lecture Notes in Computer Science*, pages 475–506, Zagreb, Croatia, May 10–14, 2020. Springer, Heidelberg, Germany.

[Aum11] Jean-Philippe Aumasson. Eve's SHA3 candidate: malicious hashing. 2011.

[BFS86] László Babai, Peter Frankl, and Janos Simon. Complexity classes in communication complexity theory (preliminary version). In *27th Annual Symposium on Foundations of Computer Science*, pages 337–347, Toronto, Ontario, Canada, October 27–29, 1986. IEEE Computer Society Press.

[BKOV17] Saikrishna Badrinarayanan, Dakshita Khurana, Rafail Ostrovsky, and Ivan Visconti. Unconditional UC-secure computation with (stronger-malicious) PUFs. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 382–411, Paris, France, April 30 – May 4, 2017. Springer, Heidelberg, Germany.

[BBG13] James Ball, Julian Borger, and Glenn Greenwald. Revealed: how US and UK spy agencies defeat internet privacy and security. `http://www.theguardian.com/world/2013/sep/05/nsa-gchq-encryption-codes-security`, September 2013.

[BYJKS02] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. In *43rd Annual Symposium on Foundations of Computer Science*, pages 209–218, Vancouver, BC, Canada, November 16–19, 2002. IEEE Computer Society Press.

[BKS$^+$05] Boaz Barak, Guy Kindler, Ronen Shaltiel, Benny Sudakov, and Avi Wigderson. Simulating independence: new constructions of condensers, ramsey graphs, dispersers, and extractors. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 1–10, Baltimore, MA, USA, May 22–24, 2005. ACM Press.

[BFM18a] Balthazar Bauer, Pooya Farshim, and Sogol Mazaheri. Combiners for backdoored random oracles. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, pages 272–302, 2018. **Part of this thesis.**

[BFM18b] Balthazar Bauer, Pooya Farshim, and Sogol Mazaheri. Combiners for backdoored random oracles. Cryptology ePrint Archive, Report 2018/770, 2018. `https://eprint.iacr.org/2018/770`, **Part of this thesis.**

[Bel06] Mihir Bellare. New proofs for NMAC and HMAC: Security without collision-resistance. In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 602–619, Santa Barbara, CA, USA, August 20–24, 2006. Springer, Heidelberg, Germany.

[Bel15] Mihir Bellare. New proofs for NMAC and HMAC: Security without collision resistance. *Journal of Cryptology*, 28(4):844–878, October 2015.

[BCK96a] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In Neal Koblitz, editor, *Advances in Cryptology – CRYPTO'96*, volume 1109 of *Lecture Notes in Computer Science*, pages 1–15, Santa Barbara, CA, USA, August 18–22, 1996. Springer, Heidelberg, Germany.

[BCK96b] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Pseudorandom functions revisited: The cascade construction and its concrete security. In *37th Annual Symposium on Foundations of Computer Science*, pages 514–523, Burlington, Vermont, October 14–16, 1996. IEEE Computer Society Press.

[BHSV98] Mihir Bellare, Shai Halevi, Amit Sahai, and Salil P. Vadhan. Many-to-one trapdoor functions and their relation to public-key cryptosystems. In Hugo Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 283–298, Santa Barbara, CA, USA, August 23–27, 1998. Springer, Heidelberg, Germany.

[BH15] Mihir Bellare and Viet Tung Hoang. Resisting randomness subversion: Fast deterministic and hedged public-key encryption in the standard model. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 627–656, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.

[BJK15] Mihir Bellare, Joseph Jaeger, and Daniel Kane. Mass-surveillance without the state: Strongly undetectable algorithm-substitution attacks. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015: 22nd Conference on Computer and Communications Security*, pages 1431–1440, Denver, CO, USA, October 12–16, 2015. ACM Press.

[BKR16] Mihir Bellare, Daniel Kane, and Phillip Rogaway. Big-key symmetric encryption: Resisting key exfiltration. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 373–402, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany.

[BPR14] Mihir Bellare, Kenneth G. Paterson, and Phillip Rogaway. Security of symmetric encryption against mass surveillance. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 1–19, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany.

[BR14] Mihir Bellare and Todor Ristov. A characterization of chameleon hash functions and new, efficient designs. *Journal of Cryptology*, 27(4):799–823, October 2014.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press.

[BR95] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In Alfredo De Santis, editor, *Advances in Cryptology – EUROCRYPT'94*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111, Perugia, Italy, May 9–12, 1995. Springer, Heidelberg, Germany.

[BR06] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Heidelberg, Germany.

[BCC$^+$15] Daniel J. Bernstein, Tung Chou, Chitchanok Chuengsatiansup, Andreas Hülsing, Eran Lambooij, Tanja Lange, Ruben Niederhagen, and Christine van Vredendaal. How to manipulate curve standards: A white paper for the black hat http: //bada55.cr.yp.to. In *Security Standardisation Research - Second International Conference, SSR 2015, Tokyo, Japan, December 15-16, 2015, Proceedings*, pages 109–139, 2015.

[BLN16] Daniel J. Bernstein, Tanja Lange, and Ruben Niederhagen. Dual EC: A standardized back door. In *The New Codebreakers - Essays Dedicated to David Kahn on the Occasion of His 85th Birthday*, pages 256–281, 2016.

[BDPV08] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. On the indifferentiability of the sponge construction. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 181–197, Istanbul, Turkey, April 13–17, 2008. Springer, Heidelberg, Germany.

[BDPVA09] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Keccak sponge function family main document. *Submission to NIST (Round 2)*, 3:30, 2009.

[BDPVA11] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. Cryptographic sponge functions. *Submission to NIST (Round 3)*, 2011.

[BK89] Manuel Blum and Sampath Kannan. Designing programs that check their work. In *21st Annual ACM Symposium on Theory of Computing*, pages 86–97, Seattle, WA, USA, May 15–17, 1989. ACM Press.

[BLR90] Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. In *22nd Annual ACM Symposium on Theory of Computing*, pages 73–83, Baltimore, MD, USA, May 14–16, 1990. ACM Press.

[BB06] Dan Boneh and Xavier Boyen. On the impossibility of efficiently combining collision resistant hash functions. In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 570–583, Santa Barbara, CA, USA, August 20–24, 2006. Springer, Heidelberg, Germany.

[BFSK11] Christina Brzuska, Marc Fischlin, Heike Schröder, and Stefan Katzenbeisser. Physically uncloneable functions in the universal composition framework. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 51–70, Santa Barbara, CA, USA, August 14–18, 2011. Springer, Heidelberg, Germany.

[Buc17] Ben Buchanan. Nobody But Us: the rise and fall of the golden age of signals intelligence. *Hoover Institution*, 2017.

[CDL17] Jan Camenisch, Manu Drijvers, and Anja Lehmann. Anonymous attestation with subverted TPMs. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part III*, volume 10403 of *Lecture Notes in Computer Science*, pages 427–461, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145, Las Vegas, NV, USA, October 14–17, 2001. IEEE Computer Society Press.

[CRS$^+$07] Ran Canetti, Ronald L. Rivest, Madhu Sudan, Luca Trevisan, Salil P. Vadhan, and Hoeteck Wee. Amplifying collision resistance: A complexity-theoretic treatment. In

Alfred Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 264–283, Santa Barbara, CA, USA, August 19–23, 2007. Springer, Heidelberg, Germany.

[CP10] Arkadev Chattopadhyay and Toniann Pitassi. The story of set disjointness. *SIGACT News*, 41(3):59–85, September 2010.

[CMG$^+$16] Stephen Checkoway, Jacob Maskiewicz, Christina Garman, Joshua Fried, Shaanan Cohney, Matthew Green, Nadia Heninger, Ralf-Philipp Weinmann, Eric Rescorla, and Hovav Shacham. A systematic analysis of the juniper dual EC incident. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016: 23rd Conference on Computer and Communications Security*, pages 468–479, Vienna, Austria, October 24–28, 2016. ACM Press.

[CNE$^+$14] Stephen Checkoway, Ruben Niederhagen, Adam Everspaugh, Matthew Green, Tanja Lange, Thomas Ristenpart, Daniel J. Bernstein, Jake Maskiewicz, Hovav Shacham, and Matthew Fredrikson. On the practical exploitability of dual EC in TLS implementations. In Kevin Fu and Jaeyeon Jung, editors, *USENIX Security 2014: 23rd USENIX Security Symposium*, pages 319–335, San Diego, CA, USA, August 20–22, 2014. USENIX Association.

[CMY$^+$16] Rongmao Chen, Yi Mu, Guomin Yang, Willy Susilo, Fuchun Guo, and Mingwu Zhang. Cryptographic reverse firewall via malleable smooth projective hash functions. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016, Part I*, volume 10031 of *Lecture Notes in Computer Science*, pages 844–876, Hanoi, Vietnam, December 4–8, 2016. Springer, Heidelberg, Germany.

[Con20] Wikipedia Contributors. FBI–Apple encryption dispute — Wikipedia, the Free Encyclopedia. `https://en.wikipedia.org/wiki/FBI%E2%80%93Apple_encryption_dispute`, 2020.

[CDG18] Sandro Coretti, Yevgeniy Dodis, and Siyao Guo. Non-uniform bounds in the random-permutation, ideal-cipher, and generic-group models. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 693–721, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany.

[CDGS18] Sandro Coretti, Yevgeniy Dodis, Siyao Guo, and John P. Steinberger. Random oracles and non-uniformity. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part I*, volume 10820 of *Lecture Notes in Computer Science*, pages 227–258, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany.

[CDMP05] Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-Damgård revisited: How to construct a hash function. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 430–448, Santa Barbara, CA, USA, August 14–18, 2005. Springer, Heidelberg, Germany.

[CPS08] Jean-Sébastien Coron, Jacques Patarin, and Yannick Seurin. The random oracle model and the ideal cipher model are equivalent. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 1–20, Santa Barbara, CA, USA, August 17–21, 2008. Springer, Heidelberg, Germany.

[CK18] Henry Corrigan-Gibbs and Dmitry Kogan. The discrete-logarithm problem with preprocessing. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 415–447, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany.

[DFK$^+$14] Dana Dachman-Soled, Nils Fleischhacker, Jonathan Katz, Anna Lysyanskaya, and Dominique Schröder. Feasibility and infeasibility of secure computation with malicious PUFs. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part II*, volume 8617 of *Lecture Notes in Computer Science*, pages 405–420, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany.

[Dam90] Ivan Damgård. A design principle for hash functions. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 416–427, Santa Barbara, CA, USA, August 20–24, 1990. Springer, Heidelberg, Germany.

[DFP15] Jean Paul Degabriele, Pooya Farshim, and Bertram Poettering. A more cautious approach to security against mass surveillance. In Gregor Leander, editor, *Fast Software Encryption – FSE 2015*, volume 9054 of *Lecture Notes in Computer Science*, pages 579–598, Istanbul, Turkey, March 8–11, 2015. Springer, Heidelberg, Germany.

[DPSW16] Jean Paul Degabriele, Kenneth G. Paterson, Jacob C. N. Schuldt, and Joanne Woodage. Backdoors in pseudorandom number generators: Possibility and impossibility results. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 403–432, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[Din16] Itai Dinur. New attacks on the concatenation and XOR hash combiners. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 484–508, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.

[DFMT20a] Yevgeniy Dodis, Pooya Farshim, Sogol Mazaheri, and Stefano Tessaro. Towards defeating backdoored random oracles: Indifferentiability with bounded adaptivity. In *TCC 2020: 18th Theory of Cryptography Conference*, 2020. **Part of this thesis.**

[DFMT20b] Yevgeniy Dodis, Pooya Farshim, Sogol Mazaheri, and Stefano Tessaro. Towards defeating backdoored random oracles: Indifferentiability with bounded adaptivity. Cryptology ePrint Archive, Report 2020/1199, 2020. `https://eprint.iacr.org/2020/1199`, **Part of this thesis.**

[DGG$^+$15] Yevgeniy Dodis, Chaya Ganesh, Alexander Golovnev, Ari Juels, and Thomas Ristenpart. A formal treatment of backdoored pseudorandom generators. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 101–126, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.

[DGK17] Yevgeniy Dodis, Siyao Guo, and Jonathan Katz. Fixing cracks in the concrete: Random oracles with auxiliary input, revisited. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part II*, volume 10211 of *Lecture Notes in Computer Science*, pages 473–495, Paris, France, April 30 – May 4, 2017. Springer, Heidelberg, Germany.

[DMS16] Yevgeniy Dodis, Ilya Mironov, and Noah Stephens-Davidowitz. Message transmission with reverse firewalls—secure communication on corrupted machines. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 341–372, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany.

[DRS04] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 523–540, Interlaken, Switzerland, May 2–6, 2004. Springer, Heidelberg, Germany.

[DSSL16] Yevgeniy Dodis, Martijn Stam, John P. Steinberger, and Tianren Liu. Indifferentiability of confusion-diffusion networks. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 679–704, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.

[DFGS15] Benjamin Dowling, Marc Fischlin, Felix Günther, and Douglas Stebila. A cryptographic analysis of the TLS 1.3 handshake protocol candidates. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015: 22nd Conference on Computer and Communications Security*, pages 1197–1210, Denver, CO, USA, October 12–16, 2015. ACM Press.

[DFGS16] Benjamin Dowling, Marc Fischlin, Felix Günther, and Douglas Stebila. A cryptographic analysis of the TLS 1.3 draft-10 full and pre-shared key handshake protocol. Cryptology ePrint Archive, Report 2016/081, 2016. `http://eprint.iacr.org/2016/081`.

[DFS16] Stefan Dziembowski, Sebastian Faust, and François-Xavier Standaert. Private circuits III: Hardware trojan-resilience via testing amplification. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016: 23rd Conference on Computer and Communications Security*, pages 142–153, Vienna, Austria, October 24–28, 2016. ACM Press.

[ElG84] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18, Santa Barbara, CA, USA, August 19–23, 1984. Springer, Heidelberg, Germany.

[EPS15] C. Evans, C. Palmer, and R. Sleevi. Public key pinning extension for HTTP. `https://tools.ietf.org/html/rfc7469`, April 2015. RFC 7469.

[FN91] Amos Fiat and Moni Naor. Rigorous time/space tradeoffs for inverting functions. In *23rd Annual ACM Symposium on Theory of Computing*, pages 534–541, New Orleans, LA, USA, May 6–8, 1991. ACM Press.

[FS87] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194, Santa Barbara, CA, USA, August 1987. Springer, Heidelberg, Germany.

[FIP02] NIST FIPS. 198: The keyed-hash message authentication code (HMAC). *National Institute of Standards and Technology, Federal Information Processing Standards*, page 29, 2002.

[FJM18a] Marc Fischlin, Christian Janson, and Sogol Mazaheri. Backdoored hash functions: Immunizing HMAC and HKDF. In *31st IEEE Computer Security Foundations Symposium, CSF 2018, Oxford, United Kingdom, July 9-12, 2018*, pages 105–118, 2018. **Part of this thesis.**

[FJM18b] Marc Fischlin, Christian Janson, and Sogol Mazaheri. Backdoored hash functions: Immunizing HMAC and HKDF. Cryptology ePrint Archive, Report 2018/362, 2018. `https://eprint.iacr.org/2018/362`, **Part of this thesis.**

[FL07] Marc Fischlin and Anja Lehmann. Security-amplifying combiners for collision-resistant hash functions. In Alfred Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 224–243, Santa Barbara, CA, USA, August 19–23, 2007. Springer, Heidelberg, Germany.

[FLP14] Marc Fischlin, Anja Lehmann, and Krzysztof Pietrzak. Robust multi-property combiners for hash functions. *Journal of Cryptology*, 27(3):397–428, July 2014.

[FM18] Marc Fischlin and Sogol Mazaheri. Self-guarding cryptographic protocols against algorithm substitution attacks. In *31st IEEE Computer Security Foundations Symposium, CSF 2018, Oxford, United Kingdom, July 9-12, 2018*, pages 76–90, 2018. **Part of this thesis.**

[GPR14] Peter Gaži, Krzysztof Pietrzak, and Michal Rybár. The exact PRF-security of NMAC and HMAC. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 113–130, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany.

[GP13] Barton Gellman and Laura Poitras. U.S., British intelligence mining data from nine U.S. Internet companies in broad secret program. `http://www.washingtonpost.com/investigations/us-intelligence-mining-data-from-nine-us-internet-companies-in-broad-secret-program/2013/06/06/3a0c0da8-cebf-11e2-8845-d970ccb04497_story.html`, August 2013.

[GGP10] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 465–482, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Heidelberg, Germany.

[Gol01] Oded Goldreich. *Foundations of Cryptography: Basic Tools*, volume 1. Cambridge University Press, Cambridge, UK, 2001.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[GGH$^+$19] Alexander Golovnev, Siyao Guo, Thibaut Horel, Sunoo Park, and Vinod Vaikuntanathan. 3sum with preprocessing: Algorithms, lower bounds and cryptographic applications. *CoRR*, abs/1907.08355, 2019.

[GGH$^+$20] Alexander Golovnev, Siyao Guo, Thibaut Horel, Sunoo Park, and Vinod Vaikuntanathan. Data structures meet cryptography: 3SUM with preprocessing. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *52nd Annual ACM Symposium on Theory of Computing*, pages 294–307, Chicago, IL, USA, June 22–26, 2020. ACM Press.

[GLM$^+$15] Mika Göös, Shachar Lovett, Raghu Meka, Thomas Watson, and David Zuckerman. Rectangles are nonnegative juntas. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th Annual ACM Symposium on Theory of Computing*, pages 257–266, Portland, OR, USA, June 14–17, 2015. ACM Press.

[Gre14] Glenn Greenwald. *No Place to Hide: Edward Snowden, the NSA, and the U.S. Surveillance State.* Metropolitan Books, USA, 2014.

[GC13] Venkatesan Guruswami and Mahdi Cheraghchi. Set disjointness lower bound via product distribution. Scribes for Information theory and its applications in theory of computation, 2013. `http://www.cs.cmu.edu/~venkatg/teaching/ITCS-spr2013/`.

[HH09] Iftach Haitner and Thomas Holenstein. On the (im)possibility of key dependent encryption. In Omer Reingold, editor, *TCC 2009: 6th Theory of Cryptography Conference*, volume 5444 of *Lecture Notes in Computer Science*, pages 202–219. Springer, Heidelberg, Germany, March 15–17, 2009.

[HK06] Shai Halevi and Hugo Krawczyk. Strengthening digital signatures via randomized hashing. In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 41–59, Santa Barbara, CA, USA, August 20–24, 2006. Springer, Heidelberg, Germany.

[Hel80] Martin E. Hellman. A cryptanalytic time-memory trade-off. *IEEE Trans. Inf. Theory*, 26(4):401–406, 1980.

[Her02] Amir Herzberg. Folklore, practice and theory of robust combiners. Cryptology ePrint Archive, Report 2002/135, 2002. `http://eprint.iacr.org/2002/135`.

[HS08] Jonathan J. Hoch and Adi Shamir. On the strength of the concatenated hash combiner when all the hash functions are weak. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP 2008: 35th International Colloquium on Automata, Languages and Programming, Part II*, volume 5126 of *Lecture Notes in Computer Science*, pages 616–630, Reykjavik, Iceland, July 7–11, 2008. Springer, Heidelberg, Germany.

[HKT11] Thomas Holenstein, Robin Künzler, and Stefano Tessaro. The equivalence of the random oracle model and the ideal cipher model, revisited. In Lance Fortnow and Salil P. Vadhan, editors, *43rd Annual ACM Symposium on Theory of Computing*, pages 89–98, San Jose, CA, USA, June 6–8, 2011. ACM Press.

[HPRV19] Thibaut Horel, Sunoo Park, Silas Richelson, and Vinod Vaikuntanathan. How to subvert backdoored encryption: Security against adversaries that decrypt all ciphertexts. In Avrim Blum, editor, *ITCS 2019: 10th Innovations in Theoretical Computer Science Conference*, volume 124, pages 42:1–42:20, San Diego, CA, USA, January 10–12, 2019. LIPIcs.

[Imp95] Russell Impagliazzo. A personal view of average-case complexity. In *Proceedings of the Tenth Annual Structure in Complexity Theory Conference, Minneapolis, Minnesota, USA, June 19-22, 1995*, pages 134–147. IEEE Computer Society, 1995.

[Jea16] Jérémy Jean. TikZ for Cryptographers. `http://www.iacr.org/authors/tikz/`, 2016.

[Jou04] Antoine Joux. Multicollisions in iterated hash functions. Application to cascaded constructions. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 306–316, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Heidelberg, Germany.

[KLT15] Jonathan Katz, Stefan Lucks, and Aishwarya Thiruvengadam. Hash functions from defective ideal ciphers. In Kaisa Nyberg, editor, *Topics in Cryptology – CT-RSA 2015*, volume 9048 of *Lecture Notes in Computer Science*, pages 273–290, San Francisco, CA, USA, April 20–24, 2015. Springer, Heidelberg, Germany.

[KNTX10] Akinori Kawachi, Akira Numayama, Keisuke Tanaka, and Keita Xagawa. Security of encryption schemes in weakened random oracle models. In Phong Q. Nguyen and David Pointcheval, editors, *PKC 2010: 13th International Conference on Theory and Practice of Public Key Cryptography*, volume 6056 of *Lecture Notes in Computer Science*, pages 403–419, Paris, France, May 26–28, 2010. Springer, Heidelberg, Germany.

[KM13] Neal Koblitz and Alfred Menezes. Another look at HMAC. *J. Mathematical Cryptology*, 7(3):225–251, 2013.

[KMR17] Pravesh K. Kothari, Raghu Meka, and Prasad Raghavendra. Approximating rectangles by juntas and weakly-exponential lower bounds for LP relaxations of CSPs. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *49th Annual ACM Symposium on Theory of Computing*, pages 590–603, Montreal, QC, Canada, June 19–23, 2017. ACM Press.

[Kra10] Hugo Krawczyk. Cryptographic extraction and key derivation: The HKDF scheme. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 631–648, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Heidelberg, Germany.

[KBC97] Hugo Krawczyk, Mihir Bellare, and Ran Canetti. HMAC: Keyed-hashing for message authentication. *RFC 2104*, 1997.

[KE10] Hugo Krawczyk and Pasi Eronen. HMAC-based extract-and-expand key derivation function (HKDF). *RFC 5869*, 2010.

[KR98] Hugo Krawczyk and Tal Rabin. Chameleon hashing and signatures. Cryptology ePrint Archive, Report 1998/010, 1998. `http://eprint.iacr.org/1998/010`.

[KN97] Eyal Kushilevitz and Noam NisanY. *Communication complexity.* Cambridge University Press, 1997.

[LLTT05] Chia-Jung Lee, Chi-Jen Lu, Shi-Chun Tsai, and Wen-Guey Tzeng. Extracting randomness from multiple independent sources. *IEEE Trans. Information Theory*, 51(6):2224–2227, 2005.

[Leh10] Anja Lehmann. *On the Security of Hash Function Combiners.* PhD thesis, TU Darmstadt, März 2010.

[LP19] Gaëtan Leurent and Thomas Peyrin. From collisions to chosen-prefix collisions application to full SHA-1. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part III*, volume 11478 of *Lecture Notes in Computer Science*, pages 527–555, Darmstadt, Germany, May 19–23, 2019. Springer, Heidelberg, Germany.

[LW15] Gaëtan Leurent and Lei Wang. The sum can be weaker than each part. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 345–367, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.

[Li15] Xin Li. Three-source extractors for polylogarithmic min-entropy. In Venkatesan Guruswami, editor, *56th Annual Symposium on Foundations of Computer Science*, pages 863–882, Berkeley, CA, USA, October 17–20, 2015. IEEE Computer Society Press.

[Lis07] Moses Liskov. Constructing an ideal hash function from weak ideal compression functions. In Eli Biham and Amr M. Youssef, editors, *SAC 2006: 13th Annual International Workshop on Selected Areas in Cryptography*, volume 4356 of *Lecture Notes in Computer Science*, pages 358–375, Montreal, Canada, August 17–18, 2007. Springer, Heidelberg, Germany.

[Luc05] Stefan Lucks. A failure-friendly design principle for hash functions. In Bimal K. Roy, editor, *Advances in Cryptology – ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 474–494, Chennai, India, December 4–8, 2005. Springer, Heidelberg, Germany.

[MBH^{+}13] Ewen MacAskill, Julian Borger, Nick Hopkins, Nick Davies, and James Ball. GCHQ taps fibre-optic cables for secret access to world's communications. `https://www.theguardian.com/uk/2013/jun/21/gchq-cables-secret-world-communications-nsa`, June 2013.

[Mau92] Ueli M. Maurer. Conditionally-perfect secrecy and a provably-secure randomized cipher. *Journal of Cryptology*, 5(1):53–66, January 1992.

[MRH04] Ueli M. Maurer, Renato Renner, and Clemens Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 21–39, Cambridge, MA, USA, February 19–21, 2004. Springer, Heidelberg, Germany.

[MS07] Ueli M. Maurer and Johan Sjödin. A fast and key-efficient reduction of chosen-ciphertext to known-plaintext security. In Moni Naor, editor, *Advances in Cryptology – EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 498–516, Barcelona, Spain, May 20–24, 2007. Springer, Heidelberg, Germany.

[MT08] Ueli M. Maurer and Stefano Tessaro. Basing PRFs on constant-query weak PRFs: Minimizing assumptions for efficient symmetric cryptography. In Josef Pieprzyk, editor, *Advances in Cryptology – ASIACRYPT 2008*, volume 5350 of *Lecture Notes in Computer Science*, pages 161–178, Melbourne, Australia, December 7–11, 2008. Springer, Heidelberg, Germany.

[MT10] Ueli M. Maurer and Stefano Tessaro. A hardcore lemma for computational indistinguishability: Security amplification for arbitrarily weak PRGs with optimal stretch. In Daniele Micciancio, editor, *TCC 2010: 7th Theory of Cryptography Conference*, volume 5978 of *Lecture Notes in Computer Science*, pages 237–254, Zurich, Switzerland, February 9–11, 2010. Springer, Heidelberg, Germany.

[MRS09] Florian Mendel, Christian Rechberger, and Martin Schläffer. MD5 is weaker than weak: Attacks on concatenated combiners. In Mitsuru Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 144–161, Tokyo, Japan, December 6–10, 2009. Springer, Heidelberg, Germany.

[Men13] Joseph Menn. Exclusive: Secret contract tied NSA and security industry pioneer. `https://www.reuters.com/article/us-usa-security-rsa-idUSBRE9BJ1C220131220`, December 2013.

[Mer90] Ralph C. Merkle. One way hash functions and DES. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 428–446, Santa Barbara, CA, USA, August 20–24, 1990. Springer, Heidelberg, Germany.

[Mil20] Greg Miller. ‘The intelligence coup of the century’ For decades, the CIA read the encrypted communications of allies and adversaries. `https://www.washingtonpost.com/graphics/2020/world/national-security/cia-crypto-encryption-machines-espionage`, February 2020.

[MS15] Ilya Mironov and Noah Stephens-Davidowitz. Cryptographic reverse firewalls. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 657–686, Sofia, Bulgaria, April 26–30, 2015. Springer, Heidelberg, Germany.

[Mit13] Arno Mittelbach. Cryptophia’s short combiner for collision-resistant hash functions. In Michael J. Jacobson Jr., Michael E. Locasto, Payman Mohassel, and Reihaneh Safavi-Naini, editors, *ACNS 13: 11th International Conference on Applied Cryptography and Network Security*, volume 7954 of *Lecture Notes in Computer Science*, pages 136–153, Banff, AB, Canada, June 25–28, 2013. Springer, Heidelberg, Germany.

[Mor15] Pawel Morawiecki. Malicious Keccak. Cryptology ePrint Archive, Report 2015/1085, 2015. `http://eprint.iacr.org/2015/1085`.

[MB12] Dana Moshkovitz and Boaz Barak. Communication complexity. Scribes for Advanced Complexity Theory, 2012. `https://people.csail.mit.edu/dmoshkov/courses/adv-comp/`.

[NS06] M. Nandi and D. R. Stinson. Multicollision attacks on some generalized sequential hash functions. Cryptology ePrint Archive, Report 2006/055, 2006. `http://eprint.iacr.org/2006/055`.

[NR99] Moni Naor and Omer Reingold. Synthesizers and their application to the parallel construction of pseudo-random functions. *Journal of Computer and System Sciences*, 58(2):336–375, 1999.

[NIT08] Akira Numayama, Toshiyuki Isshiki, and Keisuke Tanaka. Security of digital signature schemes in weakened random oracle models. In Ronald Cramer, editor, *PKC 2008: 11th International Workshop on Theory and Practice in Public Key Cryptography*, volume 4939 of *Lecture Notes in Computer Science*, pages 268–287, Barcelona, Spain, March 9–12, 2008. Springer, Heidelberg, Germany.

[OSVW13] Rafail Ostrovsky, Alessandra Scafuro, Ivan Visconti, and Akshay Wadia. Universally composable secure computation with (malicious) physically uncloneable functions. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 702–718, Athens, Greece, May 26–30, 2013. Springer, Heidelberg, Germany.

[PLS13] Nicole Perlroth, Jeff Larson, and Scott Shane. N.S.A. Able to Foil Basic Safeguards of Privacy on Web. `https://www.nytimes.com/2013/09/06/us/nsa-foils-much-internet-encryption.html`, September 2013.

[PW01] Birgit Pfitzmann and Michael Waidner. A model for asynchronous reactive systems and its application to secure message transmission. In *2001 IEEE Symposium on Security and Privacy*, pages 184–200, Oakland, CA, USA, May 2001. IEEE Computer Society Press.

[PS08] Krzysztof Pietrzak and Johan Sjödin. Weak pseudorandom functions in minicrypt. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP 2008: 35th International Colloquium on Automata, Languages and Programming, Part II*, volume 5126 of *Lecture Notes in Computer Science*, pages 423–436, Reykjavik, Iceland, July 7–11, 2008. Springer, Heidelberg, Germany.

[PS96] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli M. Maurer, editor, *Advances in Cryptology – EUROCRYPT'96*, volume 1070 of *Lecture Notes in Computer Science*, pages 387–398, Saragossa, Spain, May 12–16, 1996. Springer, Heidelberg, Germany.

[RY20] Anup Rao and Amir Yehudayoff. *Communication complexity and applications*. 2020.

[Raz05] Ran Raz. Extractors with weak random seeds. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 11–20, Baltimore, MA, USA, May 22–24, 2005. ACM Press.

[RTV04] Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Notions of reducibility between cryptographic primitives. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 1–20, Cambridge, MA, USA, February 19–21, 2004. Springer, Heidelberg, Germany.

[Res18] Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. `https://tools.ietf.org/html/rfc8446`, August 2018.

[Res14] IACR Statement On Mass Surveillance ("Copenhagen Resolution"). `https://www.iacr.org/misc/statement-May2014.html`, 2014. A statement adopted unanimously by the IACR Members meeting held at Eurocrypt 2014 in Copenhagen on May 14th 2014.

[Rog15] Phillip Rogaway. The moral character of cryptographic work. Cryptology ePrint Archive, Report 2015/1162, 2015. `http://eprint.iacr.org/2015/1162`.

[RS04] Phillip Rogaway and Thomas Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In Bimal K. Roy and Willi Meier, editors, *Fast Software Encryption – FSE 2004*, volume 3017 of *Lecture Notes in Computer Science*, pages 371–388, New Delhi, India, February 5–7, 2004. Springer, Heidelberg, Germany.

[Rüh11] Ulrich Rührmair. Physical turing machines and the formalization of physical cryptography. Cryptology ePrint Archive, Report 2011/188, 2011. `http://eprint.iacr.org/2011/188`.

[Rüh16] Ulrich Rührmair. On the security of PUF protocols under bad PUFs and PUFs-inside-PUFs attacks. Cryptology ePrint Archive, Report 2016/322, 2016. `http://eprint.iacr.org/2016/322`.

[RTYZ16] Alexander Russell, Qiang Tang, Moti Yung, and Hong-Sheng Zhou. Cliptography: Clipping the power of kleptographic attacks. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016, Part II*, volume 10032 of *Lecture Notes in Computer Science*, pages 34–64, Hanoi, Vietnam, December 4–8, 2016. Springer, Heidelberg, Germany.

[RTYZ17] Alexander Russell, Qiang Tang, Moti Yung, and Hong-Sheng Zhou. Generic semantic security against a kleptographic adversary. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017: 24th Conference on Computer and Communications Security*, pages 907–922, Dallas, TX, USA, October 31 – November 2, 2017. ACM Press.

[RTYZ18] Alexander Russell, Qiang Tang, Moti Yung, and Hong-Sheng Zhou. Correcting subverted random oracles. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 241–271, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany.

[SFKR15] Bruce Schneier, Matthew Fredrikson, Tadayoshi Kohno, and Thomas Ristenpart. Surreptitiously weakening cryptographic systems. Cryptology ePrint Archive, Report 2015/097, 2015. `http://eprint.iacr.org/2015/097`.

[SF07] Dan Shumow and Niels Ferguson. On the possibility of a back door in the NIST SP800-90 Dual Ec Prng. `http://rump2007.cr.yp.to/15-shumow.pdf`, 2007. CRYPTO 2007 Rump Session.

[Sim83] Gustavus J. Simmons. The prisoners' problem and the subliminal channel. In David Chaum, editor, *Advances in Cryptology – CRYPTO'83*, pages 51–67, Santa Barbara, CA, USA, 1983. Plenum Press, New York, USA.

[Sim98] Daniel R. Simon. Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In Kaisa Nyberg, editor, *Advances in Cryptology – EUROCRYPT'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 334–345, Espoo, Finland, May 31 – June 4, 1998. Springer, Heidelberg, Germany.

[SBK$^+$17] Marc Stevens, Elie Bursztein, Pierre Karpman, Ange Albertini, and Yarik Markov. The first collision for full SHA-1. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 570–596, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

[Unr07] Dominique Unruh. Random oracles and auxiliary input. In Alfred Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer*

Science, pages 205–223, Santa Barbara, CA, USA, August 19–23, 2007. Springer, Heidelberg, Germany.

[vR12] Marten van Dijk and Ulrich Rührmair. Physical unclonable functions in cryptographic protocols: Security proofs and impossibility results. Cryptology ePrint Archive, Report 2012/228, 2012. `http://eprint.iacr.org/2012/228`.

[vD10] M.E. van Dijk. System and method of reliable forward secret key sharing with physical random functions. `https://www.google.ch/patents/US7653197`, January 26 2010. US Patent 7,653,197.

[WS11] Adam Waksman and Simha Sethumadhavan. Silencing hardware backdoors. In *2011 IEEE Symposium on Security and Privacy*, pages 49–63, Berkeley, CA, USA, May 22–25, 2011. IEEE Computer Society Press.

[Yao79] Andrew Chi-Chih Yao. Some complexity questions related to distributive computing (preliminary report). In *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing*, STOC '79, pages 209–213, New York, NY, USA, 1979. ACM.

[Yao82] Andrew Chi-Chih Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science*, pages 80–91, Chicago, Illinois, November 3–5, 1982. IEEE Computer Society Press.

[YY96] Adam Young and Moti Yung. The dark side of "black-box" cryptography, or: Should we trust capstone? In Neal Koblitz, editor, *Advances in Cryptology – CRYPTO'96*, volume 1109 of *Lecture Notes in Computer Science*, pages 89–103, Santa Barbara, CA, USA, August 18–22, 1996. Springer, Heidelberg, Germany.

[YY97a] Adam Young and Moti Yung. Kleptography: Using cryptography against cryptography. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 62–74, Konstanz, Germany, May 11–15, 1997. Springer, Heidelberg, Germany.

[YY97b] Adam Young and Moti Yung. The prevalence of kleptographic attacks on discrete-log based cryptosystems. In Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pages 264–276, Santa Barbara, CA, USA, August 17–21, 1997. Springer, Heidelberg, Germany.

[YY04] Adam Young and Moti Yung. A subliminal channel in secret block ciphers. In Helena Handschuh and Anwar Hasan, editors, *SAC 2004: 11th Annual International Workshop on Selected Areas in Cryptography*, volume 3357 of *Lecture Notes in Computer Science*, pages 198–211, Waterloo, Ontario, Canada, August 9–10, 2004. Springer, Heidelberg, Germany.

[YY06] Adam Young and Moti Yung. A space efficient backdoor in RSA and its applications. In Bart Preneel and Stafford Tavares, editors, *SAC 2005: 12th Annual International Workshop on Selected Areas in Cryptography*, volume 3897 of *Lecture Notes in Computer Science*, pages 128–143, Kingston, Ontario, Canada, August 11–12, 2006. Springer, Heidelberg, Germany.